Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Nuevos Modelos de Redes Neuronales Evolutivas y Regresión Logística Generalizada utilizando Funciones de Base. Aplicaciones

Tesis Doctoral

Pedro Antonio Gutiérrez Peña

Granada, junio 2009

Editor: Editorial de la Universidad de Granada
Autor: Pedro Antonio Gutiérrez Peña
D.L.: GR. 2610-2009
ISBN: 978-84-692-3875-2

Universidad de Granada

Nuevos Modelos de Redes Neuronales Evolutivas y Regresión Logística Generalizada utilizando Funciones de Base. Aplicaciones

MEMORIA QUE PRESENTA

Pedro Antonio Gutiérrez Peña

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Junio del 2009

DIRECTOR

César Hervás Martínez

TUTOR

Manuel Lozano Márquez

Departamento de Ciencias de la Computación
e Inteligencia Artificial

La memoria titulada “Nuevos Modelos de Redes Neuronales Evolutivas y Regresión Logística Generalizada utilizando Funciones de Base. Aplicaciones”, que presenta D. Pedro Antonio Gutiérrez Peña para optar al grado de Doctor, ha sido realizada dentro del programa de doctorado “Diseño, Análisis y Aplicaciones de Sistemas Inteligentes” del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del Doctor D. César Hervás Martínez.

Granada, junio de 2009

El Doctorando	El Director
Fdo: Pedro Antonio Gutiérrez Peña	Fdo: César Hervás Martínez

Esta Tesis Doctoral ha sido financiada en parte con cargo a los Proyectos **TIN 2005-08386-C05-02** y **TIN2008-06681-C06-03** de la Comisión Interministerial de Ciencia y Tecnología (CICYT), al Proyecto de Excelencia **P08-TIC-3745** de la Junta de Andalucía y con fondos FEDER.

La investigación de Pedro Antonio Gutiérrez Peña fue subvencionada hasta marzo del 2009 por el programa predoctoral de Formación de Personal Universitario (FPU, referencia de beca **AP-2006-01746**) del Ministerio de Educación y Ciencia (MEC).

Agradecimientos

Resulta complicado mostrar el agradecimiento que debo a tanta gente en tan pocas líneas.

En primer lugar, debo dar las gracias al principal responsable de que esta Tesis haya podido realizarse, mi director D. César Hervás. Su apoyo y orientación me han acompañado en mi trabajo desde tiempo atrás y, gracias a su dedicación, he logrado introducirme en el mundo de la investigación, además de hacer de ello una profesión. La realización de este trabajo tampoco hubiera sido posible sin las ideas y el esfuerzo de D. Manuel Lozano.

No puedo dejar de agradecer a mis padres, Rafael y Chary, a mis hermanos, Rafalín, Javier y Charo, a mi tío Pedro Antonio, y, en general, a toda mi familia, la confianza depositada en mí. Esa confianza, junto con su afecto, su motivación, sus ánimos y la educación que me han prestado, supone otro de los pilares fundamentales de este trabajo.

El apoyo de los miembros del Grupo de Investigación AYRNA de la Universidad de Córdoba ha sido indispensable para desarrollar esta investigación, debiendo destacar especialmente la labor de D. Francisco Martínez-Estudillo por aportar muchas de las ideas aquí presentadas y proporcionar una sólida base teórica para el trabajo desarrollado. También debo dar las gracias a su hermano, D. Alfonso Carlos Martínez-Estudillo, ya que su labor de investigación es la base de la de mi Tesis. Por otro lado, los consejos y la ayuda de D. Sebastián Ventura Soto han sido muy importantes en toda mi trabajo. Agradezco también a mis compañeros de laboratorio, Francisco, Manolo, Antonio, Amelia, José Luis, y en especial a Juan Carlos, las ideas y el ambiente de trabajo que, sin duda, me han ayudado a realizar esta Tesis.

Debo acordarme también de los investigadores del Instituto de Agricultura Sostenible, D.ª Francisca López, D. Luis García, D.ª Montserrat Jurado y D. José Manuel Peña, por su ayuda y su paciencia al introducirme en un área totalmente desconocida para mi. Agradezco también la inestimable ayuda de D. Francisco Herrera para la realización de esta Tesis, y los ratos compartidos en congresos, seminarios y cursos con mis compañeros de Granada, Salva, Alberto, Jesús, Julián y Nacho.

Y por último, pero no por ello menos importante, quisiera agradecer a todos mis amigos su amistad, que me ha dado la posibilidad de desconectar, descansar y reponer fuerzas cada día.

Sinceramente, gracias.

Índice

Índice de Acrónimos

Término	Acrónimo
Análisis Discriminante Cuadrático	ADC
Análisis Discriminante Lineal	ADL
Algoritmo Evolutivo	AE
Algoritmo de Retropropagación (*BackPropagation*)	BP
Búsqueda hacia atrás con vuelta a Estados Anteriores (*Backtracking Backward Search*)	BBS
Computación Evolutiva	CE
Función de Base Radial Generalizada (*Generalized RBF*)	GRBFs
Algoritmo de Mínimos Cuadrados con Reasignación de Pesos (*Iterative Reweighted Least Squares*)	IRLS
Levenberg-Marquardt	LM
Clasificador basado en Árboles de Decisión con Modelo Logístico (*Logistic Model Tree*)	LMT
Regresión Logística con todas las Covariables Iniciales y Funciones de Base Radial (*MultiLogistic Initial-RBF regression*)	MLIRBF
Perceptrón Multicapa	MLP
Modelado de Sistemas	MS
Algoritmo de Programación Evolutiva para el entrenamiento de Redes Neuronales (*Neural Net Evolutionary Programming*)	NNEP
Programación Evolutiva	PE
Programación Evolutiva Híbrida	PEH
Programación Evolutiva Híbrida con Clustering	PEHC
Programación Evolutiva Híbrida con Clustering Dinámica	PEHCD
Red Neuronal utilizando Unidades Producto y Funciones de Base Radial	PRBF
Función de Base Radial (*Radial Basis Function*)	RBF
Regresión Logística	RL
Regresión Logística utilizando las Covariables Iniciales y Funciones de Base Radial	RLIRBF
Regresión Logística utilizando las Covariables Iniciales y Unidades Producto	RLIUP
Regresión Logística utilizando sólo Unidades Producto	RLUP
Red Neuronal Artificial	RNA
Red Neuronal de Funciones de Base Radial	RNRBF
Red Neuronal de Unidades Producto	RNUP
Búsqueda hacia atrás con vuelta a Estados Anteriores incorporando Enfriamiento Simulado (*Simulated Annealing Backtracking Backward Search*)	SABBS
Regresión Logística con todas las Covariables Iniciales y Funciones de Base Radial y utilizando un Algoritmo de Simplificación (*SimpleLogistic Initial-RBF regression*)	SLIRBF
Error Estándar de Predicción (*Standard Error of Prediction*) en el Conjunto de Generalización	SEP_G
Red Neuronal con Unidades Sigmoide y Funciones de Base Radial	SRBF

Máquina de Soporte Vectorial (*Support Vector Machine*)	SVM
Unidad Producto	UP
Unidad Sigmoide	US
Red Neuronal con Unidades Sigmoide y Producto	USP

Capítulo 1

Memoria

1.1. Introducción

El Modelado de Sistemas (MS) consiste en cuantificar la relación que existe entre una variable de respuesta o variable dependiente que resulta de interés y una serie de variables independientes o predictoras que están posiblemente relacionadas con dicha variable. Éste es uno de los problemas fundamentales tratados en Estadística: un problema que convencionalmente es diferenciado en dos tareas distintas, *regresión* y *clasificación*, donde la *regresión* implica que la variable de respuesta es continua, mientras que la *clasificación* es utilizada cuando dicha variable es categórica, nominal u ordinal. El resultado del proceso de MS es la generación de modelos, es decir, de abstracciones de la realidad que pueden ser aplicadas tanto para predecir valores de la variable dependiente dados nuevos valores de las variables independientes, como para mejorar nuestra comprensión de la misma.

En ambos tipos problemas (*clasificación* y *regresión*), el objetivo se centra en determinar una relación funcional entre las variables predictoras y la variable (o variables) de respuesta. Tradicionalmente, la resolución de estos problemas se ha abordado usando técnicas de optimización para minimizar una determinada función de error, previo establecimiento por parte del investigador del tipo de modelo a aplicar. En general, no será posible determinar la relación funcional sino es mediante el uso de un conjunto de datos de ejemplo. De esta forma, la relación se modela en términos de alguna función matemática que contiene una serie de parámetros ajustables, cuyos valores se determinan con la ayuda de los datos. De manera genérica, podríamos escribir estas funciones de la forma $y(\mathbf{x}, \boldsymbol{\theta})$, donde $\boldsymbol{\theta}$ denota el vector de parámetros del modelo, y es la variable de respuesta y $\mathbf{x}$ el vector de variables independientes. Una forma de aproximación al MS pasa por considerar funciones lineales en la relación causa-efecto entre $\mathbf{x}$ e y (es decir, utilizar hiper-planos), pero, a pesar de su amplio uso y popularidad, los modelos lineales son habitualmente demasiado restrictivos para capturar de forma precisa la relación que subyace en el problema.

En principio, la búsqueda de modelos adecuados para *clasificación* y *regresión* fue un campo de trabajo prácticamente exclusivo de la Estadística, pero actualmente ha sido retomado por multitud de investigadores de otras áreas, especialmente del área de Ciencias de la Computación e Inteligencia Artificial. Esta necesidad de modelos más avanzados fue reconocida hace años por varios investigadores pioneros (por ejemplo, Rosenblatt, 1956; Whittaker, 1922) que sugirieron modelos de mayor complejidad que podrían haber superado a los modelos lineales, pero que, dada la potencia computacional disponible en aquella época, no tenían la posibilidad de ser implementados. Sin embargo, con el incremento substancial de la potencia computacional disponible actualmente, estos modelos, llamados genéricamente modelos *no lineales*, han ganado mucha popularidad. A pesar de ello, la complejidad que suponen los modelos no lineales unida a la alta dimensionalidad del espacio de las variables independientes de los problemas reales, complica considerablemente el proceso de modelado.

Otra consecuencia de la capacidad de cómputo de la que actualmente se dispone es la facilidad con que los datos pueden almacenarse. Cada día es más importante el análisis de los datos y la extracción de conocimiento de los mismos. De esta forma, el análisis de datos cobra protagonismo en un amplio conjunto de áreas (Hastie y Tibshirani, 1996), como puede ser la Sociología (por ejemplo, en el análisis de los resultados de encuestas en una campaña de mercado), la Medicina (por ejemplo, en la clasificación de trazas provenientes de electro-encefalogramas o en la detección de secuencias genéticas relevantes), la Economía (por ejemplo, en la predicción de la volatilidad de un determinado producto o en la asignación de carteras de inversión óptimas) o el Marketing (por ejemplo, en la predicción de la rentabilidad de un cliente para una compañía o en la asignación de crédito económico para una persona). Los investigadores de Ingeniería Eléctrica, de Ciencias de la Computación o de Bioinformática, entre otras áreas, se ven involucrados en problemas de *clasificación* y *regresión*, conceptos que actualmente se engloban en una amplia colección de términos como Reconocimiento de Patrones (*Pattern Recognition*), Análisis Inteligente de Datos (*Intelligent Data Analysis*), Descubrimiento de Conocimiento (*Knowledge Discovery*), Aprendizaje por Computador (*Machine Learning*) o Minería de Datos (*Data Mining*). Todos estos términos describen técnicas para resolver problemas que, en una gran mayoría de los casos, suponen una *clasificación* o una *regresión*.

Antes de proceder al análisis del conjunto de datos a modelar para clasificar o predecir, existen dos cuestiones fundamentales que influyen en las decisiones a llevar a cabo:

1. ¿Qué tipo de funciones de aproximación deberíamos utilizar o cuáles tenemos disponibles?
2. ¿Cómo sabemos que hemos llegado a la mejor aproximación posible?

Respecto a la primera cuestión, es importante señalar que actualmente existen una enorme cantidad de métodos y modelos y ninguno de ellos se puede considerar que sea mejor que el resto para cualquier problema al que se enfrente. Como ejemplo se puede consultar la discusión planteada por Friedman (1993). Entre los modelos más comúnmente utilizados se encuentran los modelos lineales y los modelos lineales generalizados, interpolación mediante *splines*, Redes Neuronales Artificiales (RNAs), bases de Fourier, *wavelets*, árboles de decisión o modelos de tipo *kernel*. Todos ellos nos proporcionan abstracciones de la realidad que, de una forma explícita, relacionan las variables de respuesta y las variables predictoras.

Dentro del campo específico de la *clasificación*, Lippmann (1996) propone una taxonomía de clasificadores considerando cinco grupos diferenciados: las funciones de densidad de probabilidad, los clasificadores *globales* basados en probabilidad a posteriori (es decir, basados en funciones elementales que tienen un valor distinto de cero sobre una región amplia del espacio de entrada), los clasificadores *locales* basados en probabilidad a posteriori (es decir, basados en funciones elementales que tienen un valor alto distinto de cero sólo sobre una región localizada del espacio de entrada), los métodos basados en en el vecino más cercano y las aproximaciones basadas en la creación de reglas. Todas estas posibilidades nos dan una idea del amplio abanico de modelos de los que se dispone actualmente.

Por otro lado, la aproximación estadística tradicional al campo de la *clasificación* es una aplicación directa de la Teoría de Decisión Bayesiana, y el modelo de Regresión Logística (RL) lineal es quizás uno de sus principales representantes. Como sugirieron Hastie y Tibshirani (1996), una forma obvia de generalizar el modelo de Regresión Logística Lineal es reemplazar los predictores lineales por una versión no paramétrica de los mismos, tales como superficies arbitrarias de *regresión*, superficies más estructuradas de alta dimensión (estimadas utilizando procedimientos como el método MARS de Friedman (1991)) o incluso modelos paramétricos todavía más complejos como pueden ser modelos aditivos o de funciones de base.

Respecto a la segunda cuestión planteada anteriormente (¿cómo sabemos que hemos llegado a la mejor aproximación posible?), un modelo es necesariamente una aproximación a la relación causa-efecto real. En cualquier situación real de análisis de datos, es inviable pensar que disponemos de un modelo que capture a la perfección la relación entre la variable a predecir y las variables predictoras. En todo caso, podremos considerar el "mejor" modelo entre un conjunto de modelos alternativos, que sería el que mejor captura esa relación para el propósito concreto que estamos tratando. Por ejemplo, si nuestro propósito es predecir correctamente una relación basándonos en la minimización del error cuadrático medio de las predicciones obtenidas frente a los valores reales, nuestra decisión estaría sesgada al error obtenido para un determinado conjunto de datos, pero es imposible saber de manera certera si la decisión sería la misma si hubiésemos considerado otra función de error o si esa decisión es extrapolable para nuevos datos pertenecientes a la misma relación.

Con respecto a este último punto, en el MS se consideran, en general, dos fases diferenciadas: *entrenamiento* y *generalización*. La adquisición del modelo y de todos los posibles valores de los parámetros que vayan asociados al mismo se realiza durante la fase de entrenamiento, también conocida como fase de aprendizaje en base a los datos de entrenamiento. Una vez que el modelo ha sido entrenado, éste puede predecir valores asociados a nuevos datos de la misma naturaleza de los que se han utilizado en el entrenamiento. Estos datos forman parte del conjunto de test y a la habilidad de determinar los valores de la variable predictora en dichos datos se le denomina capacidad de generalización. De este modo, uno de los principales objetivos del MS es encontrar un modelo que sea capaz de generalizar de la mejor forma posible.

El que la función real que estamos aproximando quede correctamente representada mediante una función $y(\mathbf{x}, \boldsymbol{\theta})$ lineal con respecto a $\mathbf{x}$ es poco probable. En los problemas de *regresión*, la función $y(\mathbf{x}, \boldsymbol{\theta})$ será, en general, no lineal y no aditiva en $\mathbf{x}$ y utilizar una función lineal es necesariamente una aproximación (no obstante, una aproximación conveniente en bastantes casos). De la misma forma, para los problemas de *clasificación*, una frontera óptima de decisión Bayesiana implica, en última instancia, que utilizamos una transformación monótona lineal en $\mathbf{x}$ de la probabilidad estimada. Esto es, consecuentemente, una aproximación.

El trabajo de investigación desarrollado en la presente Tesis Doctoral aborda distintos enfoques para ir más allá de esta linealidad. De esta forma, supone un estudio sobre nuevos modelos no lineales, tanto de *clasificación* como de *regresión*, capaces de superar las dificultades que los modelos lineales tradicionales encuentran cuando son enfrentados a problemas reales. En concreto, nos centraremos en la combinación de distintas disciplinas de la Inteligencia Artificial y la Estadística, como son las RNAs, la Computación Evolutiva (CE) o la RL.

1.1.1. Redes Neuronales Artificiales

Las RNAs (Bishop, 1996; Haykin, 1998) son una técnica de modelado enormemente flexible fundamentada en la *emulación de los sistemas nerviosos biológicos*, cuya capacidad de cómputo se desarrolla mediante un proceso adaptativo de aprendizaje. Del mismo modo que sucede con muchas otras aproximaciones estadísticas al MS, las RNAs proponen una forma concreta para la función $y(\mathbf{x}, \boldsymbol{\theta})$ y una serie de procedimientos concretos para optimizar los parámetros $\boldsymbol{\theta}$.

Las propiedades y características de las RNAs han hecho de ellas una herramienta usada con frecuencia en la resolución con éxito de problemas reales de gran complejidad, pertenecientes a diferentes áreas, como por ejemplo, el diagnóstico médico, el modelado de datos financieros, la predicción de datos de crecimiento microbiano, la determinación de producción agrícola o malas hierbas utilizando tecnologías de sensores remotos, la determinación de picos en Quimiometría, etc.

Desde un punto de vista formal, las RNAs de transmisión hacia delante y una sola capa oculta no son más que una forma de un modelo de regresión lineal generalizada que considera una combinación lineal de transformaciones no lineales $B_j(\mathbf{x}, \mathbf{w}_j)$ de las variables de entrada, con la siguiente expresión:

$$y(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{M} \beta_j B_j(\mathbf{x}, \mathbf{w}_j) \qquad (1.1)$$

siendo M el número de transformaciones no lineales, $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \mathbf{w}_1, ..., \mathbf{w}_M\}$ el conjunto de parámetros asociados al modelo, $\boldsymbol{\beta} = \{\beta_0, ..., \beta_M\}$ los parámetros asociados a la parte lineal del modelo, $B_j(\mathbf{x}, \mathbf{w}_j)$ cada una las *funciones de base*, $\mathbf{w}_j$ el conjunto de parámetros de cada función de base y $\mathbf{x} = \{x_1, ..., x_k\}$ las variables de entrada o variables independientes del problema. Este tipo de modelos, dentro de los cuales se engloban las RNAs, son conocidos como *modelos lineales de funciones de base* (Bishop, 2006). La regresión polinómica es un ejemplo particular de estos modelos en el que existe una única variable y cada función de base es una potencia de dicha variable, $B_j(\mathbf{x}) = x^j$. Una extensión de esta aproximación, en la que se divide el espacio de entrada en diferentes regiones y se aproxima cada región con un polinomio distinto, es lo que se denomina el método de *regresión* por funciones *spline* (Hastie et al., 2001). Existen muchas otras posibilidades en la elección de la tipología de las funciones de base, por ejemplo, las Funciones de Base Radial (*Radial Basis Functions*, RBFs), que dan lugar a lar Redes Neuronales de Función de Base Radial (RNRBFs) (Lee y Hou, 2002; Bishop, 1991), las funciones de base de Unidad Sigmoide (US) que dan lugar al Perceptrón Multi-capa (MLP) (Cybenko, 1989), o las Unidades Producto (UP) que dan lugar a las Redes Neuronales de UPs (RNUPs) (Durbin y Rumelhart, 1989).

El aprendizaje en RNAs consiste en estimar un valor para el conjunto de parámetros $\boldsymbol{\theta}$ y una arquitectura para la red (es decir, el número de transformaciones no lineales M y el número de conexiones entre los distintos nodos de la red). Suponiendo una arquitectura fija, el aprendizaje de los parámetros ha sido tradicionalmente llevado a cabo mediante algoritmos de optimización basados en *gradiente descendente* de la función de error, tales como el Algoritmo de Retropropagación (*BackPropagation*, BP).

Taxonomía de Modelos de Redes Neuronales según el Tipo de Función de Base

En una RNA, cada función de base tiene como salida el resultado de una función de transferencia cuya entrada es el valor obtenido al aplicar una función de activación sobre las entradas de la red. En general, en esta Tesis Doctoral, consideraremos dos tipos de funciones de activación de neuronas o nodos, las aditivas y las multiplicativas (Martínez-Estudillo, 2005). Estos dos tipos de funciones dan lugar a dos modelos de RNAs, cuyas características resumimos a continuación:

- El *modelo aditivo* es el caso usado con más frecuencia dentro de la teoría de Redes Neuronales. La función de salida de este tipo de nodos es la siguiente:

  $$B_j(\mathbf{x}, \mathbf{w}_j) = h(w_{j0} + w_{j1}x_1 + w_{j2}x_2 + ... + w_{jk}x_k) \tag{1.2}$$

  siendo $\mathbf{w}_j = \{w_{j0}, w_{j1}, ..., w_{jk}\}$ el valor de los coeficientes asociados al nodo j-ésimo, $h(\cdot)$ la función de transferencia y w_{j0} el valor umbral de activación del nodo o sesgo. La elección de la función de transferencia da lugar a distintos tipos de neuronas o nodos aditivos, entre los cuales caben destacar las neuronas umbral o perceptrón (McCulloch y Pitts, 1943) (que utilizan una función tipo escalón), las USs (que utilizan funciones de transferencia logísticas sigmoidales, tangente hiperbólica o arcotangente) y los nodos lineales (cuya función de transferencia es la función identidad).

- El *modelo multiplicativo* es un modelo más reciente, que intenta representar aquellos casos en los que existe una interacción entre las variables y las regiones de decisión no pueden separarse por hiperplanos (Schmitt, 2002). Dentro de este modelo multiplicativo, la alternativa más generalista es la de las UPs, es decir:

  $$B_j(\mathbf{x}, \mathbf{w}_j) = x_1^{w_{j1}} \cdot x_2^{w_{j2}} \cdot ... \cdot x_k^{w_{jk}} \tag{1.3}$$

  siendo en este caso $\mathbf{w}_j = \{w_{j1}, w_{j2}, ..., w_{jk}\}$, ya que el valor umbral carece de sentido para este modelo. Como puede observarse, la función de transferencia es la función identidad. Los pesos $w_{j1}, w_{j2}, ..., w_{jk}$ son números reales, por lo que las UPs generalizan a otras unidades de tipo multiplicativo, como las denominadas unidades de orden superior o unidades sigma-pi (basadas en monomios con exponentes positivos y enteros).

Por otro lado, se pueden considerar dos tipos fundamentales de funciones de base:

- Las *funciones locales* presentan un valor alto distinto de cero sólo sobre una región localizada del espacio de entrada. Un ejemplo de este tipo de funciones son las RBFs. Poseen una mayor capacidad de aproximar datos anómalos aislados, pero una mayor dificultad en entornos globales o cuando el número de variables de entrada es demasiado alto.

- Las *funciones de proyección* son funciones de entorno global (es decir, presentan un valor distinto de cero sobre una región amplia del espacio de entrada), como las USs o las UPs. Al ser globales, presentan dificultades en la aproximación de datos aislados pero suelen actuar mejor en problemas donde la dimensión del espacio de entrada es alta.

Donoho y Johnstone (1989) demuestran la dualidad entre la aproximación basada en funciones de proyección y la basada en funciones tipo *kernel* o locales. De este modo, una función se puede descomponer en dos funciones, una radial y la otra proyectiva, siendo ambas funciones mutuamente excluyentes.

A continuación, se resumirán las características principales de los modelos más utilizados de RNAs *feedforward* o con transmisión de información hacia delante y con una sola capa oculta.

Redes Neuronales de Unidades Sigmoide

Las redes de US o MLPs son aquellas que están formadas por nodos tipo sigmoide en su capa oculta, los cuales siguen un modelo aditivo de proyección con la siguiente función de salida:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \frac{1}{1 + e^{-(w_{j0} + w_{j1} \cdot x_1 + w_{j2} \cdot x_2 + \ldots + w_{jk} \cdot x_k)}} = \frac{1}{1 + e^{-(w_{j0} + \sum_{i=1}^{k} w_{ji} \cdot x_i)}} \tag{1.4}$$

Las redes de US poseen una importante propiedad: la familia de funciones reales que representan es un subconjunto denso en el espacio de las funciones continuas definidas sobre un compacto con la convergencia uniforme. Esta propiedad de densidad significa que pueden aproximar cualquier función continua con suficiente precisión, con tal de que el número de nodos ocultos no esté acotado, y proporciona una sólida base de carácter teórico que ha sido esencial para el desarrollo del estudio y de las aplicaciones de estas redes (Cybenko, 1989; Hornik et al., 1989).

Redes Neuronales de Unidades Producto

Las RNUPs introducidas por Durbin y Rumelhart (1989), son aquellas que están formadas por nodos de tipo UP en su capa oculta, los cuales siguen el modelo multiplicativo de proyección con la siguiente función de salida:

$$B_j(\mathbf{x}, \mathbf{w}_j) = x_1^{w_{j1}} \cdot x_2^{w_{j2}} \cdot \ldots \cdot x_k^{w_{jk}} = \prod_{i=1}^{k} x_i^{w_{ji}} \tag{1.5}$$

siendo en este caso $\mathbf{w}_j = \{w_{j1}, w_{j2}, ..., w_{jk}\}$, ya que en este tipo de RNAs no definimos un sesgo en capa de entrada. Entre las ventajas de las redes UP se encuentran las siguientes (Martínez-Estudillo, 2005):

- Como consecuencia del Teorema de Stone-Weierstrass, se ha demostrado que las redes de UP son aproximadores universales (Martínez-Estudillo et al., 2006b). Obsérvese que las funciones polinómicas de varias variables son un subconjunto de los modelos basados en UPs.
- La posibilidad de usar exponentes negativos permite expresar cocientes entre las variables.
- Durbin y Rumelhart (1989) demostraron que la capacidad de información de una única unidad de tipo producto (medida como la capacidad para el aprendizaje de patrones booleanos aleatorios) es aproximadamente igual a $3N$, comparado con el valor de $2N$ que corresponde a una unidad de tipo aditivo, siendo N el número de entradas de la unidad.
- Los exponentes del modelo son números reales. Esta característica tiene especial importancia, sobre todo si se tiene en cuenta que son frecuentes las situaciones en el modelado de datos reales en las que la relación entre las variables responde a una estructura de tipo potencial, donde las potencias no están restringidas a los números naturales o enteros (Saito y Nakano, 1997).
- Permiten implementar funciones polinómicas de orden superior. Han demostrado buenos resultados en el modelado de datos en los que existen interacciones de diferentes órdenes

entre las variables independientes del problema. De esta forma, cuando existen interacciones entre las variables que intervienen en un determinado fenómeno, las RNUPs permiten obtener modelos más simplificados que las redes MLP, ya que necesitan un número de funciones de base inferior.

- Junto a esto, es posible obtener cotas superiores de la dimensión VC (Vapnik y Chervonenkis, 1971) para redes basadas en UPs similares a las conocidas para las redes MLP (Schmitt, 2002), lo que supone que poseen una capacidad de generalización similar.

- A diferencia de lo que ocurre con las redes MLP o las RNRBFs, las funciones derivadas parciales del modelo obtenido a partir de una red de UPs son funciones del mismo tipo. Este hecho ayuda con frecuencia al estudio de las tasas de cambio de la variable dependiente del modelo con respecto a cada una de las variables, facilitando la interpretabilidad de los modelos.

Como contrapartida, las redes de UPs presentan un inconveniente importante, la superficie de error asociada es especialmente compleja con numerosos óptimos locales y regiones planas, y por tanto existe una mayor probabilidad de que los algoritmos de búsqueda queden atrapados en alguno de los óptimos locales (Engelbrecht y Ismail, 1999). La estructura potencial del modelo provoca que pequeños cambios en los exponentes tengan como consecuencia un cambio significativo en los valores de la función de activación y en la función de error (Schmitt, 2002). Así, los algoritmos de entrenamiento de redes basados en el gradiente quedan con frecuencia y, de forma especial para este tipo de redes, atrapados en óptimos locales. Por ejemplo, está estudiado el hecho de que el clásico Algoritmo de Retropropagación no funciona bien en RNUPs (Janson y Frenzel, 1993).

Esta dificultad relacionada con el entrenamiento, es una de las razones por las que, a pesar de las ventajas anteriormente señaladas, la teoría de RNUPs ha tenido poco desarrollo y han sido menos utilizadas como herramientas para problemas de *clasificación* y de *regresión* que otros tipos de redes (Lippmann, 1989).

Redes Neuronales de Función de Base Radial

Los nodos ocultos de las RNRBFs presentan funciones de transferencia de tipo local o *kernel* (Donoho y Johnstone, 1989; Bishop, 1991). Cada uno de los nodos RBF realiza una *aproximación local independiente* del espacio de búsqueda, normalmente mediante una campana de Gauss. La capa de salida de tipo lineal auna el efecto independiente de cada nodo, sumando cada valor obtenido. La idea es que cada nodo esté situado en el entorno de un punto, el centro, del espacio de búsqueda formado por las variables de entrada y además con un radio determinado. El proceso de aprendizaje de la red de tipo RBF consistirá en ir moviendo dichos nodos a lo largo del espacio de búsqueda, modificando los centros y los radios de los mismos, para ajustarlos de la mejor forma a los datos de entrenamiento.

La función de activación será equivalente a la función de distancia euclídea al patrón de entrada $\mathbf{x}$ (tomando como centro de la RBF el vector $\mathbf{w}_j$) y la función de transferencia será una función de tipo Gaussiana. Por tanto, la función de transferencia sería la siguiente:

$$B_j(\mathbf{x}, \mathbf{w}_j) = e^{-\frac{1}{2}\left(\frac{d(\mathbf{x}, \mathbf{w}_j)}{r_j}\right)^2} \tag{1.6}$$

siendo $\mathbf{w}_j = \{w_{j1}, w_{j2}, ..., w_{jk}\}$ y la distancia definida de la siguiente forma:

$$d(\mathbf{x}, \mathbf{w}_j) = \parallel \mathbf{x} - \mathbf{w}_j \parallel = \sqrt{\sum_{i=1}^{k} (x_i - w_{ji})^2} \quad (1.7)$$

Además se ha demostrado formalmente que las RNRBFs son aproximadores universales (Park y Sandberg, 1993). En comparación con las redes MLP, las redes RBF presentan la ventaja de que cada nodo en la capa oculta es un elemento local en el modelo, lo que hace que para un determinado patrón sólo algunas unidades ocultas se activen. Esta característica facilita el entrenamiento, disminuyendo el número de óptimos locales y regiones planas de la superficie de error, al desaparecer gran parte de las interacciones entre los pesos. Por último, el proceso de entrenamiento de una red MLP consta de una sola etapa, mientras que las redes RBF suelen entrenarse en dos etapas, siendo los pesos de los enlaces de capa de entrada a capa oculta, correspondientes a las funciones de base, aproximados, en primer lugar, mediante técnicas de aprendizaje *no supervisado*, y los pesos de capa oculta a capa de salida, en segundo lugar, mediante métodos *supervisados* (Bishop, 2006).

1.1.2. Modelos Funcionales de RNAs

Es el momento de especificar formalmente el modelo funcional que será considerado para problemas de *regresión* y *clasificación* en esta Tesis Doctoral. Los modelos tratados serán *redes feedforward* con una capa de entrada con las variables independientes del problema, una capa oculta que puede poseer distintos tipos de nodos y una capa de salida lineal. Se tratarán problemas de *regresión* y *clasificación*, siendo el número de salidas igual a uno si el problema es de *regresión* y pudiendo ser mayor que uno si el problema es de *clasificación*.

Modelo Funcional para Problemas de Regresión

Como hemos comentado anteriormente, cuando tratamos un problema de *regresión* solo se incluye un nodo de salida en el modelo. De esta forma, si representamos el conjunto de coeficientes asociados a la RNA con $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \mathbf{W}\}$, el modelo funcional asociado al nodo de salida es el siguiente:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{M} \beta j B_j(\mathbf{x}, \mathbf{w}_j) \quad (1.8)$$

siendo $\boldsymbol{\beta} = \{\beta_0, ..., \beta_M\}$ el conjunto de pesos desde el nodo de salida a los nodos de capa oculta, β_0 el coeficiente de sesgo o umbral de la red, $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$ el conjunto de coeficientes asociados a los nodos de la capa oculta, $\mathbf{w}_j = \{w_{j0}, w_{j1}, ..., w_{jk}\}$ el valor de las conexiones desde el nodo j de la capa oculta hasta la capa de entrada y $B_j(\mathbf{x}, \mathbf{w}_j)$ la salida del nodo j de la capa oculta, dada por la función de base j. De este modo, cualquiera de los tipos de funciones de base presentados en la Sección 1.1.1 podría considerarse en el modelo como $B_j(\mathbf{x}, \mathbf{w}_j)$, teniendo en cuenta, eso si, que dependiendo del tipo de función de base es necesario incluir el valor de sesgo w_{j0} (USs) o no (UPs). Para el caso de las RBFs, una solución común es considerar el centro de la RBF como $\mathbf{c} = \{w_{j1}, ..., w_{jk}\}$ y el radio como $r = w_{j0}$. La arquitectura de estas RNAs para *regresión* se refleja en la Figura 1.1, donde k es el número de variables de entrada del problema y M es el número de nodos de la capa oculta.

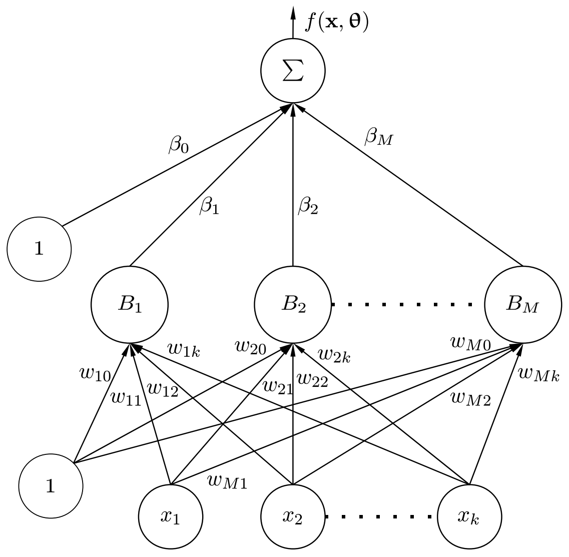


Figura 1.1: Modelo de Red utilizado para problemas de *regresión*

Modelo Funcional para Problemas de Clasificación

Cuando el problema tratado es un problema de *clasificación*, se considera un número de nodos de salida igual al número de clases o categorías de la variable a predecir menos uno. De este modo, si J es el número de clases, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_{J-1}\}$ es el conjunto de coeficientes asociados a la RNA y $\boldsymbol{\theta}_j = \{\boldsymbol{\beta}^j, \mathbf{W}\}$ es el conjunto de parámetros asociados al nodo de salida j, el modelo funcional es el siguiente:

$$f_j(\mathbf{x}, \boldsymbol{\theta}_j) = \beta_0^j + \sum_{i=1}^{M} \beta_i^j B_i(\mathbf{x}, \mathbf{w}_i), \quad 1 \leq j \leq J-1 \tag{1.9}$$

siendo $\boldsymbol{\beta}^j = \{\beta_0^j, \beta_1^j, ..., \beta_M^j\}$ el valor de las conexiones entre capa oculta y capa de salida para el nodo j.

El enfoque adoptado a la hora de interpretar las salidas de los nodos de la capa de salida es un enfoque probabilístico que considera la función de activación *softmax* en cada uno de dichos nodos, la cual viene dada por:

$$g_j(\mathbf{x}, \boldsymbol{\theta}_j) = \frac{e^{f_j(\mathbf{x}, \boldsymbol{\theta}_j)}}{\sum_{i=1}^{J} e^{f_i(\mathbf{x}, \boldsymbol{\theta}_i)}} \tag{1.10}$$

siendo $g_j(\mathbf{x}, \boldsymbol{\theta}_j)$ la probabilidad de que el patrón $\mathbf{x}$ pertenezca a la clase j. La transformación *softmax* produce estimaciones positivas en todas las salidas, siendo la suma de ellas 1, lo que hace que puedan ser interpretadas como la probabilidad de pertenencia a la clase correspondiente. Teniendo en cuenta esta consideración, se puede comprobar que la clase predicha por la RNA, $C(\mathbf{x}, \boldsymbol{\theta})$, es la correspondiente al nodo de la capa de salida cuyo valor de salida es mayor. Expresado formalmente, esto quiere decir que:

$$C(\boldsymbol{\theta}, \mathbf{x}) = \hat{l} \quad \text{donde } \hat{l} = \arg\max_j \, g_j(\mathbf{x}, \boldsymbol{\theta}_j) = \arg\max_j \, f_j(\mathbf{x}, \boldsymbol{\theta}_j), \; 1 \leq j \leq J \tag{1.11}$$

Al tratarse de probabilidades de pertenencia a una clase está claro que no es necesario calcularlas todas, ya que, por ejemplo, la probabilidad de la última salida $g_J(\mathbf{x}, \boldsymbol{\theta}_J)$ se puede

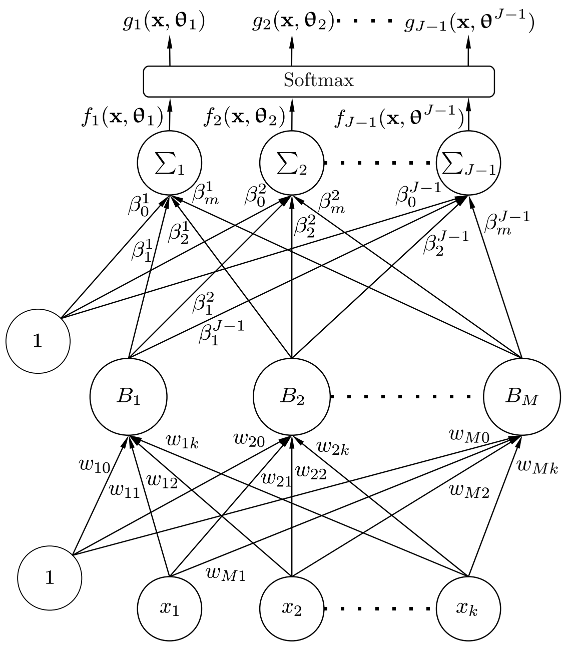


Figura 1.2: Modelo de Red utilizado para problemas de *clasificación*

obtener en función del resto como $1 - \sum_{i=1}^{J-1} g_j(\mathbf{x}, \boldsymbol{\theta}_i)$. De esta forma, podemos simplificar el modelo considerando la salida del último nodo de la capa de salida constante e igual a 0, es decir, $f_J(\mathbf{x}, \boldsymbol{\theta}_J) = 0$. Este nodo no será entrenado, para así reducir el número de coeficientes a estimar y la carga computacional del aprendizaje.

Teniendo en cuenta todas estas consideraciones, la arquitectura de RNA para problemas de *clasificación* se refleja en la Figura 1.2.

1.1.3. Combinación de Funciones de Base

Por otro lado, la utilización de modelos híbridos también ha sido propuesta en diversos trabajos, donde diferentes funciones de activación o transferencia son utilizadas para los nodos de capa oculta. Entre los trabajos existentes, se distinguen dos perspectivas: utilizar una sola capa oculta con distintas funciones de base o utilizar múltiples capas ocultas con el mismo tipo de funciones de base en cada capa aunque de distinta naturaleza entre si, estando todas interconectadas.

De acuerdo con Duch y Jankowski (2001), las funciones de transferencia mixtas dentro de una RNA puede ser introducidas de dos formas. En primer lugar, se podría hacer uso de un método constructivo que seleccione la función más prometedora de un conjunto de candidatas y la vaya añadiendo al modelo (Duch et al., 2001). La segunda forma sería comenzar con una red que contenga diversos tipos de funciones (p.ej. funciones RBF y de tipo US) y aplicar técnicas de poda o regularización para reducir el número de funciones (Duch y Jankowski, 2001).

Por otro lado, las redes neuronales de función de transferencia óptima fueron presentadas como un método para seleccionar las funciones apropiadas para un determinado problema (Jankowski y Duch, 2001), creando arquitecturas de red que aproximan bien determinado tipo de datos además de intentar reducir el tamaño del modelo.

Las redes de enlace funcional presentadas por Pao y Takefuji (1992) suponen una combinación de varios tipos de función, como pueden ser funciones polinomiales, periódicas, sigmoidales y Gaussianas. La idea fundamental tras una red de enlace funcional es la de añadir transformaciones no lineales de las variables de entrada al conjunto original de variables y suprimir la capa oculta del modelo, realizando tan solo una transformación lineal del espacio de entradas obtenido.

Existe una aproximación más compleja que considera varias capas o modelos, cada uno incorporando una estructura de funciones de base y produciendo así un sistema modular. Por ejemplo, Iulian (2002) propone una metodología incluyendo tres módulos: una red de tipo RBF, un Análisis de Componentes Principales y una red de tipo MLP. Otra propuesta de Lehtokangas y Saarinen (1998) considera dos capas ocultas en el modelo, la primera compuesta de funciones de tipo Gaussiano y la segunda de funciones de base tipo US.

Las RNAs híbridas utilizando distintos tipos de funciones de transferencia deben ser, presumiblemente, modelos con un número de nodos ocultos menor, lo que permitiría que la red sea más interpretable. Por ejemplo, un hiper-plano podría ser utilizado para dividir el espacio de entrada en dos clases y otra función Gaussiana adicional para tener en cuenta las anomalías locales. El análisis funcional de una red de tipo MLP entrenada para los mismos datos sería mucho más complejo.

En este contexto, es importante hablar del trabajo de Cohen y Intrator (2002), que se basa en las propiedad de dualidad y de complementación de las funciones basadas en proyección o funcionales *globales* (US y UP) y la tipología *kernel* (unidades RBF). Esta hibridación de modelos ha sido justificada teóricamente por Donoho y Johnstone (1989), que demostraron que cualquier función continua puede ser separada en dos funciones mutuamente excluyentes, tales como una función de tipo radial y otra de tipo cresta (basadas en proyección). Aunque de manera teórica esta descomposición está justificada, en la práctica es difícil la aplicación de métodos de tipo gradiente para separar las distintas localizaciones de una función (de manera que puedan ser ajustadas por unidades de tipo RBF) y, después, estimar el resto de la función por medio de una aproximación basada en proyecciones, todo ello sin quedar atrapados en óptimos locales en el proceso de minimización del error (Friedman, 1991).

Por último, recientemente, Wedge et al. (2006) han presentado una red neuronal híbrida utilizando unidades RBF y de tipo sigmoide, planteando un algoritmo de entrenamiento basado en tres etapas. Esta red ha sido aplicada al problema de aproximación de funciones, con el objetivo de identificar por separado aquellos aspectos de una relación funcional que se pueden expresar de forma global de aquellos que varían únicamente en regiones concretas del espacio de entrada.

1.1.4. Algoritmos Evolutivos para el Diseño de RNAs

La CE (Holland, 1975) es una rama de la ciencia que interpreta la naturaleza como una inmensa máquina de resolver problemas y trata de encontrar el origen de dicha potencialidad, para reutilizarla en sus propias aproximaciones. Los Algoritmos Evolutivos (AEs) son algoritmos de búsqueda basados en el concepto de la *selección natural*. El concepto de AE describe un conjunto de sistemas para resolver problemas mediante el ordenador, que usan un modelo computacional similar a los procesos evolutivos existentes en la naturaleza. Es decir, un modelo basado en el principio darwiniano de reproducción y supervivencia de los individuos que mejor se adaptan al entorno en que viven. Entre ellos, la Programación Evolutiva (PE), propuesta por

Fogel (1966), es un método de optimización estocástico, cuya característica fundamental es que no codifica los individuos, trabajando directamente con sus representaciones. Esta es la razón por la cual no incluye, normalmente, operador de cruce o recombinación (Fogel, 1966).

El teorema de "*No Free Lunch*" propuesto por Wolpert y Macready asegura que *no puede existir un algoritmo capaz de resolver todos los problemas en media mejor que cualquier otro algoritmo* (Wolpert y Macready, 1997). Este teorema motiva a que se sigan buscando nuevos algoritmos de optimización. La utilización de métodos de naturaleza heurística para entrenamiento de RNAs surge en la década de los 90 (Montana y Davis, 1989). Las motivaciones principales que sugieren el uso de este tipo de técnicas vienen determinadas en gran medida por las limitaciones presentadas por los algoritmos clásicos de aprendizaje:

- Imposibilidad de calcular el gradiente cuando la función de activación de los nodos no es derivable.
- Ausencia de convergencia de los algoritmos clásicos de entrenamiento, cuando el número de bits utilizados para representar la función de activación o los pesos de la red (precisión) no es suficientemente grande.
- Tendencia, por parte de los algoritmos de entrenamiento, a obtener excesivas soluciones no óptimas en cada ejecución.

Algunos de los métodos heurísticos que se han utilizado para entrenar RNAs han sido, entre otros, la búsqueda tabú (Sexton et al., 1998) o el método de enfriamiento simulado (Sexton et al., 1999). También se han propuesto diversas variantes heurísticas del algoritmo BP. Por ejemplo, la regla *delta-delta* o la regla *delta-bar-delta* (Jacobs, 1988). Otras alternativas se basan en mejorar la velocidad de convergencia del algoritmo BP (Cichocki y Unbehauen, 1993), en el uso del gradiente conjugado (Johansson et al., 1992), el algoritmo RProp (Riedmiller y Braun, 1993), algoritmos de tipo Newtoniano (Battiti, 1992) (entre los que encontramos el QuickProp, Fahlman y Lebiere, 1990) o Quasi-Newtoniano (algoritmo Davidon-Fletcher-Powell, Davidon, 1991) y métodos basados en el algoritmo de Levenberg-Marquardt (Hagan y Menhaj, 1994). Sin embargo, los algoritmos heurísticos que más impacto han tenido, relacionados con diversas arquitecturas y problemas de RNAs, han sido los basados en AEs.

De la utilización de los AEs para la optimización de RNAs surgen las denominadas Redes Neuronales Artificiales Evolutivas (Yao, 1999). Este área de investigación ha atraído mucho interés en la última década en la comunidad científica de RNAs, proporcionando una herramienta muy adecuada para la optimización tanto de la topología como de los pesos de los modelos de RNAs (Angeline et al., 1994; Blanco et al., 2001). En un gran número de casos, los resultados y las conclusiones obtenidas han resultado ser prometedores, superando la precisión obtenida por los algoritmos clásicos basados en el gradiente de la función de error (Martínez-Estudillo et al., 2006b; Yao, 1999; Angeline et al., 1994). Si tenemos en cuenta la dificultad que implica el establecimiento de una arquitectura adecuada junto con el aprendizaje de los correspondientes pesos de la red neuronal, la aplicación de AEs para el diseño de la estructura y la optimización de los coeficientes de una RNA está ampliamente justificada.

Existen tres formas de abordar el entrenamiento de una RNA con un AE:

- Evolucionar los pesos de la RNA: en este caso se fija la arquitectura de la red y se desarrolla una búsqueda global en el espacio de pesos. Los pesos son tradicionalmente codificados con codificación real. En comparación con los algoritmos BP clásicos, suelen ser computacionalmente más costosos, pero no requieren el cálculo del gradiente.

- Evolucionar la arquitectura de la RNA: en este caso se parte de una inicialización aleatoria de los pesos y, tras la ejecución del AE, se suelen realizar varias iteraciones de un algoritmo BP clásico. La codificación más natural para esta tarea es la codificación binaria. En todo caso, la optimización de la arquitectura es un proceso difícil, dando únicamente resultados moderados en problemas relativamente pequeños.

- Evolucionar simultáneamente tanto los pesos como la arquitectura de la RNA: esta estrategia pretende reunir las ventajas de las dos anteriores.

Para el caso de las redes MLP, son numerosos los trabajos en los que se aplican distintos tipos de AEs. Una buena revisión de las distintas técnicas evolutivas que han sido aplicadas para esta tipología de red se puede encontrar en el trabajo de Yao (1999). La evolución de los centros y radios de las redes de RBF también ha tenido gran aceptación (Whitehead y Choate, 1996) y recientemente han aparecido múltiples trabajos y mejoras (González et al., 2003; Manrique et al., 2006).

En el caso de las RNUPs, son menos las referencias que se pueden encontrar en las que se apliquen AEs. Janson y Frenzel (1993) diseñan un algoritmo genético para evolucionar los pesos de una red basada en UPs con una estructura predefinida. El mayor problema de este tipo de algoritmos es cómo obtener de antemano la arquitectura óptima. Ismail y Engelbrecht (1999, 2000, 2002) aplicaron cinco métodos diferentes de optimización para entrenar redes de tipo UP: un algoritmo genético estándar, un método de optimización mediante cúmulo de partículas (*Particle Swarm Optimization*, PSO), el algoritmo Leapfrog (el cual es un algoritmo de tercer orden que utilice la aceleración para estimar la posición final) y un algoritmo de poda. En definitiva, los trabajos realizados sobre RNUPs no han afrontado el problema simultáneo del diseño de la estructura de la red y la estimación de los pesos de las conexiones utilizando métodos clásicos o técnicas evolutivas, sobre todo en el área de *clasificación* (la mayoría de los trabajos han sido aplicados para *regresión*).

El punto de partida de esta Tesis Doctoral ha sido un algoritmo de PE para el entrenamiento de RNUPs para problemas de *regresión* (al que llamaremos *Neural Net Evolutionary Programming*, NNEP), basado en diversos trabajos anteriores (Martínez-Estudillo, 2005; Martínez-Estudillo et al., 2006a,b). El objetivo principal de este algoritmo es el de diseñar la estructura (número de nodos y número de conexiones) y, simultáneamente, optimizar los coeficientes de los modelos de RNUPs. El algoritmo ha demostrado obtener buenos resultados en problemas de *regresión*, tanto en términos de precisión como en complejidad (considerando como complejidad el número de nodos y conexiones) de los modelos de redes de tipo UP. Además este algoritmo supone una herramienta muy interesante para la optimización de este tipo de modelo, haciendo que sea una alternativa factible a los modelos clásicos de tipo MLP.

1.1.5. Generación de Diversidad en Algoritmos Evolutivos

Es conocido que la eficiencia de un AE depende de la correcta elección de una serie de parámetros, como son el tamaño de la población, el esquema de selección, el número de generaciones o las probabilidades de cruce y de mutación. Eiben et al. (1999) hicieron un esfuerzo por clasificar las metodologías desarrolladas para controlar los parámetros de los AEs.

Por otro lado, es importante que todo AE sea capaz de alcanzar un equilibrio entre dos factores normalmente contrapuestos:

- **Explotación o Convergencia**, es decir, capacidad del algoritmo de concentrar los esfuerzos de la búsqueda en las áreas más prometedoras, aumentando la precisión de las soluciones obtenidas.
- **Exploración o Diversidad**, es decir, capacidad del algoritmo de evitar una convergencia prematura, o lo que es lo mismo, de estancarse en un área del espacio de búsqueda que no contiene al óptimo global. Una buena diversidad aumenta la fiabilidad de las soluciones obtenidas.

La explotación está íntimamente ligada al mecanismo de competición establecido en el AE y, por tanto, a la presión selectiva incorporada mediante el operador de selección. La diversidad está relacionada con la cantidad de valores de aptitud, genotipos y fenotipos distintos que hay en la población. El problema es que ambos son objetivos contrapuestos: el aumento de la presión selectiva da lugar a una rápida pérdida de diversidad de la población, mientras que el hecho de mantener la diversidad puede llevar a una disminución de la precisión. Existen diversos trabajos cuyo objetivo fundamental es crear operadores para Algoritmos Genéticos (AGs) capaces de mantener este equilibrio (Herrera y Lozano, 1996; Lozano et al., 2008).

Tamaño de Población Variable

Un AE simple utiliza una población de tamaño constante y guía la evolución de dicha población durante un número determinado de generaciones, aplicándole operadores que modifican a los individuos en cada generación. El tamaño de la población es uno de los parámetros principales que afectan a la robustez y a la eficiencia computacional del algoritmo. Si utilizamos tamaños de población muy pequeños nos arriesgamos a provocar una convergencia prematura del algoritmo, mientras que si el tamaño es demasiado grande, el coste computacional se dispara.

Aunque existen trabajos en los que se estima el tamaño de la población óptimo basándose en la complejidad del problema (Goldberg et al., 1992), una alternativa posiblemente más interesante es utilizar un tamaño de población variable. Ejemplos de artículos anteriores en los que se sigue esta filosofía son los de Koumousis y Dimou (2002) y Smith (1993), los cuales demuestran que el esquema de tamaño de población variable mejora la capacidad de los AEs para refinar las soluciones y reduce la sensibilidad de los mismos en la elección de los parámetros.

Reinicializaciones de la Población

Por otro lado, se han propuestos muchos métodos en la literatura para intentar incrementar la diversidad de la población y evitar una convergencia prematura. Entre estos, es importante destacar los siguientes:

- Los algoritmos de Cobb y Grefenstette (1993), donde se examinan diversas alternativas de mutación, como reemplazar parte de la población en cada generación con individuos aleatorios, o aumentar fuertemente la mutación cada vez que se detecte una disminución en el rendimiento.
- El algoritmo CHC de Eshelman (1991) que consiste en un AG con selección elitista y un operador de cruce altamente perturbador, que reinicializa la población cuando la diversidad cae por debajo de un umbral.

- Otro AG que utiliza reinicializaciones de la población es el propuesto por Goldberg (1989), denominado Micro-AG o μ-AG. En líneas generales, este algoritmo utiliza una población de tamaño muy pequeño, la cual evoluciona durante muchas generaciones. Cuando, tras cierto número de generaciones, la población converge, el proceso evolutivo es reiniciado, preservando el mejor individuo y sustituyendo el resto con individuos generados aleatoriamente.

Algoritmo Genético basado en Dientes de Sierra

Un trabajo que combina los efectos del tamaño de población variable junto con los efectos de las reinicializaciones de la población, es el propuesto por Koumousis y Katsaras (2006) y que se denomina AG basado en dientes de sierra. La idea es provocar reinicializaciones periódicas, de forma que el tamaño de la población siga un esquema de dientes de sierra con una amplitud y un periodo de variación específicos. El tamaño de la población decrece linealmente conforme avanzan las generaciones hasta llegar a un mínimo en el que se produce un reinicio de la misma, añadiéndose individuos generados aleatoriamente hasta llegar al tamaño inicial de la población. El tamaño de la población evoluciona formando así dientes de sierra, con una altura (diferencia entre el tamaño de población máximo y mínimo) y periodo (número de generaciones entre cada reinicio) que deben ser especificados a priori. La dificultad surge a la hora de establecer el valor de estos parámetros para un problema concreto.

1.1.6. Modelos de Regresión Logística

El modelo de RL ha sido ampliamente empleado como método de *clasificación* en Estadística desde hace muchos años y, recientemente, ha recibido una gran aceptación en la comunidad de *Machine Learning* debido a su cercana relación con las nuevas teorías asociadas a las máquinas de soporte vectorial (*Support Vector Machine*, SVM, Vapnik, 1999) y a las técnicas de remuestreo iterativo como puede ser AdaBoost (Freund y Schapire, 1996). La RL es un modelo de regresión lineal generalizada que trata de predecir las probabilidades a posteriori de pertenencia de cada uno de los patrones de un conjunto de entrenamiento a uno de los valores que toma la variable dependiente mediante relaciones lineales con las variables predictoras (McCullagh y Nelder, 1989). El procedimiento habitual de estimación de los coeficientes de un modelo lineal de RL es el de máxima verosimilitud, en el que, para obtener el óptimo de la función de verosimilitud, se utiliza habitualmente un método de optimización local basado en un algoritmo iterativo de tipo Newton-Raphson o de mínimos cuadrados con reasignación de pesos (*Iterative Reweighted Least Squares*, IRLS) (Hastie et al., 2001) . Desde un punto de vista formal, la RL es un procedimiento sencillo y útil, pero no siempre se verifica que las probabilidades de pertenencia a cada clase transformadas mediante una transformación logarítmica presenten una relación lineal causa-efecto sobre las covariables o variables independientes.

Sea $Y \in 0, ..., J-1$, para $J > 1$, una variable de respuesta categórica o nominal (si $Y = 0$ entonces el valor de la clase es $G = 1$, y de forma similar si $Y = J - 1$ entonces la clase es $G = J$), y sea $\mathbf{X} \in \mathbb{R}^k$ el correspondiente vector de covariables. Se dice que las variables $(\mathbf{X}, Y)$ verifican un modelo de regresión logística nominal, si para cada $j = 1, ..., J-1$, $(\mathbf{X}, Y)$ satisfacen la ecuación:

$$\log \frac{P(G = j|\mathbf{X} = \mathbf{x})}{1 - P(G = J|\mathbf{X} = \mathbf{x})} = f_j(\mathbf{x}, \boldsymbol{\beta}_j) = \boldsymbol{\beta}_j^{\mathrm{T}}\mathbf{x} \tag{1.12}$$

donde $P(G = j|\mathbf{X} = \mathbf{x})/(1 - P(G = J|\mathbf{X} = \mathbf{x}))$ se define como la posibilidad de que la salida sea $Y = j - 1$.

Debido a la no linealidad del modelo, la estimación de los coeficientes deberá hacerse mediante un procedimiento iterativo. El modelo de regresión logística (Hosmer y Lemeshow, 2000) tiene por objetivo modelar las probabilidades a posteriori de pertenencia del patrón $\mathbf{x} \in \mathbf{X}$ a las J clases mediante funciones lineales en $\mathbf{x}$, asegurando siempre que la suma de la probabilidades sea la unidad. El modelo se propone como un conjunto de distribuciones de probabilidad condicional, las cuales son obtenidas utilizando el conjunto de entrenamiento $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ para $i = 1, ..., N$, donde la probabilidad de pertenencia a la clase j viene dada en forma matricial como:

$$p_j = \frac{\exp \boldsymbol{\beta}_j^{\mathrm{T}} \mathbf{x}}{1 + \sum_{i=1}^{J-1} \exp \boldsymbol{\beta}_i^{\mathrm{T}} \mathbf{x}}, \text{ para } j = 1, ..., J - 1 \qquad (1.13)$$

$$p_J = \frac{1}{1 + \sum_{i=1}^{J-1} \exp \boldsymbol{\beta}_i^{\mathrm{T}} \mathbf{x}}$$

siendo $\boldsymbol{\beta}_j = (\beta_{j0}, \beta_{j1}, ..., \beta_{jk})$, ya que se añade el valor $x_0 = 1$ al vector de características $\mathbf{x}$. Los coeficientes de regresión $\boldsymbol{\beta}_j$ se estiman a partir de los datos y se interpretan como la proporción del logaritmo de las posibilidades asociadas a cada clase j.

1.1.7. Modelos de Regresión Logística utilizando Covariables obtenidas mediante Redes Neuronales de Unidad Producto

Un método propuesto recientemente se basa en la hibridación de un modelo lineal y de RNUPs evolutivas para problemas de *clasificación* binarios (Hervás-Martínez y Martínez-Estudillo, 2007) y multi-clase (Hervás-Martínez et al., 2008). La estimación de los coeficientes del modelo se realiza en tres fases:

1. El número de funciones de base de tipo UP y sus exponentes se determina mediante el algoritmo NNEP adaptado a problemas de *clasificación*. El algoritmo se ejecuta normalmente y una vez se alcanza la última generación, se selecciona al mejor modelo de la población, se extraen sus funciones de base y se procede a realizar la segunda fase.

2. Un procedimiento estándar de optimización basada en el método de máxima verosimilitud (en concreto, el algoritmo IRLS) determina el resto de coeficientes del modelo lineal en el nuevo espacio que se obtiene al unir las covariables iniciales a las funciones de base de tipo UP estimadas previamente.

3. Un sencillo mecanismo final se incorpora para la simplificación del modelo, basado en la utilización de un algoritmo de Selección de Variables, que va eliminando una variable en cada paso, hasta que la eliminación de una variable no mejora el resultado de la *clasificación*.

Este modelo permite la generación de superficies no-lineales de *clasificación* y la identificación de las posibles interacciones de orden superior que pueden existir entre las covariables que definen el problema. Además, estos modelos son menos complejos (en cuanto a número de covariables y número de exponentes) que los modelos polinomiales alternativos de alto orden.

De esta forma, el modelo obtenido (Regresión Logística utilizando covariables Iniciales y Unidades Producto, RLIUP) es un modelo híbrido que utiliza el algoritmo NNEP para obtener

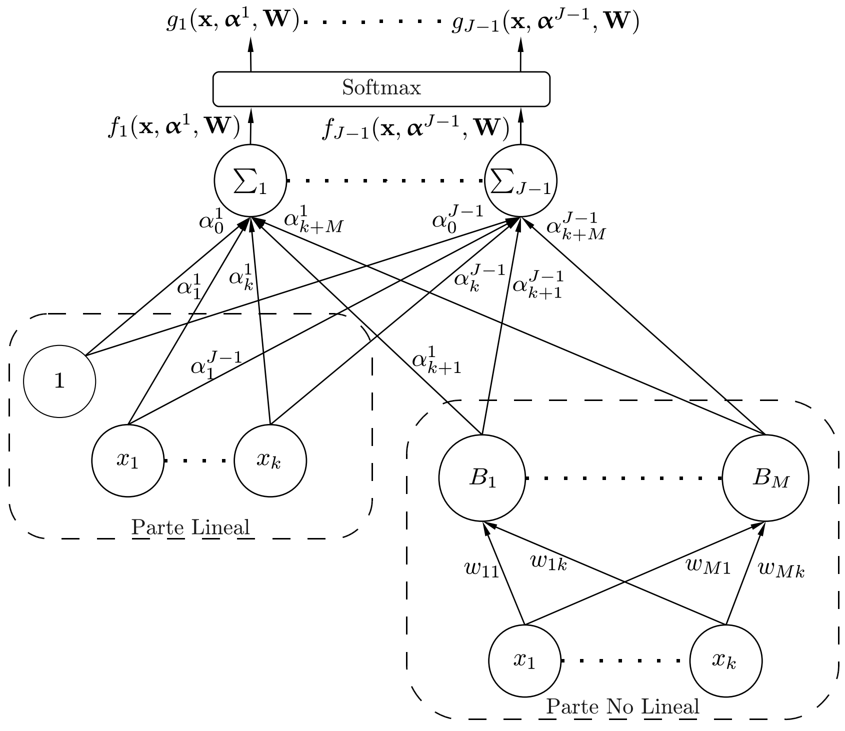


Figura 1.3: Modelo de Regresión Logística utilizando covariables Iniciales y Unidades Producto (RLIUP)

un conjunto de funciones de base de tipo UP que lleven a unos resultados lo suficientemente precisos. Una vez obtenidos, se aplica el algoritmo IRLS sobre las funciones de base de tipo UP (nuevas covariables) y las covariables iniciales del problema. Si J es el número de clases, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_{J-1}\}$ es el conjunto de coeficientes asociados al modelo y $\boldsymbol{\theta}_j = \{\boldsymbol{\alpha}^j, \mathbf{W}\}$ es el conjunto de parámetros asociados al nodo de salida j, el modelo funcional es el siguiente:

$$f_i(\mathbf{x}, \boldsymbol{\alpha}^i, \mathbf{W}) = \alpha_0^i + \sum_{j=1}^{k} \alpha_j^i x_j + \sum_{j=1}^{M} \alpha_{(k+j)}^i B_j(\mathbf{x}, \mathbf{w}_j) \tag{1.14}$$

donde $B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{n} x_i^{w_{ji}}$ son cada una de las UPs del mejor modelo de red obtenido por el algoritmo NNEP, $\boldsymbol{\alpha}^i = (\alpha_0^i, \alpha_1^i, ..., \alpha_k^i, \alpha_{k+1}^i, ..., \alpha_{k+M}^i)$ son los parámetros asociados a la parte lineal del modelo y $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M)$ son los parámetros no lineales asociados a la red de tipo UP original. Los valores ajustados por el algoritmo IRLS son los que corresponden a los vectores $\boldsymbol{\alpha}^i$, de manera que los pesos $\mathbf{W}$ son obtenidos por el algoritmo NNEP. La arquitectura de estos modelos puede verse mediante un grafo de red en la Figura 1.3.

1.1.8. Aplicaciones Reales

En esta Tesis Doctoral, hemos aplicado los modelos desarrollados a diversos problemas reales de dos áreas distintas: la Teledetección Agronómica y la Cinética Química. A continuación, resumimos brevemente los problemas abordados.

Problemas de Teledetección Agronómica

Los sistemas de Teledetección (tecnología de sensores remotos que realizan fotografías aéreas) proporcionan una gran cantidad de información con un coste razonable. El análisis de imágenes remotas permite modelar los diversos parámetros agronómicos para su aplicación en agricultura de precisión. Recientes avances en esta tecnología han planteado la necesidad de utilizar modelos más flexibles para estimar diferentes parámetros asociados a la determinación de la productividad y/o la presencia de malas hierbas en fincas.

Un aspecto relacionado con la posibilidad de minimizar el impacto de la agricultura en la calidad del medioambientales es el desarrollo de aproximaciones más eficaces para la determinación de la producción y para la aplicación inteligente de herbicidas. El motivo es que, aunque los rodales de malas hierbas se encuentran situados en zonas determinadas del cultivo, los herbicidas se aplican indiscriminadamente en toda la finca, con las consecuentes pérdidas económicas y los problemas medioambientales derivados. En este sentido, es esencial el análisis de características agronómicas que, basándose en la precisión de los mapas de rodales generados, permitan la implantación de modelos capaces de obtener un mayor rendimiento, a partir, por ejemplo, de una disminución de las cantidades de los herbicidas. Los beneficios potenciales económicos y medioambientales de la aplicación restringida de herbicidas incluyen un ahorro del consumo de pulverizador o *spray*, de los costes de los herbicidas y un incremento en el control de los rodales.

Por otro lado, en estudios malherbológicos han sido frecuentes los trabajos de competencia acerca de la influencia de las infestaciones de malas hierbas en el rendimiento de un cultivo, o sobre la obtención de mapas de emergencias con imágenes aéreas, o aquéllos que determinan la influencia de la pendiente en la existencia de rodales de malas hierbas. Con el fin de obtener el mapa de cosecha más preciso, es muy interesante aunar la influencia de todas esas variables y realizar un proceso de modelización multivariante, para así predecir mapas de rendimiento del cultivo, incorporando la elevación del terreno y los datos multiespectrales de imágenes aéreas.

De esta forma, dos ejemplos de Problemas de Teledetección Agronómica en los que la aplicación de RNAs resulta muy interesante son los siguientes:

- **Predicción de Producción**: la predicción de la producción y la elaboración de mapas de cosecha es un problema de *regresión*, en el que, a partir de los valores digitales de cada banda de la imagen aérea (Rojo, Verde, Azul e Infrarrojo) podríamos obtener un modelo capaz de aproximar la futura producción del cultivo en cuestión. Además sería interesante la incorporación de otras variables, como la elevación del terreno y el número de plantas de mala hierba existentes en cada píxel, la cual podría mejorar la precisión del modelo. Preferiblemente, las predicciones se deben hacer en fases tempranas del cultivo, de manera que se puedan detectar aquellas zonas menos productivas y el agricultor pueda actuar en consecuencia, mejorando las condiciones del cultivo.

- **Clasificación de Rodales de Malas Hierbas**: la determinación de mapas de rodales de la mala hierba partiendo de los valores digitales de las imágenes aéreas resulta del mismo modo muy interesante, ya que permitiría al agricultor tratar aquellas zonas más afectadas por este tipo de vegetación mediante herbicidas específicos. Sólo aquellas zonas más afectadas serían tratadas, lo cual redundaría en beneficios económicos y medioambientales, si se compara con un tratamiento generalizado de la finca.

Cinética Química

La Cinética Química es el campo de la Química que se ocupa de la dinámica de las reacciones químicas, así como de los mecanismos de las mismas. Dentro del área, existen una gran cantidad de estrategias para resolver y cuantificar los analitos que proporcionan curvas de señal/tiempo, altamente solapadas, obtenidas mediante instrumentos concretos de detección. Con estos modelos de previsión se puede obtener información de calidad acerca de las concentraciones de los analitos del compuesto químico (o compuestos químicos) objeto de estudio.

Del mismo modo que en la sección anterior, presentamos dos problemas concretos de Cinética Química susceptibles de ser afrontados mediante el uso de RNAs:

- **Cuantificación de picos altamente solapados provenientes de Electroforesis Capilar**: la cuantificación de picos altamente solapados obtenidos mediante un proceso de Electroforesis Capilar es un problema que ha recibido gran interés dentro de la Cinética Química. La Electroforesis Capilar es una técnica de separación basada en la migración diferencial de especies cargadas bajo la acción de un campo eléctrico o gradiente de potencial que se establece con ese fin. La idea es clasificar distintos compuestos químicos a partir de la cuantificación de los picos obtenidos mediante este proceso de Electroforesis Capilar.
- **Análisis de datos de Cromatografía de Gases acoplada con Espectrometría de Masas**: La Cromatografía de Gases es una técnica cromatográfica en la que la muestra se volatiliza y se inyecta en la cabeza de una columna cromatográfica. Dentro de las técnicas de Cromatografía de Gases se encuentra la extracción de Espacio de Cabeza. La Espectrometría de Masas es una técnica analítica para la determinación de la composición elemental de una muestra o molécula. Esta técnica se basa en ionizar los compuestos químicos para generar moléculas cargadas o fragmentos de moléculas y medir su relación masa-carga. La asociación de las dos técnicas (*Gas Chromatography*, GC, y *Mass Spectrometry*, MS) da lugar a una técnica combinada GC-MS que permite la separación e identificación de mezclas complejas. En resumen, una mezcla de compuestos inyectada en el cromatógrafo de gases se separa en la columna cromatográfica obteniendo la elución sucesiva de los componentes individuales aislados que pasan inmediatamente al espectrómetro de masas. Cada uno de estos componentes se registra en forma de pico cromatográfico y se identifica mediante su respectivo espectro de masas. Los datos resultantes son susceptibles de ser analizados mediante el uso de RNAs, para así poder obtener modelos capaces de predecir la naturaleza del compuesto.

1.2. Justificación y Motivación

Una vez presentados los principales conceptos sobre los que versa la presente Tesis Doctoral, nos planteamos una serie de problemas abiertos que justifican y motivan la investigación llevada a cabo. La Tesis ha sido dividida en dos partes diferenciadas y cada una de ellas responde a una motivación distinta. En la primera parte, hacemos referencia al campo de las Redes Neuronales Evolutivas y la justificación correspondiente comprendería los siguientes problemas abiertos:

- El trabajo de esta Tesis Doctoral parte de la Tesis de Martínez-Estudillo (2005) en la que se proponían modelos de RNUPs para problemas de *regresión*. La Tesis proponía un algoritmo de Programación Evolutiva, presentado en la Sección 1.1.4 de este documento, y que, como comentamos anteriormente, denominaremos genéricamente NNEP (*Neural Net Evolutionary Programming*). Además, en el capítulo 5 de dicha Tesis, se proponían Algoritmos Híbridos de entrenamiento de RNUPs (Martínez-Estudillo et al., 2006a), basados en la combinación del algoritmo NNEP y de un procedimiento de optimización local (el algoritmo de Levenberg-Marquardt, Marquardt, 1963). Este algoritmo de entrenamiento de RNUPs se utilizó básicamente sobre bases de datos de prueba. De esta forma, la primera motivación del trabajo realizado en la presente Tesis Doctoral es la de consolidar esta metodología en otra tipología de bases de datos correspondientes a problemas reales.

- Como ya hemos indicado, existen diferentes propuestas para el entrenamiento de RNAs utilizando funciones de base de tipo sigmoide (US) y radial (RBF) (Yao, 1999; González et al., 2003; Manrique et al., 2006). Sin embargo, en nuestro conocimiento, no existen propuestas para el entrenamiento de RNUPs para problemas de *clasificación*. De esta forma, otra motivación de la investigación realizada en esta Tesis Doctoral ha sido la de obtener un mecanismo para el entrenamiento de clasificadores basados en RNUPs. Debemos tener en cuenta que ambas tipologías de problemas (*clasificación* y *regresión*) presentan diferencias tangibles a la hora de su implementación dada la diferente naturaleza de las funciones de evaluación y de la estructura de la capa de salida de los modelos de red.

- Por otra parte, es necesario comparar los modelos basados en RNUPs, propuestos para *clasificación* y *regresión*, con los otros dos tipos habituales de redes de propagación hacia delante o *feedforward* (redes de tipo US y redes de tipo RBF). La naturaleza del problema tratado, puede hacer que estos modelos alternativos sean más adecuados que las RNUPs. Es por tanto necesario aplicar estos modelos y adaptar el algoritmo evolutivo mencionado anteriormente para su diseño, puesto que estaba orientado únicamente al entrenamiento de RNUPs en problemas de *regresión*.

- El problema de diseñar modelos de RNAs es un problema NP complejo y en el caso de las RNUPs, donde los exponentes de las funciones potenciales son números reales, la dimensión del espacio de búsqueda de soluciones aumenta de forma exponencial al aumentar tanto el número de características como el número de funciones de base. Para paliar dicha dificultad, el citado algoritmo NNEP incorpora una fuerte presión selectiva provocada por los operadores de selección y mutación paramétrica. Como analizamos en la Sección 1.1.5 de este documento, esto puede provocar que se caiga en una convergencia prematura del algoritmo, perdiendo el mismo su capacidad de exploración. Para evitar esta convergencia prematura, NNEP utiliza un tamaño de población muy elevado que asegura la exploración de múltiples zonas del espacio de búsqueda. El problema es que el coste computacional es consecuentemente muy alto, haciendo difícil el entrenamiento de bases de datos con un número de patrones y/o un número de características elevado. Esto justifica la necesidad de incorporar mecanismos de generación de diversidad en dicho algoritmo.

- Como ya hemos apuntado en la Sección 1.1.3, Donoho y Johnstone (1989) demuestran la dualidad de las aproximaciones basadas en funciones de tipo proyección y de tipo *kernel*. Siguiendo esta filosofía, los modelos de RNAs se pueden generalizar a modelos de red híbridos: redes neuronales con unidades producto y funciones de base radial (PRBF) y redes neuronales con unidades sigmoide y funciones de base radial (SRBF). Además, como red híbrida alternativa y con objeto de realizar comparaciones, se estudiarán los modelos basados en redes neuronales con unidades sigmoide y producto (USP), pese a ser ambas funciones de proyección. Los AEs aparecen aquí como una alternativa muy interesante, ya que al considerar distintos tipos de funciones de base, la complejidad del modelo aumenta, siendo necesario estimar el número de funciones de base de cada uno de los tipos considerados. De esta forma, se amplia tanto el espacio de búsqueda como el número de óptimos locales de la función de error. Sin embargo, en nuestro conocimiento, si revisamos los trabajos referentes a combinación de funciones de base (ver Sección 1.1.3), no se han propuesto AEs para el desarrollo de este tipo de modelos, siendo ésta otra de las motivaciones de los trabajos planteados en esta Tesis Doctoral.

La segunda parte de la Tesis Doctoral viene motivada por un tipo especial de modelos para problemas de *clasificación* que, aunque no conocemos autores que hayan llegado a implementarlos y/o aplicarlos, se han sugerido en varios libros y artículos de Hastie y Tibshirani (1996) y Friedman (1993). Englobaremos estos modelos bajo el término Regresión Logística Generalizada utilizando Funciones de Base. Como sugirieron Hastie y Tibshirani (1996), una forma obvia de generalizar el modelo de RL lineal es reemplazar los predictores lineales por una versión no paramétrica de los mismos, entre los que se encuentran, entre otras posibilidades, las funciones de base. Bajo esta filosofía, la idea básica sería la de ampliar un modelo estándar de RL (el cual se planteó en la Sección 1.1.6 de este documento) con funciones de base obtenidas a partir de modelos de RNAs. Las posibilidades a la hora de elegir estas funciones de base son múltiples, entre las que cabe destacar:

- Las UPs que serían capaces de aumentar el espacio de covariables inicial con nuevas covariables que convertirían la parte predictora del modelo de RL en no lineal, recogiendo en el mismo las posibles interacciones de alto orden entre las covariables iniciales. Este tipo de modelos híbridos en la estructura (lineal y no lineal) de la función predictora han demostrado ser muy eficaces para la resolución de problemas de *clasificación* binarios (Hervás-Martínez y Martínez-Estudillo, 2007) y multi-clase (Hervás-Martínez et al., 2008).
- Las unidades de tipo radial (RBFs) que llevarían a una RL con una función de predicción con dos estructuras: una lineal y la otra no lineal formada por funciones locales de tipo Gaussiano. De esta forma, la idea sería extraer localizaciones del espacio de entrada, que ayudasen a mejorar la tarea de discriminación. No existen trabajos previos en los que se haya experimentado con este tipo de modelos.

Las USs no aparecen como una alternativa interesante debido a que la estructura cuasi-hiperplanar conflictuaría con la estructura lineal de la parte predictora del modelo de RL.

Por último, es necesario indicar que es una constante en el Grupo de Investigación AYRNA el que los modelos y algoritmos desarrollados sean aplicados a la resolución de problemas reales. En este sentido, en ambas partes de la Tesis, otra motivación fundamental será la de aplicar los modelos desarrollados a la resolución de diferentes problemas reales en dos áreas tan distintas como son la Teledetección Agronómica y la Cinética Química.

1.3. Objetivos

Detectados los problemas abiertos que justifican la investigación desarrollada en esta Tesis Doctoral, pasamos a enumerar los objetivos a cumplir. Comenzamos estableciendo los objetivos generales:

1. *Plantear nuevos modelos de MS, capaces de ir más allá de la linealidad establecida en los modelos clásicos de regresión y clasificación.* En concreto, diseñaremos nuevos modelos de RNAs mediante hibridación de funciones de base (US, UP y RBF) y generalizaremos modelos clásicos de *clasificación*, como es el modelo de RL, añadiendo funciones de base (UP y RBF) como nuevas covariables.

2. *Desarrollar Algoritmos Híbridos (es decir, AEs combinados con Búsquedas Locales) para la estimación de los parámetros y la estructura de estos modelos.* Nuestro trabajo partirá de la base del algoritmo NNEP propuesto en la Tesis de Martínez-Estudillo (2005) y en diversos trabajos (Martínez-Estudillo et al., 2006a,b).

3. *Comparar las metodologías y modelos propuestos con otros métodos alternativos mediante diseño de experimentos específicos y mediante el uso de tests de hipótesis.* Los diseños de experimentos que hemos utilizado han sido el diseño *hold-out* (para aquellos casos en que únicamente se comparan algoritmos de entrenamiento evolutivo de RNAs) y el diseño *k-fold* (para aquellos casos en que comparamos nuestras propuestas con otros métodos no estocásticos de *clasificación* o *regresión*). Por otro lado, los tests de hipótesis utilizados comprenden tanto tests paramétricos (en aquellos casos en los que se satisfagan las condiciones de independencia y de normalidad de las observaciones de las variables de comparación) y tests no paramétricos (para aquellos casos en los que dichas condiciones no se satisfagan).

4. *Aplicar los modelos obtenidos en problemas reales de Teledetección Agronómica y de Cinética Química.* De esta forma, se abordarán cuatro bases de datos correspondientes a problemas de Predicción de Producción, Clasificación de Rodales de Malas Hierbas, Cuantificación de picos altamente solapados provenientes de Electroforesis Capilar y Análisis de datos de Cromatografía de Gases acoplada con Espectrometría de Masas (problemas descritos en la Sección 1.1.8 de este documento).

Los objetivos específicos motivados por los problemas abiertos establecidos en la sección anterior y organizados según las dos partes en que se divide la Tesis Doctoral, son los siguientes:

1. Redes Neuronales Evolutivas:

   a) *Desarrollar un algoritmo eficaz para el diseño de modelos de RNUPs para problemas de clasificación.* De esta forma, se pretende rediseñar el algoritmo NNEP (el cual fue planteado para problemas de *regresión*) para obtener clasificadores basados en RNUPs.

   b) *Comparar los clasificadores obtenidos con modelos de RNAs estándar, como pueden ser los modelos de redes de tipo MLP o de tipo RBF.* En este sentido, la adaptación del algoritmo NNEP a este tipo de modelos y la realización de un estudio experimental sobre las diferencias de rendimiento entre los distintos tipos de modelos, aparece como otro de los objetivos de la presente Tesis Doctoral.

   c) *Plantear un mecanismo de generación de diversidad que permita disminuir el coste computacional de los algoritmos desarrollados.* Más en concreto, nos centraremos en el mecanismo de dientes de sierra propuesto por Koumousis y Katsaras (2006), ya que, además de ser un mecanismo simple y fácil de adaptar a cualquier AE, ha demostrado una gran eficiencia, al menos para problemas de aproximación de funciones.

d) *Implementar un algoritmo para el diseño de modelos de RNAs de tipo híbrido, combinando distinto tipo de funciones de base en su capa oculta.* En concreto, nos centraremos en los modelos SRBF, PRBF y USP. Así, el diseño de RNAs de tipo híbrido será abordado mediante el AE propuesto anteriormente.

2. Regresión Logística Generalizada utilizando Funciones de Base:

   a) *Automatizar la metodología RLIUP planteada en trabajos anteriores y mejorar el mecanismo de simplificación final del modelo.* La metodología RLIUP planteada en la Sección 1.1.7 era realizada en dos etapas, utilizando en primer lugar un paquete *software* para optimizar los modelos de RNUPs y posteriormente el paquete estadístico SPSS para aplicar el algoritmo IRLS. Se pretende automatizar este proceso y además experimentar con otros mecanismos más potentes para la simplificación estructural realizada en la última etapa.

   b) *Diseñar modelos de Regresión Logística Generalizada basada en funciones de base de tipo UP.* Se pretende estudiar experimentalmente el potencial de dichos modelos sobre problemas reales.

   c) *Diseñar modelos de Regresión Logística Generalizada basada en funciones de base de tipo RBF.* Este tipo modelos serán desarrollados como una alternativa a los anteriores, y creemos que en determinado tipo de problemas, pueden suponer un aumento de la eficacia de los mismos.

1.4. Resumen

Para desarrollar los objetivos planteados, la memoria está constituida por ocho publicaciones (siete de ellas aceptadas o publicadas en revistas indexadas en el *Journal Citations Report*, JCR, y otra de ellas sometida a revisión), distribuidas en dos partes y un total de cinco secciones que se desarrollarán en los Capítulos 2 y 3. La estructura es la siguiente:

1. Redes Neuronales Evolutivas:

   a) Desarrollo de Modelos de Redes Neuronales de Unidades Producto.

   b) Diseño de Perceptrones Multi-capa utilizando un Algoritmo de Programación Evolutiva y guiando el Tamaño Poblacional mediante Dientes de Sierra Dinámicos

   c) Entrenamiento de Redes Neuronales con Funciones de Base de tipo *Kernel* y de tipo Proyección utilizando un Algoritmo de Programación Evolutiva.

2. Regresión Logística Generalizada utilizando Funciones de Base:

   a) Regresión Logística utilizando Funciones de Base de tipo Unidad Producto.

   b) Regresión Logística utilizando Funciones de Base de tipo Radial.

Por otro lado, las aplicaciones reales mencionadas serán incluidas en cada una de las partes correspondientes. Además, en esta memoria incluimos una Sección de "Discusión Conjunta de los Resultados", que proporciona una información resumida de las propuestas y los resultados más interesantes obtenidos en cada parte. La Sección "Comentarios Finales: Resultados y Conclusiones" resume los resultados obtenidos en esta memoria y presenta algunas conclusiones sobre éstos. Finalmente, en la Sección "Líneas y Propuestas Futuras" se comentarán algunos aspectos sobre trabajos futuros que quedan abiertos en la presente Tesis Doctoral.

1.5. Discusión Conjunta de Resultados

Esta sección muestra un resumen de las distintas propuestas que se recogen en la presente memoria y presenta una breve discusión sobre los resultados obtenidos por cada una de ellas.

1.5.1. Redes Neuronales Evolutivas

Entrenamiento de Redes Neuronales de Unidad Producto utilizando un Algoritmo de Programación Evolutiva

En esta parte (Sección 2.1 de este documento), proponemos algoritmos de PE para el entrenamiento de RNUPs en problemas de *clasificación* y *regresión*. Esta parte esta compuesta por dos publicaciones:

1. P.A. Gutiérrez, F. López-Granados, J.M. Peña-Barragán, M. Jurado-Expósito, M.T. Gómez-Casero, C. Hervás-Martínez. Mapping sunflower yield as affected by *Ridolfia segetum* patches and elevation by applying Evolutionary Product Unit Neural Networks to remote sensed data. Computers and Electronics in Agriculture 60:2 (2008) 122-132.
2. F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo. Evolutionary product-unit neural networks classifiers. Neurocomputing 72:1-3 (2008) 548-561.

La primera publicación es una aplicación práctica del Algoritmo Híbrido propuesto en el Capítulo 5 de la Tesis de Martínez-Estudillo et al. (2005) sobre un problema real de Predicción de Producción de cosecha en campos de girasol. Los últimos avances que se han producido en la tecnología de sensores remotos han disparado la necesidad de métodos altamente flexibles para estimar diferentes parámetros de la producción en agricultura de precisión. El objetivo del trabajo es el de determinar el potencial de las RNUPs evolutivas para la elaboración de mapas de cosecha y para actuar como sistemas de predicción de cosecha de girasol en una finca infestada de forma natural por malas hierbas. Se han tomado fotografías aéreas en la fase de crecimiento vegetativa avanzada (mitad de mayo) y utilizamos distintas aproximaciones estadísticas junto con otras basadas en RNUPs evolutivas para desarrollar los diferentes modelos de predicción. En concreto comparamos los resultados obtenidos con las siguientes siete metodologías: 1) una regresión lineal múltiple paso a paso; 2) RNUPs, entrenadas utilizando el algoritmo NNEP de PE; distintas hibridaciones del algoritmo NNEP con el procedimiento de búsqueda local de Levenberg-Marquardt (LM): 3) aplicación del algoritmo LM únicamente sobre el mejor individuo de la última generación del algoritmo NNEP (PE Híbrida, PEH), 4) realización de un análisis *cluster* basado en las salidas obtenidas por cada uno de los modelos de la última generación del algoritmo NNEP, eligiendo el mejor individuo de cada *cluster*, aplicando el algoritmo LM sobre cada uno de dichos individuos seleccionados como representantes de los *clusters* y seleccionando el mejor de los individuos optimizados (PEH con Clustering, PEHC), 5) versión dinámica del algoritmo PEHC, realizando el proceso descrito varias veces durante la evolución y seleccionando al mejor de los individuos obtenidos al terminar la evolución (PEHC Dinámica, PEHCD); realización de *ensembles* o grupos de modelos, a partir de los modelos obtenidos mediante el algoritmo PEHCD, considerando dos posibilidades: 6) predecir como valor de cosecha la media de los valores de cosecha predichos por los distintos modelos del *ensemble* y 7) predecir como valor de cosecha la mediana de los valores de cosecha predichos por los distintos modelos del *ensemble*. Incluimos una descripción del problema tratado, de la adquisición de los datos, de todas estas metodologías y del conjunto de resultados obtenidos y un análisis de los mismos basado en tests estadísticos. Además elaboramos mapas de predicción de cosecha en base a las salidas obtenidas por los distintos modelos, con objeto a comparar la calidad de dichos mapas. Los resultados obtenidos utilizando los modelos de RNEUPs muestran que el modelo funcional y los algoritmos híbridos propuestos proporcionan una predicción muy precisa comparada con las otras metodologías utilizadas para resolver este problema de *regresión* y que, en concreto, las aproximaciones basadas en *ensembles* son las que mejores resultados obtienen.

La segunda publicación propone la utilización de modelos de RNUPs para problemas de *clasificación*. Las UPs, que son nodos multiplicativos en lugar de nodos aditivos, suponen funciones de base no lineales capaces de expresar las posibles interacciones de alto orden entre las variables. En esta publicación, rediseñamos el algoritmo evolutivo NNEP para determinar la estructura básica del modelo de unidades producto y para estimar los coeficientes correspondientes. Hacemos uso de la transformación *softmax* (tal y como planteamos en la Sección 1.1.2 de este documento) como regla de decisión y la función de error de entropía cruzada debido a su interpretación probabilística. De esta forma, el modelo puede verse como una regresión logística no-lineal multinomial donde los parámetros son estimados utilizando computación evolutiva. Realizamos un conjunto de experimentos sobre siete de bases de datos de tipo *benchmark* y un problema real complejo de estimación de los límites de crecimiento microbiano de *Listeria monocytogenes*, para asegurar la seguridad y calidad de productos alimentarios. Los métodos con que comparamos los resultados obtenidos incluyen: 1) el mismo algoritmo NNEP aplicado sobre modelos clásicos de red tipo MLP (utilizando USs en su capa oculta) bajo el mismo enfoque probabilístico; 2) el algoritmo COVNET, un modelo cooperativo coevolutivo para la evolución de RNAs, propuesto por García-Pedrajas et al. (2003); 3) MPANN y SPANN, que son modelos multi-objetivo de evolución de redes neuronales artificiales (Abbass, 2003); y 4) un modelo completo de RL y RL con eliminación de variables. Los resultados empíricos obtenidos y aquellos correspondientes a los tests de comparaciones múltiples muestran que el modelo propuesto es prometedor en términos de su precisión en clasificación y del número de coeficientes del modelo, obteniendo un rendimiento acorde con el estado del arte. Además, mostramos gráficamente la tarea de *clasificación* llevada a cabo por el modelo de RNUP, junto con su capacidad de capturar las interacciones entre las variables y de reducir la dimensión del espacio de entrada.

Diseño de Perceptrones Multi-capa utilizando un Algoritmo de Programación Evolutiva basado en Dientes de Sierra Guiados

En esta parte (Sección 2.2 de este documento), el objetivo es mejorar el algoritmo NNEP, intentando incorporar un mecanismo de generación de diversidad que permita disminuir su coste computacional sin disminuir por ello la eficiencia del mismo. La publicación asociada a esta parte es la siguiente:

1. P.A. Gutiérrez, C. Hervás-Martínez, M. Lozano. Designing Multilayer Perceptrons using a Guided Saw-tooth Evolutionary Programming Algorithm. Soft Computing (2009). Aceptado.

En esta publicación, proponemos un mecanismo de generación de diversidad para el algoritmo NNEP aplicado a la determinación de la estructura básica de clasificadores basados en MLPs, y simultáneamente a la estimación los coeficientes de los modelos. La razón de haber elegido modelos MLP, es que deseábamos probar las modificaciones del algoritmo en un modelo de *clasificación* lo más estándar posible. Sobre este algoritmo, aplicamos una versión modificada de un mecanismo de mejora de diversidad basado en dientes de sierra, presentado recientemente para Algoritmos Genéticos (Koumousis y Katsaras, 2006), el cual utiliza un tamaño poblacional variable y reinicializaciones parciales periódicas de la población utilizando una función con forma de dientes de sierra. Nuestra mejora sobre este esquema estándar consiste en guiar las reinicializaciones de los dientes de sierra considerando la varianza de los mejores individuos de la población. Las reinicializaciones de la población son realizadas cuando la diferencia de varianza entre dos generaciones consecutivas es menor que un porcentaje de la varianza anterior. Comparamos las diferencias tanto en eficacia (medida como porcentaje de patrones bien clasificados en el conjunto de generalización), como en eficiencia (medida como número de evaluaciones necesarias para un mismo número de generaciones). Además, incluimos un análisis gráfico generación a generación del comportamiento del algoritmo en cuanto al máximo *fitness* obtenido y al *fitness* medio de la población. Incluimos tests estadísticos para asegurar la significación de las diferencias en eficacia existentes entre las distintas aproximaciones. Del análisis de los resultados obtenidos sobre diez bases de datos de tipo *benchmark*, podemos concluir que el coste computacional del algoritmo NNEP con un tamaño de población constante es reducido al utilizar el esquema basado en dientes de sierra. Además, el mecanismo propuesto basado en dientes de sierra guiados supone una demanda de tiempo computacional significativamente menor que el esquema original. Finalmente, ambos esquemas de dientes de sierra no implican un descenso en la precisión de los modelos obtenidos, la cual, en general, es igual o superior.

Entrenamiento de Redes Neuronales con Funciones de Base de tipo Kernel y de tipo Proyección utilizando un Algoritmo de Programación Evolutiva

En esta parte (Sección 2.3 de este documento), proponemos modelos de RNAs híbridas utilizando distinto tipo de funciones de base en su capa oculta. Además se incluye una aplicación práctica de este tipo de modelos. Esta parte esta compuesta por dos publicaciones:

1. P.A. Gutiérrez, C. Hervás-Martínez, M. Carbonero, J.C. Fernández. Combined Projection and Kernel Basis Functions for Classification in Evolutionary Neural Networks. Neurocomputing (2009). Aceptado.

2. C. Hervás-Martínez, P.A. Gutiérrez, M. Silva, J.M. Serrano. Combining classification and regression approaches for the quantification of highly overlapping capillary electrophoresis peaks by using evolutionary sigmoidal and product unit neural networks. Journal of Chemometrics 21:12 (2007) 567-577.

La primera publicación propone modelos de combinación de funciones de base para problemas de *clasificación*, utilizando el algoritmo NNEP. De esta forma, proponemos un modelo de RNA híbrida utilizando una posible combinación de diferentes funciones de transferencia de tipo proyección (US y UP) y de tipo *kernel* (RBF) en la capa oculta de una red de propagación hacia delante. Adaptamos el algoritmo NNEP a este tipo de modelos y lo aplicamos para el aprendizaje de la arquitectura, los pesos y la tipología de los nodos. Proponemos tres modelos diferentes de funciones de base combinadas que corresponden a los tres pares diferentes que se pueden obtener a partir de los nodos US, UP y RBF: redes neuronales con unidades producto y funciones de base radial (PRBF), redes neuronales con unidades sigmoide y funciones de base radial (SRBF) y redes neuronales con unidades sigmoide y producto (USP). Además, comparamos estos modelos con los correspondientes modelos puros (en los que sólo se establece un tipo de nodos en capa oculta): RNUPs, MLPs y RNRBFs. Por otro lado, analizamos de forma teórica la complejidad de las funciones de base y su capacidad de generalización para así prever las características de los modelos híbridos que combinen distintos tipos de funciones de base. Comprobamos todas las propuestas utilizando diez problemas de prueba de *clasificación* y llevamos a cabo tests estadísticos de comparaciones múltiples para probar la significación de las diferencias observadas. Los resultados indican que aquellos modelos que combinan funciones de base de tipo *kernel* y de tipo proyección se observan más eficaces que los modelos basados en un único tipo de función de base para una gran cantidad de bases de datos.

La segunda publicación de esta parte corresponde a una aplicación práctica de los modelos híbridos basados únicamente en funciones de base de tipo proyección (US y UP) sobre un problema real de *regresión* de Cuantificación de picos altamente solapados provenientes de Electroforesis Capilar. De esta forma, este trabajo es un estudio del potencial de los modelos de RNAs construidos mediante el uso de diferentes funciones transferencia de tipo proyección (MLPs, RNUPs y modelos híbridos USP) diseñados mediante el algoritmo NNEP, con el objeto de cuantificar los picos altamente solapados provenientes de los procesos de Electroforesis Capilar. Para probar esta aproximación, dos antibióticos aminoglicósidos, *amikacin* y *paramomycin*, son cuantificados a partir de muestras que contienen únicamente un componente o mezclas de los mismos, analizadas mediante Electroforesis Capilar de zona con detección fluorescente inducida por láser. Los tres modelos comprobados utilizan como datos de entrada los cuatro parámetros de la curva de Weibull asociada al perfil del pico de Electroforesis correspondiente y, en algunos casos, incluimos entre las variables de entrada una etiqueta de clase, obtenida mediante un Análisis Discriminante Lineal (ADL). Consideramos tres diseños experimentales. El primero incluye tanto los compuestos puros como la mezclas de los mismos. El segundo diseño es similar pero entre las entradas incluimos, además de los parámetros de la curva de Weibull, la etiqueta de clase obtenida por el ADL. El tercer diseño se basa en tomar como entradas sólo los parámetros de la curva de Weibull y como patrones de entrada sólo aquellos que contienen mezclas de los compuestos analizados. La combinación de técnicas de *regresión* y *clasificación* permite el establecimiento de topologías de red más simples, haciendo que los analitos sean cuantificados con una gran precisión. Los mejores modelos para las muestras mixtas son obtenidos por las RNUPs, con una arquitectura 4:4:1 (cuatro entradas, cuatro nodos en capa oculta y una salida) y 14 pesos para ambos analitos, utilizando la discriminación realizada por el análisis ADL, permitiendo que los analitos sean cuantificados con una gran precisión: con valores de error estándar de predicción en el conjunto de generalización (*Standard Error of Prediction*, SEP_G) de 8.2 % para *amikacin* y 5.6 % para *paromomycin*. Para comparar los resultados, empleamos además una regresión de mínimos cuadrados parcial, la cual proporciona una menor precisión: un error SEP_G de 11.8 y 15.7 % para *amikacin* y *paramomycin*, respectivamente. Las dimensiones reducidas de los modelos de redes neuronales seleccionados han permitido la obtención de ecuaciones simples de cuantificación para transformar las variables de entrada en la variable de salida. Estas ecuaciones pueden ser más fácilmente interpretadas desde un punto de vista químico que las proporcionadas por otros modelos de RNAs.

1.5.2. Regresión Logística Generalizada utilizando Funciones de Base

Regresión Logística utilizando Funciones de Base de tipo Unidad Producto

En esta parte (Sección 3.1 de este documento), proponemos modelos híbridos de Regresión Logística Generalizada utilizando Funciones de Base, obtenidos a partir de UPs pertenecientes a

RNUPs entrenadas mediante el algoritmo NNEP. De esta forma, partimos del modelo propuesto por Hervás-Martínez y Martínez-Estudillo (2007) y Hervás-Martínez et al. (2008), los cuales probaron su eficacia en problemas de tipo *benchmark*. Hemos automatizado el procedimiento de aprendizaje (tal y como se indicó en la Sección 1.2), y hemos incorporado diversas alternativas de simplificación estructural para la última etapa del algoritmo. Además hemos aplicado estos modelos en distintos problemas reales. Las publicaciones asociadas a esta parte son las siguientes:

1. P.A. Gutiérrez, C. Hervás-Martínez, J.C. Fernández, M. Jurado-Expósito, J.M. Peña-Barragán, F. López-Granados. Structural simplification of hybrid neuro-logistic regression models in multispectral analysis of remote sensed data. Neural Network World 19:1 (2009) 3-20.

2. P.A. Gutiérrez, F. López-Granados, J.M. Peña-Barragán, M. Jurado-Expósito, C. Hervás-Martínez. Logistic regression product-unit neural networks for mapping *Ridolfia segetum* infestations in sunflower crop using multitemporal remote sensed data. Computers and Electronics in Agriculture 64:2 (2008) 293-306.

3. C. Hervás-Martínez, M. Silva, P.A. Gutiérrez, A. Serrano. Multilogistic regression by evolutionary neural network as a classification tool to discriminate highly overlapping signals: Qualitative investigation of volatile organic compounds in polluted waters by using headspace-mass spectrometric analysis. Chemometrics and Intelligent Laboratory Systems 92:2 (2008) 179-185.

En la primera publicación investigamos diferentes metodologías de simplificación estructural de los modelos de Regresión Logística Generalizada utilizando Funciones de Base de tipo Unidad Producto, aplicándolos a la discriminación multi-espectral de rodales de mala hierba en campos de girasol. La RL se ha convertido en un método ampliamente utilizado y aceptado para analizar variables binarias o multi-clase, ya que es una herramienta flexible que puede predecir la probabilidad de que se produzca un determinado estado de una variable dicotómica. Partimos de la metodología propuesta por Hervás-Martínez y Martínez-Estudillo (2007) y Hervás-Martínez et al. (2008), basada en la hibridación de un modelo lineal y de modelos de RNUPs evolutivas para *clasificación* binaria y multi-clase. Esta metodología es automatizada, incluyendo en el software NNEP una versión optimizada el algoritmo IRLS (en lugar de utilizar para este fin el paquete software SPSS, utilizado en las publicaciones anteriores). Por otro lado, debido a que estos modelos poseen un elevado número de coeficientes, en esta publicación, presentamos dos métodos para simplificar la estructura del modelo final, reduciendo el número de covariables iniciales o UPs, ambos métodos basados en el test de Wald. El primero es una búsqueda hacia atrás con vuelta a estados anteriores (*Backtracking Backward Search*, BBS) y el segundo es un procedimiento similar pero basado en la utilización de un procedimiento estándar de enfriamiento simulado (*Simulated Annealing* BBS, SABBS). En el estudio realizado, utilizamos imágenes aéreas tomadas a mitad de mayo para evaluar el potencial de las distintas combinaciones de RL y RNUPs (RL utilizando sólo UPs, RLUP, y RL utilizando las covariables Iniciales y UPs, RLIUP), con los métodos presentados para la simplificación estructural de los modelos finales (BBS y SABBS), evaluados en el problema de discriminación de rodales de *Ridolfia segetum* (una de las malas hierbas más dominantes, competitivas y persistentes en campos de girasol) en una finca del sur de España infestada de forma natural por dicha mala hierba. Además, comparamos el rendimiento de estos seis métodos con el rendimiento de otros seis algoritmos comúnmente utilizados en *clasificación*: RL con un modelo completo, RL incorporando selección de características, árboles de decisión con modelos logísticos (*Logistic Model Trees*, LMT), árboles de decisión C4.5, árboles bayesianos (*Naïve Bayesian Trees*) y el algoritmo AdaBoost para la creación de *ensembles* de clasificadores C4.5. Analizamos las diferencias en eficacia y en número de coeficientes de las distintas metodologías propuestas e incorporamos diversos análisis estadísticos de los resultados para establecer la significación de dichas diferencias. Los métodos de simplificación estructural (BBS y SABBS) demuestran una reducción considerable en el número de coeficientes, llevando a una precisión similar o mejor. Por otro lado, al comparar estas metodologías con otras metodologías de *clasificación*, llegamos a la conclusión de que nuestras propuestas obtienen un rendimiento muy competitivo y un menor número de coeficientes.

En la segunda publicación abordamos el mismo problema real: la Clasificación de Rodales de Malas Hierbas. La Teledetección, los sistemas de información geográfica, y los sistemas de posicionamiento global nos pueden proporcionar las tecnologías necesitadas por los agricultores para maximizar los beneficios económicos y medioambientales de la agricultura de precisión. El tratamiento específico de malas hierbas es capaz de minimizar el impacto de la aplicación de herbicidas en la calidad medioambiental e incrementa la necesidad de aproximaciones más precisas para la determinación de rodales de mala hierba. La mala hierba *Ridolfia segetum* es una de las más dominantes, competitivas y persistentes en los campos de girasol al sur de España. En este publicación, utilizamos imágenes aéreas tomadas a mitad de mayo, a mitad de junio y a mitad de julio, de acuerdo a los distintos estados fenológicos de la mala hierba *R. segetum* y del girasol, para evaluar el potencial de las RNUPs evolutivas, la regresión logística (RL) y dos combinaciones de ambos métodos (RLUP y RLIUP) para la discriminación de rodales de *R. segetum* y la elaboración de mapas de probabilidad de crecimiento de esta mala hierba en sembrados de girasol de dos fincas infestadas de forma natural. El objetivo es conocer en qué fechas esta discriminación es viable y, además, determinar si estos métodos obtienen la suficiente precisión para realizar esta discriminación en agricultura de precisión. Además, comparamos el rendimiento de estos métodos en cada fecha con dos modelos muy competitivos en *clasificación* (máquinas de soporte vectorial, *Support Vector Machines*, SVM, y LMT). Por otro lado, comparamos gráficamente la calidad de los mapas obtenidos con cada una de las metodologías, estableciendo en cada pixel de la imagen la probabilidad de presencia de mala hierba obtenida por el modelo. Los resultados presentan los modelos propuestos como herramientas muy útiles para la discriminación de malas hierbas en ambas fincas, obteniendo el mejor modelo (RLIUP) una precisión en el conjunto de generalización de 99,2 % y 98,7 % en mitad de junio. Estos resultados sugieren que es posible implementar una estrategia para el tratamiento específico de malas hierbas con errores de omisión y comisión mínimos, y, de este modo, con una muy baja probabilidad de no detectar correctamente los rodales de *R. segetum*. El artículo propone la aplicación de una nueva metodología que, en nuestro conocimiento, no ha sido previamente aplicada en Teledetección, y que obtiene una mejor precisión que otras metodologías de *clasificación* habituales en el área de Teledetección, como los índices de vegetación o el método del mapeo de ángulo espectral.

La tercera publicación aborda un problema de *clasificación* en el campo del Análisis de datos de Cromatografía de Gases acoplada con Espectrometría de Masas. Este trabajo investiga la capacidad de los modelos de RL utilizando transformaciones no lineales de las covariables como una técnica de reconocimiento de patrones multi-clase para discriminar señales analíticas altamente solapadas partiendo de un número muy pequeño de covariables de entrada. Para este propósito, aplicamos RNUPs evolutivas, el modelo RLUP, y el modelo RLIUP, basándonos en la combinación de términos lineales y no lineales, estos últimos siendo son una transformación de los lineales utilizando funciones de base de tipo UP. Para comprobar esta aproximación, distintas muestras de agua potable contaminadas con compuestos volátiles orgánicos tales como el benceno, el tolueno, el xileno y de mezclas de los mismos, fueron clasificados en siete clases a partir de los datos proporcionados por su análisis espectrométrico de masas basado en espacio de cabeza. En lugar de utilizar el perfil total iónico proporcionado por el detector espectrométrico de masas como variables de entrada, los tres parámetros de la curva de Weibull ajustada a dichos datos, fueron empleados como covariables lineales para los modelos estándar de RL múltiple, mientras que las funciones de base de UP o su combinación con las covariables iniciales, fueron utilizadas para los modelos no lineales. El modelo híbrido no lineal, simplificado por un procedimiento por pasos hacia atrás, proporcionó los mejores resultados de *clasificación* con un porcentaje de patrones bien clasificados para los conjuntos de entrenamiento y generalización de un 100 y un 76,2 %, respectivamente. Realizamos la comparación los resultados obtenidos con otras metodologías estándar de *clasificación* como son: el ADL y el Análisis Discriminante Cuadrático (ADC), SVM, árboles de decisión C4.5 y RL estándar. Las dimensiones reducidas del modelo propuesto, formado por sólo tres términos, en concreto una covariable inicial y dos UPs, permitieron la inferencia de interpretaciones interesantes desde un punto de vista químico.

Regresión Logística utilizando Funciones de Base de tipo Radial

En esta parte (Sección 3.2 de este documento), se propone un nuevo modelo híbrido de Regresión Logística Generalizada, esta vez utilizando funciones de base de tipo radial (RBF). A esta parte asociamos la siguiente publicación, la cual abre la investigación de nuevos modelos de Regresión Logística Generalizada:

1. P.A. Gutiérrez, C. Hervás-Martínez, F.J. Martínez-Estudillo. Logistic Regression by means of Evolutionary Radial Basis Function Neural Networks. Enviado a IEEE Transactions on Neural Networks.

Esta publicación propone una nueva metodología híbrida multi-logística, a la que se denomina Regresión Logística utilizando las covariables Iniciales y Funciones de Base Radial (RLIRBF). El proceso para obtener los coeficientes se organiza en tres pasos. Primero, aplicamos el algoritmo NNEP, orientado a obtener una RNRBF con un número reducido de transformaciones RBF y la estructura más simple posible. Posteriormente, ampliamos el espacio de las covariables iniciales, añadiendo las transformaciones no lineales de las covariables de entrada dadas por las RBFs del mejor individuo en la última generación del algoritmo. Finalmente, aplicamos un procedimiento de optimización basado en máxima verosimilitud que determina los coeficientes asociados con un modelo de regresión multi-logística construido sobre este espacio de covariables iniciales y de base radial. En esta publicación, en lugar de utilizar el algoritmo IRLS junto con técnicas de simplificación estructural, hemos elegido un algoritmo (presente en el software WEKA) que construye el modelo de RL de forma incremental empezando con el modelo vacío y aplicando validación cruzada para decidir cuando dejar de añadir nuevas variables. De esta forma, aplicamos dos algoritmos diferentes de regresión multi-logística, uno que considera todas las covariables iniciales y RBF (*MultiLogistic Initial-RBF regression*, MLIRBF) y el mencionado anteriormente, que incrementalmente construye el modelo y aplica validación cruzada, realizando una selección de variables automática (*SimpleLogistic Initial-RBF regression*, SLIRBF). Comprobamos la metodología propuesta utilizando 18 bases datos de tipo *benchmark* y dos problemas reales agronómicos. Los resultados son comparados con los correspondientes métodos de regresión multi-logística aplicados sobre el espacio inicial de covariables, con las RNRBFs obtenidas por el algoritmo NNEP y con otros métodos clásicos de *clasificación* (LMT, árboles de decisión C4.5, *Naïve Bayes*, el algoritmo AdaBoost.M1 utilizando árboles C4.5 como clasificadores base, una RNRBF de tipo Gaussiano y el algoritmo C-SVM para la obtención de SVMs, con *kernels* de tipo Gaussiano). Incorporamos tests estadísticos para establecer la significación de las diferencias de eficiencia obtenidas por los distintos métodos. Los resultados indican que los modelos SLIRBF son mejores que los correspondientes modelos de regresión multi-logística y que las RNRBFs obtenidas por el algoritmo NNEP para casi todas las bases de datos. Además, estos modelos obtienen el ranking medio más alto en precisión en el conjunto de generalización cuando los comparamos con el resto de métodos en todas las bases de datos.

1.6. Comentarios Finales: Resultados y Conclusiones

Esta sección está dedicada a resumir brevemente los resultados obtenidos y a destacar las conclusiones que esta Tesis Doctoral aporta.

En general, hemos desarrollado nuevos modelos de *clasificación* y *regresión* basados en RNAs y en técnicas estadísticas clásicas como la RL. Estos modelos se han mostrado como alternativas muy interesantes a los modelos lineales, siendo capaces de resolver aquellos problemas en que los modelos lineales no son capaces de obtener unos resultados aceptables. La dificultad que supone la optimización tanto de la estructura de estos modelos como de los coeficientes que llevan asociados, ha llevado a desarrollar diversos AEs con este fin, tomando como algoritmo de partida el algoritmo NNEP propuesto por Martínez-Estudillo et al. (2005). La eficiencia de estos modelos y su aplicabilidad en problemas reales ha sido demostrada con diversas aplicaciones en Teledetección Agronómica y de Cinética Química, obteniendo estos nuevos modelos mejores resultados que otras técnicas estadísticas estándar o metodologías específicas utilizadas en cada una de las áreas. La significación de las diferencias obtenidas ha sido comprobada mediante la utilización de tests de hipótesis paramétricos o no paramétricos, según el caso.

Los siguientes apartados resumen brevemente los resultados obtenidos en cada una de las partes de la Tesis Doctoral y presentan algunas conclusiones sobre los mismos.

1.6.1. Redes Neuronales Evolutivas

En esta sección, hemos propuesto nuevos modelos de RNAs de tipo *feedforward* o de propagación hacia delante entrenados con AEs. Además, hemos aplicado los mismos en problemas reales. Los resultados indican que las propuestas son eficaces, tanto en los problemas de prueba considerados como en las aplicaciones reales.

Entrenamiento de Redes Neuronales de Unidad Producto utilizando un Algoritmo de Programación Evolutiva

Las conclusiones obtenidas en las dos publicaciones asociadas a esta parte son las siguientes:

- Hemos aplicado los distintos algoritmos híbridos propuestos en el Capítulo 5 de la Tesis Doctoral de Martínez-Estudillo et al. (2005) a un problema real de *regresión* de Predicción de Cosecha en Agricultura de Precisión. El estudio realizado demuestra la capacidad de las RNUPs evolutivas para analizar imágenes multi-espectrales y datos de presencia de malas hierbas y de elevación del terreno a la hora de predecir la producción de girasol en las distintas fincas analizadas.

- Hemos propuesto la utilización de los modelos de RNUPs obtenidos a lo largo de la evolución por el algoritmo PEHCD para así formar *ensembles* o conjuntos de modelos, considerando como salida del *ensemble* la media o la moda de las predicciones de los modelos que lo forman. Los resultados estadísticos y los tests de comparaciones señalan que, para el problema de Predicción de Producción evaluado, estos *ensembles* consiguieron el menor error de predicción junto con la mayor estabilidad en los resultados (con una desviación típica muy pequeña).

- Hemos estudiado los modelos de RNUPs obtenidos desde un punto de vista agronómico, demostrando que la Producción en campos de girasol infestados de *R. segetum* es influida por esta infestación, pero que también viene influida por ciertas relaciones complejas entre la elevación y/o los datos espectrales. Dado que el error de predicción en agricultura de precisión debe ser mínimo, la utilización de las metodologías propuestas en la primera publicación es altamente recomendable frente a métodos clásicos estadísticos de *regresión*, pese a tener asociadas un mayor coste computacional.

- Hemos propuesto un método de *clasificación*, que combina modelos de RNUPs y un algoritmo de PE para estimar tanto su estructura como sus coeficientes. De esta forma, la segunda publicación de esta parte supone el primer estudio de aplicación de RNUPs a problemas de multi-clasificación. Los resultados son prometedores tanto en términos de precisión en la clasificación, como en número de conexiones y número de nodos de las redes obtenidas.

- Hemos mostrado la superioridad de los clasificadores basados en RNUPs frente a los modelos clásicos MLP entrenados con el mismo algoritmo para los problemas *benchmark* y el problema real analizado. Además, concluimos que el modelo evolutivo propuesto de entrenamiento de clasificadores de RNUPs proporciona modelos comparables a otros modelos evolutivos complejos de entrenamiento de redes de tipo MLP, incluyendo el algoritmo cooperativo coevolutivo COVNET y los algoritmos multi-objetivo MPANN y SPANN.

Diseño de Perceptrones Multi-capa utilizando un Algoritmo de Programación Evolutiva basado en Dientes de Sierra Guiados

A partir de la publicación asociada a esta parte, se pueden obtener las siguientes conclusiones:

- La aplicación del esquema basado en dientes de sierra al algoritmo NNEP ha probado ser viable en el diseño de clasificadores de tipo MLP. Además hemos diseñado una versión mejorada dinámica de dichos dientes de sierra, realizando reinicializaciones de la población cuando la diferencia de la varianza de la aptitud de los mejores individuos entre dos generaciones consecutivas del algoritmo es menor que un determinado porcentaje de la varianza de la generación anterior. Hemos incorporado este mecanismo dinámico en el algoritmo NNEP, del mismo modo que podría aplicarse en otros AEs.

- El coste computacional del algoritmo NNEP se reduce al utilizar el mecanismo de dientes de sierra, y el coste obtenido al aplicar la versión dinámica propuesta es aún menor que el asociado a la versión original. Todo ello sin que esta disminución del coste computacional suponga un descenso en la precisión de los modelos obtenidos.

- Por último, a partir de los resultados del trabajo, podemos afirmar que la varianza de la distribución de la aptitud de los mejores individuos puede ser considerada como una medida adecuada para guiar los reinicios e introducir diversidad en el algoritmo NNEP. Considerando además que los experimentos realizados muestran la necesidad de un tamaño poblacional de 1.000 individuos para muchas bases de datos, esta necesaria diversidad puede ser obtenida a partir de los esquemas propuestos en este artículo, evitando el alto coste computacional asociado a un tamaño poblacional tan alto.

Entrenamiento de Redes Neuronales con Funciones de Base de tipo *kernel* y de tipo Proyección utilizando un Algoritmo de Programación Evolutiva

En esta parte, se han obtenido las siguientes conclusiones a partir de los resultados de las dos publicaciones asociadas:

- Los modelos híbridos de RNAs para *clasificación* utilizando una combinación de funciones de base para la capa oculta de la red son una alternativa interesante. Las funciones de base consideradas han incluido USs, UPs y RBFs, y hemos diseñado todos los posibles modelos que se pueden obtener al tomar dichas funciones de base por pares: redes neuronales híbridas PRBF, SRBF y USP. Hemos entrenado estos modelos utilizando un algoritmo específico de PE, que tuvo en cuenta las características especiales de cada tipo de función de base.

- Los resultados obtenidos en la primera publicación nos confirman la teoría de que una tarea de *clasificación* puede ser llevada a cabo utilizando un modelo con dos componentes diferenciadas (Donoho y Johnstone, 1989): una asociada a funciones de proyección (US o UP) y otro asociado a funciones de tipo *kernel* (RBF).

- Determinar cuál es la función de proyección más adecuada depende en gran medida de la base de datos que se esté utilizando, pero, en general, los modelos híbridos SRBF han llevado a una mayor precisión que los modelos híbridos PRBF, especialmente en aquellas bases de datos con dos clases, en las que existe una única función discriminante. La capacidad de generalización obtenida al utilizar funciones de base que aunan los efectos locales y globales de las variables de entrada al realizar la *clasificación* (es decir, los modelos SRBF y PRBF) es similar o mayor que la capacidad de generalización de las RNUPs, las redes tipo MLP o las redes de tipo RBF. Por otro lado, la hibridación de las dos funciones de proyección consideradas (UP y US) no ha presentado mejores resultados que el resto de hibridaciones pero si que presenta un número de enlaces mucho menor (permitiendo modelos mucho más interpretables) en algunas bases de datos. Podemos considerar que la capacidad de generalización de los modelos USP es equiparable a la obtenida con los correspondientes modelos puros de RNUPs o de redes de tipo MLP y que, en general, todos aquellos modelos híbridos que incorporan UPs (PRBF y USP) obtienen un número de enlaces menor que los modelos SRBF, debido a que estas funciones de base suelen necesitar un número menor de nodos para obtener una cantidad de información similar a la obtenida con USs o RBFs.

- Por otro lado, en la segunda publicación, hemos mostrado que los modelos de RNAs evolutivas son capaces de realizar una análisis cuantitativo de los picos obtenidos a partir de Electroforesis Capilar, incluso si estos están altamente solapados. Hemos probado como los cuatro parámetros de las curvas de Weibull asociadas a estos picos son muy adecuados como entradas para los tres tipos de redes neuronales evaluadas (RNUPs, redes de tipo MLP y redes de tipo USP) y como, además, una etiqueta de clase incluida para cada una de las muestras a partir de un ADL puede mejorar los resultados para algunos casos.

- Los resultados obtenidos indican que los modelos de redes neuronales evaluados son muy adecuados para caracterizar los picos obtenidos a partir de Electroforesis Capilar con un error aceptable. Del análisis de los resultados de esta segunda publicación, hemos concluido que los modelos de RNUPs son los que proporcionan para este problema los resultados mejores y más robustos, además de presentar arquitecturas más simples que pueden ser más fácilmente interpretables desde un punto de vista químico.

- De manera general, los resultados de la segunda publicación nos indican también que las RNAs son una herramienta útil en el análisis de los picos obtenidos a partir de Electroforesis Capilar, ya que evitan el consumo de tiempo que puede suponer el encontrar las condiciones óptimas para la resolución de dichos picos.

1.6.2. Regresión Logística Generalizada utilizando Funciones de Base

En esta sección, hemos propuesto un nuevo tipo de modelos, que englobamos bajo el término Regresión Logística Generalizada utilizando Funciones de Base y distintas aplicaciones de los mismos. Los resultados muestran que las propuestas son eficaces, tanto en los problemas de prueba considerados como en las aplicaciones reales.

Regresión Logística utilizando Funciones de Base de tipo Unidad Producto

Las conclusiones obtenidas a partir de las tres publicaciones asociadas a esta parte son las siguientes:

- Ha sido apropiado automatizar la metodología descrita en artículos anteriores (Hervás-Martínez y Martínez-Estudillo, 2007; Hervás-Martínez et al., 2008), para así obtener un paquete software único capaz de generar modelos RLUP y RLIUP.

- Hemos investigado diversas alternativas en la etapa de simplificación de los modelos obtenidos: una búsqueda hacia atrás con vuelta a estados anteriores (BBS) y un procedimiento similar pero basado en la utilización de un procedimiento estándar de enfriamiento simulado (SABBS). La comparación muestra que estos algoritmos reducen el número de coeficientes con una precisión similar o mejor, presentando modelos más interpretables. Esto es importante para el problema tratado (Clasificación de Rodales de Malas Hierbas) ya que puede redundar en un menor coste para futuras imágenes tomadas en el área con el mismo fin. Por otro lado, la comparación de los modelos implementados frente a los seis algoritmos comunes de *clasificación* escogidos muestra que estos modelos son competitivos y que poseen un número de coeficientes mucho menor.

- En la segunda publicación, hemos mostrado la capacidad de los modelos de Regresión Logística Generalizada utilizando UPs para analizar imágenes multi-espectrales a la hora de predecir la probabilidad de presencia de *R. segetum* en diversas fincas de estudio y mapear los rodales de *R. segetum* basándonos en dichas probabilidades. Los modelos de RNUPs evolutivas y los modelos RLIUP han obtenido mejores resultados que la RL estándar tanto en entrenamiento como en generalización. Hemos obtenido los mejores resultados para este problema mediante la aplicación del modelo RLIUP a mitad de junio en ambas localizaciones. Para resolver este mismo problema, hemos aplicado otras técnicas de *clasificación* (SVM y LMT), y hemos demostrado como el modelo RLIUP obtiene mejores resultados de generalización para casi todos las fechas y localizaciones evaluadas. De la observación de mapas de rodales de mala hierba generados, se puede deducir también que la calidad de los mapas generados con los modelos propuestos es mejor que la de los mapas generados con un modelo estándar de RL.

- Además, esta segunda publicación corrobora que la fecha o estado fenológico en que se toman las fotografías aéreas influye significativamente en la precisión con la que se pueden discriminar los rodales de *R. segetum* en los dos campos de girasol considerados, siendo el orden de precisión el siguiente: mitad de junio > mitad de julio > mitad de mayo. De esta forma, recomendamos mapear los rodales de mala hierba a mitad de junio, para así aplicar una política de gestión in-situ de malas hierbas para los años siguientes.

- Los modelos RLIUP también han demostrado ser una herramienta muy útil en el Análisis de datos de Cromatografía de Gases acoplada con Espectrometría de Masas, como hemos mostrado en la tercera publicación asociada a esta parte. De esta forma, el uso de los modelos RLIUP mejora los resultados obtenidos con la RL estándar al considerar los efectos no lineales de las covariables de UPs. Los modelos propuestos han mejorado los resultados obtenidos por los dos modelos estándar de RL considerados y aquellos alcanzados por el ADL y el ADC y por los modelos C4.5 y SVM. Debemos tener en cuenta, además, que la dificultad del problema es muy alta debido a que el número de muestras que se tiene es pequeño. También es importante señalar que de nuevo en este caso, el uso de los parámetros de la curva de Weibull asociada al perfil iónico total proporcionado por el espectrómetro de masas, ha propiciado la generación de modelos más pequeños e interpretables.

Regresión Logística utilizando Funciones de Base de tipo Radial

En esta parte, hemos comprobado las múltiples posibilidades que ofrecen los modelos de Regresión Logística Generalizada. A partir de la publicación asociada se obtienen las siguientes conclusiones:

- La combinación de la RL, los AEs y las RNRBFs para resolver problemas de *clasificación* multi-clase, construyendo un modelo de RL que combina las covariables iniciales y transformaciones Gaussianas de tipo RBF para la función de predicción, ha resultado muy útil para obtener clasificadores más eficaces.

- Con respecto a los trabajos anteriores donde se proponen modelos similares basados en RNUPs (Hervás-Martínez y Martínez-Estudillo, 2007; Hervás-Martínez et al., 2008), la aportación principal (además del uso de otro tipo de funciones de base) es la utilización de algoritmos más avanzados para el último paso de construcción del modelo final de RL. Este algoritmo realiza una selección automática de covariables y construye modelos más simples que además generalizan mejor.

- Del análisis de los resultados obtenidos, obtenemos las siguientes conclusiones. En primer lugar, la selección de covariables incorporada por el algoritmo mencionado en los modelos SLIRBF y SLRBF es necesaria en algunas bases de datos para evitar el sobreentrenamiento. Además, los resultados obtenidos por los métodos MLIRBF y SLIRBF (construidos en el conjunto de las covariables iniciales y las covariables RBF) son mejores para casi todas las bases de datos que aquellos obtenidos por sus modelos equivalentes de RL, MLogistic y SLogistic (construidos únicamente en las covariables iniciales). Finalmente, SLIRBF es un método muy competitivo, obteniendo valores de precisión muy altos cuando se compara con otros métodos de *clasificación* como LMT o SVM.

1.7. Líneas y Propuestas Futuras

A continuación, se presentan algunas líneas de trabajo futuras que se plantean a partir de los métodos propuestos en esta memoria.

Aplicación en otros Problemas Reales

Los modelos y algoritmos propuestos para su entrenamiento se pueden aplicar a problemas de *clasificación* multi-clase y de *regresión* no lineal en muy diferentes áreas de conocimiento y han sido diseñados como modelos de propósito general. De esta forma, ésta es la primera línea sobre la que continuaremos la investigación llevada a cabo en esta Tesis Doctoral. En concreto, continuamos desarrollando y aplicando nuestros modelos en diversos problemas reales de otras áreas:

- En colaboración con la Unidad de Agricultura de Precisión y Teledetección del Departamento de Protección de Cultivos del Instituto de Agricultura Sostenible (Consejo Superior de Investigaciones Científicas, CSIC), se ha abierto una línea de investigación que versa sobre la clasificación de cubiertas vegetales mediante Teledetección. Actualmente, los agricultores olivareros sólo reciben subvenciones de la Unión Europea si no labran y mantienen entre las calles del cultivo cubiertas vegetales. Estas cubiertas aumentan la retención del agua de lluvia y evitan la erosión y la pérdida de suelo fértil. Las inspecciones de campo para controlar la percepción de ayudas se realiza con visitas sólo al 1 % del total de fincas. Obviamente este método es inexacto y para solventar estos problemas, es viable la utilización de técnicas de Teledetección junto con clasificadores lo más precisos posible.

- En colaboración con el grupo de investigación "Heurísticos modernos de optimización y diseño de redes de comunicaciones" (GHEODE) de la Universidad de Alcalá, Madrid, se han abierto dos líneas nuevas. La primera versa sobre modelos para la *clasificación* de crisis financieras en base a un conjunto de variables macroeconómicas y financieras. La segunda corresponde a un problema de predicción o *regresión* de la velocidad del viento en varios aerogeneradores de un parque eólico, necesaria para estimar la producción de potencia del parque a 24 y 48 horas vista.

- En colaboración con el Profesor Ricardo Grau del Departamento de Informática de la Universidad de Las Villas en Santa Clara, Cuba, se está trabajando en un problema de *clasificación* de análisis de secuencias en Bioinformática. En concreto, se pretende utilizar una base de datos con secuencias de mutaciones de la proteasa HIV-1, cada una con 99 codones.

- En colaboración con el Doctor Javier Briceño Delgado de la Unidad de Trasplantes Hepáticos del Hospital Reina Sofía, se plantea una nueva línea de investigación consistente en la asignación de donantes y receptores de trasplantes hepáticos, valorando los donantes a través de distintas variables biológicas antes de realizar la extracción de órganos y asignarlos al mejor receptor.

- En colaboración con el Profesor Carlos García Alonso de la Facultad de Económicas y Empresariales (ETEA) de Córdoba y miembro del Grupo de Investigación AYRNA, plantearemos la resolución de un problema de *clasificación* consistente en la asignación de la eficiencia económica de diversas explotaciones agrarias, en base a los datos provenientes de un amplio conjunto de encuestas realizadas a las mismas.

Utilización de otros Modelos de Regresión Logística

El modelo de RL presentado en la Sección 1.1.6 es el modelo estándar basado en el uso de la transformación *logit* y utilizado para variables categóricas nominales, es decir, en las que las distintas categorías a clasificar no tienen un orden natural establecido. Sin embargo, en la literatura especializada, se plantean diversas alternativas a este modelo:

- Regresión Probit: la transformación *probit*, cuyo nombre procede de la unión de los términos *probability* y *unit*, es muy similar a la transformación *logit* utilizada en el modelo de RL estándar, pero utiliza una distribución normal de probabilidad acumulada para estimar la probabilidad del suceso de interés dados los valores del vector de características.
- Regresión Logística Ordinal: este modelo extiende el modelo de RL a aquellos casos en que la variable categórica no es nominal sino ordinal, es decir, las categorías poseen un orden natural (por ejemplo, un grado de cáncer tumoral que va desde tumores muy diferenciados hasta tumores vagamente reconocidos). En este caso, el modelo se basa en transformaciones *logit* acumuladas.

Los modelos planteados en esta Tesis Doctoral a través de las publicaciones incluidas en el Capítulo 3 podrían ser extendidos a este tipo de modelos de RL, utilizando las funciones de base obtenidas a partir de los distintos modelos de RNAs como transformaciones adicionales a las covariables iniciales.

Por otro lado, los modelos planteados en el Capítulo 3 han utilizado un sólo tipo de función de base (UP o RBF) en la parte del modelo correspondiente a las transformaciones no lineales. Una posibilidad adicional consistiría en el uso de distintos tipos de funciones de base para construir la parte no lineal, utilizando los modelos de RNAs híbridos desarrollados en la Sección 2.3.

Incorporación de Información obtenida a partir de la Aptitud de los Mejores Individuos

Otra línea de investigación abierta con esta Tesis Doctoral (más en concreto con la publicación incorporada en la Sección 2.2) supondría la utilización de la información que suponen los valores de aptitud de los mejores individuos en los AEs. De este modo, la caracterización de dichos valores de aptitud utilizando distintos tipos de distribuciones estadísticas puede propiciar un entrenamiento más rápido y preciso. Esto ha quedado reflejado en la utilización de la varianza de la aptitud de los mejores individuos para guiar los dientes de sierra, lo que ha llevado a una generación de diversidad dinámica, capaz de decidir en que momentos de la evolución los nuevos individuos añadidos en el siguiente reinicio pueden ayudar al algoritmo a salir de una posible situación de estancamiento en cuanto a los valores de aptitud actuales.

Utilización de Funciones de Base Radial Generalizadas

Cuando se analizan datos con una alta dimensionalidad (es decir, con un número de variables de entrada muy elevado, caso bastante común en la aplicaciones reales de Minería de Datos), los *kernel* de tipo Gaussiano pierden su interpretación en términos de localidad alrededor del centro de los mismos. La necesidad de abarcar más o menos el rango efectivo de las distancias entre los patrones en un problema real mediante la parte efectiva del *kernel* (es decir, la parte con pendiente decreciente) requiere el uso de un parámetro adicional en el *kernel* utilizado. Las Funciones de Base Radial Generalizadas (*Generalized RBFs*, GRBFs) son funciones de base, en las cuales se añade un exponente capaz de controlar la distancia más pequeña que corresponde a la parte decreciente del *kernel.* De esta forma, creemos que este tipo de funciones de base pueden obtener resultados muy buenos en problemas de *clasificación* y, por tanto, el diseño de modelos basados en GRBFs mediante AEs es otra de las futuras líneas de investigación de esta Tesis Doctoral.

Modelos Multiobjetivo Evolutivos de RNAs

Otra línea abierta a partir de los algoritmos y modelos propuestos en esta Tesis Doctoral, consiste en la utilización de algoritmos evolutivos muti-objetivo para el diseño de los modelos planteados. De esta forma, siguiendo esta línea ya iniciada por el Profesor Juan Carlos Fernández Caballero de la Universidad de Córdoba, se han propuesto modelos mono-objetivo (Martínez-Estudillo et al., 2008) y multi-objetivo (Fernández et al., 2008) en los que se optimizan dos medidas de bondad de los clasificadores que a menudo se encuentran en conflicto: su precisión total en cuanto a porcentaje de patrones bien clasificados y su sensitividad o, lo que es lo mismo, su precisión obtenida en la clase que peor se clasifica.

Bibliografía

Abbass, H.: 2003, Speeding up backpropagation using multiobjective evolutionary algorithms, *Neural Computation* **15**, 2705–2726.

Angeline, P. J., Sauders, G. M. y Pollack, J. B.: 1994, An evolutionary algorithm that constructs recurren neural networks, *IEEE Transactions on Neural Networks* **5**, 54–65.

Battiti, T.: 1992, First and second-order methods for learning: Between steepest descent and Newton's method, *Neural Computation* **4**(2), 141–166.

Bishop, C. M.: 1991, Improving the generalization properties of radial basis function neural networks, *Neural Computation* **3**(4), 579–581.

Bishop, C. M.: 1996, *Neural networks for pattern recognition*, Oxford University Press, Oxford, UK.

Bishop, C. M.: 2006, *Pattern Recognition and Machine Learning*, Springer.

Blanco, A., Delgado, M. y Pegalajar, M. C.: 2001, A real-coded genetic algorithm for training recurrent neural networks, *Neural Networks* **14**(1), 93–105.

Cichocki, A. y Unbehauen, R.: 1993, *Neural Networks for Optimization and Signal Processing*, John Wiley.

Cobb, H. G. y Grefenstette, J. J.: 1993, Genetic algorithms for tracking changing environments, *Proceedings of the 5th International Conf. on Genetic Algorithms*, pp. 523–530.

Cohen, S. y Intrator, N.: 2002, A hybrid projection-based and radial basis function architecture: initial values and global optimization, *Pattern Anal. Appl* **5**, 113–120.

Cybenko, G.: 1989, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals, and Systems* **2**(4), 303–314.

Davidon, W. C.: 1991, Variable metric method for minimization, *SIAM Journal on Optimization* **1**, 1–17.

Donoho, D. L. y Johnstone, I. M.: 1989, Projection-based approximation and a duality with kernel methods, *The Annals of Statistics* **17**(1), 58–106.

Duch, W., Adamczak, R. y Diercksen, G.: 2001, Constructive density estimation network based on several different separable transfer functions, *Proceedings of the 9th European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 107–112.

Duch, W. y Jankowski, N.: 2001, Transfer functions hidden possibilities for better neural networks, *Proceedings of the 9th European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 81–94.

Durbin, R. y Rumelhart, D.: 1989, Products units: A computationally powerful and biologically plausible extension to backpropagation networks, *Neural Computation* **1**(1), 133–142.

Eiben, A. E., Hinterding, R. y Michalewicz, Z.: 1999, Parameter control in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation* **3**(2), 124–141.

Engelbrecht, A. y Ismail, A.: 1999, Training product unit neural networks, *Stability and Control: Theory and Applications* **2**(1–2), 59–74.

Eshelman, L. J.: 1991, The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, *Proceedings of the 1st Workshop on Foundations on Genetic Algorithms*, pp. 257–266.

Fahlman, S. E. y Lebiere, C.: 1990, The cascade-correlation learning architecture, *in* D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, Vol. 2, Morgan Kaufmann, San Mateo Inc., San Francisco, CA, pp. 524–532.

Fernández, J. C., Gutiérrez, P. A., Hervás-Martínez, C. y Martínez-Estudillo, F. J.: 2008, Memetic pareto evolutionary artificial neural networks for the determination of growth limits of listeria monocytogenes, *Proceedings of the 8th International Conference on Hybrid Intelligent Systems*, Barcelona (Spain), pp. 631–636.

Fogel, L. J.: 1966, *Artificial Intelligence through Simulated Evolution*, 1st edn, John Wiley & Sons, New York.

Freund, Y. y Schapire, R. E.: 1996, Experiments with a new boosting algorithm, *Proceedings of the Thirteenth International Conference on Machine Learning*, Morgan Kaufmann, pp. 148–156.

Friedman, J.: 1991, Multivariate adaptive regression splines (with discussion), *Annals of Statistics* **19**, 1–141.

Friedman, J. H.: 1993, *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, Springer, chapter An Overview of Predictive Learning and Function Approximation, pp. 1–61.

García-Pedrajas, N., Hervás-Martínez, C. y Muñoz-Pérez, J.: 2003, Covnet: A cooperative coevolutionary model for evolving artificial neural networks, *IEEE Transaction on Neural Networks* **14**(3), 575–596.

Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional.

Goldberg, D. E., Deb, K. y Clark, J. H.: 1992, Genetic algorithms, noise, and the sizing of populations, *Complex Systems* **6**(4), 333–362.

González, J., Rojas, I., Ortega, J., Pomares, H., Fernández, F. J. y Días, A. F.: 2003, Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation, *IEEE Transactions Neural Networks* **14**(6), 1478–1495.

Hagan, M. T. y Menhaj, M. B.: 1994, Training feedforward networks with the Marquardt algorithm, *IEEE Transactions on Neural Networks* **5**(6), 989–993.

Hastie, T. J. y Tibshirani, R. J.: 1996, Nonparametric regression and classification. Part II - nonparametric classification, *in* V. Cherkassky, J. H. Friedman y H. Wechsler (eds), *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, Vol. 136 of *Computer and System Sciences*, Springer, pp. 70–82.

Hastie, T., Tibshirani, R. y Friedman, J. H.: 2001, *The Elements of Statistical Learning*, Springer.

Haykin, S.: 1998, *Neural Networks: A comprehensive Foundation*, Prentice Hall, 2nd edition, Upper Saddle River, NJ, USA.

Herrera, F. y Lozano, M.: 1996, Adaptation of genetic algorithm parameters based on fuzzy logic controllers, *in* F. Herrera y J. L. Verdegay (eds), *Genetic Algorithms and Soft Computing*, Physica-Verlag, Heidelberg, pp. 95–125.

Hervás-Martínez, C., Martínez-Estudillo, F. J. y Carbonero-Ruz, M.: 2008, Multilogistic regression by means of evolutionary product-unit neural networks, *Neural Networks* **21**(7), 951–961.

Hervás-Martínez, C. y Martínez-Estudillo, F.: 2007, Logistic regression using covariates obtained by product-unit neural network models, *Pattern Recognition* **40**(1), 52–64.

Holland, J.: 1975, *Adaptation in Natural and Artificial Systems*, University of Michigan Press (SecondEdition: MIT Press 1992).

Hornik, K., Stinchcombe, M. y White, H.: 1989, Multilayer feedforward networks are universal approximators, *Neural Networks* **2**(5), 359–366.

Hosmer, D. W. y Lemeshow, S.: 2000, *Applied logistic regression*, Wiley Series in Probability and Statistics, Wiley Interscience, New York; Chichester.

Ismail, A. y Engelbrecht, A. P.: 1999, Training product units in feedforward neural networks using particle swarm optimization, *in* V. Bajic y D. Sha (eds), *Proceedings of the 1999 International Conference on Articial Intelligence*, Durban, South Africa, pp. 36–40.

Ismail, A. y Engelbrecht, A. P.: 2000, Global optimization algorithms for training product unit neural networks, *Proceedings of the 2000 International Joint Conference on Neural Networks*, pp. 132–137.

Ismail, A. y Engelbrecht, A. P.: 2002, Pruning product unit neural networks, *Proceedings of the 2002 International Joint Conference on Neural Networks*, pp. 257–262.

Iulian, B. C.: 2002, Hybrid feedforward neural networks for solving classification problems, *Neural Process. Lett.* **16**(1), 81–91. 607865.

Jacobs, R. A.: 1988, Increased rates of convergence through learning rate adaptation, *Neural Networks* **1**(4), 359–366.

Jankowski, N. y Duch, W.: 2001, Optimal transfer function neural networks, *Procedings of the 9th European Symposium on Artificial Neural Networks*, Bruges, Belgium, pp. 101–106.

Janson, D. J. y Frenzel, J. F.: 1993, Training product unit neural networks with genetic algorithms, *IEEE Expert* **8**(5), 26–33.

Johansson, E. M., Dowla, F. U. y Goodman, D. M.: 1992, Backpropagation learning for multilayer feedforward neural networks using the conjugate gradient method, *International Journal of Neural Systems* **2**(4), 291–301.

Koumousis, V. K. y Dimou, C. K.: 2002, The effect of oscillating population size on the performance of genetic algorithms, *Proceedings of the 4th GRACM Congress on Computational Mechanics.*

Koumousis, V. K. y Katsaras, C. P.: 2006, A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance, *IEEE Transactions on Evolutionary Computation* **10**(1), 19–28.

Lee, S. H. y Hou, C. L.: 2002, An art-based construction of RBF networks, *IEEE Transactions on Neural Networks* **13**(6), 1308–1321.

Lehtokangas, M. y Saarinen, J.: 1998, Centroid based multilayer perceptron networks, *Neural Processing Letters* **7**, 101–106. [1] [2].

Lippmann, R.: 1989, Pattern classification using neural networks, *IEEE Com. Mag* **27**, 47–64.

Lippmann, R. P.: 1996, Neural networks, bayesian a posteriori probabilities, and pattern classification, *in* V. Cherkassky, J. H. Friedman y H. Wechsler (eds), *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, Vol. 136 of *Computer and System Sciences*, Springer, pp. 83–104.

Lozano, M., Herrera, F. y Cano, J. R.: 2008, Replacement strategies to preserve useful diversity in steady-state genetic algorithms, *Information Sciences* **178**(23), 4421–4433.

Manrique, D., Ríos, J. y Rodríguez-Patón, A.: 2006, Evolutionary system for automatically constructing and adapting radial basis function networks, *Neurocomputing* **69**(16-18), 2268–2283.

Marquardt, D.: 1963, An algorithm for least-squares estimation of nonlinear parameters, *SIAM J. Appl. Math* **11**, 431–441.

Martínez-Estudillo, A. C.: 2005, *Modelos de Regresión Basados en Redes Neuronales de Unidades Producto Diseñadas y Entrenadas Mediante Algoritmos de Optimización Híbrida. Aplicaciones*, Tesis doctoral, Universidad de Granada.

Martínez-Estudillo, A. C., Hervás-Martínez, C., Martínez-Estudillo, F. J. y García-Pedrajas, N.: 2006a, Hybridization of evolutionary algorithms and local search by means of a clustering method, *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics* **36**(3), 534–545.

Martínez-Estudillo, A. C., Martínez-Estudillo, F. J., Hervás-Martínez, C. y García, N.: 2006b, Evolutionary product unit based neural networks for regression, *Neural Networks* **19**(4), 477–486.

Martínez-Estudillo, F., Hervás-Martinez, C., Martínez-Estudillo, A. y Ortiz-Boyer, D.: 2005, Memetic algorithms to product-unit neural networks for regression, *Computational Intelligence and Bioinspired Systems, Proceedings* **3512**, 83–90.

Martínez-Estudillo, F. J., Gutiérrez, P. A., Hervás-Martínez, C. y Fernández, J. C.: 2008, Evolutionary learning by a sensitivity-accuracy approach for multi-class problems, *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC'08)*, IEEE Press, Hong Kong, China, pp. 1581–1588.

McCullagh, P. y Nelder, J. A.: 1989, *Generalized Linear Models*, Monographs on Statistics and Applied Probability, 2nd edn, Chapman & Hall/CRC.

McCulloch, W. S. y Pitts, W. H.: 1943, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics* **5**, 115–133.

Montana, D. y Davis, L.: 1989, Training feedforward neural networks using genetic algorithms, *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 762–767.

Pao, Y. H. y Takefuji, Y.: 1992, Functional-link net computing: theory, system architecture, and functionalities, *IEEE Computer* **25**(5), 76–79.

Park, J. y Sandberg, I. W.: 1993, Approximation and radial-basis-function networks, *Neural Computation* **5**(2), 305–316.

Riedmiller, M. y Braun, H.: 1993, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, *Proceedings of the 1993 IEEE International Conference on Neural Networks*, San Francisco, CA, pp. 586–591.

Rosenblatt, M.: 1956, Remarks on some nonparametric estimates of a density function, *The Annals of Mathematical Statistics* **27**(3), 832–837.

Saito, K. y Nakano, R.: 1997, Numeric law discovery using neural networks, *Proceedings of the IVth International Conference on Neural Information Processing (ICONIP97)*, pp. 843–846.

Schmitt, M.: 2002, On the complexity of computing and learning with multiplicative neural networks, *Neural Computation* **14**, 241–301.

Sexton, R., Allidae, B., Dorsey, R. y Johnson, J.: 1998, Global optimization for artificial neural networks: A tabu search application, *European Journal of Operational Research* **106**(2–3), 570–584.

Sexton, R., Dorsey, R. y Johnson, J.: 1999, Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing, *European Journal of Operational Research* **114**(3), 589–601.

Smith, R. E.: 1993, Adaptively resizing populations: An algorithm and analysis, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 653–653.

Vapnik, V.Ñ.: 1999, *The Nature of Statistical Learning Theory*, Springer.

Vapnik, V.Ñ. y Chervonenkis, A. Y.: 1971, On the uniform convergence of relative frequencies of events to their probabilities, *Theory of Probability and its Applications* **16**, 264–280.

Wedge, D., Ingram, D., McLean, D., Mingham, C. y Bandar, Z.: 2006, On global-local artificial neural networks for function approximation, *IEEE Transactions on Neural Networks* **17**(4), 942–952.

Whitehead, B. A. y Choate, T. D.: 1996, Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction, *IEEE Transactions on Neural Networks* **7**(4), 869–880.

Whittaker, E. T.: 1922, On a new method of graduation, *Proceedings of the Edinburgh Mathematical Society* **41**, 63–75.

Wolpert, D. H. y Macready, W. G.: 1997, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Yao, X.: 1999, Evolving artificial neural networks, *Proceedings of the IEEE* **87**(9), 1423–1447.

Capítulo 2

Trabajos Publicados y Aceptados sobre Redes Neuronales Evolutivas

2.1. Desarrollo de Modelos de Redes Neuronales de Unidades Producto

Las publicaciones asociadas a esta parte son:

2.1.1. Elaboración de Mapas de Cosecha de Girasol Afectado por Rodales de *Ridolfia segetum* y por la Elevación del Terreno mediante la Aplicación de Redes Neuronales Evolutivas de Unidad Producto a Datos obtenidos mediante Sensores Remotos - *Mapping sunflower yield as affected by Ridolfia segetum patches and elevation by applying evolutionary product unit neural networks to remote sensed data*

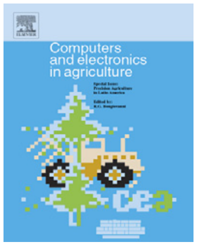
- P.A. Gutiérrez, F. López-Granados, J.M. Peña-Barragán, M. Jurado-Expósito, M.T. Gómez-Casero, C. Hervás-Martínez. Mapping sunflower yield as affected by *Ridolfia segetum* patches and elevation by applying Evolutionary Product Unit Neural Networks to remote sensed data. Computers and Electronics in Agriculture 60:2 (2008) 122-132.
    - Estado: Publicado.
    - Índice de Impacto (JCR 2007): 1.242.
    - Área de conocimiento: *Agriculture, Multidisciplinary.* Ranking 7/35. Primer Cuartil.
    - Área de conocimiento: *Computer Science, Interdisciplinary Applications.* Ranking 30/92. Segundo Cuartil.

computers and electronics in agriculture 60 (2008) 122–132

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/compag



Mapping sunflower yield as affected by *Ridolfia segetum* patches and elevation by applying evolutionary product unit neural networks to remote sensed data

P.A. Gutiérrez[b], F. López-Granados[b], J.M. Peña-Barragán[b,*], M. Jurado-Expósito[b], M.T. Gómez-Casero[b], C. Hervás-Martínez[a]

[a] *Department of Computing and Numerical Analysis, University of Cordoba, Campus de Rabanales, 14071 Cordoba, Spain*
[b] *Institute for Sustainable Agriculture, C.S.I.C., 14080 Cordoba, Spain*

ARTICLE INFO

Article history:
Received 7 February 2007
Received in revised form
18 July 2007
Accepted 20 July 2007

Keywords:
Remote sensing
Artificial neural networks
Yield prediction
Evolutionary algorithms
Sunflower

ABSTRACT

Recent advances in remote sensing technology have triggered the need for highly flexible modelling methods to estimate several crop parameters in precision farming. The aim of this work was to determine the potential of evolutionary product unit neural networks (EPUNNs) for mapping in-season yield and forecasting systems of sunflower crop in a natural weed-infested farm. Aerial photographs were taken at the late vegetative (mid-May) growth stage. Yield, elevation and weed data were combined with multispectral imagery to obtain the dataset. Statistical and EPUNNs approaches were used to develop different yield prediction models. The results obtained using different EPUNN models show that the functional model and the hybrid algorithms proposed provide very accurate prediction compared to other statistical methodologies used to solve that regression problem.

1. Introduction

One aspect of overcoming the possibility of minimizing the impact of agriculture on environmental quality is the development of more efficient approaches for crop production. The site-specific application of agricultural inputs (e.g., fertilizer or herbicide) based on accurate field maps is an essential component of the successful implementation of precision farming, leading to a reduction in the overall quantities applied (Karimi et al., 2005). Although the benefits obtained through yield mapping depend largely on the crop and the environmental conditions (Yang et al., 2000a), harvester-mounted crop yield monitoring systems are now being extensively used by farmers to maximize the yields under highly variable field conditions (Uno et al., 2005). As a first step to finding out the field variability, yield maps play an important role in the decision making process in precision agriculture.

One of the main field characteristics causing yield variability is the field topography or elevation (Kravchenko and Bullock, 2002a,b). Several authors have reported that the Elevation affects the spatial variability of yield (Kravchenko and Bullock, 2002a; Pilesjö et al., 2005) and weed intensity (Jurado-Expósito et al., 2005; Liu et al., 2002).

Sunflower (*Helianthus annuus* L.) is one of the most abundant crops in Andalusia, Southern Spain, grown on over 320,000 ha annually. It is usually grown under dry land conditions; sowing time is February–March and harvesting time July–August. *Ridolfia segetum* Moris is an umbelliferous weed, which is frequent and abundant in clay soils in Andalusia. Once established, the growth rate of *R. segetum* is very

* Corresponding author. Tel.: +34 957 499 219; fax: +34 957 499 252.
E-mail address: pa2pebaj@uco.es (J.M. Peña-Barragán).
0168-1699/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.compag.2007.07.011

rapid and its life cycle coincides with that of sunflower yield, which enhances the competitive ability of weed and results in an average yield reduction of about 32% when infestation is two *R. segetum* plants m^{-2} (Castro-Tendero and García-Torres, 1995; Carranza et al., 1995; Peña-Barragán et al., 2006). Although weed control is commonly achieved with preplanting incorporated and pre-emergence herbicides, none of these herbicides control *R. segetum* and consequently postemergence tillage or hand-weeding are frequently used to control this weed. Classical competition models not taking into account the spatial component of *R. segetum* infestations were developed by Carranza et al. (1995).

Remote sensing systems can provide a large amount of continuous field information at a reasonable cost, offering great advantages to understand the yield maps created by the harvester-mounted yield-monitoring units (Uno et al., 2005). The importance of remote sensing in site-specific agriculture has been widely reviewed for yield mapping of different crops (Plant, 2001; Yang et al., 2001, 2006). Remotely sensed imagery shows great potential in establishing the impact of seasonally changeable factors (e.g., precipitation, temperature and sunshine) in crop production (Uno et al., 2005), therefore one of the main challenges of remote imagery analysis is to determine how variations in the spectral information are related to differences in the crop state, in order to predict accurate yield maps long before the harvest. Peña-Barragán et al. (2007) obtained maps of *R. segetum* patches in sunflower using aerial photography, taking into account specific timeframes. However, no information has been found about the yield estimation of sunflower considering the presence of *R. segetum* patches and their multispectral characteristics on a field scale.

Different statistical, artificial intelligence and machine learning algorithms have proved to be quite useful in the interpretation of remotely sensed data. Kenkel et al. (2002) reported that multivariate statistical approaches have only recently started to be used in agricultural domain. The application of discriminant analysis to agricultural remote sensing in weed discrimination has been reported in many recent studies (Burks et al., 2002; Cho et al., 2002). Furthermore, the stepwise multiple linear regression (SMLR) is one of the most commonly used multivariate analytical techniques to develop empirical models from large data sets, as has been done for a number of canopy-level crop condition parameters (Shibayama and Akiyama, 1991; Osborne et al., 2002). This model fitting process is quite stable, resulting in low variance but in a potentially high bias. A traditional technique to overcome these difficulties is augmenting/replacing the input vector with new variables, the basis functions, which are transformations of the input variables and then using linear models in this new space of derived input features. Methods like artificial neural networks (ANNs) (Bishop, 1995), projection pursuit learning (Friedman and Stuetzle, 1981) and generalized additive models (Hastie and Tibshirani, 1990), can be seen as different basis function models.

Approaches based on ANNs have been applied in agricultural remote sensing (Goel et al., 2003; Karimi et al., 2005; Shanin et al., 2002; Uno et al., 2005; Yang et al., 2000a,b) to associate complicated spectral information with target attributes without any constraints for sample distribution (Mather, 2000). According to Kimes et al. (1998) and Lillesand and Kiefer (2000), ANNs described the intricate and complex non-linear relationships which exists between canopy-level spectral imagery and several crop conditions. Although, they have mainly been applied in classification problems, this technique has shown a great potential for continuous variable prediction problems, such as soil moisture estimation (Chang and Islam, 2000), water quality assessment (Zhang et al., 2002), biomass estimation (Jin and Liu, 1997) and yield prediction (Drummond et al., 2003; Liu et al., 2001). However, sigmoidal feed-forward neural networks or multilayer perceptrons (MLPs), which are the ANNs most used, have serious difficulties in reflecting accurately the strong interactions between input variables.

Product unit neural network (PUNN) models are an alternative to MLPs and are based on multiplicative nodes instead of additive ones. They correspond to a special class of feed-forward neural network introduced by Durbin and Rumelhart (1989). They aim to overcome the non-linear effects of variables by means of non-linear basis functions, constructed with the product of the inputs raised to arbitrary powers. These basis functions express the possible strong interactions between the variables, where the exponents may even take on real values and are suitable for automatic adjustment.

One of the main problems involved in the application of MLP and PUNN models is the selection of the most appropriate net architecture to be used. Classical neural network training algorithms assume a fixed architecture; however it is very difficult to establish beforehand the best structure of the network for a given problem. There have been many attempts to design the architecture automatically (Reed, 1993; Setiono and Hui, 1995). Evolutionary algorithms (EAs), which are stochastic search algorithms that execute a global search in the input space preventing the fall to local optimum (Angeline et al., 1994; García-Pedrajas et al., 2003; Yao, 1999), have demonstrated great accuracy in designing a near optimal architecture. This fact, together with the complexity of the error surface associated with a PUNN, justifies the use of an EA to design the structure and adjust the weights of these models (Martínez-Estudillo et al., 2006a,b).

Many researchers (Houck et al., 1996, 1997; Michalewicz, 1994) have shown that EAs perform well for global searching because they are capable of quickly finding and exploring promising regions in the search space, but they take a relatively long time to converge to a local optimum. Recently, new methods have been developed in order to improve the precision of the EAs by adding local optimization algorithms. These methods are commonly known as hybrid algorithms (Bersini and Renders, 1994). Martínez-Estudillo et al. (2006b) proposed the hybrid combination of three methods for the design of Evolutionary PUNNs (EPUNNs) for regression: an EA, a clustering process and a local search procedure. Clustering methods create groups (clusters) of mutually close points that could correspond to relevant regions of attraction. Then, local search procedures can be started once in every such region, e.g., from its centroid. They reported that the application of a clustering algorithm for selecting individuals representing the different regions in the search space was more efficient than using the optimization algorithm for every individual in the population. Their experiments in microbial growth showed that the proposed hybrid learning approach is able to obtain good results in hard real-world problems. On the other hand, this methodology provides us with a pool of different solutions, each of them with distinct characteristics. The information contributed by all models could be combined to obtain a more effective solution of the problem.

Ensembles are another promising research field, where several models are combined to produce an answer (Dietterich, 2000). Often, the combination of individual learners outperforms the best individual among them. There are different approaches for building ANN ensembles. One of them is based on considering heterogeneous topologies, where a family of ANNs with distinct structures (and therefore complexities) are combined (Rocha et al., 2004). Since EAs design the structure of the neural nets, the models obtained at different moments in the evolution are suitable for being combined, because each of them has distinct topological characteristics.

The main objective of this work was to assess the potential of EPUNNs trained with different hybrid algorithms for reaching the best approach to the prediction of sunflower yield at field-scale using digital elevation model (DEM), *R. segetum* weed map and remote sensed information. Seven models were applied: one multivariate statistical approach (SMLR); four EPUNN models such as evolutionary programming (EP), hybrid EP (HEP), HEP with clustering (HEPC) and dynamic HEPC (DHEPC); additionally two proposals are presented for the combination of the obtained models with DHEPC, including mean DHEPC ensembles (E_{mean}) and median DHEPC ensembles (E_{median}).

2. Materials and methods

2.1. *Materials and experimental design*

Different types of information extracted from previous research at the study area reaching the best approach were analyzed, including aerial photography, sunflower yield data, elevation data and weed patches. The data analyzed correspond to a study conducted in 2003 at the 42 ha farm Matabueyes, located in Andalusia, Southern Spain (37.8°N, 4.8°W, WGS84), naturally infested by *R. segetum*. With a clay content of nearly 60%, the soil of Matabueyes was classified as Typic Chromoxerert (USDA-NRCS, 1998) and it is representative of the extensive dry-land of Southern Spain. Sunflower crop cultivar Jalisco was seeded at 4 kg ha^{-1} in rows 0.7 m apart in mid-March and then harvested in mid-August. Tillage production methods were applied to manage the field site. In order to control annual weed seedlings in sunflower, glyphosate (Roundup, isopropylamine salt, 360 g a.i. l^{-1}, Monsanto) was applied at pre-emergence 180 g a.i. l^{-1}, but at this rate this herbicide had no significant activity on *R. segetum* plants.

Conventional-colour (CC) and colour-infrared (CIR) aerial photographs of the studied field were taken at 15 May 2003. The photographs were taken by a turboprop twin-engine plane CESSNA 402, using an automatic pilot for managing both photographic equipment and GPS. The camera was a RMK TOP 15, with a Zeiss objective, a focal distance of 153.76 mm and Kodak Aerocolor III 2444 and Kodak Aerochrome S0734 film for CC and CIR photographs, respectively. All these photographs were taken on cloudless days between 12 and 14 h standard time and the average flight height was 1525 m to obtain photographs at a scale 1:10,000. Then, the photographs were digitalized with an AGFA Horizon A3 scanner, considering a resolution of 635 dots per inch (dpi). It is important to note that brightness and contrast were not adjusted on the digitalized images. The next step was to orthorectify the digitised images, using the fiducial marks of the aerial calibration certificate, 40 ground control points taken with a differential GPS TRIMBLE PRO-XRS equipped with a TDC-1 unit (centimetre precision) and a 10 m resolution raster DEM. Finally, images were resampled to a pixel size representing 40 cm × 40 cm ground area.

Blue (B, 400–500 nm), green (G, 500–600 nm) and red (R, 600–700 nm) bands of the electromagnetic spectrum are represented by CC photographs and green, red and near-infrared (NIR, 700–900 nm) are represented by CIR photographs. The scanner produced a RGB digital image with eight-bit true colour, so pixels of the image showed digital counts within the range of 0–255 values. These digital values are considered as being directly proportional to the total light reflected from the scene (Flowers et al., 2001).

Sunflower crop was combine-harvested in August 2003 using a Massey Fergusson® harvester equipped with a calibrated Fieldstar® yield monitor and a differential global positioning system (DGPS) receiver (Blackmore and Moore, 1999). Yield data varied from 0.50 to 2.30 tonnes ha^{-1}. The DPGS also described the elevation data (z coordinate).

R. segetum patches were mapped as described in Peña-Barragán et al. (2007), applying the Specter angle mapper (SAM) supervised classification method to the multispectral imagery, at 94% overall accuracy. SAM method considers the angle between consecutive *n*-band values as an *n*-dimensional vector, which is representative of the spectral signature of each cover-class. Smaller angles between training reference spectra and pixel to be classified represent closer coincidences. A maximum angle threshold (MAT) parameter must be specified in radians, so any pixels further away than the MAT are not classified (Kruse et al., 1993).

All spatial data from images and maps were grouping and saved to an unique multiband file, taking into account two requirements: (a) the georeference error between images was less than one pixel, so similar pixels had the same coordinate and (b) the NIR digital values of CIR photograph were corrected to digital values of CC photograph, considering the differences between the G and R bands of both original photographs. Further information about the acquisition of the photographs is described in Peña-Barragán et al. (2007).

The dataset used to train and validate the different prediction methods included all the information extracted from the different pixels of the images: red, green, blue and near-infrared digital values, elevation data, *R. segetum* infestation and yield data. Due to the very large number of pixels, the dataset was reduced by obtaining mean values for each 20 × 20 pixels of the image, resulting in a total of 1056 instances. Therefore, weed density was calculated in every grid assuming the presence of two plants per every infested pixel, which provides a conservative estimate.

The experimental design was conducted using a hold-out cross-validation procedure, where the size of training set was approximately $3n/4$ and $n/4$ for the generalization set, n being the size of the full dataset. Consequently, the above-mentioned dataset was randomly split in two datasets. A 739 instances dataset was used for model training and the remaining 317 instances formed the generalization dataset.

2.2. *Methods*

2.2.1. *Statistical methods: stepwise multiple linear regression*

SPSS 13.0 software for Windows (SPSS, 2005) was used for the SMLR model development. Regression equations were calculated for all input variables of the dataset, using a backward SMLR with the following stepwise criteria: $P \leq 0.05$ for entry and $P > 0.40$ for removal.

2.2.2. *Evolutionary product unit neural networks*

2.2.2.1. Product unit neural networks. PUNNs are an alternative to MLPs, and are based on multiplicative nodes instead of additives ones. A multiplicative node is given by $\prod_{i=1}^{k} x_i^{w_{ji}}$, where k is the number of the inputs. When the exponents w_{ji} are $\{0,1\}$ a higher-order unit is obtained, namely the sigma-pi unit. The output of a polynomial or sigma-pi unit is a function of the linear sum of some monomial. In contrast to sigma-pi unit, in the product unit the exponents are not fixed and may even take real values.

Product unit based neural networks have several advantages, including increased information capacity and the ability to express strong interactions between input variables. Furthermore, it is possible to obtain upper bounds of the Vapnik–Chervonenkis (VC) dimension (Vapnik, 1999) of product unit neural networks similar to those obtained for MLP (Schmitt, 2002).

Despite these advantages, PUNNs have a major handicap: they have more local minima and a higher probability of becoming trapped in them (Ismail and Engelbrecht, 2000). The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface and therefore their training is more difficult than the training of standard MLPs. Several efforts have been made to carry out learning methods for product units (Ismail and Engelbrecht, 1999, 2000; Janson and Frenzel, 1993). The back propagation algorithm, which is the most common algorithm for training multilayer neural networks, does not work very well with the product units because of its complex error surface.

The structure of the neural network considered is described in Fig. 1: an input layer with k nodes, a node for every input variable, a hidden layer with m nodes and an output layer with one node. There are no connections between the nodes of a layer and none between the input and output layers either. The activation function of the jth node in the hidden layer is given by $\prod_{i=1}^{k} x_i^{w_{ji}}$, where w_{ji} is the weight of the connection between input node i and hidden node j and $\mathbf{w}_j = (w_{j1}, \ldots, w_{jk})$ is the weight vector. The activation function of the output node is given by:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{m} \beta_j \prod\nolimits_j (\mathbf{x}, \mathbf{w}_j) \quad (1)$$

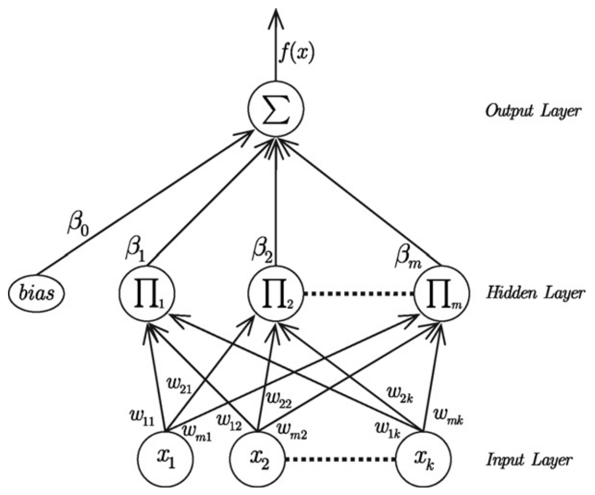


Fig. 1 – Model of a product unit based neural network.

where β_j is the weight of the connection between the hidden node j and the output node. The transfer function of all hidden and output nodes is the identity function.

2.2.2.2. Evolutionary algorithm. In this section we present the EA used to estimate the parameters and the structure of the PUNNs that minimizes the prediction error function. The algorithm, similar to that proposed by Martínez-Estudillo et al. (2006a), begins with the random generation of N_P individuals. Then the evolution process start and a population-update algorithm is applied. Since the algorithm falls into the class of EP paradigm (Fogel, 1995), population is subjected to the replication and mutation operations, but crossover is not considered, as this operation is usually regarded as being less effective for ANNs evolution (Angeline et al., 1994). Although there are different training methodologies for this purpose, most researchers agree that EP is the most suitable evolutionary computation paradigm for evolving neural nets (Koehn, 1994).

The general structure of the EA is detailed next:

Evolutionary programming (EP):

```
(1) Generate a random population of size N_P.
(2) Repeat until the stopping criterion is fulfilled.
    (2a) Apply parametric mutation to the best 10% of individuals. Apply structural mutation to the remaining 90% of individuals.
    (2b) Calculate the fitness of every individual in the population.
    (2c) Add best fitness individual of the last generation (elitist algorithm).
    (2d) Rank the individuals with respect to their fitness.
    (2e) Best 10% of population individuals are replicated and substitute the worst 10% of individuals.
(3) Select the best individual of the population in the last generation and return it as the final solution.
```

First, the initial population is generated: the algorithm begins with the random generation of a larger number of networks than the number used during the evolutionary process. Ten N_P networks are generated, where N_P is the number of individuals of the population to be trained during the evolutionary process. Fitness of a neural network of the population that implements a function $f(x)$, is calculated using a $D = \{(\mathbf{x}_l, y_l); l = 1, 2, \ldots, n_T\}$ training dataset, where the number of samples is n_T. Mean squared error (MSE) of $f(x)$ is considered:

$$\mathrm{MSE}(f) = \frac{1}{n_T} \sum_{l=1}^{n_T} (y_l - f(x_l))^2 \tag{2}$$

where y_l are the observed values and $f(x_l)$ are the predicted values. The fitness function $A(f)$ is defined by means of a strictly decreasing transformation of the MSE:

$$A(f) = \frac{1}{1 + \mathrm{MSE}(f)}, \quad 0 < A(f) \le 1 \tag{3}$$

The adjustment of both weights and structure of the PUNNs is performed by the complementary action of two mutation operators: parametric and structural mutation. Parametric mutation implies a modification in the coefficients (β_j) and the exponents (w_{ji}) of the model, using a self-adaptive simulated annealing algorithm (Kirkpatrick et al., 1983). Structural mutation modifies the topology of the neural nets, helping the algorithm to avoid local minima and increasing the diversity of the trained individuals. Five structural mutations are applied sequentially to each network: node deletion, connection deletion, node addition, connection addition and node fusion. When node deletion is applied, number of hidden nodes to be removed is obtained as a uniform value in a previous specified interval. Apart from this mutation, if connection deletion is applied, the number of connections to be deleted in the neural net is also obtained as a uniform value, but in this case, as the mutation is less disruptive, the selected interval uses to be a wider one. More details about the EA can be found in Martínez-Estudillo et al. (2006a).

2.2.2.3. *Hybrid evolutionary programming algorithms*. In this work, different variants of hybrid EAs have been applied, all of them proposed by Martínez-Estudillo et al. (2006b). The EP algorithm is the EA exposed in the previous section without neither a local search nor a clustering process. In the hybrid EP (HEP), the EP is run without the local optimization algorithm and then it is applied to the best solution obtained by the EP in the final generation. This allows the precise local optimum around the final solution to be found. Another version of hybrid EA is the HEP with the clustering algorithm (HEPC), which applies the clustering process over a large enough subset of the best individuals in the final population. The number of individuals in this subset and number of clusters to be created are critical parameters of the clustering process. Once clusters have been determined, the best individual in each cluster is selected and then optimized using the local search algorithm. Finally, dynamic HEP with clustering (DHEPC) is the most sophisticated algorithm. It carries out both the clustering process and local search dynamically during the evolution every G_0, where G_0 must be fixed a priori. All optimized individuals are extracted and stored and the final solution is the best local optimum among them.

The main objective of these methodologies is to reduce the number of times it is necessary to apply the local optimization procedure, since local search algorithms commonly involve a high computational cost. The clustering process selects the most representative groups of the population, providing a subset of individuals with different features. The selected clustering method selected is k-means clustering, using a distance measure defined for the vectors of the different values obtained for each individual over the training dataset. Further information can be found in Martínez-Estudillo et al. (2006b).

The local optimization procedure considered is the *Levenberg–Marquardt* (L–M) *algorithm* (Marquardt, 1963), a gradient-based methodology, designed specifically for minimizing a MSE (Bishop, 1995). The LM method can be trapped in a local optimum. Moreover, PUNNs have a very complex error surface with many minimum optima and local plateau. That is the reason why it is necessary to apply the *L–M algorithm* at different points of the evolution, so global optimum can be discovered with a higher probability.

The hybrid algorithms applied are summarized as follows:

Hybrid evolutionary programming (HEP):

(1) Generate a random population of size N_P.
(2) Repeat EP algorithm until the stopping criterion is fulfilled.
(3) Apply *L–M algorithm* to best solution obtained in the EP algorithm.
(4) Return the optimized individual as the final solution.

Hybrid evolutionary programming with clustering (HEPC):

(1) Generate a random population of size N_P.
(2) Repeat EP algorithm until the stopping criterion is fulfilled.
(3) Apply k-means process to best N_C individuals of the population in the last generation and assign a cluster to each individual.
(4) Apply *L–M algorithm* to best solution obtained in each cluster.
(5) Select the best individual among optimized ones and return it as the final solution.

Dynamic hybrid evolutionary programming with clustering (DHEPC):

(1) Generate a random population of size N_P.
(2) Repeat EP algorithm until the stopping criterion is fulfilled, applying the following process every G_0 generations:
    (2a) Apply k-means process to best N_C individuals of the population in current generation, assigning a cluster to each individual.
    (2b) Apply *L–M algorithm* to best solution obtained in each cluster.
    (2c) Select the best individual among optimized ones and add it to B set.
(3) Select best individual in B set and return it as the final solution.

2.2.2.4. *Ensemble approaches*. All above-mentioned methodologies share the risk of over fitting training dataset. A local search procedure leads us to the closer local optimum from the current location in the training set, but sometimes the optimized model loses its generalization capability. In this paper, ensembles are presented as a way of avoiding this overfitting, by the combination of all different EPUNN models obtained through the evolution process. Considering the models obtained in different generations, a pool of individuals with different structures can be constructed and applied to the data as an ensemble. Models are extracted from the DHEPC algorithm at different moments in the evolution, so the set of individuals obtained will presumably gather neural nets with the most efficient and varied topologies for solving the problem.

For combining outputs of all regression models of the ensemble, two different proposals are considered: mean DHEPC ensembles (E_{mean}) and median DHEPC ensembles (E_{median}). Let $B = \{f_k\}$, for $k = 1, \ldots, s$, be the set of optimized solutions obtained throughout the evolution process in a run of the DHEPC algorithm. The output of E_{mean} ensemble is calculated as the mean value of all the outputs of the elements of B and E_{median} output is obtained as the median of the outputs of the elements of B:

$$E_{\text{mean}}(B, x_l) = \overline{f(x_l)}, \qquad E_{\text{median}}(B, x_l) = f_M(x_l) \tag{4}$$

where $\overline{f(x_l)}$ and $f_M(x_l)$ represents the mean an median values of $f_k(x_l)$ values, which are obtained using Eq. (1).

2.2.3. *Model development and evaluation*

To start processing data, each of the input variables was scaled in the ranks [0.1,1.1]. The lower bound is chosen to avoid inputs values near to 0 that can produce very large values of the function for negative exponents. The upper bound is chosen near 1 to avoid dramatic changes in the outputs of the network when there are weights with large values (especially in the exponents). The new scaled variables were named Weed*, Z^*, R^*, G^*, B^* and NIR*. For example, Z^* is calculated as follows:

$$Z^* = \frac{Z - Z_{\text{min}}}{Z_{\text{max}} - Z_{\text{min}}} + 0.1 \tag{5}$$

where Z is the original elevation. Z_{min} and Z_{max} are the minimum and maximum values in the whole dataset.

As was mentioned before, SPSS 13.0 software for Windows (SPSS, 2005) was used for the SMLR model development. The different EPUNN experiments were conducted using a software package developed in JAVA by the authors, as an extension of JCLEC framework (http://jclec.sourceforge.net/) (Ventura et al., 2007). The software package is available in the non-commercial JAVA tool named KEEL (http://www.keel.es). The parameters used in the evolutionary algorithm for learning the EPUNN models are common for all methodologies: the $\mathbf{w}_j$ vector and the coefficients β_j are initialized in the [−5,5] interval; the maximum number of hidden nodes is $m = 6$; the size of the population is $N_P = 1000$. The number of nodes that can be added or removed in a structural mutation is within the [1,3] interval, whereas the number of connections that can be added or removed in a structural mutation is within the [1,7] interval. The only parameter of the L–M algorithm is the tolerance of the error to stop the algorithm; in our experiment, this parameter has the value 0.01. The k-means algorithm is applied to $N_C = 250$ best individuals in the population. The number of clusters is 4 for both the HEPC and DHEPC algorithms. For this latter algorithm, $G_0 = 600$. Finally, it is important to note that, as an evolutionary method has been applied for optimizing weights of the neural nets, a learning rate parameter is not considered and weight optimization is achieved through the double effect of parametric mutations and *Levenberg–Marquardt* local search algorithm.

Models obtained with the different evolutionary and statistical methodologies are evaluated using their prediction error over the training and generalization sets. Two error measurements are considered: the root mean squared error (RMSE, $\text{RMSE}(f) = \sqrt{\text{MSE}(f)}$, see Eq. (2)) and the standard error of prediction (SEP). Let $D = \{(x_l, y_l);\ l = 1, \ldots, n_T\}$ be the dataset. The SEP of a model that implements a function $f(x)$ represents a percentage error over the dataset and can be expressed as:

$$\text{SEP}(f) = \frac{100}{|\bar{y}|}\sqrt{\text{MSE}(f)} \tag{6}$$

where $\bar{y}$ is the average output of all patterns in dataset.

3. Results and discussion

Results obtained with the different modelling approaches were evaluated by using both RMSE and SEP, and a regression between observed and simulated yields was performed to verify that compared models did a comparable job. Table 1 shows the statistical results of the four EPUNN algorithms (EP, HEP, HEPC and DHEPC) and the combination of the obtained models with DHEPC (E_{mean} and E_{median}) over 30 runs. Based on these results, it is can be concluded that DHEPC, E_{mean} and E_{median} clearly outperform the remainder methodologies, so we used the analysis of variance (ANOVA) technique to ascertain the statistical significance of observed differences between the three corresponding means, assuming that the RMSE$_G$ values obtained have a normal distribution; a Kolmogorov–Smirnov test for normality was reached for DHEPC, E_{mean} and E_{median} with P-values of 0.650, 0.965 and 0.999, respectively. The ANOVA involves a linear regression model in which, RMSE$_G$ is the dependent variable and the independent variable is the type of algorithm. The comparisons were made in terms of a critical level for Snedecor's F. If the significance level, α, was higher than this critical level, P, we rejected the hypothesis of identical RMSE$_G$'s means. In our case this hypothesis is accepted because the P-value is 0.178, higher than a standard $\alpha = 0.05$. Finally, we used an independent t-test for analyzing the significant difference between the means of DHEPC and E_{mean} algorithms. It can be seen, for $\alpha = 0.1$, that differences in variance existed from a prior Levene's test (P-value = 0.061), and that there were differences in mean (P-value = 0.08). Based on these results, the E_{mean} methodology should be adopted for predicting sunflower yield production, since it is significantly more accurate (lower values) than DHEPC, both in terms of mean and standard deviation. Comparing E_{mean} and E_{median} algorithms, there are no significant differences neither in terms of mean nor standard deviation, but the best result (RMSE$_G$ = 0.2119 tonnes ha^{-1}) is obtained by E_{mean} methodology.

A performance comparison between proposed EPUNN models and a SMLR model measured by using the mean RMSE

Table 1 – Statistical results over 30 runs of the different evolutionary methodologies

Algorithms	RMSE (tonnes ha^{-1})							
	Learning				Generalization			
	Mean	S.D.	Best	Worst	Mean	S.D.	Best	Worst
EP	0.2356	0.0469	0.2274	0.2427	0.228	0.0566	0.2177	0.2406
HEP	0.2254	0.0374	0.2214	0.2349	0.22	0.04	0.2138	0.23
HEPC	0.2249	0.0374	**0.2209**[a]	0.2349	0.22	0.04	0.2131	0.2298
DHEPC	**0.2238**	**0.03**	**0.2209**	**0.2280**	0.2191	0.0346	0.2152	0.2241
E_{mean}	0.2243	0.0316	0.2211	0.2302	**0.2179**	**0.03**	**0.2119**	**0.2214**
E_{median}	0.2245	0.0332	0.2211	0.2315	0.2184	**0.03**	0.2131	0.222

RMSE, Root mean squared error; S.D., standard deviation; EP, evolutionary programming; HEP, hybrid evolutionary programming; HEPC, hybrid evolutionary programming with clustering; DHEPC, dynamic hybrid evolutionary programming; E_{mean}, mean DHEPC ensembles; E_{median}, median DHEPC ensembles.

[a] The values given in bold represent 'best RMSE'.

value and SEP in the generalization and learning dataset is shown in Table 2. The best single EPUNN obtained in all runs of the different algorithms is considered, corresponding to DHEPC methodology. In both cases, the models developed produced relatively low values for RMSE (<0.2594 tonnes ha^{-1}), which means that relatively low sunflower yield prediction error was achieved. However, the lowest mean RMSE and SEP for the learning and generalization dataset were obtained with the EPUNN model, SEP being around 16%. The highest RMSE was achieved for SMLR (SEP about 19% for learning and generalization). These results support the findings of other authors, which obtained similar SEP values as reported by Uno et al. (2005). They compared SMLR and ANN approaches along with various vegetation indices to develop corn yield prediction methods. They did not find clear differences between ANNs and SMLR (about 20% validation MSE in both models), although greater prediction accuracy was obtained with ANNs than with empirically derived vegetation indices. Since RMSE values depend on the magnitude of the data, SEP values provided better comparisons between different models. In our case, difference between EPUNN and SMLR models in sunflower yield prediction was about 0.0403 tonnes ha^{-1} (3.02%) in generalization RMSE (Table 2).

Regression parameters for observed and simulated sunflower yields for the worst and best models are shown in Fig. 2. The EPUNN model, with an approximately two times higher R^2 value, showed a higher goodness of fit with a closer cluster along the best-fit line than SMLR. However, both models presented high deviations between observed and predicted values, these differences being more noticeable for lowest yield values. The EPUNN best-fit regression equation slope was closer to 1 than the SMLR slope and the intercept closer to 0 than that of SMLR. The SMLR model performed the worst probably due to the higher values of sunflower yield being overestimated, i.e. they were higher than the best-fit line as can be seen in Fig. 2a. Nevertheless, higher predicted yields were at a similar distance from the regression line of EPUNN model (Fig. 2b).

Regression equations of the best and worst models for yield prediction are presented in Table 3. SMLR model identified the elevation (Z) to be the top important factor, meaning that sunflower yield was lower at higher elevation. This can be observed in Fig. 3, where a yield map with the predictions of the EPUNN model and an *R. segetum* weed map are presented. The yield reduction at higher elevation was probably due to the limited water supply in these regions, as has previously been reported for corn and soybean yield (Kravchenko and Bullock, 2002a,b; Miao et al., 2006). Weed infestation was selected as being the second most important factor to predict sunflower yield. Classic *R. segetum*–sunflower competition models for predicting sunflower yield without considering any spatial or spectral components assumed a non-linear hyperbolic relationship between mean *R. segetum* densities, obtaining a much higher RMSE (about 1.1 tonnes ha^{-1}) (Carranza et al., 1995) than RMSE of methods herein compared. Fig. 3 clearly shows that lower sunflower yield is located where *R. segetum* patches appear. For the SMLR model, regression equations shown in Table 3 indicate that one of the factors affecting sunflower

Table 2 – Comparative performance of the best EPUNN model and the SMLR model

Models	Model performance			
	Learning		Generalization	
	RMSE (tonnes ha^{-1})	SEP (%)	RMSE (tonnes ha^{-1})	SEP (%)
SMLR	0.2594	19.37	0.2534	19.07
EPUNN	**0.2256**[a]	**16.85**	**0.2131**	**16.05**

Regression equations of the best EPUNN model and the SMLR model. RMSE, Root mean squared error; SEP, standard error of prediction; SMLR, stepwise multiple linear regression; EPUNN, evolutionary product unit neural network.

[a] The values given in bold represent 'best performance'.

Table 3 – Regression equations of the best EPUNN model and the SMLR model

Models	Regression equation
SMLR	Yield (tonnes ha^{-1}) = 3.7031 − 0.0097Weed − 0.0133Z − 0.0033R + 0.0003NIR
EPUNN	Yield (tonnes ha^{-1}) = 1.6973 + 0.7396h_1 − 0.0709h_2 − 13.1973h_3 + 33.8768h_4 − 22.3513h_5 $h_1 = (G^*)^{6.1364}(Z^*)^{-1.1831}(\text{Weed}^*)^{0.4289}$ $h_2 = (NIR^*)^{-0.7263}(R^*)^{6.2797}(B^*)^{-2.3923}(Z^*)^{-2.1374}(\text{Weed}^*)^{0.5795}$ $h_3 = (NIR^*)^{-0.0902}(R^*)^{2.8042}(G^*)^{-0.3956}(B^*)^{1.6049}(Z^*)^{0.9406}$ $h_4 = (R^*)^{1.5069}(B^*)^{2.9757}(Z^*)^{0.5926}(\text{Weed}^*)^{0.0552}$ $h_5 = (NIR^*)^{0.0758}(G^*)^{0.9941}(B^*)^{3.7274}(Z^*)^{0.3144}(\text{Weed}^*)^{0.1106}$

SMLR, Stepwise multiple linear regression; EPUNN, evolutionary product unit neural network; Weed, *R. segetum* density (number of plants m^{-2}); Z, elevation (m); R, G, B, NIR, scaled digital values of red (R), green (G), blue (B) and near infrared (NIR) bands. Scaled variables Weed*, Z^*, R^*, G^*, B^* and NIR* ∈ [0.1,1.1].

yield is weed infestation but it is not the only one or the most influential. It is widely accepted that vigorous and healthy crops usually show high reflectance in near infrared and low in red (Hatfield and Pinter, 1993); that is represented in the SMLR equation, where *R* and NIR variables also affect to the prediction, although they have a more reduced influence.

The EPUNN model produced a regression equation with higher level of complexity and provided much more information about the relationship between spectral factors and crop yield than the SMLR model. In general, a neural network produces equations that are difficult to understand. The structure of the interconnected neurons depends on the complexity of a given problem and the number of neurons in the hidden layers is related to the performance of a neural network. Too few hidden neurons limit the ability of the network to model the problem, and too many result in overtraining of the input/output pair patterns presented in the training process. In our case, EPUNN formulas are not particularly complex, since they have neither a very high number of neurons nor too many variables associated in each neuron. As can be observed in Table 3, the EPUNN model has six elements, the first one (1.6973 tonnes ha^{-1}) corresponding to the bias value and the other corresponding to the hidden neurons (h_i, $1 \leq i \leq 5$). The influence of the input parameters on the output depends not only on the value of their exponents but also on the coefficient corresponding to the hidden neurons in which they

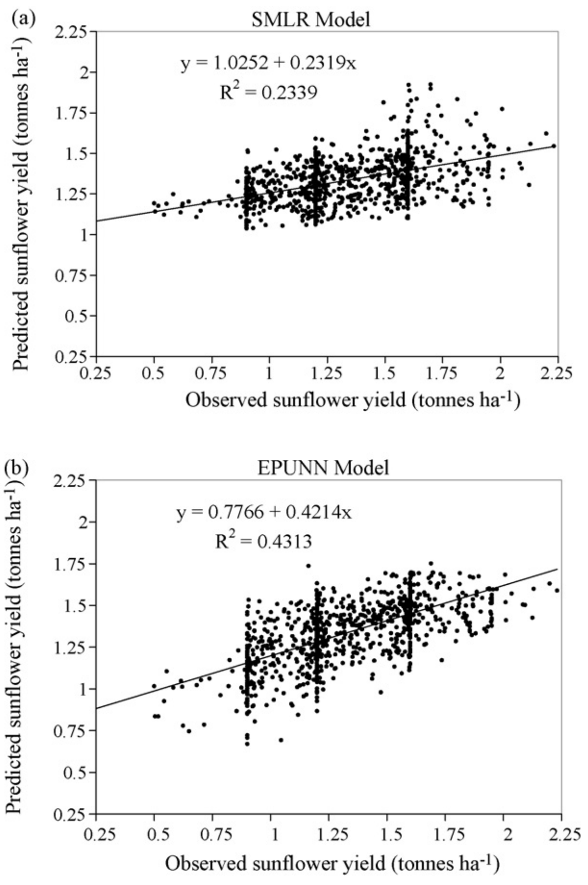


Fig. 2 – Scatter-plot of observed and computed sunflower yield using the developed models on complete set of data: (a) SMLR model and (b) EPUNN model.

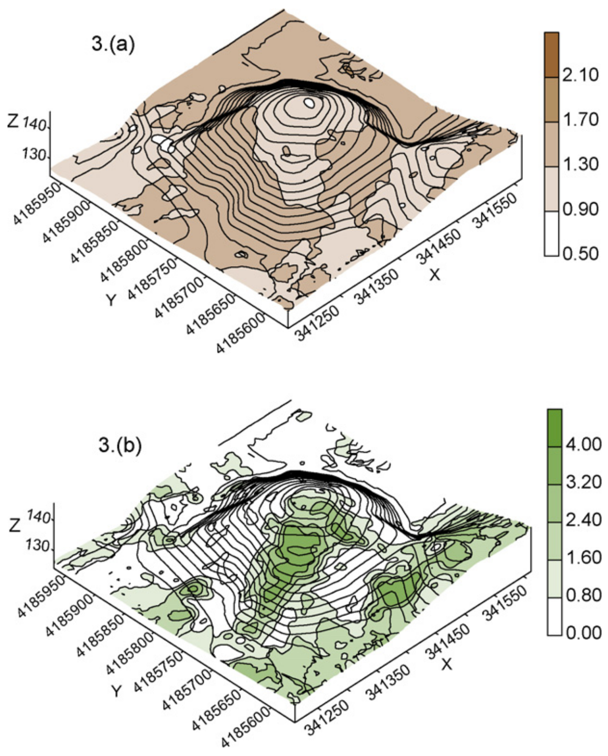


Fig. 3 – (a) Sunflower yield map predicted with EPUNN model (tonnes ha^{-1}) and (b) *R. segetum* weed map (plants m^{-2}).

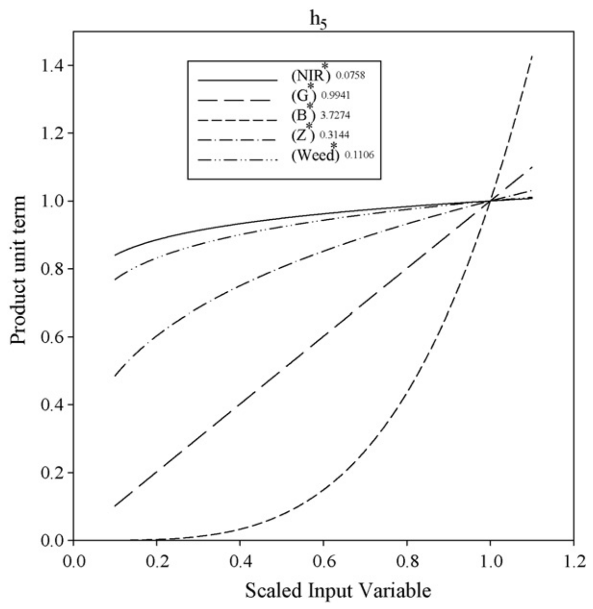


Fig. 4 – Relative contribution of h_5 hidden neuron terms used by the EPUNN model, h_5 bearing an inverse linear relationship with sunflower crop yield.

are represented. Consequently, in order to derive agronomical information by using the EPUNN model, the response of each PU hidden neuron must be independently studied. As an example, h_5 product unit terms affected by its exponent were plotted versus the scaled input variables (Weed*, Z^*, G^*, B^* and NIR*) over the range [0.1,1.1] (see Fig. 4). From the EPUNN equation shown in Table 3, it follows that the sunflower crop yield depends mainly on the interaction between the variables R^*, G^*, B^* and Z^*, the contribution of Weed* and NIR* being practically negligible, as inferred from the low exponential values of these variables in all hidden neurons. As can be observed in Fig. 4, h_5 value, which bear an inverse relationship with sunflower crop yield, is larger when Z^*, B^* and G^* increases and exponents of NIR* and Weed* variables make their contribution in the hidden neuron output non-significant, as their weighted values are near to 1. From the equation shown in Table 3, it follows that R^* variable only appears in the hidden neurons with negative coefficient, and its exponents are always greater than 0. Therefore, crop yield is inversely related to R^* value. In summary, although NIR* and weed infestation (Weed*) were also selected in EPUNN model, the wavebands of visible spectral window (B^*, G^* and R^*) and elevation (Z^*) were identified as the most influent parameters. The parameters R^*, G^*, B^* and Z^* are the key for the determination of sunflower crop yield when using EPUNN model, being yield larger when Z^*, B^*, R^* and G^* decreases. Results from other authors agree with these relationships between sunflower crop and visible domain (Peña-Barragán et al., 2006) and Z values (Ruiz et al., 2007).

Over the years, the creation of accurate maps using harvested-mounted crop yield-monitoring systems has been essential for successful implementation of precision farming (Blackmore and Moore, 1999; Reitz and Kutzbach, 1996). Our results demonstrate that models integrating spectral data, *R. segetum* spatial infestation and elevation, fit good models for predicting sunflower yield over a large area. It can be concluded that the SMLR model is capable of achieving a higher overall performance than previous classic studies, but does not achieve such high prediction accuracy as EPUNN. It is important to consider that DHEPC models are obtained from EP models, the additional computational requirements being nearly insignificant regarding the achieved improvement in accuracy performance. Once EPUNNs has been shown to successfully predict sunflower yield maps by using primarily spectral data from airborne multispectral imagery, the next investigation could address the examination of QuickBird satellite imagery for mapping sunflower yield in larger areas (of at least over 64 km^2), since QuickBird provides four channels (*B*, *G*, *R* and NIR) of multispectral imagery with 2.4 m of spatial resolution and has shown to be a useful data source for determining sorghum yield (Yang et al., 2006).

4. Conclusions

This study demonstrated the capability of EPUNN to analyze multispectral imagery, weed and elevation data for predicting sunflower yield in early growth stage. EPUNN provided better accuracy than linear SMLR models both in training set (16.85%) and generalization set (16.05%). Moreover, four different EPUNN algorithms (EP, HEP, HEPC and DHEPC) were evaluated in training PUNN for the sunflower yield prediction problem and two methods for combining resulting DHEPC models were proposed and compared to the remaining EPUNN algorithms. The statistical results of the multiple comparison tests carried out show that E_{mean} and E_{median} yielded better results than the other algorithms, with lower RMSE mean and standard deviation.

From an agronomic point of view, our study demonstrated that sunflower prediction yield in infested *R. segetum* fields is affected by weed infestation, but also implies complex relationship among parameters such as elevation or digital (or spectral) data. Granted that, computational requirements for EP were much higher than for SMLR, those necessary for DHEPC were nearly insignificant. Thus, taking into account that precision agriculture management requires a great accuracy, the criteria for selection SMLR or EPUNNs models should not be based on decreasing computational requirements and complexity, but on the accuracy of prediction. Although the results of this study are promising in considering the multivariate agronomic context, more research is needed to work on a larger field surface using high spatial resolution satellite imagery.

Acknowledgements

This work has partially been financed by TIN2005-08386-C05-02 and AGL-2005-06180-CO3-02, projects of the Spanish Inter-Ministerial Commission of Science and Technology and FEDER funds. The research of Dr. M. Jurado-Expósito and P.A. Gutiérrez has been supported by "Ramón y Cajal" (MEC) and I3P-Predoctoral Programs, respectively.

REFERENCES

Angeline, P.J., Saunders, G.M., Pollack, J.B., 1994. An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. Neural Networks 5 (1), 54–65.

Bersini, H., Renders, B., 1994. Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. In: Proceedings of the IEEE International Symposium on Evolutionary Computation, Orlando, Florida, pp. 312–317.

Bishop, M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, UK.

Blackmore, S., Moore, M., 1999. Remedial correction of yield map data. Precis. Agric. 1 (1), 53–66.

Burks, T.F., Shearer, S.A., Green, J.D., Heath, J.R., 2002. Influence of weed maturity levels on species classification using machine vision. Weed Sci. 50 (6), 802–811.

Carranza, P., Saavedra, M., García-Torres, L., 1995. Competition between *Ridolfia segetum* and sunflower. Weed Res. 35 (5), 369–376.

Castro-Tendero, A.J., García-Torres, L., 1995. SEMAGI—an expert system for weed control decision. Crop Protect. 14 (7), 543–548.

Chang, D.H., Islam, S., 2000. Estimation of soil physical properties using remote sensing and artificial neural network. Remote Sens. Env. 74 (3), 534–544.

Cho, S.I., Lee, D.S., Jeong, J.Y., 2002. Weed-plant discrimination by machine vision and artificial neural networks. Biosyst. Eng. 83 (3), 275–280.

Dietterich, T.G., 2000. Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (Eds.), Proceedings of the First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, 1857. Springer Verlag, New York, pp. 1–15.

Drummond, S.T., Sudduth, K.A., Joshi, A., Birrell, S.J., Kitchen, N.R., 2003. Statistical and neural methods for site-specific yield prediction. Trans. ASAE 46 (1), 5–14.

Durbin, R., Rumelhart, D., 1989. Products units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Comput. 1 (1), 133–142.

Flowers, M., Weisz, R., Heiniger, R., 2001. Remote sensing of winter wheat tiller density for early nitrogen application decisions. Agron. J. 93 (4), 783–789.

Fogel, D.B., 1995. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, New York.

Friedman, J., Stuetzle, W., 1981. Projection pursuit regression. J. Am. Stat. Assoc. 76, 817–823.

García-Pedrajas, N., Hervás-Martínez, C., Muñoz-Pérez, J., 2003. Covnet: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Trans. Neural Networks 14 (3), 575–596.

Goel, P.K., Prasher, S.O., Patel, R.M., Landry, J.A., Bonnell, R.B., Viau, A.A., 2003. Classification of hyperspectral data by decision trees and artificial neural networks to identify weed stress and nitrogen status of corn. Comput. Electron. Agric. 39 (2), 67–93.

Hastie, T.J., Tibshirani, R.J., 1990. Generalized Additive Models. Chapman & Hall, London.

Hatfield, J.L., Pinter, P.J., 1993. Remote sensing for crop protection. Crop Protect. 12 (6), 403–413.

Houck, C.R., Joines, J.A., Kay, M.G., 1996. Comparison of genetic algorithms, random start, and two-opt switching for solving large location–allocation problems. Comput. Oper. Res. 23 (6), 587–596.

Houck, C.R., Joines, J.A., Kay, M.G., Wilson, J.R., 1997. Empirical investigation of the benefits of partial Lamarckianism. Evol. Comput. 5 (1), 31–60.

Ismail, A., Engelbrecht, A.P., 1999. Training product units in feedforward neural networks using particle swarm optimisation. In: Proceedings of the International Conference on Artificial Intelligence, Durban, South Africa.

Ismail, A., Engelbrecht, A.P., 2000. Global optimization algorithms for training product unit neural networks. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN'2000), vol. 1, Como, Italy, pp. 132–137.

Janson, D.J., Frenzel, J.F., 1993. Training product unit neural networks with genetic algorithms. IEEE Exp. 8 (5), 26–33.

Jin, Y.Q., Liu, C., 1997. Biomass retrieval from high-dimensional active/passive remote sensing data by using artificial neural networks. Int. J. Remote Sens. 18 (4), 971–979.

Jurado-Expósito, M., López-Granados, F., Peña-Barragán, J.M., García-Torres, L., 2005. Weed density prediction with secondary input of DEM information. In: Stafford, J. (Ed.), Proceedings of the Fifth European Conference on Precision Agriculture. Uppsala, Sweden, pp. 115–122.

Karimi, Y., Prasher, S.O., McNairn, H., Bonnell, R.B., Dutilleul, P., Goel, P.K., 2005. Classification accuracy of discriminant analysis, artificial neural networks, and decision trees for weed and nitrogen stress detection in corn. Trans. ASAE 48 (3), 1261–1268.

Kenkel, N.C., Derksen, D.A., Thomas, A.G., Walson, P.R., 2002. Review: multivariate analysis in weed science research. Weed Sci. 50 (3), 281–292.

Kimes, D.S., Nelson, R.F., Manry, M.T., Fung, A.K., 1998. Attributes of neural networks for extracting continuous vegetation variables from optical and radar measurements. Int. J. Remote Sens. 19 (14), 2639–2663.

Kirkpatrick, S., Gelatt, C.P., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.

Koehn, P., 1994. Combining genetic algorithms and neural networks: the encoding problem. Ph.D. Thesis. University of Erlangen and University of Tennessee, Knoxville.

Kravchenko, A.N., Bullock, D.G., 2002a. Spatial variability of soybean quality data as a function of field topography. I. Spatial data analysis. Crop Sci. 42 (3), 804–815.

Kravchenko, A.N., Bullock, D.G., 2002b. Spatial variability of soybean quality data as a function of field topography. II. A proposed technique for calculating the size of the area for differential soybean harvest. Crop Sci. 42 (3), 816–821.

Kruse, F.A., Lefkoff, A.B., Boardman, J.B., Heidebrecht, K.B., Shapiro, A.T., Barloon, P.J., Goetz, A.F.H., 1993. The spectral image processing system (SIPS)—interactive visualization and analysis of imaging spectrometer data. Remote Sens. Env. 44 (2), 145–163.

Lillesand, T.M., Kiefer, R.W., 2000. Remote Sensing and Image Interpretation. Wiley, New York.

Liu, J., Goering, C.E., Tian, L., 2001. A neural network for setting target corn yields. Trans. ASAE 44 (3), 705–713.

Liu, Z., Clay, S.A., Clay, D.A., 2002. Spatial variability of atrazine and alachlor efficacy and mineralization in an eastern South Dakota field. Weed Sci. 50 (5), 662–671.

Marquardt, D.W., 1963. An algorithm for least-squares estimation of non-linear parameters. J. Soc. Ind. Appl. Math. 11 (2), 431–441.

Martínez-Estudillo, A.C., Mártínez-Estudillo, F.J., Hervás-Martínez, C., García-Pedrajas, N., 2006a. Evolutionary product unit based neural networks for regression. Neural Networks 19 (4.), 477–486.

Martínez-Estudillo, A.C., Hervás-Martínez, C., Martínez-Estudillo, F.J., García-Pedrajas, N., 2006b. Hybridization of evolutionary algorithms and local search by means of a clustering method. IEEE Trans. Syst. Man Cybernet. 36 (3), 534–545.

Mather, P.M., 2000. Computer Processing of Remotely Sensed Images: An Introduction. Wiley, Chichester, UK.

Miao, Y., Mulla, D.J., Robert, P.C., 2006. Identifying important factors influencing corn yield and grain quality variability using artificial neural networks. Precis. Agric. 7 (2), 117–135.

Michalewicz, Z., 1994. Genetic Algorithms + Data Structures = Evolution Programs. Springer Verlag, New York.

Osborne, S.L., Schepers, J.S., Francis, D.D., Schlemmer, M.R., 2002. Use of spectral radiance to estimate in-season biomass and grain yield in nitrogen- and water-stressed corn. Crop Sci. 42 (1), 165–171.

Peña-Barragán, J.M., López-Granados, F., Jurado-Expósito, M., García-Torres, L., 2006. Spectral discrimination of *Ridolfia segetum* and sunflower as affected by phenological stage. Weed Res. 46 (1), 10–21.

Peña-Barragán, J.M., López-Granados, F., Jurado-Expósito, M., García-Torres, L., 2007. Mapping *Ridolfia segetum* Moris patches in sunflower (*Helianthus annuus* L.) crop using aerial photographs. Weed Res. 47 (2), 164–172.

Pilesjö, P., Thylén, L., Persson, A., 2005. Topographical data for delineation of agricultural management zones. In: Stafford, J. (Ed.), Proceedings of the Fifth European Conference on Precision Agriculture. Uppsala, Sweden, pp. 819–826.

Plant, R.E., 2001. Site-specific management: the application of information technology to crop production. Comput. Electron. Agric. 30 (1), 9–29.

Reed, R., 1993. Pruning algorithms: a survey. IEEE Trans. Neural Networks 4 (5), 740–747.

Reitz, P., Kutzbach, H.D., 1996. Investigations on a particular yield mapping system for combine harvesters. Comp. Electron. Agric. 14 (2/3), 137–150.

Rocha, M., Cortez, P., Neves, J., 2004. Ensembles of artificial neural networks with heterogeneous topologies. In: Proceedings of the Fourth Symposium on Engineering of Intelligent Systems, Madeira.

Ruiz, D., Escribano, C., Fernández-Quintanilla, C., 2007. Identifying associations among sterile oat (*Avena sterilis*) infestation level, landscape characteristics, and crop yields. Weed Sci. 54 (6), 1113–1121.

Schmitt, M., 2002. On the complexity of computing and learning with multiplicative neural networks. Neural Comput. 14 (2), 241–301.

Setiono, R., Hui, L.C.K., 1995. Use of quasinewton method in a feedforward neural-network construction algorithm. IEEE Trans. Neural Networks 6 (1), 273–277.

Shanin, M.A., Tollner, E.W., McClendon, R.W., Arabnia, H.R., 2002. Apple classification based on surface bruises using image processing and neural networks. Trans. ASAE 45 (5), 1619–1627.

Shibayama, M., Akiyama, T., 1991. Estimating grain yield of maturing rice canopies using high spectral resolution reflectance measurements. Remote Sens. Env. 36 (1), 45–53.

SPSS, 2005. Advanced Models. Copyright 13.0 SPSS Inc., Chicago, IL.

Uno, Y., Prasher, S.O., Lacroix, R., Goel, P.K., Karimi, Y., Viau, A., Patel, R.M., 2005. Artificial neural networks to predict corn yield from compact airborne spectrographic imager data. Comput. Electron. Agric. 47 (2), 149–161.

USDA-NRCS, 1998. Keys to Soil Taxonomy, eighth ed. USDA-NRCS, Washington, DC, USA.

Vapnik, V.N., 1999. The Nature of Statistical Learning Theory. Springer, New York.

Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervás-Martínez, C. JCLEC: a JAVA framework for evolutionary computation. Soft Computing (2007), published on line.

Yang, C., Prasher, S.O., Landry, J.A., Ramaswamy, H.S., Ditommaso, A., 2000a. Application of artificial neural networks in image recognition and classification of crop and weeds. Can. Agric. Eng. 42 (3), 147–152.

Yang, C., Everitt, J.H., Bradford, J.M., Escobar, D.E., 2000b. Mapping grain sorghum growth and yield variations using airborne multispectral digital imagery. Trans. ASAE 43 (6), 1927–1938.

Yang, C., Bradford, J.M., Wiegand, C.L., 2001. Airborne multispectral imagery for mapping variable growing conditions and yields of cotton, grain sorghum, and corn. Trans. ASAE 44 (6), 1983–1994.

Yang, C., Everitt, J.H., Bradford, J.M., 2006. Comparison of QuickBird satellite imagery and airborne imagery for mapping grain sorghum yield patterns. Precis. Agric. 7 (1), 33–44.

Yao, X., 1999. Evolving artificial neural networks. Proc. IEEE 87 (9), 1423–1447.

Zhang, Y., Pulliainen, J., Koponen, S., Hallikainen, M., 2002. Application of an empirical neural network to surface water quality estimation in the Gulf of Finland using combined optical data and microwave data. Remote Sens. Env. 81 (2/3), 327–336.

2.1.2. Clasificadores basados en Redes Neuronales Evolutivas de Unidad Producto - *Evolutionary product-unit neural networks classifiers*

- F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo. Evolutionary Product-Unit Neural Networks Classifiers. Neurocomputing 72:1-3 (2008) 548-561.
    - Estado: Publicado.
    - Índice de Impacto (JCR 2007): 0.865.
    - Área de conocimiento: *Computer Science, Artificial Intelligence.* Ranking 50/93. Tercer Cuartil.

Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 72 (2008) 548–561
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Evolutionary product-unit neural networks classifiers

F.J. Martínez-Estudillo[a,*], C. Hervás-Martínez[b], P.A. Gutiérrez[b], A.C. Martínez-Estudillo[a]

[a]Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14005 Córdoba, Spain
[b]Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, 14071 Córdoba, Spain

Received 13 April 2007; received in revised form 26 October 2007; accepted 25 November 2007
Communicated by G.-B. Huang
Available online 15 January 2008

Abstract

This paper proposes a classification method based on a special class of feed-forward neural network, namely product-unit neural networks. Product-units are based on multiplicative nodes instead of additive ones, where the nonlinear basis functions express the possible strong interactions between variables. We apply an evolutionary algorithm to determine the basic structure of the product-unit model and to estimate the coefficients of the model. We use softmax transformation as the decision rule and the cross-entropy error function because of its probabilistic interpretation. The approach can be seen as nonlinear multinomial logistic regression where the parameters are estimated using evolutionary computation. The empirical and specific multiple comparison statistical test results, carried out over several benchmark data sets and a complex real microbial *Listeria* growth/no growth problem, show that the proposed model is promising in terms of its classification accuracy and the number of the model coefficients, yielding a state-of-the-art performance.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Classification; Product-unit neural networks; Evolutionary neural networks

1. Introduction

The simplest method for the classification of patterns provides the class level given their observations via linear functions in the predictor variables. This process of model fitting is quite stable, resulting in low variance but a potentially high bias. Frequently, in a real-problem of classification, we cannot make the stringent assumption of additive and purely linear effects of the variables. A traditional technique to overcome these difficulties is augmenting/replacing the input vector with new variables, the basis functions, which are transformations of the input variables, and then using linear models in this new space of derived input features. One first approach is to augment the inputs with polynomial terms to achieve higher-order Taylor expansions, for example, with quadratic terms and multiplicative interactions. Once the number and the structure of the basis functions have been determined, the models are linear in these new variables and their fitting is a standard procedure. Methods like sigmoidal feed-forward neural networks [6], projection pursuit learning [23], generalized additive models [31], and PolyMARS [43], a hybrid of multivariate adaptive splines (multiadaptive regression splines, MARS) [22], specifically designed to handle classification problems, can be seen as different basis function models. The major drawback of these approaches is stating the optimal number and typology of corresponding basis functions.

We tackle this problem proposing a nonlinear model along with an evolutionary algorithm (EA) that finds the optimal structure of the model and estimates its corresponding coefficients. Concretely, our approach tries to overcome the nonlinear effects of the input variables by means of a model based on nonlinear basis functions constructed with the product of the inputs raised to arbitrary powers. These basis functions express possible strong interactions between the variables, where the exponents may even take on real values and are suitable for automatic adjustment. The model proposed

*Corresponding author.
E-mail addresses: fjmestud@etea.com (F.J. Martínez-Estudillo), chervas@uco.es (C. Hervás-Martínez), i02gupep@uco.es (P.A. Gutiérrez), acme@etea.com (A.C. Martínez-Estudillo).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.11.019

corresponds to a special class of feed-forward neural network, namely product-unit based neural networks (PUNN) introduced by Durbin and Rumelhart [16]. They are an alternative to sigmoidal neural networks and are based on multiplicative nodes instead of additive ones. Unfortunately, the error surface associated with PUNNs is extremely convoluted with numerous local optima and plateaus. The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface. Because of this, their training is more difficult than the training of standard sigmoidal-based networks. For example, it is well known [7] that back-propagation is not efficient in the training of product-units. Section 3 will briefly show the most relevant techniques that have been used so far to apply learning methods to product-unit networks.

On the other hand, the evolutionary approach is used to optimize both the weights and the architecture of the network simultaneously. In general, classical neural networks training algorithms assume a fixed architecture; nevertheless, it is very difficult to know beforehand what the most suitable structure of the network for a given problem will be. There have been many attempts to design the architecture automatically, such as constructive and pruning algorithms [51,56]. We used EAs to design a nearly optimal neural network architecture because better results have been obtained using this heuristic [4,62]. This fact, together with the complexity of the error surface associated with a PUNN, justifies the use of an EA to design the topology of the network and to train its corresponding weights. The evolutionary process determines the number of basis functions, associated coefficients and corresponding exponents in the model.

We use the softmax activation function and the cross-entropy error function [6]. Therefore, from a statistical point of view, the approach can be seen as a nonlinear multinomial logistic regression [30], where we optimize log-likelihood using evolutionary computation. Actually, we attempt to estimate conditional class probabilities using a multilogistic model with the nonlinear model given by PUNNs.

We evaluate the performance of our methodology on seven data sets taken from the UCI repository [7], and on a real microbial growth/no growth problem in order to determine the growth limits of *Listeria monocytogenes* [2,18] to assure microbial safety and quality in foods. Empirical and statistical test results show that the proposed method performs well when compared to several other learning classification techniques. We obtain a classifier with interesting results in terms of classification accuracy and number of hidden nodes. Moreover, we show graphically the classification task carried out by the product-unit model together with its capability to both capture the interactions between the variables and to reduce the dimension of the input space. This reduction of dimensionality facilitates the study of the behavior of corresponding basis functions and the relevance of each input variable in the final model. This paper is organized as follows: Section 2 shows the main related works; Section 3 is devoted to a description of PUNNs; Section 4 describes the evolution of PUNNs; Section 5 explains the experiments and the comparison test carried out; and finally, Section 6 summarizes the conclusions of our work.

2. Related works

We start by giving a brief overview of the different methods that use basis functions to move beyond linearity. The first method cited to solve classification problems is conventional statistical discriminant analysis [30], which assumes that the measurement vectors in each class follow a normal multivariate distribution. If the covariance matrices of the measurements in each class are the same, the method shows that the regions created by Bayes' decision rule are separated by boundaries, which are linear in the input variables. When the conventional assumption of the equality of covariate matrices is dropped, Bayes' decision rule gives quadratic boundaries. In many examples, the inadequacy of linear or quadratic discriminant analysis for the purpose of classification made it necessary to look for approaches that could approximate highly nonlinear class boundaries. Instead of assuming specific distributions for the inputs and using them to calculate conditional class probabilities, one can estimate these classes directly from training sample cases.

A number of methods based on nonparametric regression [30], which are capable of approximating highly nonlinear class boundaries in classification problems, have been developed in the last few years.

Generalized additive models [31] comprise automatic and flexible statistical methods that may be used to identify and characterize nonlinear effects. The generalized additive model approximates multidimensional functions as a sum of univariate curves. Univariate functions are estimated in a flexible manner, using an algorithm whose basic building block is a scatter plot smoother, for example, the cubic smoothing spline. The additive model manages to retain interpretability by restricting nonlinear effects in the predictors in order to enter them into the model independently of each other. Generalized additive models provide a natural first approach to relaxing strong linear assumptions.

Bose [8] presented a method, classification using splines (CUS), somewhat similar to the neural network method, which uses additive cubic splines to estimate conditional class probabilities. Afterwards, the same author presented a modification of CUS, named "the method of successive projections", to solve more complex classification problems [9]. Although this method was presented using CUS, it is possible to replace CUS by any nonparametric regression-based classifier.

Kooperberg et al. [43] propose an automatic procedure that uses linear splines and their tensor products. This method is a hybrid of the MARS [22] called PolyMars, specifically designed to handle classification problems. It grows the model in a forward stage-wise fashion like MARS, but at each stage it uses quadratic approximation to multinomial log-likelihood to search for the next basis function pair. Once found, the enlarged model is fit according to the maximum likelihood method, and the process is repeated.

From a different point of view, neural networks have been an object of renewed interest among researchers, both in statistics and computer science, owing to the significant results obtained in a wide range of classification problems. Many different types of neural network architectures have been used, but the most popular one has been that of the single-hidden-layer feed-forward network. Amongst the numerous approaches using neural networks in classification problems, we focus our attention on that of evolutionary artificial neural networks (EANNs). EANNs have been a key research area in the past decade providing a better platform for simultaneously optimizing both network performance and architecture. Miller et al. [48] proposed evolutionary computation as a very good candidate to search the space of architectures because the fitness function associated with that space is complex, noisy, nondifferentiable, multimodal and deceptive. Since then, many evolutionary methods have been developed to evolve artificial neural networks, see, for example, [12,19,24,52,60,62,63]. In these works we find several methods that combine architectural evolution with weight learning and use different mutation operators, including, in some cases, partial training after each architectural mutation or approaches that hybridize EANNs with a local search technique to improve the slowness of the convergence. The problem of finding suitable architecture and the corresponding weights of the network is a very complex task (for a very interesting review of the matter the reader can consult [61]). Among the EANN paradigm it is worthwhile to point out two different approaches. Cooperative co-evolution [50] is a recent paradigm in the area of evolutionary computation focused on the evolution of co-adapted subcomponents without external interaction [24]. COVNET is a new cooperative co-evolutionary model for evolving artificial neural networks [25]. The method is based on the idea of co-evolving subnetworks that must cooperate to form a solution for a specific problem, instead of evolving complete networks.

On the other hand, multiobjective evolutionary optimization has recently appeared as an enhanced approach to optimize both the structure and weights of the neural network [5,27]. The idea of designing neural networks within a multiobjective setup was first considered by Abbass in [1]. Here, the multiobjective problem formulation essentially involves setting up two objectives: complexity of the network (number of weights, number of connections or a combination of both) and the training error. Finally, an interesting idea has been developed in subsequent works related to the topic of network ensembles [11,13,14]. In these papers, the authors tackle the ensemble-learning problem within a multiobjective setup where diversity and accuracy objectives are in conflict and the evolutionary process searches for a good trade-off between them.

Finally, a new learning algorithm called extreme learning machine (ELM) for single hidden-layer feed-forward neural networks has been recently proposed [35,36]. This novel procedure, unlike the conventional implementations of gradient-based learning algorithms, chooses randomly hidden nodes and analytically determines the output weights of the network. This algorithm provides good generalization performances at extremely fast learning speeds and it has been proved in theory that the universal approximator property holds [34]. However, ELM may need a higher number of hidden nodes due to random determination of input weights and hidden biases. Several algorithms based on the ELM method (hybrid proposals which use the differential evolution algorithm [64] and a convex optimization method [33]) have been developed to achieve good generalization performances with more compact networks.

3. Product-unit neural networks

In this section we present the family of product-unit basis functions used in the classification process and its representation by means of a neural network structure. This class of multiplicative neural networks comprises such types as sigma–pi networks and product-unit networks. A multiplicative node is given by

$$y_j = \prod_{i=1}^{k} x_i^{w_{ji}}, \tag{1}$$

where k is the number of the inputs. If the exponents w_{ji} in (1) are $\{0,1\}$ we obtain a higher-order unit, also known as sigma–pi unit. In contrast to the sigma–pi unit, in the product-unit the exponents are not fixed and may even take real values.

Some advantages of PUNNs are their increased information capacity and ability to form higher-order input combinations. Durbin and Rumelhart [16] determined empirically that the information capacity of product-units (measured by their capacity for learning random Boolean patterns) is approximately $3N$, as compared to $2N$ in a network with additive units for a single threshold logic function, where N denotes the number of inputs to the network. Besides, it is possible to obtain the upper bounds of the VC dimension [58] in PUNNs similar to those obtained in sigmoidal neural networks [55]. Schmitt [55] derives the upper bounds of the VC dimension and the pseudo-dimension for various types of networks with multiplicative units. As the most general case, Schmitt [55] considers feed-forward networks consisting of product and sigmoidal units, and showing that their pseudo-dimension is bounded from above by a polynomial with the same order of magnitude as the currently best known bound for purely sigmoidal networks. Concretely, he shows that a product-unit network has a pseudo-dimension at least $O(W^2k^2)$, where the number of parameters is W and k is the number of hidden nodes (see Theorem 1 and Corollary 2). As a consequence, the complexity of a PUNN (from the point of view of the VC dimension or the pseudo-dimension) depends on the number of parameters and the number of hidden nodes instead of on the values of the weights.

Finally, it is a straightforward consequence of the Stone–Weierstrass theorem to prove that PUNNs are universal approximators [46] (observe that polynomial functions in several variables are a subset of product-unit models).

Despite these advantages, PUNNs have a major drawback. They have more local minima and a higher probability of becoming trapped in them [38]. Several efforts have been made to carry out learning methods for product-units. Janson and Frenzel [41] developed a genetic algorithm for evolving the weights of a network based on product-units with a predefined architecture. The major problem with this kind of algorithm is how to obtain the optimal architecture beforehand. Ismail and Engelbrecht [38,39] applied four different optimization methods to train PUNNs: random search, particle swarm optimization, genetic algorithms and leapfrog optimization. They concluded that random search is not efficient for training this type of network, and that the other three methods show an acceptable performance in three problems of function approximation with low dimensionality. In a later paper [40] they used a pruning algorithm to develop both the structure and the training of the weights in a PUNN. Leerink et al. [44] tested different local and global optimization methods for PUNNs. Their results show that local methods, such as back-propagation, are prone to be trapped in local minima, and that global optimization methods, such as simulated annealing and random search, are impractical for larger networks. They suggested some heuristics to improve back-propagation, and the combination of local and global search methods. In short, the studies carried out on PUNNs have not tackled the problem of the simultaneous design of the structure and weights in this kind of neural network, using either classical or evolutionary based methods. Moreover, PUNNs have been applied mainly to solve regression problems so far [17,46,47,53,54].

On the other hand, it is interesting to note that a problem arises with networks containing product-units that receive negative inputs and have weights that are not integers. A negative number raised to some noninteger power yields a complex number. Since neural networks with complex outputs are rarely used in applications, Durbin and Rumelhart [16] suggest discarding the imaginary part and using only the real component for further processing. This manipulation would have disastrous consequences for the VC dimension when we consider real-valued inputs. No finite dimension bounds can, in general, be derived for networks containing such units [55].

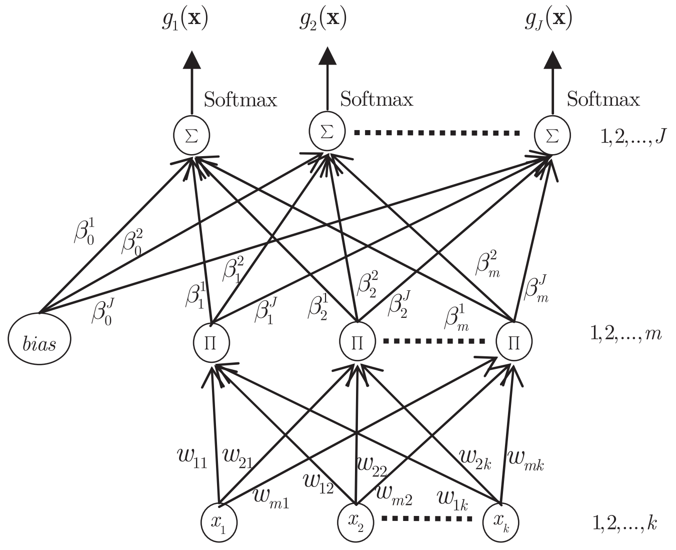


Fig. 1. Model of a product-unit neural network.

To avoid this problem, the input domain is restricted, and we define the set given by $\{\mathbf{x} = (x_1, x_2, ..., x_k) \in \mathbb{R}^k : x_i > 0, i = 1, 2, \ldots, k\}$.

We consider a PUNN with the following structure (Fig. 1): an input layer with k nodes, a node for every input variable, a hidden layer with m nodes and an output layer with J nodes, one for each class level. There are no connections between the nodes of a layer, and none between the input and output layers either. The activation function of the jth node in the hidden layer is given by $B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{k} x_i^{w_{ji}}$, where w_{ji} is the weight of the connection between input node i and hidden node j and $\mathbf{w}_j = (w_{j1}, \ldots, w_{jk})$ the weights vector. The activation function of the output node l is given by $\beta_0^l + \sum_{j=1}^{m} \beta_j^l B(\mathbf{x}, \mathbf{w}_j)$, where β_j^l is the weight of the connection between the hidden node j and the output node l and β_0^l the corresponding bias. The transfer function of all hidden and output nodes is the identity function. In this way, the estimated function $f_l(\mathbf{x}; \boldsymbol{\theta}_l)$ from each output is given by

$$f_l(\mathbf{x}; \boldsymbol{\theta}_l) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l B_j(\mathbf{x}, \mathbf{w}_j), \quad l = 1, 2, \ldots, J, \tag{2}$$

where $\boldsymbol{\theta}_l = (\boldsymbol{\beta}^l, \mathbf{w}_1, \ldots, \mathbf{w}_m)$ and $\boldsymbol{\beta}^l = (\beta_0^l, \beta_1^l, \ldots, \beta_m^l)$.

4. Classification problem

In a classification problem, measurements x_i, $i = 1,2,\ldots,k$, are taken on a single individual (or object), and the individuals are to be classified into one of the J classes based on these measurements. It is assumed that J is finite, and the measurements x_i are random observations from these classes.

A training sample $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, \ldots, N\}$ is available, where $\mathbf{x}_n = (x_{1n}, \ldots, x_{kn})$ is the random vector of measurements taking values in $\Omega \subset \mathbb{R}^k$, and $\mathbf{y}_n$ is the class level of the n-th individual. We adopt the common technique of representing the class levels using a "1-of-J" encoding vector $\mathbf{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(J)})$, such as $y^{(l)} = 1$ if $\mathbf{x}$ corresponds to an example belonging to class l, and otherwise $y^{(l)} = 0$. Based on the training sample we wish to find a decision function $C : \Omega \to \{1, 2, \ldots, J\}$ for classifying individuals. In other words, Ω provides a partition, say $D_1, D_2, \ldots, D_J$, of Ω, where D_l corresponds to the l class, $l = 1,2,\ldots,J$, and measurements belonging to D_l will be classified as coming from the l-th class. A misclassification occurs when a decision rule Ω assigns an individual (based on measurements vector) to a class j when it actually comes from a class $l \neq j$. We define the corrected classification rate (CCR) by $\mathrm{CCR} = (1/N)\sum_{n=1}^{N} I(C(\mathbf{x}_n) = \mathbf{y}_n)$, where $I(\bullet)$ is the zero-one loss function. A good classifier tries to achieve the highest possible CCR in a given problem.

We consider the softmax activation function [6] given by

$$g_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{l=1}^{J} \exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}, \quad l = 1, 2, \ldots, J. \tag{3}$$

If we use the training data set $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}$, where $x_{in} > 0$ $\forall i,n$, then the cross-entropy error function (K-class multinomial deviance) for those observations is

$$\begin{aligned} l(\boldsymbol{\theta}) &= -\frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{J} y_n^{(l)} \log g_l(\mathbf{x}_n, \boldsymbol{\theta}_l) \\ &= \frac{1}{N}\sum_{n=1}^{N}\left[-\sum_{l=1}^{J} y_n^{(l)} f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) + \log \sum_{l=1}^{J} \exp f_l(\mathbf{x}_n, \boldsymbol{\theta}_l)\right], \end{aligned} \tag{4}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)$. The error surface associated with the model is very convoluted with numerous local optimums and the Hessian matrix of the error function $l(\boldsymbol{\theta})$ is, in general, indefinite. Moreover, the optimal number of basis functions in the model (i.e. the number of hidden nodes in the neural network) is unknown. Thus, we determine the estimation of the vector parameters $\hat{\boldsymbol{\theta}}$ by means of an EA (see Section 5).

The optimum rule $C(\mathbf{x})$ is the following:

$$C(\mathbf{x}) = \hat{l}, \quad \text{where } \hat{l} = \arg\max_{l} g_l(\mathbf{x}, \hat{\boldsymbol{\theta}}),$$
$$\text{for } l = 1, 2, \ldots, J. \tag{5}$$

From a statistical point of view, with the softmax activation function (3) and the cross-entropy error (4), the neural network model can be seen as a multilogistic regression model. Nevertheless, the nonlinearity of the model with respect to the parameters $\boldsymbol{\theta}_j$ and the indefinite character of the associated Hessian matrix do not recommend the use of gradient-based methods (for example, iteratively reweighted least squares (IRLS) commonly used in the optimization of log-likelihood in linear multinomial logistic regression) to minimize the negative log-likelihood function.

Observe that softmax transformation produces positive estimates that sum to one and, therefore, the outputs can be interpreted as conditional probability of class membership. Specifically, the probability that $\mathbf{x}$ belongs to class l is written as

$$p(y^{(l)} = 1|\mathbf{x}, \boldsymbol{\theta}_j) = g_l(\mathbf{x}, \boldsymbol{\theta}_l), \quad l = 1, 2, \ldots, J \tag{6}$$

and the classification rule (5) coincides with the optimal Bayes rule. In other words, an individual should be assigned to the class which has the maximum probability, given vector measurement $\mathbf{x}$. On the other hand, because of the normalization condition

$$\sum_{l=1}^{J} p(y^{(l)} = 1|\mathbf{x}, \boldsymbol{\theta}_l) = 1, \tag{7}$$

the probability for one of the classes does not need to be estimated. There is a redundancy in the functions $f_l(\mathbf{x}, \boldsymbol{\theta}_l)$, since adding an arbitrary $h(\mathbf{x})$ to each output leaves the model (3) unchanged. Traditionally one of them is set to zero and we reduce the number of parameters to estimate. With loss generality, we set $f_J(\mathbf{x}, \boldsymbol{\theta}_J) = 0$.

5. Evolutionary algorithm

We apply an evolutionary neural networks algorithm to estimate the parameter that minimizes the cross-entropy error function. EA designs the structure and learns the weights of PUNNs. The search begins with an initial population of PUNNs, and in each iteration the population is updated using a population-update algorithm. The population is subjected to the operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving artificial networks [4,62]. With these features the algorithm falls into the class of evolutionary programming [20,21]. The general structure of EA is similar to the one presented in [32]:

(1) Generate a random population of size N_{P}.
(2) Repeat until the stopping criterion is fulfilled.
  (a) Calculate the fitness of every individual in the population.
  (b) Rank the individuals with respect to their fitness.
  (c) The best individual is copied into the new population.
  (d) The best 10% of population individuals are replicated and substitute the worst 10% of individuals.
  Over that intermediate population we
  (e) Apply parametric mutation to the best 10% of individuals.
  (f) Apply structural mutation to the remaining 90% of individuals.

In the current approach, $l(\boldsymbol{\theta})$ is the error function of an individual g of the population, where g is a PUNN given by the multivaluated function $g(\mathbf{x}, \boldsymbol{\theta}) = (g_1(\mathbf{x}, \boldsymbol{\theta}_1), \ldots, g_l(\mathbf{x}, \boldsymbol{\theta}_l))$ and the fitness measure is a strictly decreasing transformation of the error function, $l(\boldsymbol{\theta})$ given by $A(g) = 1/(1 + l(\boldsymbol{\theta}))$.

Parametric mutation is accomplished for each coefficient w_{ji}, β_j^l of the model with Gaussian noise, where the variances of the normal distribution are updated throughout the evolution of the algorithm. Once the mutation is performed, the fitness of the individual is recalculated and the usual simulated annealing process [42,49] is applied. On the other hand, structural mutation implies a modification in the neural network structure and allows explorations of different regions in the search space while helping to keep up the diversity of the population. There are five different structural mutations: node deletion, connection deletion, node addition, connection addition and node fusion. These five mutations are applied sequentially to each network. For further details about the parametric and structural mutations see [32,46].

The stop criterion is reached if one of the following conditions is fulfilled: a number of generations are reached or the variance of the fitness of the best 10% of the population is less than 10^{-4}.

The parameters used in EA are common for all problems. The exponents w_{ji} are initialized in the $[-5,5]$ interval, the coefficients β_j^l are initialized in $[-5,5]$. The maximum number of hidden nodes is $m=6$. The size of the population is $N_P=1000$. The number of nodes that can be added or removed in a structural mutation is within the [1,2] interval. The number of connections that can be added or removed in a structural mutation is within the $[1,c]$ interval, where c is a third of the number of connections of the model. We consider 400 as the maximum number of generations.

We have done a simple linear rescaling of the input variables in the interval [1,2], X_i^* being the transformed variables. The lower bound is chosen to avoid input values near 0, which can produce very large values of the outputs for negative exponents. The upper bound is chosen to avoid dramatic changes in the outputs of the network when there are weights with large values (especially in the exponents).

6. Experiments

In this section we compare the Evolutionary Product Unit Neural Network method (EPUNN) with different evolutionary neural network learning algorithms. First of all, in order to show the abilities of the product-unit models, we run the same evolutionary setup with standard neural networks with sigmoidal units, and we compare the results of the models based on multiplicative units against additive unit models. Next, we compare our model EPUNN to COVNET, a new cooperative co-evolutionary neural network method. Afterwards, the comparison is made with respect to two recent approaches based on multiobjective evolutionary neural networks: MPANN and SPANN. Furthermore, we analyze in detail two classification models obtained by EA in two data sets, to show graphically the task of classification carried out by the product-unit model and how well it can capture the interactions between the variables as well as reduce the dimension of the input space. Finally, the performance of our model is tested in a complex real microbial *Listeria* growth/no growth problem.

The different experiments were conducted using a software package developed in JAVA by the authors as an extension of JCLEC framework (http://jclec.sourceforge.net/) [59]. The software package is available in the noncommercial JAVA tool named KEEL (http://www.keel.es) [3].

6.1. Neural network models comparisons

6.1.1. Product-units versus additive sigmoidal units

In order to justify the benefits of applying the product-unit model, we evolve the traditional feed-forward neural networks with sigmoidal additive units with the same EA. This approach will be known as evolutionary sigmoidal unit neural networks (ESUNN). The same evolutionary setup (EA and parameters) will be considered for the EPUNN and ESUNN models in seven classification problems with different features (see Table 1). The seven data sets are available by anonymous ftp from [7].

We use a 10-fold stratified cross-validation and we carry out 10 runs of each fold for every data set. This gives a 100 data points for each data set, from which the average classification accuracy in the generalization set (CCR_G) and standard deviation are calculated.

The comparison between the predictive ability of the two models of artificial neural networks (EPUNN and ESUNN) was carried out using statistical tests in terms of accuracy (mean of the correct classification rate, CCR_G), homogeneity (standard deviation, SD, of the CCR_G), best

Table 1
Data sets used for the experiments, sorted by size

Data sets	Instances	Missing values (%)	Numeric attributes	Binary attributes	Nominal attributes	Classes
Heart-statlog	270	0.0	13	0	0	2
Ionosphere	351	0.0	33	1	0	2
Balance	625	0.0	4	0	0	3
Australian	690	0.6[a]	6	4	5	2
Diabetes	768	0.0	8	0	0	2
German	1000	0.0	6	3	11	2
Hypothyroid	3772	5.5[a]	7	20	2	4

[a]We have used original data sets without transforming the missing values.

and worst results and topology (mean and SD of the number of connections). See Table 2.

First, we study the normality of CCR_G and the number of connections by means of a Kolmogorov–Smyrnov test [10]. For this and the remaining comparisons, all tests were obtained using the SPSS statistical package [57].

From the results for CCR_G (see Table 3) a normal distribution can be assumed for Diabetes, German and Hypothyroid because the observed significance levels, p-Values, were higher than the significance level of the test, $\alpha = 0.05$. The normality hypothesis of the accuracy distribution cannot be assumed in Heart-statlog, Ionosphere, Balance, and Australian data sets. Regarding the number of connections, it is possible to assume the normal distribution hypothesis in all cases because p-Values were higher than 0.05. Therefore, under the hypothesis of normal distribution, the comparisons between EPUNN and ESUNN models were done by using the t-test for equal or different variance, according to the results previously obtained by the Levene test [45], in both cases with $\alpha = 0.05$. On the other hand, we use a nonparametric Mann–Whitney test [10] for differences between means in the non-normal distribution hypothesis. The statistical results obtained are shown in Table 4. We observe significant differences, in favor of EPUNN, at a significance level $\alpha = 0.05$ in Balance, Diabetes and German data sets in CCR_G results, while there are not significant differences in the remaining ones. As for the number of connections, there are a significantly lower number of connections for EPUNN in the Heart-statlog, Balance, Australian and Hypothyroid data sets, and there are a significantly lower

Table 4
Statistical comparison (p-Values of the Levene and Student's t-tests or Mann–Whitney test (M–W)) of the generalization ability (CCR_G) and number of connections (# Conn.) for EPUNN and ESUNN models

EPUNN versus ESUNN	CCR_G		♯ Conn.	
	Levene test	t-Test or M–W test	Levene test	t-Test
Heart-statlog	–	0.525	0.000	0.000[a]
Ionosphere	–	0.185	0.000	0.149
Balance	–	0.000[a]	0.300	0.000[a]
Australian	–	0.525	0.003	0.009[a]
Diabetes	0.086	0.047[a]	0.082	0.010[a]
German	0.129	0.000[a]	0.080	0.000[a]
Hypothyroid	0.080	0.618	0.212	0.001[a]

[a]There are significant differences on average using a significance level $\alpha = 0.05$.

Table 2
Statistical results for EPUNN and ESUNN in seven data sets

Data sets	Training				Generalization				♯ Conn.	
	Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
ESUNN										
Heart-statlog	86.21	1.15	88.89	83.54	83.22	6.61	92.59	66.67	16.99	2.52
Ionosphere	91.75	1.36	94.30	88.29	88.66	5.22	100.00	74.29	46.35	8.80
Balance	92.38	1.22	95.74	90.05	91.03	4.15	98.39	79.03	30.31	2.33
Australian	87.81	0.81	90.02	85.35	85.49	3.92	94.20	78.26	49.58	12.66
Diabetes	79.15	0.75	80.78	77.75	75.93	5.25	84.21	63.16	17.02	3.00
German	77.32	1.52	79.67	73.22	73.00	5.56	86.00	60.00	62.84	13.96
Hypothyroid	94.34	0.23	94.82	93.76	94.18	0.95	96.02	92.57	76.73	11.30
EPUNN										
Heart-statlog	84.65	1.63	88.48	80.25	81.89	6.90	96.30	62.96	14.78	3.83
Ionosphere	93.79	1.46	97.15	90.19	89.63	5.52	100.00	74.29	43.97	13.87
Balance	97.26	0.98	99.47	94.32	95.69	2.36	100.00	90.32	25.62	2.18
Australian	87.01	0.82	88.57	85.02	85.74	3.90	95.65	78.26	44.13	16.26
Diabetes	77.79	0.75	79.62	76.16	77.40	4.38	84.21	68.42	18.08	2.31
German	77.33	1.34	80.56	73.56	76.28	4.82	90.00	65.00	74.16	18.82
Hypothyroid	94.51	0.55	96.97	93.64	94.25	1.08	96.55	92.31	71.13	12.69

Table 3
Normality Kolmogorov–Smirnov test of the generalization ability (CCR_G) and number of connections (# Conn.) for EPUNN and ESUNN models

K–S test	Heart-statlog		Ionosphere		Balance		Australian		Diabetes		German		Hypothyroid	
	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN	EPUNN	ESUNN
p-Value CCR_G	0.010[a]	0.001[a]	0.066	0.018[a]	0.002[a]	0.010[a]	0.034[a]	0.255	0.330	0.304	0.647	0.522	0.123	0.076
p-Value # Conn.	0.222	0.157	0.602	0.465	0.108	0.205	0.436	0.689	0.492	0.080	0.624	0.354	0.108	0.205

[a]Non-Gaussian distribution for a significance level $\alpha = 0.05$.

number of connections for ESUNN in the Diabetes and German ones, using a significance level $\alpha = 0.05$

We conclude that for the Heart-statlog, Australian and Hypothyroid data sets, the models using EPUNN have a significantly lower number of connections without significantly worsening or improving the mean value of CCR_G, as compared to those found with the ESUNN model.

For the Ionosphere data set there are no significant differences in the CCR_G and in the number of connections with respect to the type of base functions used.

The Balance data set has a significantly higher mean CCR_G value when using EPUNN models than when using ESUNN models; these models also manifesting a significantly lower mean number of connections. For Diabetes and German data sets the mean CCR_G values obtained with EPUNN models are significantly higher than when ESUNN models are used, although the mean number of connections is also significantly higher in the EPUNN models.

6.1.2. *Comparison with a cooperative co-evolutionary neural network method: COVNET*

In this section we compare our approach with COVNET, a new cooperative co-evolutionary model for evolving artificial neural networks [25]. The method is based on the idea of co-evolving subnetworks that must cooperate to form a solution for a specific problem, instead of evolving complete networks.

To be consistent with [25], the comparison is tested on three classification data sets: Diabetes, Australian and Heart disease. The features of Diabetes and Australian data sets can be seen in Table 1. Heart disease is a data set from the Cleveland Clinic Foundation and contains 13 attributes, 5 classes and 270 examples (the problem is described more deeply in [15]). We have carried out the same experimental design: 75% of patterns of each data set were used for training purposes and the remaining 25% were used as generalization assessment set. Three different random permutations of the patterns were made, and the evolutionary process was repeated 10 times for each permutation.

Table 5 shows, for each permutation of the data sets, the averaged CCR over 10 repetitions for training and generalization sets, the standard deviation, the best and worst individuals, and the mean and standard deviation of the number of connections of the best networks obtained for each experiment. Each permutation of the data set is labelled I, II and III in the table. The mean results over the three permutations are labelled All.

In order to verify the true difference between the performance in EPUNN and the COVNET network, we conducted statistical tests (see Table 6). First, we corroborated by means of a Kolmogorov–Smyrnov test that the

Table 5
Statistical results for EPUNN and COVNET in three data sets

Data sets	Partition	Algorithm	Training				Generalization				♯ Conn.	
			Mean	SD	Best	Worst	Mean	SD	Best	Worst	Mean	SD
Diabetes	I	COVNET	77.74	0.73	78.65	76.56	80.05	2.84	83.85	75.52	–	–
		EPUNN	77.48	0.91	78.99	76.22	79.84	0.95	81.25	78.65	20.43	2.80
	II	COVNET	77.31	0.70	78.30	76.04	79.37	2.36	82.29	76.04	–	–
		EPUNN	76.82	0.56	78.99	75.87	81.30	0.79	82.29	80.21	20.90	2.77
	III	COVNET	77.57	0.42	78.12	76.74	80.89	0.82	82.29	79.69	–	–
		EPUNN	76.91	0.61	78.47	76.39	82.03	1.21	84.38	80.73	21.42	2.91
	All	COVNET	77.54	0.63	–	–	80.10	2.20	–	–	24.60	–
		EPUNN	77.07	0.75	–	–	81.06	1.34	–	–	20.94	2.76
Heart disease	I	COVNET	87.43	0.63	88.61	86.63	85	1.35	86.76	82.35	–	–
		EPUNN	86.63	0.70	87.13	85.64	86.62	1.29	88.24	83.82	25.61	3.81
	II	COVNET	86.34	1.42	89.11	84.16	87.94	2.17	91.18	85.29	–	–
		EPUNN	83.76	1.04	87.129	82.18	90.88	1.52	94.12	88.24	28.20	4.10
	III	COVNET	86.88	0.75	87.62	85.64	84.26	3.18	88.24	79.41	–	–
		EPUNN	85.55	1.21	87.62	84.16	83.52	1.67	85.29	80.88	27.30	2.26
	All	COVNET	86.88	1.06	–	–	85.74	2.79	–	–	33.07	–
		EPUNN	85.31	1.55	–	–	87.01	3.39	–	–	27.03	3.54
Australian	I	COVNET	86.25	0.75	87.45	84.94	88.02	0.88	88.95	86.05	–	–
		EPUNN	84.17	0.32	84.36	83.59	88.66	0.63	88.95	87.21	11.31	11.84
	II	COVNET	85.69	0.92	87.64	84.36	88.95	1.19	90.71	86.63	–	–
		EPUNN	84.40	0.30	84.94	83.78	88.84	0.25	88.95	88.37	3.97	1.37
	III	COVNET	85.89	0.59	86.87	85.14	88.31	0.89	89.53	86.63	–	–
		EPUNN	84.48	0.24	84.94	84.36	88.72	0.30	88.95	88.37	6.40	4.95
	All	COVNET	85.95	0.78	–	–	88.43	1.04	–	–	34.23	–
		EPUNN	85.24	0.64	–	–	88.55	1.17	–	–	31.40	12.16

Table 6
Statistical test in three data sets

Data sets	Partition	K–S (p-Values)	t-Test (p-Values)
Diabetes	All	0.637	0.046[a]
Heart disease	All	0.702	0.125
Australian	All	0.061	0.685

[a]There are significant differences between the EPUNN and the COVNET means, for $\alpha = 0.05$.

distribution of the CCR_G values is normal; p-Values in the table are higher than $\alpha = 0.05$, and in this way, the normality hypothesis for the distribution of the CCR_G obtained using EPUNN is accepted. In [25], the normality hypothesis for the distribution of the CCR_G values for COVNET is accepted using the Kolmogorov–Smyrnov test, regarding p-Values presented for Diabetes data set ($\alpha = 0.44$), for Heart disease data set ($\alpha = 0.07$), and for Australian data set, ($\alpha = 0.56$). Then, we tested the hypothesis that the means of the CCR_G obtained from the experiments with EPUNN and COVNET networks do not result in significant differences, and therefore we performed a t-test that allowed us to ascertain if the mean of the CCR_G obtained with EPUNN was significantly higher than the mean of the CCR_G obtained with the COVNET network at a significance level $\alpha = 0.05$.

Table 6 shows that, when we consider the 30 results obtained in the three folds, the improvements on mean for the CCR_G of EPUNN are significant in Diabetes (for $\alpha = 0.05$), while there are not significant differences for the Australian and Heart disease data sets.

6.1.3. Multiobjective neural networks

Now we compare our proposal with some of the most recent algorithms that use the evolutionary neural network paradigm: MPANN (memetic pareto artificial neural network) and SPANN (a self-adaptive version of MPANN) [1]. We have tested EPUNN on two benchmark problems used in the references quoted: the Australian credit card assessment problem and the Diabetes problem. To be consistent with [1,13], we use a 10-fold stratified cross-validation for the Australian data set and a 12-fold stratified cross-validation for the Diabetes data set. For every data set we performed 10 runs of each fold. This gives 100 data points for the Australian, and 120 for the Diabetes data set, from which the average classification accuracy CCR_G and the standard deviation are calculated.

We carry out a standard comparison test to verify if there are significant differences in the CCR_G when comparing the EPUNN model to MPANN/SPANN methodologies. Following the hypothesis of the normality of the results, we carry out a t-test to compare the mean results of CCR_G in the EPUNN and MPANN/SPANN methodologies. We use the mean and standard deviation of the (100, 120) runs for MPANN and SPANN, 10 for each fold, and (100, 120) runs for EPUNN, 10 for each fold.

Table 7
Mean classification accuracy and standard deviation for generalization CCR_G for MPANN, SPANN and EPUNN

	MPANN	SPANN	EPUNN
Australian			
Test error rate	86.40 ± 4.50	86.90 ± 4.60	85.74 ± 3.90
Hidden units	5.00 ± 1.94	6.00 ± 1.83	2.90 ± 0.09
Diabetes			
Test error rate	74.90 ± 6.20	70.70 ± 5.00	77.89 ± 3.48
Hidden units	6.60 ± 1.51	7.17 ± 2.21	2.98 ± 0.14

Table 7 shows the statistical results of EPUNN, MPANN and SPANN for Australian and Diabetes data sets. The p-Values obtained for Australian (0.276 for MPANN and 0.058 for SPANN) and for Diabetes (0.000 for MPANN and 0.000 for SPANN) show that there are no significant differences between the mean results for MPANN and EPUNN algorithms and for SPANN and EPUNN in the Australian data set for a significant coefficient $\alpha = 0.05$; but there are significant differences between the mean results for MPANN and EPUNN algorithms and for SPANN and EPUNN in the Diabetes data set for the same significant coefficient. We conclude that in the latter data set, the EPUNN mean is higher than the MPANN and SPANN means. Moreover, there are differences between the EPUNN model and the MPANN and SPANN methodologies with respect to the number of hidden nodes in both databases, where this number is lower for EPUNN. Table 7 shows that the models constructed by EPUNN are smaller than the models built by the other algorithms. Therefore, the experiments show that the EPUNN algorithm produces models with an interesting trade-off between the accuracy and complexity of the classifier, determined by the low number of hidden nodes, possibly outperforming its interpretability.

Now we study in detail the best product-unit model obtained for the Australian and Diabetes data sets. For the Diabetes data set, we consider the best model of one of the 10-fold used in the experiments (specifically the seventh fold). These models can be easily implemented and the reader can reproduce and compare the results.[1]

The model for the Diabetes data set is determined by three basis functions:

$$B_1 = (X_2^*)^{2.749}(X_6^*)^{0.528},$$
$$B_2 = (X_1^*)^{-1.0785}(X_3^*)^{1.672}(X_5^*)^{0.904}(X_6^*)^{-3.319}(X_7^*)^{-1.349}(X_8^*)^{-1.662},$$
$$B_3 = (X_1^*)^{2.296}(X_2^*)^{-4.677}(X_4^*)^{0.634}(X_6^*)^{3.682}(X_7^*)^{2.832},$$

and the output of the softmax transformation is

$$\hat{g}_1(\mathbf{x}) = \frac{\exp\{-4.179 + 0.961B_1 - 4.569B_2 + 0.262B_3\}}{1 + \exp\{-4.179 + 0.961B_1 - 4.569B_2 + 0.262B_3\}}.$$

[1]The folds used in the cross-validation experiments and the EPUNN models obtained are available for the reader in http://www.uco.es/grupos/ayrna/.

By using the properties of the softmax, the decision rule can be expressed in a more simplified way:

$$C(x) = \begin{cases} 1 & \text{if } 0.961B_1 - 4.569B_2 + 0.262B_3 > 4.179, \\ 0 & \text{if } 0.961B_1 - 4.569B_2 + 0.262B_3 < 4.179. \end{cases}$$

If we observe the best model obtained for the Diabetes data set, and taking into account that the transformed variables X_i^* take values in the same interval $[1, 2]$, we see that the most relevant variables are X_2, X_6 and X_7; likewise, X_4 is the least important variable in the model.

On the other hand, we observe that the product-unit model transforms the eight-dimensional input space into a three-dimensional space given by the basis functions $B_1(\mathbf{x})$, $B_2(\mathbf{x})$, and $B_3(\mathbf{x})$. The model tries to capture the interactions among the variables and carries out a reduction in the dimensionality of the space. It is interesting to point out that this reduction allows us to depict the separation of the two classes into training and test points by means of linear functions in the transformed space. Fig. 2 shows the graphics for the training and test points for the Diabetes data set problem and the plane that separates the points in the two classes.

A similar analysis can be carried out for the Australian data set. We consider the best model of one of the 10-fold used in the experiments (the first fold) given by the two basis functions:

$$B_1 = (X_9^*)^{0.783}(X_{11}^*)^{3.142}(X_{12}^*)^{0.426}(X_{18}^*)^{0.661}(X_{26}^*)^{1.065}$$
$$(X_{42}^*)^{2.745}(X_{43}^*)^{0.419}(X_{45}^*)^{0.112}(X_{49}^*)^{-2.320}(X_{51}^*)^{0.858},$$

$$B_2 = (X_1^*)^{-2.687}(X_3^*)^{-2.005}(X_4^*)^{-2.871}(X_6^*)^{2.676}(X_9^*)^{4.618}$$
$$(X_{14}^*)^{1.935}(X_{15}^*)^{-1.099}(X_{16}^*)^{-2.957}(X_{17}^*)^{3.366}(X_{25}^*)^{3.688}$$
$$(X_{27}^*)^{4.551}(X_{28}^*)^{1.164}(X_{30}^*)^{-1.679}(X_{38}^*)^{-0.029}(X_{39}^*)^{1.510}$$
$$(X_{40}^*)^{3.462}(X_{41}^*)^{-2.613}(X_{42}^*)^{2.415}(X_{43}^*)^{-4.390}(X_{46}^*)^{-4.592}$$
$$(X_{47}^*)^{-0.070}$$

and the output of the softmax transformation is

$$\hat{g}_1(\mathbf{x}) = \frac{\exp\{-3.230 + 0.416B_1 + 0.119B_2\}}{1 + \exp\{-3.230 + 0.416B_1 + 0.119B_2\}}.$$

The decision rule can be expressed as

$$C(\mathbf{x}) = \begin{cases} 1 & \text{if } 0.416B_1 + 0.119B_2 > 3.230, \\ 0 & \text{if } 0.416B_1 + 0.119B_2 < 3.230. \end{cases}$$

Fig. 3 shows the training and test points graphics for the Australian data set problem and the linear decision boundary that separates the points in the two classes. Observe that we have carried out a logarithmic transformation in the vertical axis to improve the graph. It is important to point out that, in this case, the product-unit model projects the 50-dimensional input space onto a two-dimensional space given by the basis functions $B_1(\mathbf{x})$ and $B_2(\mathbf{x})$. This reduction allows us a graphical analysis of the classification problem, facilitating the study of basis function behavior as well as the relevance of the input variables in the model. For example, the graphics for the Australian data set show that the basis function $B_1(\mathbf{x})$ is more important than the $B_2(\mathbf{x})$ ones in the model. Taking into account, again, that the transformed variables X_i^* take values in the same interval $[1, 2]$, we can see that the most

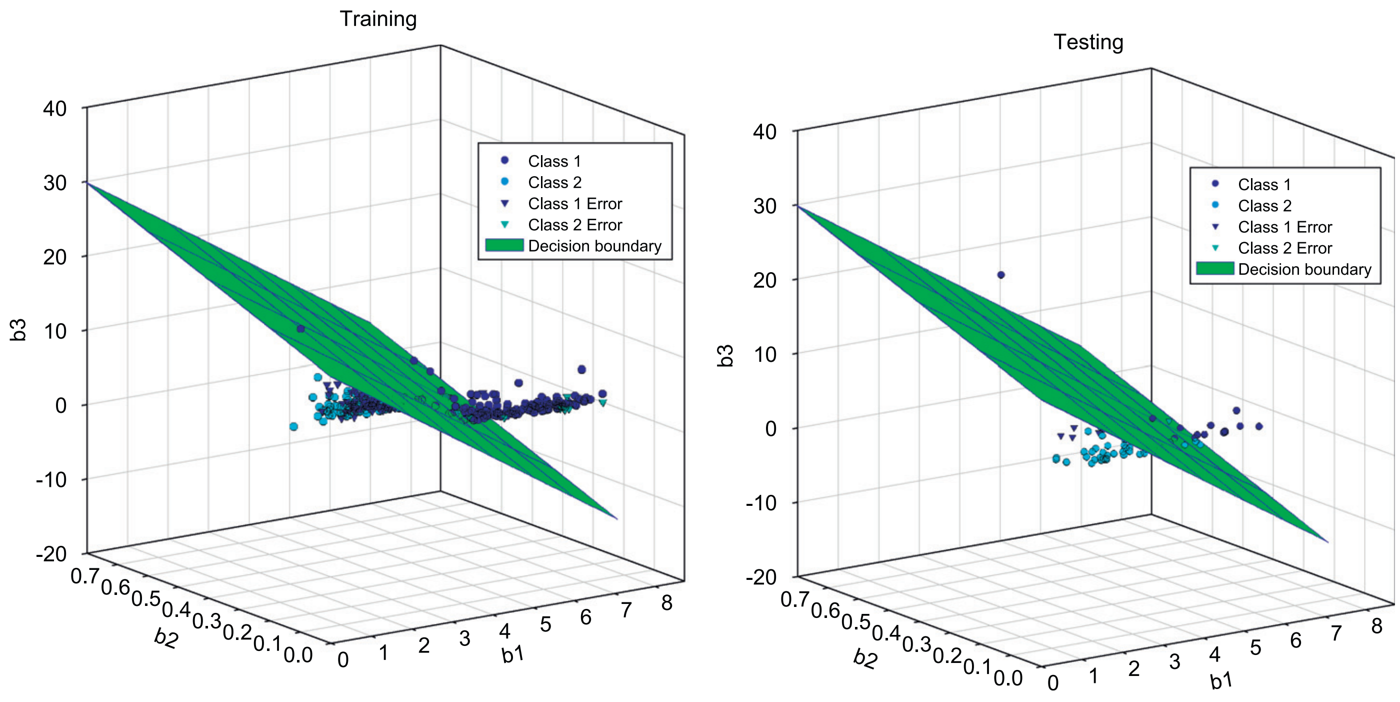


Fig. 2. Graphics of training and test points and decision boundary for the Diabetes data set problem.

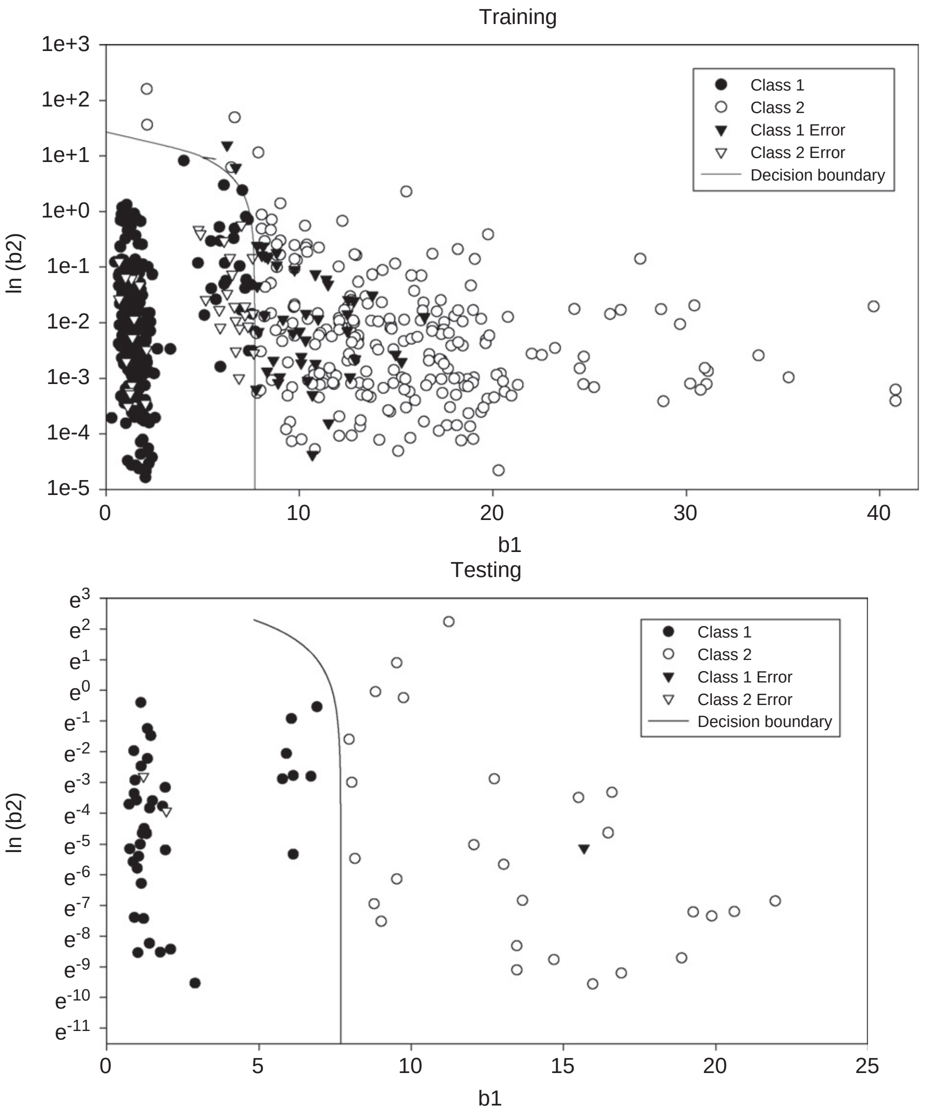


Fig. 3. Graphics of training and test points and decision boundary for the Australian data set problem.

relevant variables are X_{11} and X_{42}. Moreover, a value of $B_1(\mathbf{x})$ higher than 7 provides us with a simple and accurate rule of classification (see Fig. 3). Observe that the graphical study above can be carried out when the final model has two or three nodes in the hidden layer.

6.2. *A real classification problem: Listeria growth/no growth*

The main purpose of this section is to present and compare the efficacy, in terms of good classification, of different logistic regression approaches and the EPUNN on a data set of *L. monocytogenes* growth prediction, as a function of storage temperature *T*, pH, citric acid (CA) and ascorbic acid (AA).

L. monocytogenes has been a serious problem that has concerned food industries due to its ubiquity in the natural environment [2,37] and the specific growth conditions of the pathogen, which lead to its high prevalence in different kinds of food products. One impetus for this research was the problem of listeriosis [26]; so different strategies have been proposed to limit levels of contamination at the time of consumption to less than 100 CFU/g [18].

A fractional factorial design was followed in order to find the growth limits of *L. monocytogenes*. A number of 232 different conditions were chosen for the model with eight replicates per condition, from which we have eliminated those that were far removed from the growth/no-growth range, so that we have considered 305 data to form the training group, 57%, and 234 data to form the generalization group. This experimental design was intended to explore the survival/death interface. Data were collected at concentrations of CA and AA between 0% and 0.4% (w/v), at 4, 7, 10, 15 and 30 °C with a pH range of 4.5–6.

We used two logistic regression methods, a full logistic model, MLogistic, and a backward selected variables

Table 8
CCR obtained for the growth limits of *L. monocytogenes* for MLogistic, SLogistic and EPUNN models

Models	CCR_T	CCR_G
SLogistic	82.30	76.10
Mlogistic	82.30	74.40
EPUNN	92.83	86.72

method, SLogistic, using SPSS software [57], in order to obtain the significant variables of the logistic regression model by stepwise selection. These methods are considered because there has been a renewed interest in the use of these statistical techniques in predictive microbiology (see, for example, [28]). Then, the classification efficiency of the training and generalization data sets of the EPUNN model was compared with MLogistic and SLogistic models. Table 8 shows a mean, in 30 runs, of the 86.72% of the CCR_G from the EPUNN model, a great improvement on the percentage obtained using the SLogistic (76.10%) or MLogistic (74.40%) models. These results are in line with those obtained by Hajmeer and Basheer [29].

7. Conclusions

We propose a classification method that combines a nonlinear model based on a special class of feed-forward neural network, namely PUNNs, and a learning EA that finds the optimal structure of the model and estimates the corresponding coefficients. To the best of our knowledge, this is the first study that applies evolutionary PUNNs to solve a wide range of multiclassification problems evolving both structure and weights. A review of the related literature shows that, up to now, research on product-units has mainly been applied to solve regression problems.

Our method uses softmax transformation and the cross-entropy error function. From a statistical point of view, the approach can be seen as nonlinear multinomial logistic regression where we use evolutionary computation to optimize log-likelihood. In fact, we attempt to estimate conditional class probabilities using a multilogistic model with nonlinear models given by product-units. The coefficients that minimize the cross-entropy error function are estimated by means of an EA. The algorithm proposed evolves both the weights and the structure of the network using evolutionary programming. The difficulty entailed in ascertaining the most suitable network structure at the outset of a given problem is well known; the evolution of the structure, however, partially alleviates this problem. The performance of our approach is estimated by means of comparison with other methodologies from different points of view. First, the results obtained in the comparison of EPUNN with the evolution of traditional feed-forward neural networks with sigmoidal additive units shows the capabilities of product-unit models versus the standard neural network model. Next, we compare our model EPUNN to a new cooperative co-evolutionary method (COVNET), and with respect to two recent approaches based on multiobjective evolutionary neural networks (MPANN and SPANN). In both cases, the empirical and statistical results show that EPUNN model performs well compared to other evolutionary neural network techniques for classification. Promising results are obtained in terms of classification accuracy, number of connections and number of nodes of the classifier. Moreover, we show the best model for each problem and graphically use two examples to illustrate the classification task carried out by the PUNN model as well as the capability of the product-unit to both capture the interactions between the variables and reduce the dimensionality of the problem.

Finally, for a nontrivial real problem of predictive microbiology, the result of the accuracy obtained using the EPUNN model outperforms the results obtained by consolidated statistical techniques for classification.

As future work, it would be of interest to state the relationship among the level of interaction of the data, the complexity level of each data set and the level of performance obtained in the product-unit models for the a corresponding problem.

Acknowledgments

This work has been financed in part by TIN 2005-08386-C05-02 projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT) and FEDER funds. The authors would like to thank the HIBRO Research Group of the University of Córdoba, Spain, which obtained the experimental data set used in Section 6.2 of this paper. The research of P.A. Gutiérrez has been subsidized by the FPU Predoctoral Program (Spanish Ministry of Education and Science), Grant reference AP2006-01746.

References

[1] H.A. Abbass, Speeding up backpropagation using multiobjective evolutionary algorithms, Neural Comput. 15 (11) (2003) 2705–2726.
[2] N. Ahamad, E.H. Marth, Behaviour of *Listeria monocytogenes* at 7, 13, 21 and 35 °C in tryptose broth acidified with acetic citric or lactic acid, J. Food Prot. 52 (1989) 688–695.
[3] J. Alcala-Fdez, L. Sánchez, S. García, M.J.D. Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. 2007, accepted for publication.
[4] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Trans. Neural Networks 5 (1) (1994) 54–65.
[5] M. Azevedo, A.D. Padua, B. Rodrigues, Improving generalization of MLPs with sliding mode control and the Levenberg–Marquardt algorithm, Neurocomputing 70 (7–9) (2007) 1342–1347.
[6] M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[7] C. Blake, C.J. Merz, UCI repository of machine learning data bases, 1998 ⟨http://www.ics.uci.edu/mlearn/MLRepository.thmlwww.ics.uci.edu/mlearn/MLRepository.thml⟩.

[8] S. Bose, Classification using splines, Comput. Stat. Data Anal. 22 (1996) 505–525.
[9] S. Bose, Multilayer statistical classifiers, Comput. Stat. Data Anal. 42 (2003) 685–701.
[10] W.J. Conover, Practical Nonparametric Statistics, Wiley, New York, 1971.
[11] A. Chandra, X. Yao. DIVACE: diverse and accurate ensemble learning algorithm, in: Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning, Lecture Notes in Computer Science, vol. 3177, Springer, Berlin, 2004.
[12] A. Chandra, X. Yao, Evolutionary framework for the construction of diverse hybrid ensembles, in: Proceedings of the 13th European Symposium on Artificial Neural Networks, d-side, Brugge, Belgium, 2005.
[13] A. Chandra, X. Yao, Ensemble learning using multi-objective evolutionary algorithms, J. Math. Modelling Algorithms 5 (4) (2006) 417–445.
[14] A. Chandra, X. Yao, Evolving hybrid ensembles of learning machines for better generalization, Neurocomputing 69 (7–9) (2006) 686–700.
[15] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, V. Froelicher, International application of a new probability algorithm for the diagnosis of coronary artery disease, Am. J. Cardiol. 64 (1989) 304–310.
[16] R. Durbin, D. Rumelhart, Products units: a computationally powerful and biologically plausible extension to backpropagation networks, Neural Comput. 1 (1989) 133–142.
[17] A.P. Engelbrecht, A. Ismail, Training product unit neural networks, Stability Control: Theory Appl. 2 (1–2) (1999) 59–74.
[18] European Commission, Opinion of the scientific committee on veterinary measures relating to public health on *Listeria monocytogenes*, 1999 ⟨http://www.europa.eu.int/comm/food/fs/sc/scv/out25⟩.
[19] D.B. Fogel, Using evolutionary programming to greater neural networks that are capable of playing Tic-Tac-Toe, in: International Conference on Neural Networks, IEEE Press, San Francisco, CA, 1993.
[20] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, New York, 1995.
[21] D.B. Fogel, A.J. Owens, M.J. Wals, Artificial Intelligence Through Simulated Evolution, Wiley, New York, 1966.
[22] J. Friedman, Multivariate adaptive regression splines (with discussion), Ann. Stat. 19 (1991) 1–141.
[23] J. Friedman, W. Stuetzle, Proyection pursuit regression, J. Am. Stat. Assoc. 76 (376) (1981) 817–823.
[24] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, Multi-objective cooperative coevolution of artificial neural networks, Neural Networks 15 (10) (2002) 1255–1274.
[25] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, COVNET: a cooperative coevolutionary model for evolving artificial neural networks, IEEE Trans. Neural Networks 14 (3) (2003) 575–596.
[26] S.M. George, B.M. Lund, T.F. Brocklehurst, The effect of pH and temperature on the initiation of growth of *Listeria monocytogenes*, Lett. Appl. Microbiol. 6 (1988) 153–156.
[27] A. Gepperth, S. Roth, Applications of multi-objective structure optimization, Neurocomputing 69 (7–9) (2006) 701–713.
[28] K.P.M. Gysemans, K. Bernaerts, A. Vermeulen, A.H. Geeraerd, J. Debevere, F. Devlieghere, J.F.V. Impe, Exploring the performance of logistic regression model types on growth/no growth data of *Listeria monocytogenes*, Int. J. Food Microbiol. 114 (3) (2007) 316–331.
[29] M.N. Hajmeer, I.A. Basheer, Comparison of logistic regression and neural network-based classifier for bacterial growth, Food Microbiol. 20 (2003) 43–55.
[30] T. Hastie, R.J. Tibshirani, J. Friedman, The elements of statistical learning, in: Data Mining, Inference and Prediction, Springer, Berlin, 2001.
[31] T.J. Hastie, R.J. Tibshirani, Generalized Additive Models, Chapman & Hall, London, 1990.
[32] C. Hervás, F.J. Martínez-Estudillo, Logistic regression using covariates obtained by product-unit neural network models, Pattern Recognition 40 (2007) 52–64.
[33] G.-B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.
[34] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Networks 17 (4) (2006) 879–892.
[35] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[36] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN2004), Budapest, Hungary, 2004.
[37] C.-A. Hwang, M.L. Tamplin, The influence of mayonnaise pH and storage temperature on the growth of *Listeria monocytogenes* in seafood salad, Int. J. Food Microbiol. 102 (2005) 277–285.
[38] A. Ismail, A.P. Engelbrecht, Training products units in feedforward neural networks using particle swarm optimization, in: V.B. Bajic, D. Sha (Eds.), Development and Practice of Artificial Intelligence Techniques, Proceeding of the International Conference on Artificial Intelligence, Durban, South Africa, 1999.
[39] A. Ismail, A.P. Engelbrecht, Global optimization algorithms for training product units neural networks, in: International Joint Conference on Neural Networks IJCNN'2000, Como, Italy, 2000.
[40] A. Ismail, A.P. Engelbrecht, Pruning product unit neural networks, in: Proceedings of the International Conference on Neural Networks, Honolulu, Hawaii, 2002.
[41] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (5) (1993) 26–33.
[42] S. Kirkpatric, C.D.J. Gellat, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[43] C. Kooperberg, S. Bose, C.J. Stone, Polychotomous regression, J. Am. Stat. Assoc. 92 (1997) 117–127.
[44] L.R. Leerink, C.L. Giles, B.G. Horne, M.A. Jabri, Learning with products units, Adv. Neural Networks Process. Syst. 7 (1995) 537–544.
[45] H. Levene, Essays in honor of Harold Hotelling, in: Contributions to Probability and Statistics, 1960, pp. 278–292.
[46] A.C. Martinez-Estudillo, F.J. Martinez-Estudillo, C. Hervas-Martinez, N. Garcia-Pedrajas, Evolutionary product unit based neural networks for regression, Neural Networks 19 (4) (2006) 477–486.
[47] A.C. Martinez-Estudillo, C. Hervas-Martinez, F.J. Martinez-Estudillo, N. Garcia-Pedrajas, Hybridization of evolutionary algorithms and local search by means of a clustering method, IEEE Trans. Syst. Man Cybern. 36 (3) (2006) 534–545.
[48] G.F. Miller, P.M. Todd, S.U. Hedge, Designing neural networks using genetic algorithms, in: Proceedings of the Third International Conference Genetic Algorithms and their Applications, Morgan Kaufmann, San Mateo, CA, 1989.
[49] R.H.J.M. Otten, L.P.P.P. van Ginneken, The Annealing Algorithm, Kluwer, Boston, MA, 1989.
[50] M.A. Potter, K.A. de Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (1) (2000) 1–29.
[51] R. Reed, Pruning algorithms—a survey, IEEE Trans. Neural Networks 4 (1993) 740–747.
[52] M. Rocha, P. Cortez, J. Neves, Evolution of neural networks for classification and regression, Neurocomputing 70 (2007) 2809–2816.
[53] K. Saito, R. Nakano, Numeric law discovery using neural networks, in: Proceedings of the Fourth International Conference on Neural Information Processing (ICONIP97), 1997.
[54] K. Saito, R. Nakano, Extracting regression rules from neural networks, Neural Networks 15 (2002) 1279–1288.
[55] M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, Neural Comput. 14 (2001) 241–301.

[56] R. Setiono, L.C.K. Hui, Use of quasi-Newton method in a feedforward neural-network construction algorithm, IEEE Trans. Neural Networks 6 (1995) 273–277.
[57] I. SPSS, SPSS© para Windows 11.0.1, SPSS Inc., Chicago, IL, 1989–2001.
[58] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1999.
[59] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás-Martínez, JCLEC: a JAVA framework for evolutionary computation, Soft Computing—A Fusion of Foundations, Methodologies and Applications 12 (4) (2007) 381–392.
[60] W. Yan, Z. Zhu, R. Hu, Hybrid genetic /BP algorithm and its application for radar target classification, in: Proceedings of the IEEE National Aerospace Electronics Conference, IEEE Press, Piscataway, NJ, 1997.
[61] X. Yao, Evolving artificial neural network, Proc. IEEE 9 (87) (1999) 1423–1447.
[62] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Trans. Neural Networks 8 (3) (1997) 694–713.
[63] X. Yao, Y. Liu, Making use of population information in evolutionary artificial neural networks, IEEE Trans. Syst. Man Cybern. 28 (3) (1998) 417–425.
[64] Q.-Y. Zhu, A.K. Qin, P.N. Suganthan, G.-B. Huang, Evolutionary extreme learning machine, Pattern Recognition 38 (2005) 1759–1763.

Francisco J. Martínez-Estudillo was born in Villacarrillo, Jaén. He received the B.S. degree in mathematics in 1987 and the Ph. D. degree in Mathematics in 1991, speciality Differential Geometry, both from the University of Granada, Granada, Spain. From 1987 to 2002, he developed his research in non-Euclidean geometry, Lorentz spaces and maximal surfaces. He is currently a professor in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain. His current research interests include structure optimization of neural networks, evolutionary algorithms and multiobjective optimization.

César Hervás-Martínez was born in Cuenca, Spain. He received the B.S. degree in statistics and operating research from the Universidad Complutense, Madrid, Spain, in 1978 and the Ph.D. degree in mathematics from the University of Seville, Seville, Spain, in 1986. He is a Professor with the University of Córdoba in the Department of Computing and Numerical Analysis in the area of computer science and artificial intelligence and an Associate Professor in the Department of Quantitative Methods in the School of Economics. His current research interests include neural networks, evolutionary computation and the modelling of natural systems.

Pedro A. Gutiérrez-Peña was born in Córdoba, Spain, in 1982. He received the B.S. degree in Computer Science from the University of Sevilla, Spain, in 2006. He is currently working toward the Ph.D. Degree at the Department of Computer Science and Numerical Analysis (University of Cordoba, Spain), in the area of Computer Science and Artificial Intelligence. His current interests include neural networks and their applications, evolutionary computation and hybrid algorithms.

Alfonso C. Martínez-Estudillo was born in Villacarrillo, Jaén. He received the B.S. degree in Computer Science in 1995 and the Ph. D. degree in 2005, both from the University of Granada, Granada, Spain. He is currently a lecturer in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain. His current research interests include neural networks, evolutionary algorithms and multiobjective optimization.

2.2. Diseño de Perceptrones Multi-capa utilizando un Algoritmo de Programación Evolutiva y guiando el Tamaño Poblacional mediante Dientes de Sierra Dinámicos - *Designing multilayer perceptrons using a Guided Saw-tooth Evolutionary Programming Algorithm*

- P.A. Gutiérrez, C. Hervás-Martínez, M. Lozano. Designing Multilayer Perceptrons using a Guided Saw-tooth Evolutionary Programming Algorithm. Soft Computing (2009). Aceptado.
  - Estado: Aceptado.
  - Índice de Impacto (JCR 2007): 0.607.
  - Área de conocimiento: *Computer Science, Artificial Intelligence.* Ranking 66/93. Tercer Cuartil.
  - Área de conocimiento: *Computer Science, Interdisciplinary Applications.* Ranking 61/92. Tercer Cuartil.

Fecha: 6 Apr 2009 05:14:45 -0400 [06/04/09 11:14:45 CEST]

De: Editorial Office Soft Computing <bgerla@unisa.it>

Para: i02gupep@uco.es

Asunto: SOCO: Editor Decision - Accept

Ref.: Ms. No. SOCO-D-08-00228R1
Designing Multilayer Perceptrons using a Guided Saw-tooth Evolutionary Programming Algorithm
Soft Computing

Dear Mr. Gutiérrez,

I am pleased to tell you that your work has now been accepted for publication in Soft Computing.

It was accepted on 05-04-2009

Comments from the Editor and Reviewers can be found below.

Thank you for submitting your work to this journal.

With kind regards

Brunella Gerla, Ph.D
Managing Editor (Editorial Office)
Soft Computing

Comments from the Editors and Reviewers:

Reviewer #2: In this second revision the authors front successfully all the comments proposed by the reviewer.

Soft Comput
DOI 10.1007/s00500-009-0429-x

ORIGINAL PAPER

Designing multilayer perceptrons using a Guided Saw-tooth Evolutionary Programming Algorithm

Pedro Antonio Gutiérrez · César Hervás · Manuel Lozano

Abstract In this paper, a diversity generating mechanism is proposed for an Evolutionary Programming (EP) algorithm that determines the basic structure of Multilayer Perceptron classifiers and simultaneously estimates the coefficients of the models. We apply a modified version of a saw-tooth diversity enhancement mechanism recently presented for Genetic Algorithms, which uses a variable population size and periodic partial reinitializations of the population in the form of a saw-tooth function. Our improvement on this standard scheme consists of guiding saw-tooth reinitializations by considering the variance of the best individuals in the population. The population restarts are performed when the difference of variance between two consecutive generations is lower than a percentage of the previous variance. From the analysis of the results over ten benchmark datasets, it can be concluded that the computational cost of the EP algorithm with a constant population size is reduced by using the original saw-tooth scheme. Moreover, the guided saw-tooth mechanism involves a significantly lower computer time demand than the original scheme. Finally, both saw-tooth schemes do not involve an accuracy decrease and, in general, they obtain a better or similar precision.

Keywords Artificial neural networks · Multilayer perceptrons · Evolutionary algorithms · Evolutionary programming · Population reinitializations · Saw-tooth algorithm

This paper is a significant extension of a work originally reported in the Proceedings of the 8th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'07).

P. A. Gutiérrez (✉) · C. Hervás
Department of Computer Science and Numerical Analysis, University of Córdoba, 14071 Cordoba, Spain
e-mail: i02gupep@uco.es

C. Hervás
e-mail: chervas@uco.es

M. Lozano
University of Granada, 18071 Granada, Spain
e-mail: lozano@decsai.ugr.es

1 Introduction

In the past decade, Evolutionary Algorithms (EAs) (Back et al. 1997; Eiben and Smith 2003) and Artificial Neural Networks (ANNs) (Bishop 2006) have been combined as a key research area, providing an interesting platform for simultaneously optimizing both the weights and the architecture of Multilayer Perceptrons (MLPs) (Tsai et al. 2006; Leung et al. 2003; Yao 1999; Maniezzo 1993) while avoiding the shortcomings of traditional Backpropagation (Angeline et al. 1994). Evolutionary Programming (EP) was originally proposed by Fogel et al. (1966) and it is one of the most renowned branches of EAs whose main variation operator is mutation. Its emphasis on the behavioural link between parents and offspring can increase the efficiency of ANN's evolution, preventing the mapping problem (Angeline et al. 1994) by representing the individuals at the species level (Fogel 1995). This reason together with the potential disadvantages of the recombination operator in ANNs has contributed in the development of several EP algorithms for evolving ANNs (Palmes et al. 2005; Yao 1997; Yao and Liu 1997; Angeline et al. 1994; McDonnell and Waagen 1993; Fogel et al. 1990).

In general, an EA must be able to maintain a trade-off between two different issues that affect its performance: *exploitation* (related to the competition mechanism

Journal : Large 500 | Dispatch : 27-4-2009 | Pages : 15
Article No. : 429 | ☐ LE | ☐ TYPESET
MS Code : SOCO-D-08-00228 | ✓ CP | ✓ DISK

established in the EA and, therefore, to the selective pressure incorporated through the selection operator) and *diversity* (related to the number of fitness values, genotypes or phenotypes that evolve in the population). *Variable population size* (Smith 1993; Koumousis and Dimou 2002) is an interesting method in EAs for improving capacity to refine the solutions obtained and to reduce sensitivity in selecting the parameters. Different *population reinitialization* methodologies have also been proposed in EA literature [e.g. alternative mutation strategies (Cobb and Grefenstette 1993), the CHC algorithm (Eshelman 1991) or the micro-Genetic Algorithms, μ-GAs (Krishnakumar 1989)], all of them aimed at attempting to increase the diversity of the population and to avoid a premature convergence. One methodology combining the effects of the two strategies is the saw-tooth GA, proposed by Koumousis and Katsaras (2006), which follows a saw-tooth population scheme with a specific *amplitude* and *period* of variation. In each period, the population size decreases linearly and, at the beginning of the next period, randomly generated individuals are appended to the population. The saw-tooth scheme enhanced the overall performance of a standard GA in a variety of benchmark problems and a problem generator (Koumousis and Katsaras 2006).

In previous works (Martínez-Estudillo et al. 2008; Hervás et al. 2006), an EP algorithm has been proposed for the evolution of the weights and structure of ANN classifiers. This algorithm has a very strong selective pressure and considers a very high population size in order to avoid premature convergence, which allows to explore a wide enough area of the search space. The problem is that the computational cost is also very high, making difficult the application of the algorithm over datasets with a high number of patterns and/or characteristics. With the motivation of increasing the diversity of the algorithm without using a high population size and computational time demand, the present paper derives a new EP methodology to design the structure and weights of MLP classifiers by the application of the saw-tooth population size scheme (Saw-tooth Evolutionary Programming, SEP). The competitive results obtained by the Saw-tooth GA in multimodal function optimization (Koumousis and Katsaras 2006) suggest that this computational cost reduction can be achieved without losing the accuracy previously reported by the EP algorithm. Moreover, to the best of our knowledge, this scheme has not previously been considered in an EP algorithm. Among the different existing ANN architectures, the MLP model has been chosen because it is one of the most standard and successful model of neural network in the context of pattern recognition (Bishop 2006).

The main difficulty associated with the saw-tooth scheme is the selection of saw-teeth parameters (*amplitude* and *period* of variation) that affects the performance of the algorithm. A possibility for reducing this sensitivity is the development of a dynamic scheme, in which, the restarts (i.e. the transitions between one saw-tooth and the next) could be performed taking into consideration the degree of diversity in each generation of evolution. In order to evaluate this diversity, the fitness variance of the individuals can be considered. When a high population size is involved, it is more suitable to use only the variance of the best individuals in the population, given that the variance of the complete population can take values in a more wide range from one generation to another and its control is more difficult. A dynamic version of the SEP algorithm is proposed in this work, in which, we guide the period of each saw-tooth according to the variance of the fitness of the best individuals in the population (Guided Saw-tooth Evolutionary Programming, GSEP). The underlying idea is that reinitializations should be performed when the best individuals have converged and no further optimization is being achieved, so randomly generated individuals can help the algorithm to explore new regions in the search space.

In order to test the performance and suitability of the two proposed methodologies (SEP and GSEP), a comparative study using the original EP algorithm with constant population size and the SEP and GSEP algorithms has been applied to ten datasets taken from the UCI repository (Asuncion and Newman 2007).

The rest of the paper is organized as follows: Sect. 2 analyzes previous works related to the diversity-convergence trade-off in EAs and the main ideas associated with the original saw-tooth GA, which is the starting point of this work; Sect. 3 is dedicated to a description of the selected EP algorithm; Sect. 4 describes the application of the saw-tooth scheme in the EP algorithm; the aim of Sect. 5 is to describe the proposed variance guided version of the saw-tooth scheme; Sect. 6 explains the experiments carried out; and finally, Sect. 7 shows the conclusions of our work and outlines some future research lines.

2 The diversity-convergence trade-off: related works

The performance and the efficiency of an EA is related to two different factors:

- *Diversity* (*exploration*): algorithm capability of avoiding premature convergence, i.e. being trapped in a search area which does not contain the global optimum.
- *Convergence* (*exploitation*): algorithm capability of directing search efforts to the most promising areas, increasing the accuracy of the solutions obtained.

The main problem is that these two objectives are opposite: increasing the selective pressure results in

Journal : Large 500 | Dispatch : 27-4-2009 | Pages : 15
Article No. : 429 | ☐ LE | ☐ TYPESET
MS Code : SOCO-D-08-00228 | ✓ CP | ✓ DISK

decreasing diversity (Goldberg and Deb 1990), while keeping the diversity in the population can result in decreasing accuracy (Whitley 1989). The EAs literature consistently cites the importance of maintaining diversity as being crucial in avoiding premature convergence toward local optima (Eshelman and Schaffer 1991; Goldberg 1989; Chaiyaratana et al. 2007). Moreover, several efforts have been made to create a set of operators capable of maintaining the convergence-diversity trade-off (Herrera and Lozano 1996; Lozano et al. 2008), to derive specialized operators controlling and assisting the selection procedure [e.g. crowding (De Jong 1975), deterministic crowding (Mahfoud 1992), sharing (Goldberg and Richardson 1987) or fitness uniform selection (Hutter and Legg 2006), among others] and to reintroduce genetic material [e.g. random immigrants (Cobb and Grefenstette 1993) and mass extinction (Greewood et al. 1999)].

Diversity measures are traditionally used to analyze and to control EAs (Ursem 2002). In Herrera et al. (1996), an adaptive real-coded genetic algorithm is designed and discussed based on the use of fuzzy logic controllers. The Diversity Control Oriented Genetic Algorithm uses a diversity measure based on Hamming distance to calculate a survival probability for individuals (Shimodaira 2001). Another approach (Oppacher and Wineberg 1999) is based on a containment factor between two subpopulations, which is based on Hamming distances between all members of the two populations. The Forking GA uses specialized diversity measures to turn a subset of the population into a subpopulation (Tsutsui et al. 1997).

Analysis of selection schemes such as those by Prügel-Bennett and Shapiro (1994, 1997) and Rattray (1996) show that the change in mean fitness at each generation is a function of the population fitness variance. At each generation, this variance is reduced due to two factors. One factor is selection pressure producing multiple copies of fitter population members while the other factor is independent of population member fitness and is due to the stochastic nature of genetic operators (genetic drift). Moreover, a method for calculating genetic drift in terms of changing population fitness variance has been presented (Rogers and Pruegel-Bennett 1999). All these studies suggest that the fitness variance can be considered a suitable measure for estimating the diversity of the population in each generation.

Population size is one of the most important parameters affecting the robustness and computational efficiency of EAs. Small population sizes may result in premature convergence to nonoptimal solutions, whereas large population sizes require a considerable increase in computational effort. Although some authors try to estimate an optimal population size regarding the complexity of the problem (Goldberg et al. 1992), *variable population size* is a more interesting alternative, improving the EA capability of obtaining better solutions and reducing sensitivity in the selection of different parameters. For example, Smith (1993) proposed an algorithm that adjusts the population size based on the probability of selection error, and Koumousis and Dimou (2002) applied a sinusoidal oscillating population size for structural problems.

On the other hand, several methods using a constant population size have been proposed in the literature that attempt to increase the diversity of the population and avoid premature convergence, including, among others, alternative mutation strategies (Cobb and Grefenstette 1993), the well-known Eshelman's CHC algorithm (Eshelman 1991) or the micro-Genetic Algorithm or μ-GA, suggested by Goldberg (1989) and then implemented by Krishnakumar (1989). Both CHC and μ-GA algorithms are based on *periodical reinitializations* of population adding randomly generated individuals when the diversity drops below a threshold.

2.1 The Saw-tooth Genetic Algorithm

Combining the effects of population size variation and reinitialization alters the evolution dynamics of a GA in a way that is expected to enhance overall performance (De Jong et al. 2001). The saw-tooth GA (Koumousis and Katsaras 2006) utilizes a variable population size following a periodic scheme with a mean population size $\overline{n}$, which corresponds to the constant population size GA having the same computational cost. Moreover, the scheme is characterized by amplitude D and period of variation T. Thus, at a specific generation t, the population size n_t is determined as:

$$n_t = \text{int}\left\{\overline{n} + D - \frac{2D}{T-1}\left[t - T\text{int}\left(\frac{t-1}{T}\right) - 1\right]\right\} \qquad (1)$$

where int$\{\cdot\}$ is the floor function. Therefore, $n_1 = \overline{n} + D, n_T = \overline{n} - D, n_{T+1} = \overline{n} + D$, etc. The selection of the $\overline{n}, T$ and D parameters affects the performance of the algorithm. For amplitude, if $D = 0$, regardless of the period t, the algorithm is reduced to a constant population size GA. For higher amplitude D values, the population size decreases with constant decay and reinitialization is enforced every T generations. The effect of population reinitialization is more drastic as the amplitude D increases from 0 to $\overline{n} - 1$. Moreover, the selection of the period T is critical as it controls the duration of the decay process before reinitialization occurs.

In the cited paper, advisable values of the saw-tooth t, D and $\overline{n}$ parameters are obtained experimentally for unimodal and multimodal problems. As a conclusion, the more adequate normalized amplitude $D/\overline{n}$ is estimated from 0.9 to 0.96 and the advisable normalized period ranges from

Journal : Large 500 | Dispatch : 27-4-2009 | Pages : 15
Article No. : 429 | ☐ LE | ☐ TYPESET
MS Code : SOCO-D-08-00228 | ✓ CP | ✓ DISK

$T/\bar{n} = 0.5$ to 0.7 for multimodal optimization problems and a standard real coded GA (Koumousis and Katsaras 2006). Using these guidelines, Koumousis and Katsaras carried out a comparison to a standard GA and a μ-GA over different multimodal test functions, concluding that the saw-tooth GA is more robust than the standard GA and the μ-GA, for 15 out of the 18 selected functions.

3 Evolutionary Programming Algorithm

In this section, we present the base algorithm used to estimate the parameters and the structure of the MLP neural network classifiers. Given its characteristics, the algorithm falls into the class of EP (Fogel et al. 1966). The main objective of this EP algorithm is to design a neural network with the better structure and weights possible for each classification problem tackled. The search begins with an initial population of MLPs, to which a population-update algorithm is applied in each iteration. The algorithm shares many characteristics and properties with other previous algorithms (Angeline et al. 1994; Bornholdt and Graudenz 1992; Yao and Liu 1997). Individuals are subject to the operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving ANNs (Angeline et al. 1994). The basic steps applied in each iteration of the EP algorithm are represented in Fig. 1, where CCR is the Correctly Classified Rate or classification accuracy. The algorithm includes the application of two different mutations, a double elitism maintenance, the sort of population and finally the replication of the best individuals. Using this basic iteration loop, the general framework of the EP algorithm is shown in Fig. 2, where G is the maximum number of generations and N_I is the maximum number of individuals of the algorithm (which is constant in this case). The initialization process (Fig. 2, step 2) ensures a good enough initial population.

The EP algorithm was implemented using the Evolutionary Computation framework JCLEC (Ventura et al. 2008) (http://jclec.sourceforge.net) and is part of the the non-commercial JAVA tool named KEEL (Alcala-Fdez et al. 2008) (http://www.keel.es). It was originally designed

```
1: EP Iteration:
2: Apply parametric mutation to the best 10% of individuals and obtain the fitness of mutated individuals
3: Apply structural mutation to the remaining 90% of individuals and obtain the fitness of mutated individuals
4: Add the best fitness individual and the best CCR individual from the last generation (double elitism)
5: Rank the individuals with respect to their fitness
6: The best 10% of population individuals are replicated and substitute the worst 10% of individuals
```

Fig. 1 Basic EP iteration

```
1: EP Algorithm:
2: Generate a random population of size $10N_I$ and select the best $N_I$ individuals
3: $t \leftarrow 0$
4: while $t < G$ do
5:     Apply EP Iteration
6:     $t \leftarrow (t+1)$
7: end while
8: Select the best CCR individual and the best Entropy individual in the final population
9: return Best CCR and Entropy individuals
```

Fig. 2 EP training algorithm framework

for the evolution of Product Unit Neural Networks, being adapted in this paper to MLP neural networks in order to better evaluate the effect of the different diversity enhancement mechanisms over a standard ANN model.

In the following subsections, we describe some of the most important characteristics of the algorithm. First, the ANN model and the fitness function used by the algorithm are presented. A brief description of the representation of the neural networks is also given and the elitism and mutation operators are briefly described. Finally, some conclusions about the selective pressure of the algorithm are derived. More extended details of the algorithm can be consulted in Martínez-Estudillo et al. (2008), Martínez-Estudillo et al. (2006a, b) and Hervás et al. (2006).

3.1 MLP classifiers

We consider standard feed forward MLP neural networks with one hidden layer, M hidden nodes in the hidden layer and J output nodes, where J is the number of classes of the problem. A scheme of the MLP models considered in this work is given in Fig. 3. We interpret the outputs of the neurons in the output layer from the point of view of probability which considers the softmax activation function given by the following expression:

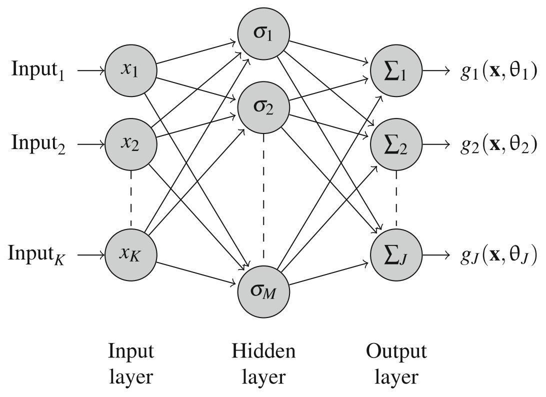


Fig. 3 MLP model with softmax activation function in the output layer

$$g_l(\mathbf{x}, \theta_l) = \frac{\exp f_l(\mathbf{x}, \theta_l)}{\sum_{j=1}^{J} \exp f_j(\mathbf{x}, \theta_j)}; 1 \le l \le J$$

where $g_l(\mathbf{x}, \theta_l)$ is the probability that pattern $\mathbf{x}$ has of belonging to class l, $\theta_l = (\beta_0^l, \ldots, \beta_M^l, \mathbf{w}_1, \ldots, \mathbf{w}_M)$ is the vector of weights of the output node l, $\mathbf{w}_j = (w_{j0}, \ldots, w_{jK})$ is the vector of input weights of the hidden node J and $f_l(\mathbf{x}, \theta_l)$ is the output of output node l for pattern $\mathbf{x}$ given by:

$$f_l(\mathbf{x}, \theta_l) = \beta_0^l + \sum_{j=1}^{M} \beta_j^l \sigma\left(w_{j0} + \sum_{i=1}^{K} w_{ji} x_i\right)$$

where $\sigma(\cdot)$ is the sigmoidal activation function.

The classification rule $C(\mathbf{x})$ of the MLP coincides with the optimal Bayes' rule. In other words, a pattern $\mathbf{x}$ should be assigned to the class which has the maximum probability:

$$C(\mathbf{x}) = \arg\max_l g_l(\mathbf{x}, \theta_l), \quad \text{for } l = 1, 2, \ldots, J$$

On the other hand, due to the normalization condition, we have $\sum_{l=1}^{J} g_l(\mathbf{x}, \theta_l) = 1$. In this way, the probability for one of the classes (the last one, in our case) can be obtained from the other probabilities. This allows us to consider $f_J(\mathbf{x}, \theta_J) = 0$, thus reducing the number of parameters to be adjusted.

3.2 Fitness function

We define the CCR or classification accuracy by:

$$\mathrm{CCR} = (1/N) \sum_{n=1}^{N} (I(C(\mathbf{x}_n) = \mathbf{y}_n)$$

where $I(\cdot)$ is the zero-one loss function, $\mathbf{y}_n$ is the desired output for pattern n and N is the total number of patterns in the dataset. A good classifier tries to achieve the highest possible CCR in a given problem. However, the CCR is a discontinuous function, which makes convergence very difficult in neural network optimization. This is the reason why the function used in the EP algorithm to obtain the fitness of individuals is a strictly decreasing transformation of the cross entropy error function [as previously considered in Martínez-Estudillo et al. (2008)], which is given by the following expression:

$$A(g) = \frac{1}{1 + l(\theta)} \tag{2}$$

where $\theta = (\theta_1, \ldots, \theta_J)$ and $l(\theta)$ is the Entropy error function of the model and is obtained as:

$$l(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J} y_n^{(l)} \log g_l(\mathbf{x}_n, \theta_l)$$

3.3 Representation of the individuals

The algorithm evolves architectures and connection weights simultaneously, each individual being an ANN. The neural networks are represented using an object oriented approach and the algorithm deals directly with the ANN phenotype. Each connection is specified by a binary value indicating if the connection exists, and a real value representing its weight. As the crossover is not considered, this object-oriented representation does not assume a fixed order between the different hidden nodes.

In order to define the topology of the neural networks, two parameters are considered: $M_{\min}$ and $M_{\max}$. They correspond, respectively, to the minimum and maximum number of hidden nodes in the whole evolutionary process. The number of hidden nodes of the model M is bounded by these parameters, $M_{\min} \le M \le M_{\max}$.

3.4 Double elitism

As mentioned previously, fitness is a decreasing transformation of the Entropy error because the Entropy function is continuous and helps the algorithm to enhance the classifiers more gradually than with the use of CCR. However, the relationship between CCR and Entropy error strongly depends on the dataset structure. Hence, regarding experimental results, using Entropy elitism is more suitable for some datasets to result in a higher test accuracy, but maintaining the best CCR individual can be more appropriate for some other datasets. For this reason, the EP algorithm maintains both the best CCR and the best fitness individuals as solutions (Fig. 1, step 4), the best approach for each problem being difficult to ascertain a priori. In order to reduce the additional computational cost of the CCR calculation, the best CCR individual is obtained only from the best 10% of individuals (previously sorted using the Entropy error function).

3.5 Parametric and structural mutations

The severity of both structural and parametric mutations depends on the temperature $T(g)$ of the neural network model, defined by $T(g) = 1 - A(g)$, $0 \le T(g) \le 1$, where $A(g)$ is the fitness function defined in Eq. 2.

Parametric mutation (Fig. 1, step 2) is accomplished for each weight $w \in \theta$ of the neural network with Gaussian noise $w(t+1) = w(t) + \xi(t)$, where $\xi(t)$ represents a one dimensional normally distributed random variable, $N(0, \alpha T(g))$. The α value is updated throughout the evolutionary process, applying the simplest heuristic 1/5 success rule of Rechenberg (1973). The weights are sequentially mutated, hidden node after hidden node, and a standard simulated annealing process is applied to accept or reject the modifications in each node.

On the other hand, *structural mutation* (Fig. 1, step 3) implies a modification in the neural network structure and allows explorations of different regions in the search space

Journal : **Large 500**	Dispatch : **27-4-2009**	Pages : **15**
Article No. : **429**	☐ LE	☐ TYPESET
MS Code : **SOCO-D-08-00228**	✓ CP	✓ DISK

while helping to keep up the diversity of the population. There are five different structural mutations: node deletion, connection deletion, node addition, connection addition and node fusion. These five mutations are applied sequentially to each network, each one with a given probability, which is given by the temperature of the neural net $T(g)$ (Martínez-Estudillo et al. 2006b).

For further details about parametric and structural mutations, see Martínez-Estudillo et al. (2008, 2006b).

3.6 Selective pressure

The EP algorithm analyzed has a very strong selective pressure mainly due to two reasons:

- The selection operator replaces the worst 10% of population individuals with the best 10% (Fig. 1, step 6), which enables the networks with a better classification capability to quickly breed.
- Parametric mutation incorporates a simulated annealing process which leaves diversity generation in the hands of structural mutation, when the changes that involve a lower fitness are not accepted.

In order to avoid premature convergence, the EP algorithm considers a very high population size ($N_I = 1{,}000$), which allows to explore a wide enough area of the search space. This high population size is necessary for obtaining results with an acceptable accuracy level, as will be shown in the experimental section of this paper. The problem is that the computational cost is also very high, hindering the application of the algorithm over datasets with a high number of patterns and/or characteristics. For this reason, in the following sections, two different population size schemes will be presented, both attempting a trade-off between convergence and diversity, combining a variable population size and periodical reinitializations of the population.

4 Saw-tooth Evolutionary Programming Algorithm

The competitive results obtained by the saw-tooth GA in multimodal function optimization (Koumousis and Katsaras 2006) have encouraged us to adapt this scheme to the previously presented EP algorithm, defining what we call saw-tooth Evolutionary Programming (SEP). In this way, we try to reduce the computational cost of the algorithm, without losing the efficiency it has previously achieved designing neural networks classifiers (Martínez-Estudillo et al. 2008; Hervás et al. 2006). Instead of using Eq. 1 and with the motivation of being able to consider a dynamic behaviour for the guided version of the scheme, the population size in each generation T is calculated from the last generation $t-1$ population size as:

$$n_t = n_{t-1} - N_S$$

N_S being the saw-tooth slope and $n_0 = N_I$ being the number of individuals in the initial population. In this way, the saw-tooth shape depends on the saw-tooth slope N_S, which also depends on the parameters D and T.

The parameter D is calculated according to the guidelines described in Sect. 2.1. For the parameter T, we estimate its value from the maximum number of generations G and a parameter r that defines the maximum number of restarts, $T = \mathrm{int}(G/r)$. The use of this new parameter r is justified, because a better control of the increase in the diversity of the algorithm is achieved by defining the maximum number of restarts (that is, the maximum number of saw-tooth oscillations) than by defining the amplitude T of each saw-tooth. With these two parameters, the saw-tooth slope is obtained as $N_S = (2D)/T$. Finally, a minimum number of individuals N_M in the population can be derived from the amplitude D of the saw-teeth, $N_M = N_I - 2D$.

The EP algorithm is easily adapted to this scheme, adding a population size updating step in each generation and resulting in the algorithm presented in Fig. 4, where the basic EP iteration is the same as introduced in Fig. 1. Double elitism ensures that the best individuals are copied to the subsequent population after the population reinitialization, when the good solutions may constitute a small proportion of the population.

The proposed SEP algorithm uses two parameters for controlling the population size: N_M and r. Figure 5 is an example of the saw-tooth scheme, including these parameters. In the figure, it is shown how N_M and r parameters can be used instead of T and D, which, in our opinion, provides a better control of the increase in diversity once the maximum number of generations G is defined.

```
1:  SEP Algorithm:
2:  Generate a random population of size $10N_I$ and select the best $N_I$
    individuals
3:  $t \leftarrow 0$; $n_0 \leftarrow N_I$
4:  while $t < G$ do
5:      if $(n_{t-1} - N_S) \geq N_M$ then
6:          $n_t = n_{t-1} - N_S$ {Decrement population size}
7:      else
8:          $n_t = N_I$ {Perform a reinitialization}
9:      end if
10:     Apply EP Iteration
11:     $t \leftarrow (t+1)$
12: end while
13: Select the best $CCR$ individual and the Entropy individual in the
    final population
14: return Best $CCR$ and Entropy individuals
```

Fig. 4 SEP training algorithm framework

Journal : Large 500 | Dispatch : 27-4-2009 | Pages : 15
Article No. : 429 | ☐ LE | ☐ TYPESET
MS Code : SOCO-D-08-00228 | ✓ CP | ✓ DISK

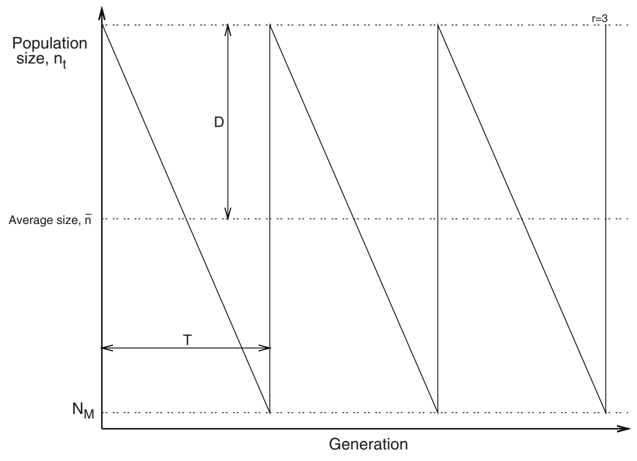


Fig. 5 Population size using the SEP scheme

5 Guided Saw-tooth Evolutionary Programming Algorithm

The previously defined scheme reduces the computational cost of the EP algorithm without losing its efficiency, as will be shown later in the experimental section. However, the reinitializations are always performed in the same generations, regardless of the specific situation of the population at each stage in the evolution. Moreover, the SEP algorithm is very sensitive to the values of the parameters r and N_M. A possibility for reducing this sensitivity is the development of a dynamic scheme, in which, the restarts (i.e. the transitions between one saw-tooth and the next) could be performed taking into consideration the degree of diversity in each generation of evolution. In this way, the algorithm can be dynamically adapted to the specific situation of the current solutions examined within the search space, and the reinitializations can be brought forward, resulting in a more efficient algorithm because fewer generations are needed to obtain similar results.

With this motivation, our second proposal consists of enhancing the saw-tooth scheme considering the variance of the best individual distributions, forcing the beginning of a new saw-tooth oscillation when the best individuals have converged and the performance of the algorithm decreases. Consequently, if the variance of the fitness of the best individuals has significantly decreased, the population size falls. Otherwise a restart is performed, new individuals being added to the current population. This methodology has been called Guided Saw-tooth Evolutionary Programming (GSEP). The reason why only the best individuals have been examined to guide the algorithm is that the variance of the complete population can take values in a wider range from one generation to another and its control is more difficult, specially when a high population size is involved.

In this way, the population size updating step will be slightly modified, performing reinitializations not only when the saw-tooth has achieved the minimum population size value, N_M, but also when the population is considered to have converged. The condition that must be fulfilled in order to force the restart of the population is that *the difference of variance of the fitness of the best individuals between two consecutive generations decreases by a specific percentage (λ) with respect to the previous generation's variance*, previously establishing the most appropriate distribution to represent the fitness of the best individuals. As a first approximation, the best individual fitness values have been characterized using two distinct distributions, uniform distribution and normal distribution:

- Considering the uniform distribution, the variance is estimated by:

$$\hat{\sigma}^2 = \frac{1}{12}(f_{\max} - f_{\min})^2$$

$f_{\max}$ and $f_{\min}$ being the maximum and minimum fitness value of the best individual set B.

- Considering the normal distribution, the variance is estimated as:

$$\hat{\sigma}^2 = \frac{1}{|B|-1}\sum_{i=1}^{|B|}(f_i - \bar{f})^2$$

f_i being the fitness of the individual i of the set of best individuals B and $\bar{f}$ the mean fitness value of all individuals in B.

It is important to establish the number of individuals in the best individual set ($|B|$) whose variance will be studied to guide the algorithm. This number of individuals is constant during the evolution process and has been fixed to the minimum number of individuals ($|B| = N_M$), since the algorithm always maintain at least these individuals.

The EP algorithm can also be easily adapted to this guided scheme, adding a population size updating step in each generation and resulting in the algorithm presented in Fig. 6, where the basic EP iteration is the same introduced in Fig. 1. In order to limit computational cost, the number of individuals that will be added in each restart is equal to half of the number of missing individuals (Fig. 6, step 9), since a lower number of individuals is needed as the evolution progresses.

The GSEP algorithm uses the percentage of variance (λ) and the same two parameters than SEP (N_M and r) for controlling the population size, but it is less sensitive to their values, given its dynamic behaviour. Figure 7 is an example of the guided saw-tooth scheme. The algorithm results in a lower population size, converging to the minimum in the final generations. In this way, the computational cost is reduced.

```
 1: GSEP Algorithm:
 2: Generate a random population of size $10N_\mathrm{I}$ and select the best $N_\mathrm{I}$
    individuals
 3: $t \leftarrow 0$; $n_0 \leftarrow N_\mathrm{I}$; $s \leftarrow 0$
 4: while $t < G$ do
 5:     if $s > r$ then
 6:         $n_t = N_\mathrm{M}$ {Keep minimum population size}
 7:     else if $(\widehat{\sigma}_t^2 < \widehat{\sigma}_{t-1}^2$ and $(\widehat{\sigma}_{t-1}^2 - \widehat{\sigma}_t^2) < \lambda \widehat{\sigma}_{t-1}^2)$
          or $(n_{t-1} - N_\mathrm{S}) \leq N_\mathrm{M}$ then
 8:         $s \leftarrow (s+1)$
 9:         $n_t = n_{t-1} + \frac{N_\mathrm{I} - n_{t-1}}{2}$ {Perform a reinitialization}
10:     else
11:         $n_t = n_{t-1} - N_\mathrm{S}$ {Decrement population size}
12:     end if
13:     Apply EP Iteration
14:     $t \leftarrow (t+1)$
15: end while
16: Select the best CCR individual and the best Entropy individual in
    the final population
17: return  Best CCR and Entropy individuals
```

Fig. 6 GSEP training algorithm framework

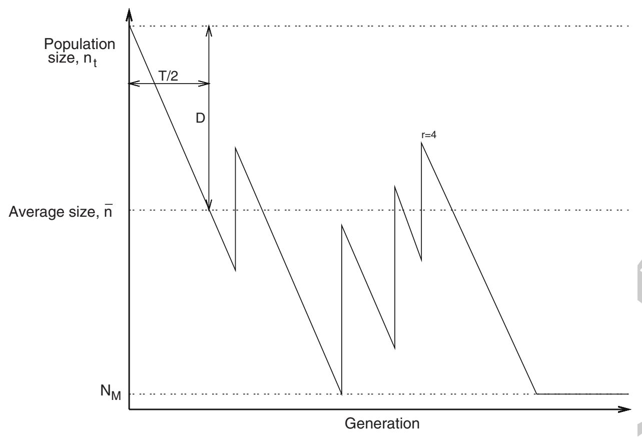


Fig. 7 Population size using the Guided Saw-tooth Evolutionary Programming scheme

6 Experiments

The proposed GSEP and SEP algorithms are applied to ten datasets taken from the UCI repository (Asuncion and Newman 2007), to test their overall performance as compared to the EP algorithm. The selected datasets include both binary classification problems (German, Heart-statlog, Ionosphere, KrVsKp and Pima) and multiclass classification problems (Anneal, Balance, Glass, Vehicle and Yeast) with different numbers of instances, features and classes (see Table 1). Following the guidelines of Pretchel (1994), the experimental design was conducted using a holdout cross-validation procedure, with 30 runs and 75% of instances for the training set and 25% for the test set.

Regarding the common configuration parameters of the different experiments, the weight range for the initialization of input connections is $[-5, 5]$ and for output connections is $[-10, 10]$, the normalization range for input variables is $[0.1, 0.9]$, the minimum and maximum number of nodes to be added or deleted in structural mutations is $[1, 2]$, the percentage of input and output links to be added or deleted in structural mutations are 30 and 5%, respectively, and the initial α value for parametric mutations is 0.5. This α parameter defines, together with the temperature of the neural nets, the variances of the normal distributions used for updating the weights.

Some other parameters are specific for each dataset and include: the maximum number of generations (G), minimum (M_min) and maximum (M_max) number of hidden nodes in the whole evolutionary process, maximum number of restarts (r) and percentage of variance to consider that a restart must be performed (λ). For all the parameters, except the number of hidden nodes, we have considered a small, medium and high value: $G \in \{100, 300, 1,000\}, r \in \{1, 4, 8\}$ and $\lambda \in \{0.0010, 0.0050, 0.0075\}$. The better experimental values of these parameters are summarized in Table 2.

Table 1 Characteristics of the different datasets: total number of instances, number of instances of the training and test sets, number of real (R), binary (B) or nominal (N) features, number of inputs (#In) and number of classes or outputs (#Out)

Dataset	#Instances			#Features				
	Total	Train.	Test	R	B	N	#In	#Out
Anneal	898	674	224	9	–	29	59	6
Balance	625	469	156	4	–	–	4	3
German	1,000	750	250	6	3	11	61	2
Glass	214	161	53	9	–	–	9	6
Heart	270	202	68	6	1	6	13	2
Ionos.	351	264	87	33	1	–	34	2
KrVsKp	3,196	2,397	799	35	1	–	38	2
Pima	768	576	192	8	–	–	8	2
Vehicle	946	635	211	18	–	–	18	4
Yeast	1,484	1,113	371	8	–	–	8	10

All nominal variables are transformed to binary variables

Table 2 Specific configuration parameter values

Dataset	G	$[M_\mathrm{min}, M_\mathrm{max}]$	r	λ
Anneal	300	[5,7]	1	0.0075
Balance	300	[3,5]	4	0.0050
German	100	[2,4]	1	0.0075
Glass	300	[7,9]	1	0.0075
Heart	100	[1,2]	1	0.0075
Ionos	300	[3,5]	1	0.0075
KrVsKp	1,000	[7,8]	4	0.0010
Pima	100	[1,3]	1	0.0050
Vehicle	1,000	[6,8]	4	0.0010
Yeast	1,000	[12,16]	8	0.0050

Journal : Large 500 | Dispatch : 27-4-2009 | Pages : 15
Article No. : 429 | ☐ LE | ☐ TYPESET
MS Code : SOCO-D-08-00228 | ✓ CP | ✓ DISK

Considering the guidelines proposed by Koumousis and Katsaras (2006) and summarized in Sect. 2.1 and an estimated mean population size $\overline{n}$ of 500, D parameter value should range between 480 and 490. Consequently, we have considered two different values of the N_{M} parameter, 20 and 40, for SEP and GSEP algorithms. The period T of saw-teeth is determined by the maximum number of restarts r and maximum number of generations G, both specified in Table 2. For GSEP experiments, r value has been increased in one additional restart, in order to avoid an excessive number of generations with a minimum population size.

For each dataset, a total of eight different variants of the population schemes are run: EP1000 and EP500 (Evolutionary Programming algorithm with constant population size, $N_{\mathrm{I}} = 1{,}000$ and $N_{\mathrm{I}} = 500$), SEP20 and SEP40 (Saw-tooth Evolutionary Programming with two different minimum numbers of individuals, $N_{\mathrm{M}} = 20$ and $N_{\mathrm{M}} = 40$) and GSEP20N, GSEP40N, GSEP20U and GSEP40U (Guided Saw-tooth Evolutionary Programming with two different minimum numbers of individuals, $N_{\mathrm{M}} = 20$ and $N_{\mathrm{M}} = 40$ and considering the Normal and Uniform distributions of the fitness of the best individuals). These experiments constitute a significant extension of those originally reported in Gutiérrez et al. (2007), considering additional datasets, statistical tests, the EP500 methodology and some preliminary studies about the training efficiency of the algorithms, which will be presented below.

In order to evaluate the performance of the proposed methodology from a graphical perspective, two preliminary experiments have been performed without considering the maximum number of generations G. In this way, the maximum and mean training fitness of the MLPs of the population have been plotted versus the number of fitness evaluations of each algorithm in Figs. 8 and 9. Using these representations, the efficiency of the algorithms is compared when a similar computational cost is considered, since the number of evaluations increases when the algorithms that tend to maintain a higher number of individuals are used. The depicted values correspond to the average of this maximum and mean fitness over 30 runs for the German and Pima datasets, using the EP1000 and EP500

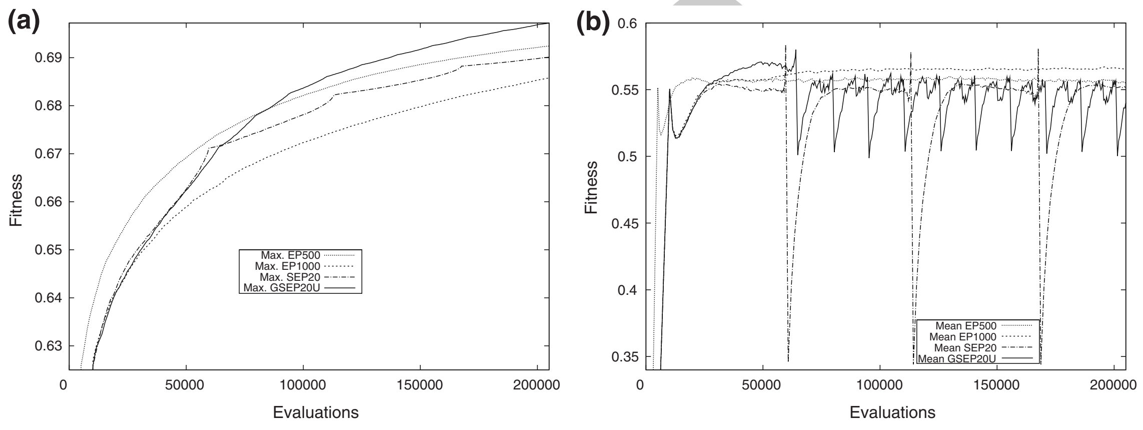


Fig. 8 Maximum (**a**) and mean (**b**) fitness of the population throughout the evaluations of the different proposed algorithms for the German dataset

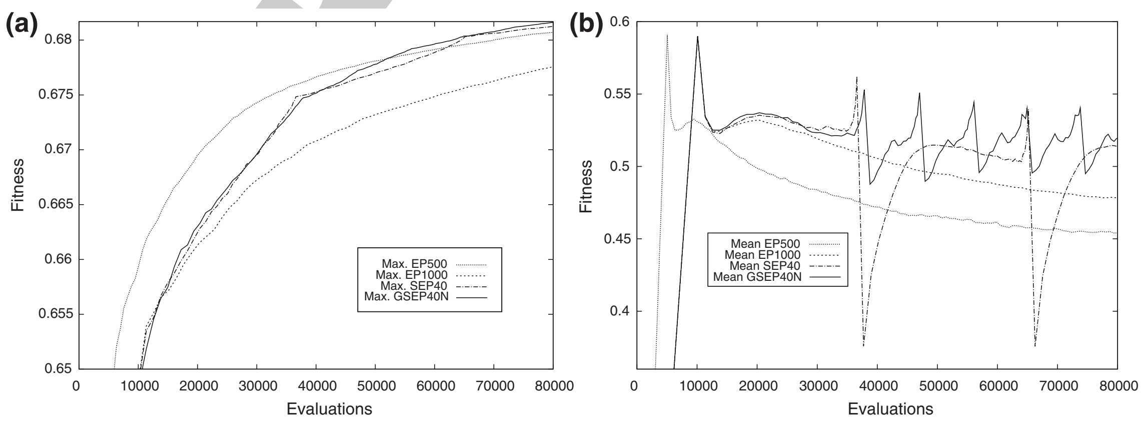


Fig. 9 Maximum (**a**) and mean (**b**) fitness of the population throughout the evaluations of the different proposed algorithms for the Pima dataset

Journal : **Large 500** Dispatch : **27-4-2009** Pages : **15**
Article No. : **429** ☐ LE ☐ TYPESET
MS Code : **SOCO-D-08-00228** ✓ CP ✓ DISK

algorithms and the best performing variant for SEP (SEP20 or SEP40) and GSEP (GSEP20N, GSEP40N, GSEP20U or GSEP40U). Saw-tooth algorithms (SEP and GSEP) achieved a better mean and maximum fitness than constant population size algorithms. Although in the initial evaluations, the maximum fitness achieved by EP500 is higher, the dynamic GSEP algorithm achieves an overall better final fitness for both datasets. Moreover, mean fitness of the population is higher using the GSEP scheme than using the SEP for both datasets.

Once the performance of the algorithm has been compared from the point of view of their training fitness efficiency, all the algorithms are run using the same number of generations G for each dataset, in order to compare the differences of their test classification accuracies and their computational cost. These results will be discussed from the the points of view of accuracy and efficiency in the following subsections.

6.1 Accuracy analysis

As previously stated, the main objective of the saw-tooth algorithms is to introduce a high degree of diversity without using a high population size while obtaining better or similar accuracy. Once all the algorithms have been run 30 times with the same values for all the parameters except for population size and reinitialization dynamics, the best training CCR and Entropy individuals in the last generation are selected. Table 3 shows the mean value and the standard deviation of the training and test CCR of the best models obtained throughout the 30 runs of the experiments. These results are shown for the EP1000 and EP500 algorithms and for the best performing variant of SEP and GSEP algorithms. Moreover, as the algorithms returned both the best CCR and Entropy individuals, Table 3 only includes the most efficient approach (Entropy or CCR) in each given problem, which is always the same for all the different variants of the algorithms, since the suitability of each elitism depends on the characteristics of the datasets.

The algorithm that yields better test results in each dataset is represented in bold print. The saw-tooth algorithms (SEP and GSEP) can be observed to outperform the original EP algorithm in seven out of the ten problems analyzed. Moreover, the proposed GSEP scheme yielded better results than SEP in eight of the ten experiments evaluated.

In order to determine if there are significant differences in mean accuracy (represented by the CCR_G) of the three population size schemes (EP, SEP and GSEP), different statistical tests have been applied. First of all, the Kolmogorov–Smirnov (K–S) statistical test concluded that the CCR_G distributions of the best models follow a normal distribution for a standard asymptotic significance level, $\alpha = 0.05$. A Levene test (Levene 1960) has been applied in order to assess if there are significant differences in variance, concluding that the differences are not significant (second column in Table 4, $p-$value > 0.05). On the basis of these hypotheses, an ANOVA I analysis (Snedecor and

Table 3 Statistical results [Mean and standard deviation (SD) in the 30 runs] of the accuracy in the training (CCR_T) and test (CCR_G) sets of the best models obtained with the best configuration of the different methodologies proposed

Dataset (elitism)	Method	CCR_T (mean± SD)	CCR_G (mean± SD)
Anneal (CCR)	EP1000	99.77 ± 0.24	**98.33** ± 0.64
	EP500	99.52 ± 0.61	97.80 ± 1.30
	SEP20	99.55 ± 0.45	97.95 ± 0.95
	GSEP40U	99.57 ± 0.42	97.98 ± 0.85
Balance (CCR)	EP1000	95.47 ± 1.50	**94.10** ± 1.64
	EP500	94.66 ± 1.85	93.55 ± 1.65
	SEP40	94.48 ± 1.35	93.18 ± 1.86
	GSEP20U	93.24 ± 1.22	92.95 ± 1.72
German (entropy)	EP1000	78.66 ± 1.48	73.05 ± 1.58
	EP500	77.77 ± 1.51	71.84 ± 2.17
	SEP20	78.32 ± 1.12	73.03 ± 1.51
	GSEP20U	77.85 ± 1.42	**73.52** ± 1.81
Glass (CCR)	EP1000	72.61 ± 3.00	67.23 ± 4.10
	EP500	71.37 ± 2.51	66.29 ± 4.02
	SEP40	71.76 ± 2.96	67.99 ± 4.04
	GSEP20N	70.99 ± 2.96	**68.93** ± 4.53
Heart (entropy)	EP1000	86.06 ± 0.90	**86.91** ± 2.06
	EP500	86.34 ± 0.96	85.69 ± 1.54
	SEP20	86.34 ± 1.22	85.44 ± 1.65
	GSEP20N	86.50 ± 0.96	86.37 ± 1.85
Ionos. (CCR)	EP1000	98.27 ± 0.93	92.53 ± 1.80
	EP500	98.30 ± 0.87	92.30 ± 1.71
	SEP20	98.43 ± 0.83	92.41 ± 2.08
	GSEP40U	98.43 ± 0.75	**92.61** ± 1.80
KrVsKp (CCR)	EP1000	99.51 ± 0.31	99.17 ± 0.31
	EP500	99.18 ± 0.50	98.79 ± 0.46
	SEP20	99.39 ± 0.36	98.99 ± 0.41
	GSEP20U	99.33 ± 0.34	**99.27** ± 0.37
Pima (entropy)	EP1000	76.78 ± 0.45	79.39 ± 1.61
	EP500	76.59 ± 0.77	79.86 ± 1.59
	SEP40	76.88 ± 0.68	79.83 ± 1.34
	GSEP40N	76.53 ± 0.83	**79.88** ± 1.46
Vehicle (entropy)	EP1000	79.94 ± 1.45	78.33 ± 2.74
	EP500	79.69 ± 1.43	78.03 ± 2.90
	SEP20	80.12 ± 1.86	78.50 ± 2.93
	GSEP20N	79.88 ± 1.44	**78.80** ± 2.30
Yeast (entropy)	EP1000	61.31 ± 0.80	58.66 ± 1.20
	EP500	61.38 ± 0.70	58.22 ± 1.14
	SEP40	61.03 ± 0.76	**58.71** ± 1.13
	GSEP40U	60.95 ± 0.93	58.41 ± 1.03

Journal : Large 500 | Dispatch : 27-4-2009 | Pages : 15
Article No. : 429 | ☐ LE | ☐ TYPESET
MS Code : SOCO-D-08-00228 | ✓ CP | ✓ DISK

Table 4 *p*-values of the equality of variances Levene's test, *p*-values of the Snedecor's F ANOVA I test, coefficient of determination R^2, ranking of averages of the Tukey statistical multiple comparison tests for accuracy in the test sets using the four different methodologies and *p*-values of the *t* test comparing EP1000 and EP500 methodologies

Dataset	All methodologies				EP1000VsEP500
	Levene-test	*F* test	R^2	Ranking	*t* test
Anneal	0.110	0.179	0.96	$\mu_{EP1000} \geq \mu_{GSEP40U} \geq \mu_{SEP20} \geq \mu_{EP500}$	0.049*
Balance	0.956	0.058	0.94	$\mu_{EP1000} \geq \mu_{EP500} \geq \mu_{SEP40} \geq \mu_{GSEP20U}$	0.195
German	0.141	0.003*	0.89	$\mu_{GSEP20U} \geq \mu_{EP1000} \geq \mu_{SEP20} \geq \mu_{EP500}$; $\mu_{EP1000} > \mu_{EP500}$	0.016*
Glass	0.662	0.096	0.95	$\mu_{GSEP20N} \geq \mu_{SEP40} \geq \mu_{EP1000} \geq \mu_{EP500}$	0.372
Heart	0.501	0.007*	0.90	$\mu_{EP1000} \geq \mu_{GSEP20N} \geq \mu_{EP500} \geq \mu_{SEP40}$; $\mu_{EP1000} > \mu_{EP500}$	0.011*
Ionos	0.738	0.925	0.99	$\mu_{GSEP40U} \geq \mu_{EP1000} \geq \mu_{SEP20} \geq \mu_{EP500}$	0.615
KrVsKp	0.509	0.003*	0.88	$\mu_{GSEP20U} \geq \mu_{EP1000} \geq \mu_{SEP20} \geq \mu_{EP500}$; $\mu_{EP1000} > \mu_{EP500}$	0.000*
Pima	0.624	0.544	0.98	$\mu_{GSEP40N} \geq \mu_{EP500} \geq \mu_{SEP40} \geq \mu_{EP1000}$	0.261
Vehicle	0.214	0.737	0.99	$\mu_{GSEP20N} \geq \mu_{SEP20} \geq \mu_{EP1000} \geq \mu_{EP500}$	0.682
Yeast	0.907	0.414	0.98	$\mu_{SEP40} \geq \mu_{EP1000} \geq \mu_{GSEP40U} \geq \mu_{EP500}$	0.148

* Statistically significant differences with *p* value < 0.05

Cochran 1989) has been performed where the factor to consider is the methodology applied. In Table 4, the significance values of the Snedecor's F ANOVA I test are shown in the third column. The high values of the coefficient of determination R^2 (fourth column in Table 4) suggest that the variance of the $\mathrm{CCR_G}$ is well explained through the use of the linear models of ANOVA I. From the analysis of the significance levels, it can be concluded that for all the datasets (except for German, Heart and KrVsKp) there are no significant differences in mean $\mathrm{CCR_G}$ (p-value > 0.05).

For the German, Heart and KrVsKp datasets, the use of parametric multiple comparison contrasts is proposed for the four studied populations with the aim of ranking the averages of the $\mathrm{CCR_G}$ obtained with each methodology. The Tukey test (Miller 1986) has been applied and the results from all these tests are shown as a methodology ranking in the fifth column of Table 4. In this ranking, $\mu_a \geq \mu_b$ is used to express that, although the mean $\mathrm{CCR_G}$ of the "a" methodology is higher than that of "b", the differences are not significant and $\mu_a > \mu_b$ is used to express that the mean results from "a" methodology are significantly higher that those from "b".

Finally, Table 4 includes the *p*-values of a *t* test only comparing the EP1000 and EP500 methodologies, in order to justify that such a high population size initially selected for the EP algorithm ($N_I = 1{,}000$) is necessary. As can be observed, the EP1000 methodology outperforms EP500 in four out of the ten datasets analyzed (p-value < 0.05), indicating that this high population size is necessary for some datasets.

6.2 Efficiency analysis

On the other hand, the efficiency of all the algorithms has been estimated by obtaining the mean population size over all the generations. Both EP and SEP algorithms have a fixed population scheme and their mean population size is the same throughout the 30 runs. However, the GSEP scheme can result in a different mean population size in each run, according to the generations in which the reinitializations are performed, so the average and standard deviation of this mean population size during the 30 runs have to be obtained.

In order to better evaluate the computational cost associated with each methodology, an expression relating the population size to the number of fitness evaluations performed by the algorithm has been derived (see Eq. 4). Next, the details concerning this expression are given. As presented in Fig. 1, the EP algorithm applies structural mutation over 90% of the current population, parametric mutation over the remaining 10% of individuals and evaluates 10% of the best individuals in order to maintain the CCR elitism. Consequently, it can be concluded that in each generation T, the number of evaluations performed by the algorithm is the following:

$$e_t = e_S(0.9n_t) + e_P(0.1n_t) + e_{CCR}(0.1n_t) \qquad (3)$$

where $e_S(x)$ is the number of evaluations corresponding to x structural mutations, $e_P(x)$ is the number of evaluations corresponding to x parametric mutations, $e_{CCR}(x)$ is the number of evaluations corresponding to obtain the CCR value of x individuals and n_t is the number of individuals in generation t.

Structural mutation and CCR calculation implies only one evaluation per each individual. However, the parametric mutation implies $j+1$ evaluations per each individual, where J is the number of hidden nodes in the model, since it applies a standard simulated annealing process for accepting or rejecting the modifications of each hidden node weights (including the bias node). We suppose that the number of hidden nodes J is a uniform random variable that takes values in the interval $[M_{\min}, M_{\max}]$, whose average is $\hat{j} = (M_{\min} + M_{\max})/2$. Moreover, during the initialization process, $10N_{\mathrm{I}}$ evaluations are performed, that have to be added to the total number of evaluations. Consequently, and considering Eq. 3, the mean number of evaluations of the EP algorithm can be expressed as:

$$\begin{aligned} \bar{e} &= 10N_{\mathrm{I}} + G\left(0.9\overline{N} + 0.1\overline{N}\left(\hat{j}+1\right) + 0.1\overline{N}\right) \\ &= 10N_{\mathrm{I}} + G\overline{N}\left(1 + 0.1\left(\frac{M_{\min}+M_{\max}}{2} + 1\right)\right) \end{aligned} \tag{4}$$

where N_{I} is the maximum number of individuals in the algorithm and $\bar{n}$ is the empirical mean number of individuals obtained during all generations of the evolutionary process. In the standard EP algorithm, n_t is a constant and $\overline{N} = N_{\mathrm{I}}$.

Using this expression, the mean population size $\bar{n}$ and the corresponding number of evaluations $\bar{e}$ have been included in Table 5, for each dataset and each methodology. The algorithm resulting in a lower computational cost is presented in bold face. Moreover, the mean population size results have been graphically represented in Fig. 10.

From the analysis of the results, it can be concluded that the saw-tooth schemes (SEP and GSEP) result in a lower computational cost than EP1000. SEP methodology results in approximately one half of the computational cost obtained by EP1000 methodology and in the same computational cost as EP500. The cost associated with the GSEP methodology is significantly lower than that obtained by SEP for all datasets, except for Anneal, where the cost is more similar. The differences favouring GSEP are more noticeables for Balance, KrVsKp, Pima and Vehicle.

Regarding these results, the methodology proposed for all the datasets is GSEP, because its computational cost is significantly lower and its accuracy level is better or similar, there only existing significant differences in mean $\mathrm{CCR_G}$ between EP1000 and EP500 methodologies (as concluded in the previous section). Moreover, GSEP is the best methodology in efficiency and accuracy in six out of the ten problems analyzed.

Finally, the mean population size over the 30 runs of Vehicle dataset has been plotted for GSEP in Fig. 11, together with two example runs and the value of the different associated parameters, in order to remark on the dynamic behaviour of the GSEP method and to clarify the differences between this scheme and the SEP scheme.

Table 5 Statistical results [mean and standard deviation (SD) in the 30 runs] of the efficiency evaluated with the mean number of individuals ($\bar{n}$) and the mean number of evaluations ($\bar{e}$) for the best configuration of the different methodologies proposed

Datasets (elitism)	Method	$\overline{N}$ (mean ± SD)	$\bar{e}$ (mean ± SD)
Anneal	EP1000	1000.00	520,000
(CCR)	EP500	500.00	260,000
	SEP20	479.07	254,326
	GSEP40U	**414.79** ± 58.77	**221,543** ± 29,973
Balance	EP1000	1000.00	460,000
(CCR)	EP500	500.00	230,000
	SEP40	524.00	245,800
	GSEP20U	**245.01** ± 8.19	**120,250** ± 3,686
German	EP1000	1000.00	220,000
(entropy)	EP500	500.00	110,000
	SEP20	480.79	110,966
	GSEP20U	**373.22** ± 11.62	**62,251** ± 1,627
Ionos.	EP1000	1000.00	310,000
(entropy)	EP500	500.00	155,000
	SEP20	510.00	163,000
	GSEP20U	**392.20** ± 18.73	**186,490** ± 8,429
KrVsKp	EP1000	1000.00	1,490,000
(entropy)	EP500	500.00	745,000
	SEP20	451.27	677,880
	GSEP20U	**233.11** ± 11.21	**441,254** ± 20,739
Glass	EP1000	1000.00	390,000
(CCR)	EP500	500.00	195,000
	SEP40	544.48	216,902
	GSEP20N	**389.69** ± 18.70	**232,123** ± 10,659
Pima	EP1000	1000.00	166,000
(CCR)	EP500	500.00	83,000
	SEP40	509.30	89,451
	GSEP40N	**323.29** ± 8.86	**52,028** ± 1,152
Heart	EP1000	1000.00	135,000
(entropy)	EP500	500.00	67,500
	SEP20	520.00	75,000
	GSEP20N	**366.91** ± 35.53	**55,864** ± 4,441
Vehicle	EP1000	1000.00	2,710,000
(entropy)	EP500	500.00	1,355,000
	SEP20	421.53	1,148,131
	GSEP20N	**207.19** ± 16.08	**382,942** ± 28,944
Yeast	EP1000	1000.00	2,010,000
(entropy)	EP500	500.00	1,005,000
	SEP20	505.02	1,020,040
	GSEP20N	**344.49** ± 19.05	**871,225** ± 47,625

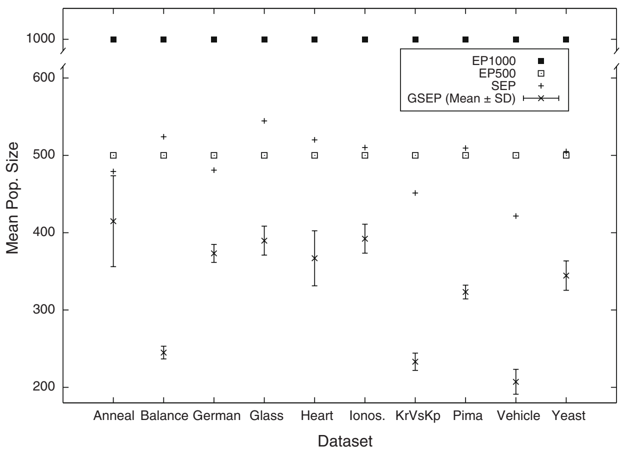


Fig. 10 Graphical representation of the efficiency statistical results for the best configuration of the different methodologies proposed

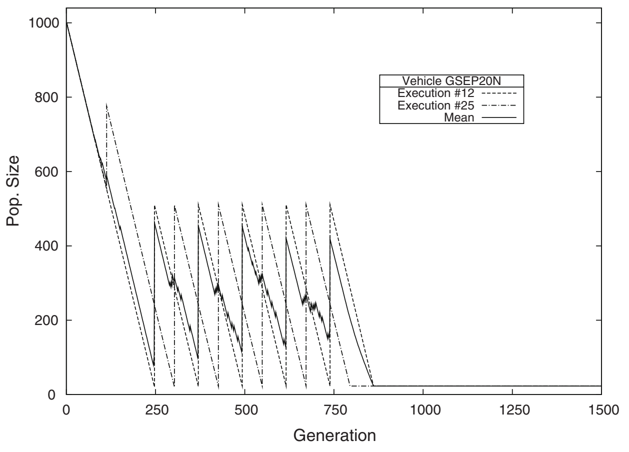


Fig. 11 Population size of two randomly selected executions and mean population size, using the GSEP20N methodology for the Vehicle dataset. The saw-tooth parameter values are the following: $T = 300$, $D = 480$, $m_n = 20$ and $r = 4$

7 Conclusions

The application of the saw-tooth scheme over a previously designed EP algorithm (SEP methodology) has proved viable in designing MLP classifiers. The proposed GSEP algorithm has been presented as an enhanced version of the standard saw-tooth scheme, performing population reinitializations when the difference of variance between two generations is lower than a percentage of the previous generation's variance. This proposed scheme could be easily introduced into any other existing EA.

From the analysis of the experimental results over ten benchmark datasets, it can be concluded that the computational cost of the EP algorithm with a constant population size is reduced by using the original saw-tooth scheme. Moreover, the guided saw-tooth mechanism involves a significantly lower computer time demand than the original scheme. Finally, both saw-tooth schemes do not involve an accuracy decrease and, in general, they obtain a better or similar precision.

In this way, it can be affirmed that the variance of the distribution of the fitness of the best individuals can be considered a suitable measure for guiding restarts and introducing diversity into the EP algorithm evaluated. Considering that the experiments have also shown that a population size of 1,000 individuals is needed by the EP algorithm in some datasets, this high degree of diversity can be supplied through the schemes proposed in this paper, avoiding the high computational cost associated with such a high population size. The evaluation of the algorithms for a wide, thought not exhaustive, range of classification problems examined have shown results that are comparable to those of other classification techniques found in machine learning literature (Landwehr et al. 2005).

On the basis of the "no free lunch" theorem (Wolpert and Macready 1997), it seems that the proposed GSEP method definitely outperforms the constant population EP algorithm. For the original SEP algorithm, the advantage of the proposed GSEP may depend on the specific problem and the set of selected parameters.

As a future research line, different "a priori" probabilistic distributions could be more realistic than the normal and uniform distributions. One possibility could be considering extreme value distributions, for example, adjusting a Weibull distribution. These distributions are more adequate for bounded fitness values. Distributions generated with bootstrapping methods could be alternatively used.

Some other future research lines comprise different extensions of the EP algorithm, that could enhance the performance of the different variants proposed. For example, the algorithm could be applied to alternative neural network structures (Radial Basis Function Neural Networks or Product Unit Neural Networks), more complex initialization processes (Guilléen et al. 2007; Bramlette 1991; Wang et al. 2002) could increase the initial fitness of the population or the proposals could be adapted to a multi-objective perspective.

Acknowledgments This work has been partially subsidized by the TIN 2005-08386-C05-02 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andaluca" (Spain). The research of P. A. Gutirrez has been funded by the FPU Predoctoral Program (Spanish Ministry of Education and Science), grant reference AP2006-01746.

References

Alcala-Fdez J, Sánchez L, García S, del Jesús MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernàndez JC, Herrera F (2008) KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Comput 13(3):307–318

Angeline PJ, Sauders GM, Pollack JB (1994) An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans Neural Netw 5:54–65

Asuncion A, Newman D (2007) UCI machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html

Back T, Fogel DB, Michalewicz Z (1997) Handbook of evolutionary computation. IOP Publishing Ltd, Bristol

Bishop CM (2006) Pattern recognition and machine learning. Springer, Berlin

Bornholdt S, Graudenz D (1992) General asymmetric neural networks and structure design by genetic algorithms. Neural Netw 5(2):327–334. doi:10.1016/S0893-6080(05)80030-9

Bramlette MF (1991) Initialization, mutation and selection methods in genetic algorithms for function optimization. In: Belew RK, Booker LB (eds) Proceedings of the 4th international conference on genetic algorithms (ICGA'91), pp 100–107

Chaiyaratana N, Piroonratana T, Sangkawelert N (2007) Effects of diversity control in single-objective and multi-objective genetic algorithms. J Heuristics 13(1):1–34

Cobb HG, Grefenstette JJ (1993) Genetic algorithms for tracking changing environments. In: Proceedings of the 5th international conference on genetic algorithms, pp 523–530

De Jong ED, Watson RA, Pollack JB (2001) Reducing bloat and promoting diversity using multi-objective methods. In: Proceedings of the 2001 genetic and evolutionary computation conference, Morgan Kaufmann, San Francisco, CA, pp 11–18

De Jong KA (1975) An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, MI, USA

Eiben AE, Smith JE (2003) Introduction to evolutionary computing (Natural Computing Series). Springer, Berlin

Eshelman LJ (1991) The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In: Proceedings of the 1st workshop on foundations on genetic algorithms, pp 257–266

Eshelman LJ, Schaffer JD (1991) Preventing premature convergence in genetic algorithms by preventing incest. In: Belew R, Booker L (eds) Proceedings of the fourth international conference on genetic algorithms, Morgan Kaufman, San Mateo, CA, pp 115–122

Fogel D (1995) Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, New York

Fogel D, Owens A, Wals MJ (1966) Artificial intelligence through simulated evolution. Wiley, New York

Fogel DB, Fogel LJ, Porto VW (1990) Evolving neural networks. Biol Cybern 63(6):487–493. 10.1007/BF00199581

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading

Goldberg DE, Deb K (1990) A comparative analysis of selection schemes used in genetic algorithms. In: Proceedings of the first workshop on foundations of genetic algorithms, Bloomington Campus, Indiana, USA, pp 69–93

Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multi-modal function optimization. In: Proceedings of the 2nd international conference on genetic algorithms, Lawrence Erlbaum Associates, pp 41–49

Goldberg DE, Deb K, Clark JH (1992) Genetic algorithms, noise, and the sizing of populations. Complex Syst 6(4):333–362

Greewood GW, Fogel GB, Ciobanu M (1999) Emphasizing extinction in evolutionary programming. In: Angeline P, Michalewicz Z, Schoenauer M, Yao X, Zalzala A (eds) Proceedings of the 1999 congress on evolutionary computation (CEC99), pp 666–671

Guillén A, Rojas I, González J, Pomares H, Herrera L, Valenzuela O, Rojas F (2007) Output value-based initialization for radial basis function neural networks. Neural Process Lett 25(3):209–225

Gutiérrez PA, Hervás-Martínez C, Lozano M (2007) Saw-tooth algorithm guided by the variance of best individual distributions for designing evolutionary neural networks. In: Proceedings of the 8th international conference on intelligent data engineering and automated learning (IDEAL'07), Springer, Birmingham, Lecture Notes in Computer Science, vol 4881, pp 1131–1140

Herrera F, Lozano M (1996) Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In: Herrera F, Verdegay JL (eds) Genetic algorithms and soft computing, Physica-Verlag, Heidelberg, pp 95–125

Herrera F, Lozano M, Verdegay JL (1996) Dynamic and heuristic fuzzy connectives-based crossover operators for controlling the diversity and convergence of real-coded genetic algorithms. Int J Intell Syst 11(12):1013–1040

Hervás C, Martínez-Estudillo FJ, Gutiérrez PA (2006) Classification by means of evolutionary product-unit neural networks. In: Proceedings of the 2006 international joint conference on neural networks, Vancouver, Canada, pp 2834–2842

Hutter M, Legg S (2006) Fitness uniform optimization. IEEE Trans Evol Comput 10(5):568–589

Koumousis VK, Dimou CK (2002) The effect of oscillating population size on the performance of genetic algorithms. In: Proceedings of the 4th GRACM congress on computational mechanics

Koumousis VK, Katsaras CP (2006) A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 10(1):19–28

Krishnakumar K (1989) Micro-genetic algorithms for stationary and non-stationary function optimization. In: Proceedings of the SPIE: intelligent control and adaptive systems, vol 1196, pp 289–296

Landwehr N, Hall M, Frank E (2005) Logistic model trees. Mach Learn 59(1–2):161–205

Leung FHF, Lam HK, Ling SH, Tam PKS (2003) Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Trans Neural Netw 14(1):79–88. doi:10.1109/TNN.2002.804317

Levene H (1960) Essays in Honor of Harold Hotelling. In: Contributions to probability and statistics. Stanford University Press, London

Lozano M, Herrera F, Cano JR (2008) Replacement strategies to preserve useful diversity in steady-state genetic algorithms. Inf Sci 178(23):4421–4433

Mahfoud SW (1992) Crowding and preselection revisited. In: Männer R, Manderick B (eds) Parallel problem solving from nature 2, North-Holland, Amsterdam, pp 27–36

Maniezzo V (1993) Searching among search spaces: hastening the genetic evolution of feedforward networks. In: Proceedings of the conference on artificial neural nets and genetic algorithms

Martínez-Estudillo AC, Hervás-Martínez C, Martínez-Estudillo FJ, García-Pedrajas N (2006a) Hybridization of evolutionary algorithms and local search by means of a clustering method. IEEE Trans Syst Man Cybern B 36(3):534–545

Martínez-Estudillo AC, Martánez-Estudillo FJ, Hervás-Martínez C, García N (2006b) Evolutionary product unit based neural networks for regression. Neural Netw 19(4):477–486

Martínez-Estudillo FJ, Hervás-Martínez C, Gutiérrez PA, Martínez-Estudillo AC (2008) Evolutionary product-unit neural networks classifiers. Neurocomputing 72(1–2):548–561

McDonnell JR, Waagen DE (1993) Evolving recurrent perceptrons. In: Ruck DW (ed) Proceedings of the SPIE conference, Science of Artificial Neural Networks II, vol 1966, pp 350–361

Miller RG (1986) Beyond ANOVA, basics of applied statistics. Wiley, London

Oppacher F, Wineberg M (1999) The shifting balance genetic algorithm: improving the GA in a dynamic environment. In: Proceedings of the genetic and evolutionary computation conference, Morgan Kaufmann, San Francisco, vol 1, pp 504–510

Palmes PP, Hayasaka T, Usui S (2005) Mutation-based genetic neural network. IEEE Trans Neural Netw 16(3):587–600

Prechelt L (1994) PROBEN1: a set of neural network benchmark problems and benchmarking rules. Tech. Rep. 21/94, Fakultät f++r Informatik (Universität Karlsruhe)

Prügel-Bennett A, Shapiro JL (1994) An analysis of genetic algorithms using statistical mechanics. Phys Rev Lett 72(9):1305–1309

Prügel-Bennett A, Shapiro JL (1997) The dynamics of a genetic algorithm for simple Ising systems. Physica D 104:75–114

Rattray M (1996) Modeling the dynamics of genetic algorithms using statistical mechanics. PhD thesis, Manchester University, Manchester, UK

Rechenberg I (1973) Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution. Frommann-Holzboog, Stuttgart

Rogers A, Pruegel-Bennett A (1999) Genetic drift in genetic algorithm selection schemes. IEEE Trans Evol Comput 3(4):298

Shimodaira H (2001) A diversity-control-oriented genetic algorithm DCGA: performance in function optimization. In: Proceedings of the 2001 congress on evolutionary computation, 2001, vol 1, pp 44–51. doi:10.1109/CEC.2001.934369

Smith RE (1993) Adaptively resizing populations: an algorithm and analysis. In: Proceedings of the fifth international conference on genetic algorithms, pp 653–653

Snedecor GW, Cochran WG (1989) Statistical methods, 6th edn. Iowa State University Press

Tsai JT, Chou JH, Liu TK (2006) Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. IEEE Trans Neural Netw 17(1):69–80

Tsutsui S, Fujimoto Y, Ghosh A (1997) Forking genetic algorithms: GAs with search space division schemes. Evol Comput 5(1):61–80

Ursem RK (2002) Diversity-guided evolutionary algorithms. In: Proceedings of parallel problem solving from nature VII (PPSN-2002), Springer, Berlin, pp 462–471

Ventura S, Romero C, Zafra A, Delgado JA, Hervas C (2008) JCLEC: a Java framework for evolutionary computation. Soft Comput 12(4):381–392

Wang L, Zheng DZ, Tang F (2002) An improved evolutionary programming for optimization. In: Proceedings of the 4th World Congress on Intelligent Control and Automation, vol 3, pp 1769–1773

Whitley D (1989) The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Schaffer JD (ed) Proceedings of the third international conference on genetic algorithms, Morgan Kaufman, San Mateo, CA

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yao X (1997) Global optimization by evolutionary algorithms. In: Proceedings of the 2nd Aizu international symposium on parallel algorithms/architecutre synthesis, (pAs'97), Aizu-Wakamatsu, Japan, pp 282–291

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447

Yao X, Liu Y (1997) A new evolutionary system for evolving artificial neural networks. IEEE Trans Neural Netw 8(3):694–713

Journal : **Large 500** Dispatch : **27-4-2009** Pages : **15**
Article No. : **429** ☐ LE ☐ TYPESET
MS Code : **SOCO-D-08-00228** ✓ CP ✓ DISK

2.3. Entrenamiento de Redes Neuronales con Funciones de Base de tipo *Kernel* y de tipo Proyección utilizando un Algoritmo de Programación Evolutiva

Las publicaciones asociadas a esta parte son:

2.3.1. Combinación de Funciones de Base de tipo Proyección y tipo *Kernel* para Clasificación mediante Redes Neuronales Evolutivas - *Combined Projection and Kernel Basis Functions for Classification in Evolutionary Neural Networks*

- P.A. Gutiérrez, C. Hervás-Martínez, M. Carbonero, J.C. Fernández. Combined Projection and Kernel Basis Functions for Classification in Evolutionary Neural Networks. Neurocomputing (2009). Aceptado.
    - Estado: Aceptado.
    - Índice de Impacto (JCR 2007): 0.865.
    - Área de conocimiento: *Computer Science, Artificial Intelligence.* Ranking 50/93. Tercer Cuartil.

Fecha: 7 Sep 2008 08:28:48 +0100 [07/09/08 11:14:45 CET]

De: Neurocomputing <neucom-eo@elsevier.com>

Para: i02gupep@uco.es

Asunto: SOCO: Editor Decision - Accept

Ref.: Ms. No. NEUCOM-D-08-00132R1
Combined Projection and Kernel Basis Functions for Classification in Evolutionary Neural Networks
Neurocomputing

Dear Mr. Gutiérrez,

We are very pleased to inform you that your paper “Combined Projection and Kernel Basis Functions for Classification in Evolutionary Neural Networks”, with manuscript number: NEUCOM-D-08-00132R1, has been accepted for publication in Neurocomputing. Many congratulations!

We will check the files of your manuscript to see whether we have all the information needed to forward your paper to Elsevier. If not we will get back to you and ask you for the information we need.

If you have not yet supplied the source files for this paper (i.e., the individual text and figure files, as well as files containing the author photographs and biographies), the publisher (Elsevier) will contact you soon to supply that material outside of the system.

Thank you for submitting your work to our journal and do consider us again in the future.
Kind regards,
Juan Corchado
Special Issue Editor
Tom Heskes
Editor in Chief
Neurocomputing

Comments from the Editors and Reviewers:

Reviewer #1: The paper is fine now. The recommendations have been incorporated to the paper.
Reviewer #2: This paper has been accepted with minor changes. This version incorporates all suggestions properly.

Neurocomputing ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Combined projection and kernel basis functions for classification in evolutionary neural networks

P.A. Gutiérrez [a],*, C. Hervás [a], M. Carbonero [b], J.C. Fernández [a]

[a] Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, 14071 Córdoba, Spain
[b] Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14005 Córdoba, Spain

ARTICLE INFO

Keywords:
Classification
Projection basis functions
Kernel basis functions
Evolutionary neural networks

ABSTRACT

This paper proposes a hybrid neural network model using a possible combination of different transfer projection functions (sigmoidal unit, SU, product unit, PU) and kernel functions (radial basis function, RBF) in the hidden layer of a feed-forward neural network. An evolutionary algorithm is adapted to this model and applied for learning the architecture, weights and node typology. Three different combined basis function models are proposed with all the different pairs that can be obtained with SU, PU and RBF nodes: product–sigmoidal unit (PSU) neural networks, product–radial basis function (PRBF) neural networks, and sigmoidal–radial basis function (SRBF) neural networks; and these are compared to the corresponding pure models: product unit neural network (PUNN), multilayer perceptron (MLP) and the RBF neural network. The proposals are tested using ten benchmark classification problems from well known machine learning problems. Combined functions using projection and kernel functions are found to be better than pure basis functions for the task of classification in several datasets.

© 2009 Published by Elsevier B.V.

1. Introduction

Q1 Some features of neural networks (NNs) make them particularly attractive and promising for applications in the modelling and control of nonlinear systems: (1) universal approximation abilities, (2) parallel distributed processing abilities, (3) learning and adaptation, (4) natural fault tolerance and (5) feasibility for hardware implementation [47]. NNs have been an important tool for classification since recent research activities identified them as a promising alternative to conventional classification methods such as linear discriminant analysis or decision trees [31].

Different types of NNs are now being used for classification purposes [31], including, among others: multilayer perceptron (MLP) NNs, where the transfer functions are sigmoidal unit (SU) basis functions; radial basis function (RBF) NNs with kernel functions, where the transfer functions are usually Gaussian [7]; general regression neural networks proposed by Specht [46]; and a class of multiplicative NNs, namely product unit neural networks (PUNNs) [16]. The network is chosen depending on the conditions involved and the prior knowledge about the system under study. The most widely used NN models are MLPs and RBF networks. One distinctive characteristic of all these models is the combination of transfer and activation functions used in the hidden layer or the hidden layers of the NN. This pair of functions (transfer and activation function) will be jointly referred as the basis function further on this paper.

The combination of different basis functions in the hidden layer of a NN has been proposed as an alternative to traditional NNs. For example, Donoho demonstrated that any continuous function can be decomposed into two mutually exclusive type of functions [13], one associated with projection functions (PU or SU) and the other associated with kernel functions (RBF).

This paper aims to obtain empirically a response to the following question: Is the combination of several projection-based and kernel-based functions appropriate for optimizing the accuracy of a NN classifier? In order to answer this question, this paper compares the performance of pure basis function models to the performance of combined basis function models. Three kinds of hidden nodes are considered (SU, PU and RBF) and the three different associated pairs of combined basis function models are proposed: product–sigmoidal unit (PSU) NNs, product–radial basis function (PRBF) NNs, and sigmoidal–radial basis function (SRBF) NNs; these are compared to the corresponding pure models (PUNNs, MLPs and RBF NNs). A hybrid neural network model is proposed, maintaining the linear structure between the hidden and the output layer. Therefore, although the model is complex (it is composed of two different type basis functions), it keeps a linear structure. In this way, this idea is similar to that proposed by Friedman and Stuetzle [20].

The problem of finding a suitable architecture and the corresponding weights for a neural network is a very complex

* Corresponding author. Tel.: +34 957 218 349; fax: +34 957 218 630.
E-mail addresses: i02gupep@uco.es (P.A. Gutiérrez), chervas@uco.es (C. Hervás), mariano@etea.com (M. Carbonero), i82fecaj@uco.es (J.C. Fernández).

0925-2312/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.neucom.2008.09.020

task (for an interesting review of this subject, the reader can consult the review of Yao [53]). This complexity increases significantly when considering different basis functions, which justifies the use of an evolutionary algorithm (EA) to design the structure and training of the weights. This EA is presented in this paper with the aim of optimizing simultaneously the weights and the structure of the two types of basis functions considered in the hidden layer.

The objectives of the study are the following:

- To develop an EA capable of overcoming the difficulties associated with the training of NNs containing combinations of the different basis functions considered in a single hidden layer.
- To experimentally check the duality of kernel and projection basis functions introduced by Donoho [13] and determine the most adequate projection function (PU or SU) for this hybridization.
- To check if the hybridization of two projection basis functions (PSU) is suitable for building a classifier.
- To compare the performance and the network size of the combined basis function models to their corresponding pure networks.

The rest of the paper is structured as follows. Section 2 analyzes previous works which are related to the training of projection based NNs, kernel based NNs or to the hybridization of both neural network basis functions. Section 3 formally presents the combined basis function model for classification considered in this work, and analyzes some important properties that have to be considered when combining the different basis functions proposed. The main characteristics of the algorithm used for training the hybrid models are described in Section 4. Section 5 presents the experiments carried out and discusses the results obtained. Finally, Section 6 completes the paper with the main conclusions and future directions suggested by this study.

2. Related works

In this section, different works related to the optimization and the designing of neural network models are presented. First, the main projection basis function neural network methods are described; next, the kernel basis function neural networks and the associated methods are introduced; and finally, the analysis of some works that combine both functional models is performed.

2.1. Neural networks based on projection basis functions

The MLP with the developed efficient back-propagation training algorithm [42] is probably the most frequently used type of neural network model in practical applications. However, due to its multilayer structure and the limitations of the back-propagation algorithm, the training process often settles in undesirable local minima of the error surface or converges too slowly. Adaptation or learning is a major focus of MLP research that provides the NN model with a degree of robustness. Additionally, building a MLP model is complex due to the presence of many training factors. Training factors traditionally involved in building a MLP model may include: the number of hidden nodes, training tolerance, initial weight distribution and the function gradient.

Product unit neural networks (PUNNs) introduced by Durbin and Rumelhart [16] are another promising alternative [25,34]. PUNN models are a class of high-order neural networks which only have unidirectional interlayer connections. In contrast to sigmoidal neural networks, PUNNs are based on multiplicative nodes instead of additive ones. PUs in the hidden layer of PUNNs provide a powerful mechanism for these models to efficiently learn higher-order combinations of inputs. Durbin and Rumelhart determined empirically that the information capacity of PUs (measured by their capacity for learning random Boolean patterns) is approximately $3N$, compared to the $2N$ of a network with additive units for a single threshold logic function, where N denotes the number of inputs to the network [16]. Besides that, it is possible to obtain the upper bounds of the Vapnik Chervonenkis (VC) dimension [50] in PUNNs, these bounds being similar to those obtained in sigmoidal neural networks [44]. It is a straightforward consequence of the Stone–Weierstrass theorem to prove that PUNNs are universal approximators [33] (observe that polynomial functions in several variables are a subset of PU models).

Despite these advantages, product-unit based networks have a major drawback. Networks based on PUs have more local minima and a higher probability of becoming trapped in them [33]. The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface. Because of this reason, their training is more difficult than the training of standard MLPs. For example, it is well known that back-propagation is not efficient for the training of PUNNs [41]. Several efforts have been made to carry out learning methods for PUs: Janson and Frenzel [25] developed a genetic algorithm for evolving the weights of a PUNN with a predefined architecture. The major problem with this kind of algorithm is how to obtain the optimal architecture beforehand. Leerink et al. [27] tested different local and global optimization methods for PUNNs. Ismail and Engelbrecht [22] applied four different optimization methods to train PUNNs: random search, particle swarm optimization, genetic algorithms, and LeapFrog optimization. They concluded that random search is not efficient to train this type of network, and that the other three methods show an acceptable performance in three problems of function approximation with low dimensionality.

2.2. Neural networks based on kernel basis functions

RBF NNs have been used in the most varied domains, from function approximation to pattern classification, time series prediction, data mining, signals processing, and nonlinear system modelling and control. They have some useful properties which render them suitable for modelling and control. First, such networks are universal approximators [40]. In addition, they belong to a class of linearly parameterized networks where the network output is connected to tuneable weights in a linear manner. The RBF uses, in general, hyper-ellipsoids to split the pattern space. This is different from MLP networks which build their classifications on hyper-planes, defined by a weighted sum. Due to their functional approximation capabilities, RBF networks have been seen as a good solution for interpolation problems. RBF can be considered a local average procedure, and the improvement in both its approximation ability as well as in the construction of its architecture has been note-worthy [6].

Their simple architecture allows the application of fast and efficient learning algorithms [38]. However, they are only applied for some problems in the theory of interpolation, because strict interpolation may sometimes not be a good strategy for the training of RBF networks due to possibly poor resulting generalization ability. In addition, if the size of the training set is too large, and/or if there is a large number of redundant data (linearly dependent vectors) in this set, the likelihood of obtaining an illconditioned correlation matrix is higher and hinders the training procedure. Moreover, the constraint of having as many RBFs as data points makes the problem over-determined. To overcome these computational difficulties, the complexity of the network has to be reduced, requiring an approximation to a regularized solution [18,37]. This approach involves the search for a suboptimal solution in a lower dimensional space [8].

The number and positions of basis functions, which correspond to the nodes in the hidden layer of the network, have an important influence on the performance of the RBF neural net. Both problems have been tackled using a variety of approaches. For instance, the number and position of the RBFs may be fixed and defined a priori [21]; they may be determined by unsupervised clustering algorithms [12]; or they can be evolved using evolutionary algorithms [8,29] or a hybrid algorithm [55].

In order to solve the problem of selecting a proper size for the hidden layer, a new category of algorithms has been introduced to automatically determine the structure of the network. These include the orthogonal least squares algorithm [9]; constructive methods, where the structure of the network is built incrementally [56]; pruning methods that start with an initial selection of a large number of hidden units which is reduced as the algorithm proceeds [36]; and the simultaneous selection of network structure and parameters by employing optimization methods based on genetic algorithms [5].

2.3. *Neural networks based on hybrid basis functions*

Hybrid models have also been proposed, where different activation/transfer functions are used for the nodes in the hidden layer. Several authors have proposed the hybridization of different basis functions, using either one single hybrid hidden layer or several connected pure layers.

According to Duch and Jankowski [15], mixed transfer functions within one network may be introduced in two ways. In the first way, a constructive method selects the most promising function from a pool of candidates in RBF-like architecture and add it to the network [14]. In the second approach, starting from a network that already contains several types of functions (such as Gaussian and sigmoidal functions), pruning or regularization techniques are used to reduce the number of functions [15].

Optimal transfer function networks were presented as a method for selecting appropriate functions for a given problem [24], creating architectures that are well matched for some given data and resulting in a very small number of hidden nodes.

In the functional link networks of Pao [39], a combination of various functions, such as polynomial, periodic, sigmoidal and Gaussian functions is used. The basic idea behind a functional link network is the use of functional links, adding nonlinear transformation of the input variables to the original set of variables, and suppressing the hidden layer, only performing a linear transformation of this derived input space. In this way, the first nonlinear mapping is fixed and the second linear mapping is adaptive.

A more complex approach considers several layers or models, each one containing a basis function structure, resulting in a modular system. For example, Iulian proposes a methodology including three distinct modules implementing a hybrid feedforward neural network, namely a Gaussian type RBF network, a principal component analysis (PCA) process, and a MLP NN [23]. Another proposal of Lehtokangas and Saarinen considers two hidden layers in the model [28], the first one composed of Gaussian functions and the second one made up of SU basis functions.

Neural networks using different transfer functions should use fewer nodes, enabling the function performed by the network to be more transparent. For example, one hyperplane may be used to divide the input space into two classes and one additional Gaussian function to account for local anomaly. Analysis of the mapping performed by an MLP network trained on the same data will not be so simple.

In this context, it is worth emphasizing the paper by Cohen and Intrator [11], which is based on the duality and complementary properties of projection-based functions (SU and PU) and kernel typology (RBF). This hybridization of models has been justified theoretically by Donoho [13], who demonstrated that any continuous function can be decomposed into two mutually exclusive functions, such as radial (kernel functions) and crest ones (based on the projection). Although theoretically this decomposition is justified, in practice it is difficult to apply gradient methods to separate the different locations of a function (in order to adjust them by means of a combination of RBFs) and then to estimate the residual function by means of a functional approach based on projections, all without getting trapped in local optima in the procedure of error minimization [19].

Recently, Wedge and collaborators [52] have presented a hybrid RBF and sigmoidal neural network using a three step training algorithm for function approximation, aiming to achieve an identification of the aspects of a relationship that are universally expressed separately from those that vary only within particular regions of the input space.

3. Combined basis function model for classification

The combined basis function (CBF) model used in the classification process is proposed in this section and it is represented by means of a neural network structure. First of all, it is important to formally define the classification task. In a classification problem, measurements x_i, $i=1,2,\dots,k$, of a single individual (or object) are taken, and the individuals are to be classified into one of the J classes based on these measurements. A training dataset $D=\{(\mathbf{x}_n,\mathbf{y}_n); n=1,2,\dots,N\}$ is available, where $\mathbf{x}_n=(x_{1n},\dots,x_{kn})$ is the random vector of measurements taking values in $\Omega\subset R^k$, and $\mathbf{y}_n$ is the class level of the n-th individual. The representation of the class level is performed using the common technique of a "1-of-J" encoding vector, $\mathbf{y}=(y^{(1)},y^{(1)}\dots,y^{(J)})$, and the correctly classified rate (CCR) is used as an accuracy measure defined by $CCR=\frac{1}{N}\sum_{n=1}^{N} I(C(\mathbf{x}_n)=\mathbf{y}_n)$, where $I(.)$ is the zero–one loss function. A good classifier tries to achieve the highest possible CCR in a given problem.

In order to tackle this classification problem, the outputs of the CBF model have been interpreted from the point of view of probability through the use of the softmax activation function [7], which is given by

$$g_l(\mathbf{x},\boldsymbol{\theta}_l)=\frac{\exp f_l(\mathbf{x},\boldsymbol{\theta}_l)}{\sum_{j=1}^{J}\exp f_j(\mathbf{x},\boldsymbol{\theta}_j)},\quad l=1,2,\dots,J \tag{1}$$

where J is the number of classes in the problem, $f_l(\mathbf{x},\boldsymbol{\theta}_l)$ is the output of the l output node for pattern $\mathbf{x}$ and $g_l(\mathbf{x},\boldsymbol{\theta}_l)$ the probability a pattern $\mathbf{x}$ has of belonging to class l. The hybrid CBF model to estimate the function $f_l(\mathbf{x},\boldsymbol{\theta}_l)$ is given by a weighted combination of a pair of basis functions

$$\begin{aligned}f_l(\mathbf{x},\boldsymbol{\theta}_l)=\beta_0^l+\sum_{j=1}^{m_1}\beta_j^{l,1}B_j^1(\mathbf{x},\mathbf{w}_j^1)\\+\sum_{j=1}^{m_2}\beta_j^{l,2}B_j^2(\mathbf{x},\mathbf{w}_j^2),\quad l=1,2,\dots,J\end{aligned} \tag{2}$$

where $\boldsymbol{\theta}_l=(\boldsymbol{\beta}^l,\mathbf{w}_1^1,\dots,\mathbf{w}_{m_1}^1,\mathbf{w}_1^2,\dots,\mathbf{w}_{m_2}^2)$, $\boldsymbol{\beta}^l=(\beta_0^l,\beta_1^{l,1},\dots,\beta_{m_1}^{l,1},\beta_1^{l,2},\dots,\beta_{m_2}^{l,2})$ and $\mathbf{w}_j^i$ is the weight vector of the connections between the input layer and the j-th hidden node of type i. In order to simplify the following expressions, $B_j(\mathbf{x};\mathbf{w}_j)$ will be used in a general way to substitute both $B_j^1(\mathbf{x};\mathbf{w}_j^1)$ or $B_j^2(\mathbf{x};\mathbf{w}_j^2)$. Two types of projection basis functions are considered in (2) to replace B_j^1 or B_j^2:

- Sigmoidal units in the form:
$$B_j(\mathbf{x};\mathbf{w}_j)=\frac{1}{1+\exp(-w_{j0}-\sum_{i=1}^{k}w_{ji}x_i)}$$
where $\mathbf{w}_j=(w_{j0},w_{j1},\ldots,w_{jk})$, w_{j0} being the bias.
- Product units have the following expression:
$$B_j(\mathbf{x};\mathbf{w}_j)=\prod_{i=1}^{k}x_i^{w_{ji}}$$
where $\mathbf{w}_j=(w_{j1},\ldots,w_{jk})$, not considering bias in the inputs.
- Kernel basis functions replacing B_j^1 or B_j^2 in (2) are RBFs, in the form:
$$B_j(\mathbf{x};(\mathbf{c}_j|r_j))=\exp\left(-\frac{|\mathbf{x}-\mathbf{c}_j|^2}{2r_j^2}\right)$$
where $\mathbf{w}_j=(w_{j0},w_{j1},\ldots,w_{jk})$, and $\mathbf{c}_j=(w_{j1},\ldots,w_{jk})$ and $r_j=w_{j0}$ are, respectively, the centre and width of the Gaussian basis function of the j-th hidden unit.

Using the softmax activation function presented in Eq. (1), the class predicted by the neural net corresponds to the node in the output layer whose output value is the greatest. In this way, the optimum classification rule $C(\mathbf{x})$ is the following:

$$C(\mathbf{x})=\hat{l}\quad\text{where } \hat{l}=\arg\max_{l} g_l(\mathbf{x},\hat{\boldsymbol{\theta}})\ \text{ for } l=1,2,\ldots,J \tag{3}$$

The function used to evaluate the CBF model is the function of cross-entropy error and is given by the following expression for J classes:

$$l(\boldsymbol{\theta})=-\frac{1}{N}\sum_{n=1}^{N}\sum_{l=1}^{J}y_n^{(l)}\log g_l(x_n,\boldsymbol{\theta}_l) \tag{4}$$

where $\boldsymbol{\theta}=(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_J)$. Moreover, because of the normalization condition

$$\sum_{l=1}^{J}g_l(\mathbf{x},\boldsymbol{\theta}_l)=1,$$

the probability for one of the classes does not need to be estimated. There is a redundancy in the functions $f_l(\mathbf{x},\boldsymbol{\theta}_l)$, since adding an arbitrary $h(\mathbf{x})$ to each output leaves the decision (3) unchanged. Traditionally one of them is set to zero ($f_J(\mathbf{x},\boldsymbol{\theta}_J)=0$) which reduces the number of parameters to estimate. With all these considerations, Fig. 1 represents the general scheme of the CBF model, not considering bias in the input layer.

From a statistical point of view, with the softmax activation function defined in (1) and the cross-entropy error defined in (4), the neural network model can be seen as a multilogistic regression model. Nevertheless, the nonlinearity of the model with respect to parameters $\boldsymbol{\theta}$, and the indefinite character of the associated Hessian matrix do not recommend the use of gradient-based methods to minimize the negative log-likelihood function (for example, Iteratively Reweighted Least Squares, which is commonly used in the optimization of log-likelihood in linear multinomial logistic regression). Thus, an evolutionary algorithm is used to determine both the number of nodes in the hidden layer as well as the weights of the nets.

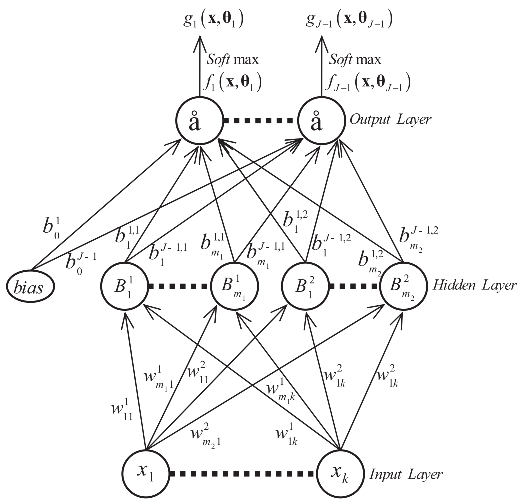


Fig. 1. Combined basis function model for classification.

3.1. Basis function complexity and generalization capability

The first consideration when using NNs in a classification problem is the type of projection or kernel basis function to be selected. When the underlying cross-entropy function is unbounded, the PU basis functions are very adequate for classification, because, if the whole space R^k is considered to be the input domain, an MLP basis function model has the following upper and lower bounds:

$$|f_{MLP}(\mathbf{x})|=\left|\sum_{j=1}^{m}\beta_j\frac{1}{1+e^{-\langle\mathbf{x},\mathbf{w}\rangle}}\right|\leqslant\sum_{j=1}^{m}|\beta_j|$$

where $\langle,\rangle$ is the inner product.

In the same way, a Gaussian RBF has the following bounds:

$$|f_{RBF}(\mathbf{x})|=\left|\sum_{j=1}^{m}\beta_j e^{-\|\mathbf{x}-\mathbf{c}_j\|/r_j}\right|\leqslant\sum_{j=1}^{m}|\beta_j|$$

where $\|\ \|$ is the norm.

However a PU basis function model is unbounded, as if $f_{PU}(\mathbf{x})=\sum_{j=1}^{m}\beta_j\prod_{i=1}^{p}x_i^{w_{ji}}$ then:

$$\lim_{\|x\|\to+\infty}f_{PU}(\mathbf{x})=\lim_{\|x\|\to+\infty}\sum_{j=1}^{m}\beta_j\prod_{i=1}^{p}x_i^{w_{ji}}=\infty$$

considering certain values for the parameters of the model.

In general, the training data available for solving a classification problem belongs to a compact set, which is the set where the parameters of the model are adjusted. In this set, the MLP models can always be applied. However, the bounded functions do not show a good generalization capability for data which do not belong to the training compact set. Consequently, the behaviour of the basis functions at training set domain borders is a characteristic which influences the decision about which basis functions are the most adequate.

The second consideration to take into account is the generalization capability of the combined basis function models. For this purpose, the combined models formed by PU projection basis functions and kernel RBFs (PRBF models) and those formed by SU basis functions and RBFs (SRBF models) are considered first. If a family of functions is an average of projection and kernel basis function, this structure enhances the capability to reduce empirical risk, that is, the error obtained with generalization patterns. The reason for this reduction is that this kind of model can be consider to be an ensemble of the two models it is made up

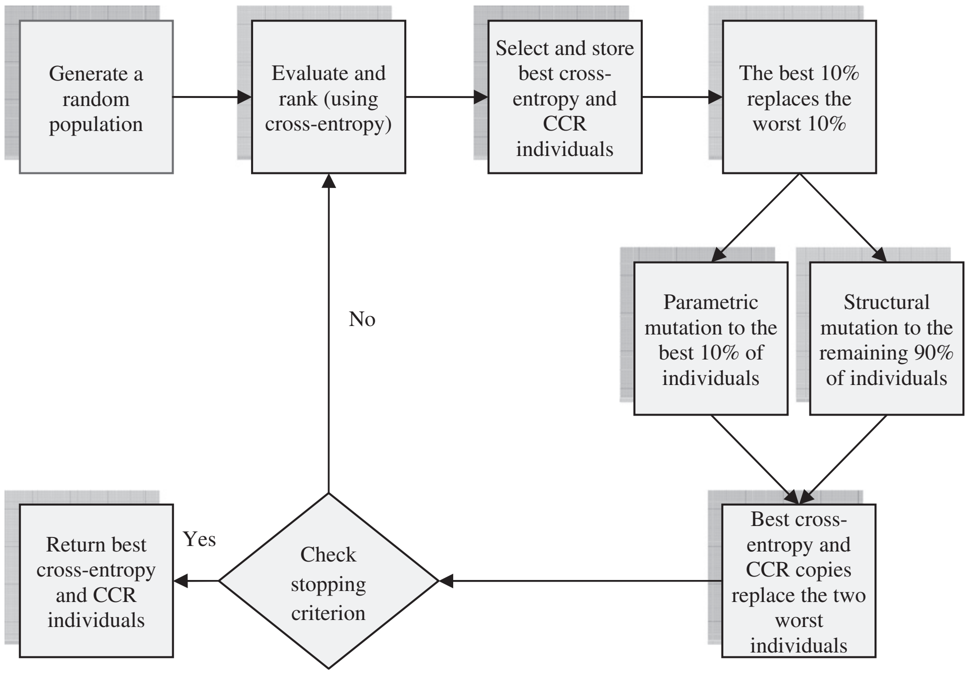

Generate a random population
Evaluate and rank (using cross-entropy)
Select and store best cross-entropy and CCR individuals
The best 10% replaces the worst 10%
Parametric mutation to the best 10% of individuals
Structural mutation to the remaining 90% of individuals
Best cross-entropy and CCR copies replace the two worst individuals
Check stopping criterion
No
Yes
Return best cross-entropy and CCR individuals

Fig. 2. Combined basis function evolutionary programming scheme.

of, a simple average of the outputs obtained from two individual classifiers using two different structures, PU or SU, and RBF. Most of the existing ensemble learning algorithms [4,49] can be interpreted to be building diverse base classifiers implicitly. When projection basis functions and kernel basis functions are used, the ensemble results in a high degree of diversity because the individual PU or SU submodel contributes a global recognition structure and will disagree with the RBF submodel in the composed network, as this model contributes a local recognition structure. Using this kind of models, generalization capability is related to the VC dimension [50]. Particularly, when a model is of high complexity, the VC dimension is higher and its generalization capacity decreases. Schmitt [45] established a super-linear lower bound on the VC dimension for a RBF neural network with W parameters, and one hidden layer of k RBF nodes, of at least $(W/12)\log(k/8)$. On the other hand, MLP neural networks have a VC dimension that is also super-linear in the number of network parameters W, and when there is one hidden layer with k MLP nodes, the VC dimension is at least $(W/18)\log(k/4)$ [43]. Thus, the cooperative network effect enhancing the computational power of sigmoidal networks is also confirmed for RBF networks, and, consequently, for SRBF and PRBF hybrid models. The latter occurs since the upper bounds of the VC dimension in a PUNN are similar to those obtained for an MLP [34].

The capacity for generalization of this ensemble-like typology using basis functions with sufficiently diverse global-local discriminator characteristics (i.e. SRBF or PRBF models) is similar to or greater than the generalization capability of pure PU, SU or RBF models, which is proven empirically in this study.

The next point is the generalization capacity of the combined models formed by product and sigmoidal projection basis functions (PSU models). In general and considering the bias-variance trade-off, the higher the complexity obtained by the models, the lower their generalization capability. In this respect, it is interesting to point out that Schmitt [44] obtains the upper bounds of VC dimension using networks where SUs and PUs are combined, and that these upper bounds are similar to those obtained for pure PU or SU neural networks. This fact guarantees good generalization capability for the PSU model, this generalization being similar to the capability of MLPs or PUNNs. However both basis functions have a projection characteristic which prevents their global discrimination characteristics from contributing sufficient diversity in some datasets, the generalization capability of these models being similar to that obtained with their corresponding pure models. This will be analyzed in the experimental section.

4. The combined basis function evolutionary programming algorithm

In this section, the evolutionary algorithm used to estimate the parameters and the structure (including the number of hidden nodes of each basis function type) of the CBF models is presented.

The objective of the algorithm is to design a CBF neural network with optimal structure and weights for each classification problem tackled. The population is subject to the operations of replication and mutation, and crossover is not used due to its potential disadvantages in evolving artificial networks [2,54]. With these features the algorithm falls into the class of evolutionary programming (EP) [17]. It is an extension of the neural net evolutionary programming (NNEP) algorithm proposed in previous works [33,34], adding the necessary functionalities to consider CBF neural networks with different basis functions in the hidden layer and resulting in what we have called *combined basis function evolutionary programming* (CBFEP). NNEP is a software package developed in JAVA by the authors, as an extension of the JCLEC[1] framework [51] and it is available in the noncommercial JAVA tool named KEEL[2] [1].

The general framework of CBFEP is shown in Algorithm 1. The CBFEP algorithm starts by generating a random population of CBF models and makes them evolve by applying different mutation and replication operations. A graphical representation of the

[1] Java Class Library for Evolutionary Computation (http://jclec.sourcepforge.net).

[2] Knowledge Extraction base on Evolutionary Learning (http://www.keel.es).

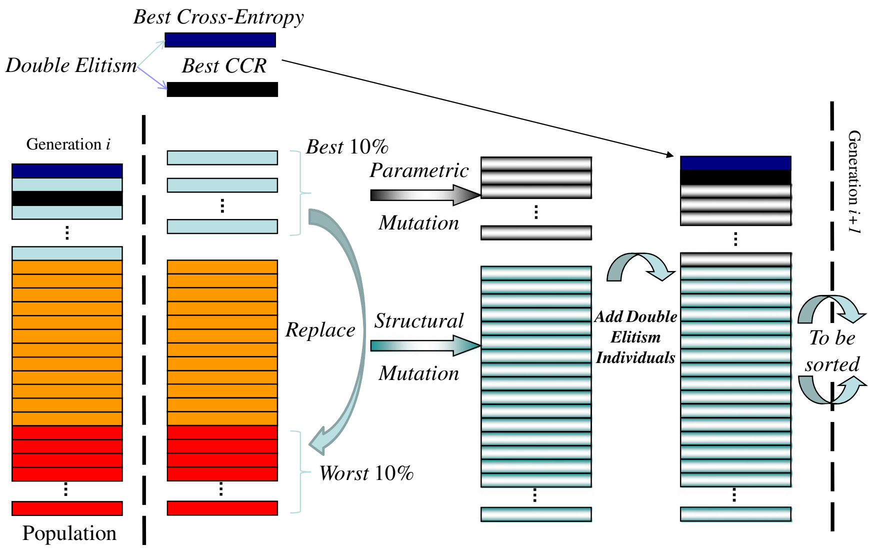


Fig. 3. Double elitism procedure scheme.

algorithm has been included in Fig. 2, where the different stages of the algorithm are more clearly differentiated.

Algorithm 1. General framework of combined basis function evolutionary programming algorithm.

```
(1) Generate a random population of size $N_P$, where each individual presents a combined basis function structure.
(2) Repeat until the stopping criterion is fulfilled.
    (2.1) Calculate the fitness (decreasing transformation of cross-entropy error) of every individual in the population and rank the individuals with respect to their cross-entropy error.
    (2.2) Select and store best cross-entropy individual and best CCR individual (double elitism).
    (2.3) The best 10% of population individuals are replicated and substitute the worst 10% of individuals.
    (2.4) Apply mutations:
        (2.4.1) Parametric mutation to the best 10% of individuals.
        (2.4.2) Structural mutation to the remaining 90% of individuals, using a modified add node mutation in order to preserve the combined basis function structure.
    (2.5) Add best cross-entropy individual and best CCR individual from previous generation and substitute the two worst individuals.
(3) Select the best CCR individual and the best cross-entropy individual in the final population and consider both as possible solutions.
```

The fitness measure is a strictly decreasing transformation of the cross-entropy error $l(\boldsymbol{\theta})$ given by $A(g) = 1/1 + l(\boldsymbol{\theta})$, where g is a CBF model given by the multivaluated function $g(\mathbf{x}, \boldsymbol{\theta}) = (g_1(\mathbf{x}, \boldsymbol{\theta}_1), \ldots, g_J(\mathbf{x}, \boldsymbol{\theta}_J))$ and $l(\boldsymbol{\theta})$ is the cross-entropy error, defined in (4). The severity of mutations depends on the temperature $T(g)$ of the neural network model, defined by $T(g) = 1 - A(g)$, $0 \leqslant T(g) \leqslant 1$. For further details about the general characteristics of the NNEP algorithm, the reader can consult previous works [32–34].

One specific characteristic of this algorithm is the *double elitism*. As can be observed, the proposed algorithm returns both the best cross-entropy and the best *CCR* individuals as feasible solutions (step 3, Algorithm 1). In general, the relationship between *CCR* and cross-entropy error strongly depends on the dataset structure. Hence, regarding experimental results, using cross-entropy elitism is more suitable in order for some datasets to result in higher generalization accuracy, but using *CCR* elitism can be more appropriate for some other datasets. For this reason, the algorithm returns the best *CCR* and the best cross-entropy individuals as solutions, the best approach for each problem being difficult to ascertain a priori. In generation *i*, both the best cross-entropy individual and the best *CCR* individual are stored (step 2.2, Algorithm 1). Then, the selection and the mutations are applied (steps 2.3 and 2.4, Algorithm 1). Finally, the population of the next generation is formed merging the double elitism individuals and the mutated individuals (step 2.5, Algorithm 1) and they will be sorted in the next generation by using the cross-entropy error function (step 2.1, Algorithm 1). In Fig. 3, it can be observed graphically how the *double elitism* procedure works and how the different steps are applied.

Next, there is an explanation of the specific details of the CBFEP algorithm which concern to the optimization of the CBF structure. We will refer to the two types of hidden nodes considered in CBF topology as t_1 and t_2.

In order to define the topology of neural networks generated in the evolution process, three parameters are considered: m, M_E and M_I. They correspond to the minimum and the maximum number of hidden nodes in the whole evolutionary process and the maximum number of hidden nodes in the initialization process, respectively. In order to obtain an initial population formed by models simpler than the most complex model possible, parameters must fulfil the condition $m \leqslant M_I \leqslant M_E$.

Networks of $10N_P$ are generated, where $N_P = 1000$ is the size of the population during the evolutionary process. Then the best N_P neural networks are selected (step 1, Algorithm 1). For the generation of a network, the number of nodes in the hidden layer is taken from a uniform distribution in the interval $[m, M_I]$. Once the number of hidden nodes is decided, each hidden node is generated using a probability of 0.5 to decide if the node corresponds to t_1 or t_2. For PU and SU hidden nodes, the number of connections between each node in the hidden layer and the input nodes is determined from a uniform distribution in the interval $(0, k]$, where k is the number of independent variables. For RBF hidden nodes, the number of connections is always k, since these connections represent the coordinates of the centre of the

Table 1
Main characteristics of each dataset tested and non-common parameter values.

Dataset	# Instances	# Inputs	Distribution	# Classes	# Gen.	m	M_I	M_E
Balance	625	4	(288,49,288)	3	500	3	4	5
Card	690	51	(307,383)	2	50	1	2	3
German	1000	61	(700,300)	2	300	2	3	4
Glass	214	9	(17,76,13,29,70,9)	6	500	7	8	9
Heart	270	13	(150,120)	2	100	1	1	2
Ionosphere	351	34	(126,225)	2	300	3	4	5
Newthyroid	215	5	(150,35,30)	3	100	1	1	4
Pima	768	8	(500,268)	2	160	1	2	3
Vote	435	16	(267,168)	2	10	1	1	2
Zoo	101	16	(41,20,5,13,4,8,10)	7	400	2	3	3

node. The number of connections between each hidden node and the output layer is determined from a uniform distribution in the interval $(0, J-1]$.

Weights are initialized differently depending on the type of hidden node generated. For PU and SU hidden nodes, weights are assigned using a uniform distribution defined throughout two intervals: $[-5, 5]$ for connections between the input layer and hidden layer and, for all kinds of nodes, $[-10, 10]$ for connections between the hidden layer and the output layer. For RBF hidden nodes, the connections between the input layer and hidden layer represent the centre of the associated Gaussian distribution and these centres are initialized using a clustering algorithm, so the EA can start the evolutionary process with well positioned centres. The main idea is to cluster input data in k groups, k being the number of hidden RBF nodes. In this way, each hidden RBF node can be positioned in the centroid of its corresponding cluster. Finally, the radius of the RBF hidden node is calculated as the geometric mean of the distance to the two closest centroids. The clustering technique chosen (similar to that proposed by Cohen and Intrator [10]) is a modification of the classic k-means algorithm, in which the initial centroids are calculated using a specific initialization algorithm that avoids local minima, increasing the probability that the initial k cluster centres not be from a smaller number of clusters.

Parametric mutation (step 2.4.1, Algorithm 1) is accomplished for PU, SU and RBF hidden nodes by adding a Gaussian noise represented by a one-dimensional normally distributed random variable with mean 0 and adaptive variance, where variances are updated throughout the evolution of the algorithm. Radius r_i of RBF hidden nodes are mutated using a similar procedure.

On the other hand, structural mutation (step 2.4.2, Algorithm 1) implies a modification in the neural network structure and allows the exploration in different regions of the search space while helping to keep up the diversity of the population. There are different structural mutations, similar to the mutations in the GNARL model [2], including connection deletion, connection addition, node deletion, node addition and node fusion. There is no reason for connection deletion and connection addition if the node mutated is a RBF. Therefore, these mutations are not taken into consideration with this kind of nodes.

In node fusion, two randomly selected hidden nodes, a and b, are replaced by a new node c, which is a combination of the two. a and b must be of the same type in order to accomplish this mutation. The connections that are common to both nodes are kept and the connections that are not shared by the nodes are inherited by c with a probability of 0.5, their weight being unchanged. The weight resulting from the fusion of common connections depends again on the type of hidden node chosen. For PU and SU nodes, the resulting weights are given by

$$\beta_c^l = \beta_a^l + \beta_b^l, \quad w_{jc} = \left(\frac{w_{ja} + w_{jb}}{2}\right)$$

while for RBF nodes, the fact that RBFs can be interpreted as circumferences with a perfectly identified center and radius obliges the resulting weights to be considered as

$$\mathbf{c}_c = \frac{r_a}{r_a + r_b}\mathbf{c}_a + \frac{r_b}{r_a + r_b}\mathbf{c}_b, \quad r_c = \frac{r_a + r_b}{2}$$

$\mathbf{c}_j$ being the center defined by the hidden node j, which is $\mathbf{c}_j = (w_{j1}, w_{j2}, \ldots, w_{jk})$, and r_j being its radius.

In node addition, once the number of hidden nodes to be added has been decided, each hidden node is generated using a probability of 0.5 to assign its type (t_1 or t_2). If the number of hidden nodes in the neural net is M_E, no hidden nodes are added.

5. Experiments

In order to analyze the performance of the CBF model and the corresponding training algorithm CBFEP, 10 datasets in the UCI repository have been tested [3]. The experimental design was conducted using a holdout cross-validation procedure with $3n/4$ instances for the training set and $n/4$ instances for the generalization set, where the size of the dataset is n. All parameters of the CBFEP algorithm are common to these ten problems, except the m, M_I, M_E and the number of generations (#Gen.) values, which are represented in Table 1 together with the main characteristics of each dataset. An analysis of the results obtained for all pure (SU, PU and RBF) or combined (PSU, PRBF and SRBF) basis functions used in the neural network model is performed for every dataset.

Table 2 shows the mean value and standard deviation (mean±SD) results of the correctly classified rate for the training (CCR_T) and generalization (CCR_G) sets and the mean and standard deviation of the number of net connections (#links) of the best models obtained in 30 runs of all the experiments performed. As the algorithm returns both the best CCR and cross-entropy individuals, two averages and standard deviations are obtained for each dataset. However, in order to reduce the extension of the table, the results included in Table 2 correspond only to the best performing variant in each CBF model, since the suitability of each elitism depends on the characteristics of the dataset and the basis function structure applied. It can be observed that the combination of basis functions produces good results with respect to CCR_G. In fact, from a purely descriptive point of view, the best result is obtained using CBF models in six out of the ten datasets analyzed, and the second best result is obtained in three out of the four remaining datasets. Moreover, in two of the 10 datasets analyzed, the first and second best results have been obtained using combined basis functions.

To ascertain the statistical significance of the differences observed in each dataset performance, an analysis of variance

Table 2
Statistical results (mean ± SD) of the correctly classified rate for 10 datasets and for the training (CCR_T) and generalization (CCR_G) sets and number of connections (#links) for 30 executions of the CBFEP algorithm using pure or combined basis functions and CCR_T (C) or cross-entropy (E) elitism.

Dataset	Func.	Elit.	CCR_T Mean ± SD	CCR_G Mean ± SD	# Links Mean ± SD	Dataset	Func.	Elit.	CCR_T Mean ± SD	CCR_G Mean ± SD	# Links Mean ± SD
Balance	PU	C	99.30 ± 0.34	96.45 ± 1.28	23.1 ± 2.6	Card	PU	E	84.40 ± 2.02	87.50 ± 2.75	24.6 ± 14.7
	SU	C	96.89 ± 1.49	95.11 ± 1.58	31.9 ± 1.8		*SU*	E	86.82 ± 0.94	*87.71 ± 1.42*	52.1 ± 10.8
	RBF	C	90.67 ± 0.61	90.77 ± 1.31	29.7 ± 3.0		RBF	C	78.51 ± 1.90	76.69 ± 3.33	124.2 ± 24.1
	PSU	C	99.44 ± 0.61	**98.01 ± 0.96**	29.3 ± 4.5		PSU	E	86.71 ± 1.00	87.38 ± 1.17	43.3 ± 15.9
	PRBF	C	98.74 ± 0.69	*97.41 ± 1.11*	26.6 ± 3.9		PRBF	C	84.40 ± 2.17	86.43 ± 4.09	61.4 ± 32.5
	SRBF	C	93.83 ± 1.85	92.91 ± 1.61	32.6 ± 2.0		**SRBF**	E	86.16 ± 0.93	**88.02 ± 1.01**	62.2 ± 21.0
German	PU	C	74.13 ± 1.37	71.24 ± 1.52	47.9 ± 19.5	Glass	PU	C	75.90 ± 4.74	65.16 ± 4.17	62.4 ± 7.1
	SU	C	81.21 ± 1.39	73.07 ± 1.64	93.9 ± 21.8		**SU**	E	75.22 ± 2.52	**67.67 ± 3.49**	83.8 ± 6.0
	RBF	C	73.71 ± 0.88	71.69 ± 1.32	213.0 ± 30.4		RBF	C	66.29 ± 2.81	64.91 ± 4.74	108.1 ± 9.4
	PSU	E	79.40 ± 1.60	*73.12 ± 1.71*	86.0 ± 21.9		PSU	E	75.78 ± 2.13	66.23 ± 3.91	82.8 ± 6.5
	PRBF	E	72.60 ± 1.65	71.25 ± 1.45	119.6 ± 43.9		PRBF	C	74.76 ± 2.96	65.03 ± 3.96	89.3 ± 9.5
	SRBF	E	79.11 ± 2.05	**73.44 ± 1.61**	105.7 ± 34.0		*SRBF*	C	74.35 ± 2.53	*67.17 ± 4.36*	97.9 ± 7.5
Heart	PU	C	86.44 ± 1.57	83.58 ± 2.15	11.0 ± 2.6	Ionos	PU	E	96.79 ± 1.13	91.15 ± 2.20	39.1 ± 9.8
	SU	E	86.06 ± 0.90	**86.91 ± 2.06**	17.2 ± 2.0		*SU*	E	98.83 ± 0.75	*92.61 ± 1.56*	73.9 ± 10.2
	RBF	C	85.78 ± 0.79	81.37 ± 2.54	27.6 ± 4.2		RBF	C	91.39 ± 1.27	90.42 ± 2.60	158.5 ± 18.9
	PSU	C	88.27 ± 0.95	*85.93 ± 2.27*	16.9 ± 2.6		PSU	C	99.07 ± 0.55	92.11 ± 1.88	67.9 ± 12.6
	PRBF	E	84.14 ± 1.64	82.79 ± 2.57	20.4 ± 4.4		PRBF	E	94.29 ± 2.18	91.34 ± 2.41	93.9 ± 16.4
	SRBF	E	85.86 ± 1.63	85.49 ± 1.96	18.9 ± 4.3		**SRBF**	C	98.13 ± 0.88	**93.22 ± 1.61**	100.2 ± 16.6
Newth	*PU*	C	99.25 ± 0.55	*96.85 ± 2.71*	16.4 ± 3.2	Pima	PU	E	77.27 ± 0.65	78.45 ± 1.29	12.5 ± 2.1
	SU	C	98.72 ± 0.65	94.88 ± 2.26	22.1 ± 3.6		**SU**	E	76.88 ± 0.60	**79.98 ± 1.53**	18.6 ± 2.0
	RBF	C	95.67 ± 0.62	95.00 ± 2.01	24.2 ± 3.8		RBF	C	74.81 ± 1.54	75.66 ± 2.56	26.8 ± 3.1
	PSU	E	99.36 ± 0.60	96.36 ± 2.77	20.3 ± 3.7		PSU	E	76.95 ± 0.77	78.89 ± 1.87	17.0 ± 2.8
	PRBF	E	98.63 ± 0.97	**97.96 ± 2.45**	19.7 ± 3.8		PRBF	E	77.46 ± 0.51	78.54 ± 1.44	17.0 ± 1.5
	SRBF	C	94.70 ± 1.38	95.62 ± 2.20	23.5 ± 3.3		*SRBF*	E	77.40 ± 0.48	*79.64 ± 1.29*	22.6 ± 3.0
Vote	*PU*	E	95.14 ± 0.97	*95.52 ± 2.26*	5.9 ± 2.4	Zoo	**PU**	C	98.20 ± 1.74	**94.80 ± 4.48**	29.7 ± 2.8
	SU	E	95.88 ± 0.68	94.26 ± 1.91	14.0 ± 4.5		*SU*	C	99.34 ± 1.02	*92.67 ± 4.34*	49.9 ± 4.5
	RBF	C	91.55 ± 1.52	87.50 ± 2.77	30.4 ± 7.6		RBF	C	78.46 ± 2.73	75.07 ± 5.00	66.0 ± 1.5
	PSU	E	95.59 ± 0.63	94.57 ± 1.96	12.8 ± 4.3		PSU	C	98.46 ± 1.80	92.13 ± 5.09	42.2 ± 6.8
	PRBF	C	95.88 ± 0.45	**96.02 ± 0.88**	13.3 ± 8.2		PRBF	E	95.44 ± 3.60	91.33 ± 5.95	35.3 ± 6.0
	SRBF	C	96.26 ± 0.42	94.54 ± 2.23	16.0 ± 7.3		SRBF	E	97.02 ± 3.86	90.40 ± 4.77	48.6 ± 5.2

Elit.: Elitism; the represented results correspond only to the best performing final mean (E: best training cross-entropy individual; C: best CCR_T individual). The best result in the generalization set has been represented in bold face and the second best result in italic face.

(ANOVA) test [35] with the CCR_G of the best models as the test variable has been carried out (previously evaluating if the CCR_G values follow normal distribution, using a Kolmogorov–Smirnov test). Based on the hypothesis of normality, ANOVA examines the effects of some quantitative or qualitative variables (called factors) on one quantitative response. Here, the objective of this analysis is to determine if the influence of the basis function used in the hidden layer of the neural nets is significant in mean with respect to the CCR_G obtained by the CBFEP algorithm. Thus, the linear model for CCR_G has the form

$$CCR_{G_{ij}} = \mu + B_i + e_{ij}$$

for $i = 1, \ldots, 6$ and $j = 1, 2, \ldots, 30$. Factor B_i analyzes the effect on the CCR_G in the i-th level of that factor, where B_i represents the typology of the basis functions used in the hidden layer of the net, with levels: $(i = 1)$ for pure PUs; $(i = 2)$ for pure SUs; $(i = 3)$ for pure RBFs; $(i = 4)$ for combined PSU; $(i = 5)$ for combined PRBF; $(i = 6)$ for combined SRBF. The term μ is the fixed effect that is common to all the populations. The term e_{ij} is the influence on the result of everything that could not be otherwise assigned, or of random factors.

Thus, 180 simulations were carried out for each dataset, corresponding to the 30 runs of each of the six levels for the factor. First, a Levene test (L) [30] is performed to evaluate the equality of variances (if $\alpha < p$-value, then the variances of CCR_G are equal) and then the Snedecor's F tests is performed for assessing if the influence of the basis function used in the hidden layer of the neural nets is significant in mean with respect to the CCR_G. The p-value of both tests are presented in Table 3. The F test shows that the effect of basis function typology used in the hidden layer is statistically significant in all datasets for the CCR_G at a significance level of 5%, i.e., $\alpha > p$-value in all datasets (see third column in Table 3).

Table 3
p-value of the Levene test (L) and the F test of ANOVA I applied to the CCR_G and # links values.

Dataset	p-Value			
	CCR_G		#links	
	L test	F test	L test	F test
Balance	0.048*	0.000*	0.008*	0.000*
Card	0.000*	0.000*	0.000*	0.000*
German	0.935	0.000*	0.000*	0.000*
Glass	0.948	0.033*	0.013*	0.000*
Ionos	0.033*	0.000*	0.001*	0.000*
Newthyroid	0.033*	0.000*	0.538	0.000*
Vote	0.000*	0.000*	0.000*	0.000*
Pima	0.000*	0.000*	0.013*	0.000*
Heart	0.591	0.000*	0.004*	0.000*
Zoo	0.425	0.000*	0.000*	0.000*

* Statistically significant differences with p-value < 0.05.

Table 4
p-value of the Tamhane and Tukey tests for CCR_G and ranking of the different basis function proposal based on this multiple comparison test.

$H_0 \equiv \hat{\mu}_{(I)} = \hat{\mu}_{(J)}$		p-Value									
(I)	(J)	Balance Tam.	Card Tam.	German Tukey	Glass Tukey	Heart Tukey	Ionos. Tam.	Newth. Tam.	Pima Tam.	Vote Tam.	Zoo Tukey
PU	SU	0.009*	1.000	0.000*	0.176	0.000*	0.110	+0*	0.002*	0.289	0.558
	RBF	0.000*	0.000*	0.866	1.000	0.003*	0.986	0.059	0.000*	0.000*	0.000*
	PSU	0.000*	1.000	0.000*	0.916	0.001*	0.690	1.000	0.995	0.735	0.303
	PRBF	0.043*	0.984	1.000	1.000	0.763	1.000	0.799	1.000	0.991	0.080
	SRBF	0.000*	0.998	0.000*	0.412	0.017*	0.002*	0.592	0.012*	0.771	0.010*
SU	PU	0.009*	1.000	0.000*	0.176	0.000*	0.110	0.050*	0.002*	0.289	0.558
	RBF	0.000*	0.000*	0.009*	0.103	0.000*	0.007*	1.000	0.000*	0.000*	0.000*
	PSU	0.000*	0.998	1.000	0.752	0.552	0.999	0.340	0.218	1.000	0.998
	PRBF	0.000*	0.839	0.000*	0.136	0.000*	0.371	0.000*	0.006*	0.001*	0.904
	SRBF	0.000*	0.998	0.937	0.997	0.153	0.676	0.967	0.998	1.000	0.489
RBF	PU	0.000*	0.000*	0.866	1.000	0.003*	0.986	0.059	0.000*	0.000*	0.000*
	SU	0.000*	0.000*	0.009*	0.103	0.000*	0.007*	1.000	0.000*	0.000*	0.000*
	PSU	0.000*	0.000*	0.006*	0.816	0.000*	0.083	0.409	0.000*	0.000*	0.000*
	PRBF	0.000*	0.000*	0.880	1.000	0.153	0.928	0.000*	0.000*	0.000*	0.000*
	SRBF	0.000*	0.000*	0.000*	0.279	0.000*	0.000*	0.989	0.000*	0.000*	0.000*
PSU	PU	0.000*	1.000	0.000*	0.916	0.001*	0.690	1.000	0.995	0.735	0.303
	SU	0.000*	0.998	1.000	0.752	0.552	0.999	0.340	0.218	1.000	0.998
	RBF	0.000*	0.000*	0.006*	0.816	0.000*	0.083	0.409	0.000*	0.000*	0.000*
	PRBF	0.360	0.980	0.000*	0.872	0.000*	0.944	0.271	1.000	0.010*	0.989
	SRBF	0.000*	0.341	0.967	0.949	0.975	0.226	0.988	0.701	1.000	0.755
PRBF	PU	0.043*	0.984	1.000	1.000	0.763	1.000	0.799	1.000	0.991	0.080
	SU	0.000*	0.839	0.000*	0.136	0.000*	0.371	0.000*	0.006*	0.001*	0.904
	RBF	0.000*	0.000*	0.880	1.000	0.153	0.928	0.000*	0.000*	0.000*	0.000*
	PSU	0.360	0.980	0.000*	0.872	0.000*	0.944	0.271	1.000	0.010*	0.989
	SRBF	0.000*	0.515	0.000*	0.342	0.000*	0.013*	0.004*	0.043*	0.025*	0.978*
SRBF	PU	0.000*	0.998	0.000*	0.412	0.017*	0.002*	0.592	0.012*	0.771	0.010*
	SU	0.000*	0.998	0.937	0.997	0.153	0.676	0.967	0.998	1.000	0.489
	RBF	0.000*	0.000*	0.000*	0.279	0.000*	0.000*	0.989	0.000*	0.000*	0.000*
	PSU	0.000*	0.341	0.967	0.949	0.975	0.226	0.988	0.701	1.000	0.755
	PRBF	0.000*	0.515	0.000*	0.342	0.000*	0.013*	0.004*	0.043*	0.025*	0.978

Dataset	Means ranking of the CCR_G
Balance	$\boldsymbol{\mu}_{\mathbf{PSU}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{PU}} > \mu_{\mathrm{SU}} > \mu_{\mathrm{SRBF}} > \mu_{\mathrm{RBF}};\ \mu_{\mathrm{PSU}} > \mu_{\mathrm{PU}};$
Card	$\boldsymbol{\mu}_{\mathbf{SRBF}} \geqslant \mu_{\mathrm{SU}} \geqslant \mu_{\mathrm{PU}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{PRBF}} > \mu_{\mathrm{RBF}}$
German	$\boldsymbol{\mu}_{\mathbf{SRBF}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{SU}} > \mu_{\mathrm{RBF}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{PRBF}}$
Glass	$\boldsymbol{\mu}_{\mathbf{SU}} \geqslant \mu_{\mathrm{SRBF}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{PU}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{RBF}}$
Heart	$\boldsymbol{\mu}_{\mathbf{SU}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{SRBF}} > \mu_{\mathrm{PU}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{RBF}};\ \mu_{\mathrm{PU}} > \mu_{\mathrm{RBF}}$
Ionos.	$\boldsymbol{\mu}_{\mathbf{SRBF}} \geqslant \mu_{\mathrm{SU}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{PU}} \geqslant \mu_{\mathrm{RBF}};\ \mu_{\mathrm{SRBF}} > \mu_{\mathrm{PRBF}};\ \mu_{\mathrm{SU}} > \mu_{\mathrm{RBF}}$
Newth.	$\boldsymbol{\mu}_{\mathbf{PRBF}} \geqslant \mu_{\mathrm{PU}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{SRBF}} \geqslant \mu_{\mathrm{RBF}} \geqslant \mu_{\mathrm{SU}};\ \mu_{\mathrm{PRBF}} > \mu_{\mathrm{SRBF}};\ \mu_{\mathrm{PU}} > \mu_{\mathrm{RBF}}$
Pima	$\boldsymbol{\mu}_{\mathbf{SU}} \geqslant \mu_{\mathrm{SRBF}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{PU}} > \mu_{\mathrm{RBF}};\ \mu_{\mathrm{SU}} > \mu_{\mathrm{PRBF}}$
Vote	$\boldsymbol{\mu}_{\mathbf{PRBF}} \geqslant \mu_{\mathrm{PU}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{SRBF}} \geqslant \mu_{\mathrm{SU}} > \mu_{\mathrm{RBF}};\ \mu_{\mathrm{PRBF}} > \mu_{\mathrm{PSU}}$
Zoo	$\boldsymbol{\mu}_{\mathbf{PU}} \geqslant \mu_{\mathrm{SU}} \geqslant \mu_{\mathrm{PSU}} \geqslant \mu_{\mathrm{PRBF}} \geqslant \mu_{\mathrm{SRBF}} > \mu_{\mathrm{RBF}};\ \mu_{\mathrm{PU}} > \mu_{\mathrm{SRBF}}$

* Statistically significant differences with p-value<0.05; Tam.: Tamhane test; Tukey: Tukey test. $\mu_A \geqslant \mu_B$: topology A yields better results than topology B, but the differences are not significant; $\mu_A > \mu_B$: topology A yields better results than topology B with significant differences. The binary relation $\geqslant$ is not transitive.

After carrying out these tests, a post-hoc multiple comparison test has been performed on the average CCR_G obtained. If the hypothesis that the variances are equal is accepted (using the Levene test presented in Table 3), a Tukey test [35] is carried out, and if not, a Tamhane test [48] is done. Both tests aim to rank the average of each level in each factor, in order to locate the level whose mean CCR_G is significantly better than the mean CCR_G of all the other levels.

Table 4 shows the results obtained following the above methodology, including the test performed (Tukey or Tamhane) and the performance ranking of the different basis functions. For the Balance dataset, the best result is obtained for PSU or PRBF combined basis functions. That is, the average CCR_G obtained with PSU is better than the averages obtained with the other combined models, except PRBF. For Card, German and Ionos, the most descriptive result is obtained with the combined model SRBF, although it is not significantly better than all the others in any case: it results in a multiple draw with four of the remaining methods for the Card dataset, and with two methods for German and Ionos. Very similar conclusions can be obtained from the analysis of almost all the other datasets, except for the Zoo dataset, where the procedures based on a single basis function type result in significantly better results than those based on CBF models.

Table 5 represents the results obtained following a similar methodology with the number of links (# links) of the obtained models, including the test performed (Tukey or Tamhane) and the number of link ranking for different basis functions. The models that have the significantly lowest number of connections are pure PU models, while pure RBF models have the highest number of connections. In this way, when using pure basis function models, the number of PU model connections is always lower than that of all the others, and when using CBF models, the number of connections of those models formed with PUs (PRBF and PSU) is also always lower than the number obtained with SRBF. This fact is due to the properties of PU basis function: PUs have the ability

Table 5
p-values of the Tamhane and Tukey tests for #links and ranking of the different basis function proposal based on this multiple comparison test.

$H_0 \equiv \hat{\mu}_{(I)} = \hat{\mu}_{(J)}$		*p*-Value									
(I)	(J)	Balance Tam.	Card Tam.	German Tam.	Glass Tam.	Heart Tam.	Ionos. Tam.	Newth. Tukey	Pima Tam.	Vote Tam.	Zoo Tam.
PU	SU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	RBF	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	PSU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	PRBF	0.003*	0.000*	0.000*	0.000*	0.000*	0.000*	0.006*	0.000*	0.001*	0.001*
	SRBF	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
SU	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	RBF	0.025*	0.000*	0.000*	0.000*	0.000*	0.000*	0.224	0.000*	0.000*	0.000*
	PSU	0.102	0.206	0.938	1.000	1.000	0.544	0.392	0.208	0.997	0.000*
	PRBF	0.000*	0.911	0.091	0.151	0.012*	0.000*	0.112	0.016*	1.000	0.000*
	SRBF	0.907	0.312	0.843	0.000*	0.621	0.000*	0.668	0.000*	0.964	0.996
RBF	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	SU	0.025*	0.000*	0.000*	0.000*	0.000*	0.000*	0.224	0.000*	0.000*	0.000*
	PSU	1.000	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	PRBF	0.020*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	SRBF	0.001*	0.000*	0.000*	0.000*	0.000*	0.000*	0.976	0.000*	0.000*	0.000*
PSU	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	SU	0.102	0.206	0.938	1.000	1.000	0.544	0.392	0.208	0.997	0.000*
	RBF	1.000	0.000*	0.000*	0.000*	0.000*	0.000*	0.001*	0.000*	0.000*	0.000*
	PRBF	0.249	0.128	0.008*	0.052	0.008*	0.000*	0.988	1.000*	1.000	0.002*
	SRBF	0.013*	0.004*	0.146	0.000*	0.451	0.000*	0.010	0.000*	0.498	0.003*
PRBF	PU	0.003*	0.000*	0.000*	0.000*	0.000*	0.000*	0.006*	0.000*	0.001*	0.001*
	SU	0.000*	0.911	0.091	0.151	0.012*	0.000*	0.112	0.016	1.000	0.000*
	RBF	0.020*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	PSU	0.249	0.128	0.008*	0.052	0.008*	0.000*	0.988	1.000*	1.000	0.002*
	SRBF	0.000*	1.000	0.945	0.004*	0.948	0.901	0.001*	0.000*	0.946	0.000*
SRBF	PU	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*	0.000*
	SU	0.907	0.312	0.843	0.000*	0.621	0.000*	0.668	0.000*	0.964	0.996
	RBF	0.001*	0.000*	0.000*	0.000*	0.000*	0.000*	0.976	0.000*	0.000*	0.000*
	PSU	0.013*	0.004*	0.146	0.000*	0.451	0.000*	0.010	0.000*	0.498	0.003*
	PRBF	0.000*	1.000	0.945	0.004*	0.948	0.901	0.001*	0.000*	0.946	0.000*

Dataset	Means Ranking of the # Links
Balance	$\mu_{PU} < \mu_{PRBF} < \mu_{PSU} \leqslant \mu_{RBF} \leqslant \mu_{SU} \leqslant \mu_{SRBF}$; $\mu_{PSU} < \mu_{SU}$; $\mu_{RBF} < \mu_{SRBF}$
Card	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{PRBF} \leqslant \mu_{SRBF} < \mu_{RBF}$; $\mu_{PSU} < \mu_{PRBF}$
German	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{SRBF} \leqslant \mu_{PRBF} < \mu_{RBF}$; $\mu_{SU} < \mu_{PRBF}$
Glass	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{PRBF} < \mu_{SRBF} < \mu_{RBF}$; $\mu_{PSU} < \mu_{PRBF}$
Heart	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{SRBF} \leqslant \mu_{PRBF} < \mu_{RBF}$; $\mu_{SU} < \mu_{PRBF}$
Ionos	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{SU} < \mu_{PRBF} \leqslant \mu_{SRBF} < \mu_{RBF}$
Newth.	$\mu_{PU} < \mu_{PRBF} \leqslant \mu_{PSU} \leqslant \mu_{SU} \leqslant \mu_{SRBF} \leqslant \mu_{RBF}$; $\mu_{PSU} < \mu_{SRBF}$
Pima	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{PRBF} \leqslant \mu_{SU} < \mu_{SRBF} < \mu_{RBF}$
Vote	$\mu_{PU} < \mu_{PSU} \leqslant \mu_{PRBF} \leqslant \mu_{SU} \leqslant \mu_{SRBF} < \mu_{RBF}$
Zoo	$\mu_{PU} < \mu_{PRBF} < \mu_{PSU} < \mu_{SRBF} \leqslant \mu_{SU} < \mu_{RBF}$

* Statistically significant differences with *p*-value < 0.05; Tam.: Tamhane test; Tukey: Tukey test. $\mu_A \geqslant \mu_B$: topology A results in a lower number of connections than topology B, but the differences are not significant; $\mu_A > \mu_B$: topology A results in a lower number of connections than topology B with significant differences. The binary relation $\geqslant$ is not transitive.

to capture interactions between input variables, which means they do not need so many connections.

In some datasets, the hybridization of projection type basis functions with RBFs yields better generalization results than corresponding pure RBF models, with a significantly lower number of connections. In general, there is no definite conclusion about the type of basis function applicable to a given dataset, based on the statistical tests. Hybridization enhances generalization accuracy in Balance and Ionosphere datasets, while using pure basis function models are suggested for Heart and Zoo datasets. Moreover, CCR_G values are significantly more homogeneous using hybrid models for the Balance, Card and Vote datasets and there are no significant differences in the remaining datasets.

6. Conclusions

A hybrid CBF neural network model has been proposed, using a possible combination of two different transfer projection functions (SU and PU) and/or kernel functions (RBF) in the hidden layer of a feed-forward neural network. The different CBF models proposed have been designed with an EA (CBFEP) constructed specifically for the typology of a combination of basis functions, taking into account the specific characteristics of each pure and CBF model. The performance in the different datasets has been improved by considering the best individuals both in *CCR* and cross-entropy training errors (*double elitism*). The evaluation of the model an the algorithm for ten datasets has shown results that are comparable to those of other classification techniques found in machine learning literature [26].

The results obtained confirm the theory that a classification task can be performed using a model with two different components [13]: one associated with projection functions (PU or SU) and the other associated with kernel functions (RBF). Determining the most adequate projection function strongly depends on the dataset tackled, but, in general, the SRBF hybrid models have better accuracy than PRBF, especially in those datasets with two classes where only one discriminant function exists. The capacity for generalization using basis functions with sufficiently diverse global-local discriminator characteristics (i.e. SRBF or PRBF models) is similar to or greater than the generalization capability of pure PU, SU or RBF models. Moreover, the hybridization of the two projection basis functions (PU and SU) did not present better accuracy than the remaining hybridizations but it presented a significantly lower number of connections (allowing smaller and more interpretable models) in the Card, German, Glass, Heart, Ionosphere and Vote datasets. The generalization capability of PSU models is similar to that obtained with their corresponding pure models.

The CBF models and the CBFEP algorithm maintain the performance and the number of links of corresponding pure models in several problems, or they even improve accuracy and/or model sizes. In general, the number of connections of pure models formed by PUs is always lower than that of all the others, and the number of connections of those CBF models using PUs (PRBF and PSU) is also lower than the number obtained with SRBF.

We are currently working on a hypothesis about the case in which a modelling structure using combined PU and RBF basis functions could possibly fit better than SU and RBF pure basis functions, associated to the existence of a big difference between the values of a relatively small number of data at the domain borders and in-domain inner data. This hypothesis is going to be evaluated empirically using a noise level measurement obtained for each dataset, and, in this way, the best combined model (SRBF or PRBF) would be determined according to the characteristics of the dataset in question.

Acknowledgments

This work has been partially subsidized by TIN 2005-08386-C05-02 projects of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT) and FEDER funds. The research done by P.A. Gutiérrez and J.C. Fernández has been financed, respectively, by the FPU (Grant reference AP-2006-01746) and the FPI (Grant reference BES-2006-12543) Predoctoral Programs (Spanish Ministry of Education and Science).

References

[1] J. Alcala-Fdez, L. Sánchez, S. García, M.J.D. Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: A software tool to assess evolutionary algorithms for data mining problems. Soft Computing, 2007 (in press). Q2
[2] P.J. Angeline, G.M. Saunders, J.B. Pollack, An evolutionary algorithm that constructs recurrent neural networks, IEEE Transactions on Neural Networks 5 (1) (1994) 54–65.
[3] A. Asuncion, D.J. Newman, UCI Maching Learning Repository, ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩, Irvine, CA: University of California, School of Information and Computer Science, 2007
[4] A.S. Atukorale, T. Downs, P.N. Suganthan, Boosting HONG networks, Neurocomputing 51 (2003) 75–86.
[5] S.A. Billings, G.L. Zheng, Radial basis function network configuration using genetic algorithms, Neural Networks 8 (1995) 877–890.
[6] C.M. Bishop, Improving the generalization properties of radial basis function neural networks, Neural Computation 3 (4) (1991) 579–581.
[7] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
[8] L.N. Castro, E.R. Hruschka, R.J.G.B. Campello, An evolutionary clustering technique with local search to design RBF neural network classifiers, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2004, pp. 2083–2088.
[9] S. Chen, S.A. Billings, C.F.N. Cowan, P.W. Grant, Practical identification of NARMAX models using radial basis functions, International Journal of Control 52 (1990) 1327–1350.
[10] S. Coen, N. Intrator, Global optimization of RBF networks, Technical Report, S.o.C. Science, Tel-Aviv University.
[11] S. Cohen, N. Intrator, A hybrid projection-based and radial basis function architecture: initial values and global optimization, Pattern Analysis and Applications 5 (2002) 113–120.
[12] C. Darken, J. Moody, Fast adaptive K-means clustering: some empirical results, in: Proceedings of the IEEE INNS International Joint Conference on Neural Networks, 1990, pp. 233–238.
[13] D. Donoho, Projection based in approximation and a duality with kernel methods, The Annals of Statistics 17 (1989) 58–106.
[14] W. Duch, R. Adamczak, G.H.F. Diercksen, Constructive density estimation network based on several different separable transfer functions, in: Proceedings of the 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 107–112.
[15] W. Duch, N. Jankowski, Transfer functions hidden possibilities for better neural networks, in: Proceedings of the 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 81–94.
[16] R. Durbin, D. Rumelhart, Products units: a computationally powerful and biologically plausible extension to backpropagation networks, Neural Computation 1 (1989) 133–142.
[17] D.B. Fogel, A.J. Owens, M.J. Wals, Artificial Intelligence Through Simulated Evolution, Wiley, New York, 1966.
[18] J.A.S. Freeman, D. Saad, Learning and generalization in radial basis function networks, Neural Computation 7 (5) (1995) 1000–1020.
[19] J. Friedman, Multivariate adaptive regression splines (with discussion), The Annals of Statistics 19 (1991) 1–141.
[20] J.H. Friedman, W. Stuetzle, Projection pursuit regression, Journal of the American Statistical Association 76 (1981) 817–823.
[21] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control system—a survey, Automatica 28 (1992) 1083–1112.
[22] A. Ismail, A.P. Engelbrecht, Global optimization algorithms for training product units neural networks, in: Proceedings of the International Joint Conference on Neural Networks IJCNN'2000, Italy, 2000, pp. 132–137. Q3
[23] B.C. Iulian, Hybrid feedforward neural networks for solving classification problems, Neural Processing Letters 16 (1) (2002) 81–91.
[24] N. Jankowski, W. Duch, Optimal transfer function neural networks, in: Procedings of the 9th European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 101–106.
[25] D.J. Janson, J.F. Frenzel, Training product unit neural networks with genetic algorithms, IEEE Expert 8 (5) (1993) 26–33.
[26] N. Landwehr, M. Hall, Logistic model trees, Machine Learning 59 (2005) 161–205.
[27] L.R. Leerink, Learning with products units, Advances in Neural Networks Processing Systems 7 (1995) 537–544.
[28] M. Lehtokangas, J. Saarinen, Centroid based Multilayer Perceptron Networks, Neural Processing Letters 7 (1998) 101–106.
[29] F.H.F. Leung, H.K. Lam, S.H. Ling, P.K.S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks 14 (1) (2003) 79–88.
[30] H. Levene, In Contributions to Probability and Statistics, Stanford University Press, 1960.
[31] R.P. Lippmann, Pattern classification using neural networks, IEEE Communications Magazine 27 (1989) 47–64.
[32] A.C. Martínez-Estudillo, C. Hervás-Martínez, F.J. Martínez-Estudillo, N. García, Hybridation of evolutionary algorithms and local search by means of a clustering method, IEEE Transaction on Systems, Man and Cybernetics, Part. B: Cybernetics 36 (3) (2006) 534–546.
[33] A.C. Martínez-Estudillo, F.J. Martínez-Estudillo, C. Hervás-Martínez, N. García, Evolutionary product unit based neural networks for regression, Neural Networks 19 (4) (2006) 477–486.
[34] F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez, A.C. Martínez-Estudillo, Evolutionary product-unit neural networks classifiers, Neurocomputing, 2008 (in press).
[35] R.G. Miller, Beyond ANOVA, Basics of Applied Statistics, Chapman & Hall, London, 1996.
[36] M.T. Musavi, W. Ahmed, K.H. Chan, K.B. Farms, D.M. Hummels, On the training of radial basis function classifiers, Neural Networks 5 (1992) 595–603.
[37] M.J.L. Orr, Regularisation in the selection of radial basis function centres, Neural Computation 7 (3) (1995) 606–623.
[38] C. Panchapakesan, M. Palaniswami, D. Ralph, C. Manzie, Effects of moving the centers in an RBF network, IEEE Transactions on Neural Networks 13 (6) (2002) 1299–1307.
[39] Y.H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, IEEE Computer 25 (5) (1992) 76–79.
[40] J. Park, I.W. Sandberg, Universal approximation using radial basis function networks, Neural Computation 3 (2) (1991) 246–257.
[41] R. Reed, Pruning algorithms. A survey, IEEE Transactions on Neural Networks 4 (1993) 740–747.
[42] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by backpropagating errors, Nature 323 (1986) 533–536.

[43] A. Sakurai, Tighter bounds of the VC-dimension of three layer networks, in: Proceedings of the World Congress on Neural Networks, Erlbaum, Hillsdale, New Jersey, 1993, pp. 540–543.
[44] M. Schmitt, On the complexity of computing and learning with multiplicative neural networks, Neural Computation 14 (2001) 241–301.
[45] M. Schmitt, Radial basis function neural networks have superlinear VC dimension, in: Proceedings of the COLT/EuroCOLT2001, Lecture Notes in Artificial Intelligence, vol. 2111, 2001, pp. 14–30.
[46] D.F. Specht, A general regression neural network, IEEE Transactions on Neural Networks 2 (6) (1991) 568–576.
[47] J.A.K. Suykens, J.P.L. Vandewalle, B.L.R.D. Moor, Artificial Neural Networks for Modelling and Control of Non-linear Systems, Kluwer Academic Publishers, USA, 1996.
[48] A.C. Tamhane, D.D. Dunlop, Statistics and Data Analysis, Prentice Hall, Englewood Cliffs, NJ, 2000.
[49] E.K. Tang, P.N. Suganthan, X. Yao, An analysis of diversity measures, Machine Learning 65 (2006) 247–271.
[50] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, 1999.
[51] S. Ventura, C. Romero, A. Zafra, J.A. Delgado, C. Hervás-Martínez, JCLEC: a JAVA framework for evolutionary computation, Soft Computing 12 (4) (2007) 381–392.
[52] D. Wedge, D. Ingram, D. McLean, C. Mingham, Z. Bandar, On global-local artificial neural networks for function approximation, IEEE Transactions on Neural Networks 17 (4) (2006) 942–952.
[53] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE 87 (9) (1999) 1423–1447.
[54] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, IEEE Transactions on Neural Networks 8 (3) (1997) 694–713.
[55] Z.Q. Zhao, D.S. Huang, A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability, Applied Mathematical Modelling 31 (2007) 1271–1281.
[56] Q. Zhu, Y. Cai, L. Liu, A global learning algorithm for a RBF network, Neural Networks 12 (1999) 527–540.

Pedro A. Gutiérrez-Peña was born in Cordoba, Spain, in 1982. He received his B.S. degree in Computer Science from the University of Seville, Seville, Spain, in 2006. He is currently working toward his Ph.D. degree in the Department of Computer Science and Numerical Analysis (University of Cordoba, Spain), in the area of computer science and artificial intelligence. His current interests include neural networks and their applications, evolutionary computation and hybrid algorithms.

César Hervás-Martínez was born in Cuenca, Spain. He received his B.S. degree in Statistics and Operating Research from the Universidad Complutense, Madrid, Spain, in 1978 and his Ph.D. degree in Mathematics from the University of Seville, Seville, Spain, in 1986. He is a Professor with the University of Córdoba in the Department of Computing and Numerical Analysis in the area of computer science and artificial intelligence and an Associate Professor in the Department of Quantitative Methods in the School of Economics. His current research interests include neural networks, evolutionary computation, and the modelling of natural systems.

Mariano Carbonero-Ruz was born in Córdoba, Spain. He received his M.Sc. degree in Mathematics in 1987 and his Ph.D. degree in Mathematics in 1995, speciality Statistics and Operational Research, both from the University of Seville, Seville, Spain. Later, he received his M.Sc. degree in Economics Sciences in 2000 from UNED, Spain. He develops his research in applied statistics in machine learning and Economic Sciences. He is currently a Professor in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain.

Juan C. Fernández-Caballero was born in Peñarroya-Pueblonuevo, Córdoba, Spain, in 1980. He received the B.S. degree in Computer Science from the University of Granada, Spain, in 2005. He is currently a Ph.D. Student in the Department of Computing and Numerical Analysis, University of Córdoba, Spain, in the area of computer science and artificial intelligence. His current areas of interest include neural networks and applications, evolutionary computation and multiobjective optimization.

2.3.2. Combinando Regresión y Clasificación para la Cuantificación de Picos Altamente Solapados obtenidos por un Proceso de Electroforesis Capilar mediante el Desarrollo de Modelos de Redes Neuronales Evolutivas de Unidades Producto y Unidades Sigmoide - *Combining classification and regression approaches for the quantification of highly overlapping capillary electrophoresis peaks by using evolutionary sigmoidal and product unit neural networks*

- C. Hervás-Martínez, P.A. Gutiérrez, M. Silva, J.M. Serrano. Combining classification and regression approaches for the quantification of highly overlapping capillary electrophoresis peaks by using evolutionary sigmoidal and product unit neural networks. Journal of Chemometrics 21:12 (2007) 567-577.
    - Estado: Publicado.
    - Índice de Impacto (JCR 2007): 1.367.
    - Área de conocimiento: *Automation & Control Systems.* Ranking 9/52. Primer Cuartil.
    - Área de conocimiento: *Chemistry, Analytical.* Ranking 40/70. Tercer Cuartil.
    - Área de conocimiento: *Computer Science, Artificial Intelligence.* Ranking 32/93. Segundo Cuartil.
    - Área de conocimiento: *Instruments & Instrumentation.* Ranking 16/55. Segundo Cuartil.
    - Área de conocimiento: *Mathematics, Interdisciplinary Applications.* Ranking 16/74. Primer Cuartil.
    - Área de conocimiento: *Statistics & Probability.* Ranking 20/91. Primer Cuartil.

JOURNAL OF CHEMOMETRICS
J. Chemometrics 2007; **21**: 567–577
Published online 4 September 2007 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/cem.1082

Combining classification and regression approaches for the quantification of highly overlapping capillary electrophoresis peaks by using evolutionary sigmoidal and product unit neural networks

Cesar Hervás[1], Pedro Antonio Gutierrez[1], Manuel Silva[2]* and Juan Manuel Serrano[2]
[1]University of Córdoba, Computer Science, Spain
[2]University of Cordoba, Analytical Chemistry, Spain

Received 27 February 2007; Revised 13 July 2007; Accepted 25 July 2007

This is a study of the potential of neural networks built by using different transfer functions (sigmoidal, product and sigmoidal–product units) designed by an evolutionary algorithm to quantify highly overlapping electrophoretic peaks. To test this approach, two aminoglycoside antibiotics, amikacin and paramomycin, were quantified from samples containing either only one component or mixtures of them though capillary zone electrophoresis (CZE) with laser-induced fluorescence (LIF) detection. The three models assayed used as input data the four-parameter Weibull curve associated with the profile of the electrophoretic peak and in some cases the class label for each sample estimated by cluster analysis. The combination of classification and regression approaches allowed the establishment of straightforward network topologies enabling the analytes to be quantified with great accuracy and precision. The best models for mixture samples were provided by product unit neural networks (PUNNs), 4:4:1 (14 weights) for both analytes, after discrimination by cluster analysis, allowing the analytes to be quantified with great accuracy: 8.2% for amikacin and 5.6% for paromomycin within the standard error of prediction for the generalization test, SEP_G. For comparison, partial least square regression was also used for the resolution of these mixtures; it provided a minor accuracy: SEP_G 11.8 and 15.7% for amikacin and paramomycin, respectively. The reduced dimensions of the neural networks models selected enabled the derivation of simple quantification equations to transform the input variables into the output variable. These equations can be more easily interpreted from a chemical point of view than those provided by other ANN models. Copyright © 2007 John Wiley & Sons, Ltd.

KEYWORDS: neural networks; evolutionary algorithm; overlapped peaks; capillary electrophoresis

1. INTRODUCTION

A variety of approaches are found in the literature for resolving and quantifying overlapped chromatographic bands using chemometric methods, in the majority of the cases based on a two-dimensional array of data, such as that generated by high performance liquid chromatography (HPLC) with diode-array detection (DAD) [1–9]. In recent years, capillary zone electrophoresis (CZE) has gained in popularity as an alternative to HPLC for routine analysis, and its application is widespread in many fields of analytical chemistry. Despite its higher resolution with respect to HPLC, sometimes the complete separation of sample components is not fully achieved. Nonetheless, the quantification difficulties from poorly resolved peaks may be overcome mathematically by using chemometric techniques. Multivariate curve resolution methods [10–12], augmented iterative target transformation factor analysis [1], wavelet transforms [13] and second-order derivative electropherograms [14,15] are, among other approaches, the most recently reported for this purpose using the second-order data from CZE–DAD. In the last decade, artificial neural networks (ANNs) have shown their unique merits regarding the great variety of chemometric approaches reported for the classification and regression purposes in many fields of analytical chemistry. Particularly in separation

*Correspondence to: M. Silva, University of Cordoba, Analytical Chemistry, Cordoba, Spain.
E-mail: qa1sirom@uco.es

science, ANNs have been used mainly as tools to optimize the experimental conditions for carrying out separation and, to a lesser extent, for quantification in overlapped HPLC [16–20] peaks and, more recently, for unresolved peaks in CZE [21–25].

Several ANN models have been used to achieve regression analysis, including quantification in overlapped peaks, in analytical chemistry. Although multilayer perceptron (MLP) modeling with sigmoidal unit basis functions (SU) as transfer functions is the most widely used approach [16,20–23], radial basis function (RBF) neural networks [19,21] and, more recently, multiplicative neural networks [26], especially the so-called product unit neural networks (PUNNs) [17,27,28] modeling with product unit basis functions (PU) are other interesting choices depending on the analytical problem addressed. In addition to this type of models where all the nodes of the hidden layer have the same type of activation/transfer functions, hybrid models have also been proposed, where different activation/transfer functions are used for the nodes in the hidden layer.

In this context, it is worth emphasizing the papers by Duch and Jankowski [29] which propose different transfer functions for the nodes in the hidden layer and those by Cohen and Intrator [30,31], supported on the duality between functions based on projection, (SU, PU, etc.) and on kernel typology (RBF). The hybridization of models has been justified theoretically by Donoho [32] who demonstrated that any continuous function can be decomposed into two mutually exclusive functions, such as radial and crest (based on the projection) ones. But to the best of our knowledge there is not any theoretical work associated with the hybridization of the SU/PU in neural networks. Although theoretically this decomposition is justified, in practice it is difficult to separate the different locations of a function and to estimate them by means of a combination of RBFs, and then to estimate the residual function by means of a functional approach based on projections without getting trapped in local optima in the procedure of minimization of error by using gradient methods [33]. A similar idea has been considered in this work: neural network architecture with several transfer functions based on projections.

Thus, this paper deals with the evaluation of different evolutionary ANN models based on SU, PU and a combination of both basis function (SPU) neural networks as powerful tools for the quantification of analytes that provide highly overlapping chromatographic peaks by using first-order data. To test proposed approaches, two aminoglycoside antibiotics, amikacin and paramomycin, were quantified in mixtures using only the analytical information provided by their overlapped CZE peaks (t_i,S_{t_i}: first-order data) registered with a laser-induced fluorescence (LIF) detector. Further details on the analytical methodology are described in a previous paper [34]. Several chemical and chemometric strategies were merged in the proposed approaches:

1. The methodology was extended to more practical and real situations considering the possibility that the analyzed samples could either contain only one component or mixtures of them.
2. A classic procedure was used for classification, such as linear discriminant analysis (LDA) [35], in order to differentiate the three types of analyzed samples.
3. Models of networks with a limited number of inputs were tested. To do so, as in previous papers [16,17], the network inputs were estimated by the Levenberg–Marquardt method in the form of a four-parameter Weibull curve associated with the profile of the electrophoretic band.
4. An additional input was introduced, namely the label associated with the class of sample analyzed, in order to improve the capacity of generalization of the models; this proposal constitutes the first precedent on the joint employment of classification and regression approaches for resolving overlapped chromatographic peaks.
5. The potential of ANNs with hybrid models for activation/transfer functions was evaluated. This paper shows the first research on the use of hybrid models associated with two specific types of functions as functional projection approximators: product and sigmoidal functions, yielding the SPUNNs hybrid model.
6. Evolutionary algorithms were employed for the optimization of the parameters of the model as an alternative to the classic choice based on gradient methods, considering the complexity of the error surfaces obtained by minimizing the squared errors derived from the fitting of the regression models in the training set.

2. THEORY

The aim of the proposed approach is to evaluate the potential of different ANN models: SU, PU and SPUNNs, for the quantification of overlapped chromatographic bands, concretely unresolved CZE peaks, in order to predict the contribution of each component to the overall analytical signal.

2.1. Selection of ANN inputs

The first step of the approach consists of extracting the information from the analytical response (CZE peak) in order to select the inputs for the ANNs. Upon examining the responses (individual and mixture samples), it can be observed that profiles of the CZE peaks (t_i, S_{t_i}) were accurately fitted by least-square regression to a four-parameter Weibull curve defined by S_m (peak height), B (dispersion of S_{t_i} values from S_m), C (associated to the inflection points of the curve and defining the concavity and convexity regions) and t_m (residence time). In addition, the samples were also classified into three categories by using LDA. Classes were defined according to the composition of the samples as follows: G_1, amikacin; G_2, paromomycin; and G_3, mixtures of both antibiotics. In some studies, the label associated with the class was used as input for ANN models.

2.2. ANN models

ANNs modeling with SU, PU and hybrid SPU as transfer functions were used and their features compared in this work. Taking into account that the latter is the most novel approach, its foundation is now described in detail.

Although the use of PUNNs in this context is very recent, the description of its theoretical basis is outside the scope of this paper, although some current papers from the authors can be considered for further explanations [17,27,28].

To estimate the concentrations of the antibiotics in the samples by using SPUNN models, we consider standard regression formulation under a general setting for predictive learning. In the regression formulation, the goal of learning is to estimate an unknown (target) real-valued function $g(x)$ in the relationship

$$y = g(\mathbf{x}) + \varepsilon \tag{1}$$

where ε is an independent and identically distributed zero mean random noise, (whose distribution can depend on x, the distribution of x in the training set also being unknown), x is a multivariate input and y is a scalar output. The estimation is made based on a finite number (n) of samples (training data): $(\mathbf{x}_1,y_1), l = 1, 2, \ldots, n$. To solve regression problems, the family of functions $f(x, w)$ (w is a set of parameters of f) used in the approximation should be previously determined by using a learning method that selects the best model, $f(x, w_0)$, from $f(x, w)$; finally, the quality of the approximation can be expressed by the loss (discrepancy) measure $L(y, f(x, w))$, the squared error being a common loss function used for regression.

The linear models given by $f(x, w) = w^{\mathrm{T}}x$ are the simplest ones used to solve regression problems. It is extremely unlikely that the true function $g(x)$ is actually linear in x. Normally, $g(x)$ is nonlinear and non-additive in x, and representing $g(x)$ by a linear model is usually a convenient, and sometimes necessary, approximation. Another class of useful approximation can be expressed as a linear basis expansion

$$f(\mathbf{x}) = \sum_{j=1}^{M} \beta_j B_j(\mathbf{x},\mathbf{w}_j) \tag{2}$$

where $B_j(\mathbf{x},\mathbf{w}_j)$ are a suitable set of functions or nonlinear transformations of the input vector $\mathbf{x} = (\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_p)$, $B_0(\mathbf{x},\mathbf{w}_j) = 1$ in order to consider bias in the model; β_j are the coefficients from lineal combination that are estimated from the data; $\mathbf{w}_j = (\mathbf{w}_{j1},\mathbf{w}_{j2},\ldots,\mathbf{w}_{jp})$ are the parameters associated with the basis functions; and M is the parameter of regularization of the model, which is associated with the number of basis functions that are necessary and sufficient to minimize some definite function of the error on this matter. In this work, two types of basis functions have been used, namely SU function

$$B_j(\mathbf{x},\mathbf{u}_j) = \frac{1}{1 + e^{-\left(\mathbf{u}_{j0} + \sum_{i=1}^{p} \mathbf{u}_{ji}\mathbf{x}_i\right)}}, \qquad j = 1, \ldots, m_1 \tag{3}$$

and PU function

$$B_k(\mathbf{x},\mathbf{w}_k) = \prod_{i=1}^{p} \mathbf{x}_i^{\mathbf{w}_{ki}}, \qquad k = 1, \ldots, m_2 \tag{4}$$

whose linear combination provided the hybrid function (SPU) used for estimation:

$$f(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{m_1} \alpha_j B_j(\mathbf{x}, \mathbf{u}_j) + \sum_{k=1}^{m_2} \beta_k B_k(\mathbf{x},\mathbf{w}_k) \tag{5}$$

The method involves finding a sufficient number of basis functions (architecture) providing a type of approximate universal function for the estimate function, in such a way, and taking into account that both types of basis functions are universal approximators [17], that for every $\varepsilon > 0$ it should be possible to find a value of m_1 and m_2 as well as the estimators of the parameters $\alpha_0,\alpha_j,\beta_k$, $\hat{\mathbf{u}}_j$ and $\hat{\mathbf{w}}_k$ for $j = 1, \ldots, m_1$ and $k = 1, \ldots, m_2$, that hold:

$$\left\| f(\mathbf{x}) - \left(\alpha_0 + \sum_{j=1}^{m_1} \alpha_j B_j(\mathbf{x},\mathbf{u}_j) + \sum_{k=1}^{m_2} \beta_k B_k(\mathbf{x},\mathbf{w}_k) \right) \right\| < \varepsilon \tag{6}$$

This optimization problem is similar to that involved in the 'projection pursuit' regression model with the special feature being that the 'ridge functions' are exclusively of two types. Evolutionary algorithms similar to those reported by Angeline *et al.* [36], Yao and Liu [37] and García *et al.*[38] and that have been used in this work to obtain the ANN architecture and to estimate model coefficients.

This kind of function topology can be represented by an ANN architecture, as shown in Figure 1. The network has k inputs that represent the independent variables of the model, five in our problem, $m_1 + m_2 = 6$ nodes in the hidden layer and one node in the output layer. The activation of the jth sigmoidal and kth product nodes in the hidden layer is given by Equations (3) and (4), respectively; where $\mathbf{u}_{ji}$ is the weight of the connection between input node i and hidden sigmoidal node j and w_{ki} is the weight of the connection between input node i and hidden product node k. The activation of the node in the output layer is given by Equation (5); where α_j and β_k are the weights of the connection between the hidden sigmoidal node j or product node k and the output node. The transfer functions of each product node in the hidden layer and the output node for regression are the identity function. In this way, each function of the $f(x, w)$ family is represented by the structure of the network.

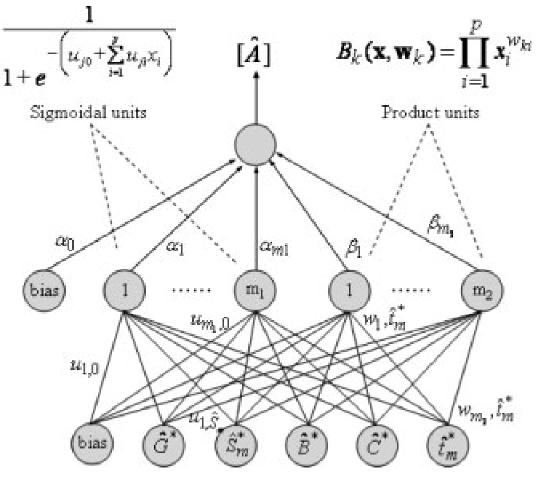


Figure 1. Functional scheme of the neural network based on sigmoidal and product units. $\alpha_0, \ldots\ldots, \alpha_{m_1}$ and $\beta_0, \ldots\ldots, \beta_{m_2}$ are the regression coefficients of the model. Other symbols are defined in the text.

2.3. Evolutionary algorithm

The optimization of the SPUNN topology consisted of the search for the structure of the sigmoidal and product unit base functions that best fit the data of the training set, by determining the values of m_1 and m_2 associated with the optimum number of base functions for each type involved. On the other hand, the estimation of the weights of the network is closed to the evolution of the $\mathbf{u}_j$ and $\mathbf{w}_j$ vectors, which determine the coefficients in each base function, as well as with α_j and β_k, coefficients involved in the linear combination of the base functions. The population is subjected to the operations of replication and structural and parametric mutations. The general structure of the algorithm is the following:

(1) Generate the initial population.
(2) Repeat until the stopping criterion is fulfilled.
(a) Calculate the fitness of every individual in the population.
(b) Rank the individuals regarding their fitness.
(c) The best individual is copied into the new population.
(d) Ten per cent of the best individuals in the population are replicated and substitute the 10% of worst individuals.
(e) Apply parametric mutation to 10% of the best individuals.
(f) Apply structural mutation to the rest of the 90% of individuals.

For the generation of the initial population, the algorithm begins with the random generation of a larger number of networks than the number used during the evolutionary process. Initially, we generate 10 000 networks from which, the best 1000 with major fitness are extracted, being it the population size along the evolutionary process. For the generation of a network, the number of nodes in the hidden layer is taken from a uniform distribution in the interval [0,6], for $m_1 + m_2 = 6$, where m_1 and m_2 are related to the maximum number of hidden nodes in each base function in the population. The number of connections between each node of the hidden layer and the input nodes is determined from a uniform distribution in the interval [0,*k*], where *k* is the number of independent variables. There is always at least one connection between the hidden layer and the output node. Once the topology of the network is defined, each connection is assigned a weight from a uniform distribution in the interval [−5,5] for the weights between the input and hidden layers, and the same interval for the weights between the hidden layer and the output node. Two types of mutations are performed in the algorithm: parametric and structural. The parametric mutations affect the weights of the network and the structural ones affect to the network topology (hidden nodes and connections). The severity of a mutation is dictated by the function's temperature $T(f(x))$ given by:

$$T(f(\mathbf{x})) = 1 - A(f(\mathbf{x}))\ 0 \le T(f(\mathbf{x})) \le 1 \tag{7}$$

where $A(f(\mathbf{x}))$ is the fitness value of the $f(x)$.

Let $D = \{(\mathbf{x}_l, y_l) : l = 1, 2, \ldots\ldots, n_T\}$ be the training data set, where the number of samples is n_T. In this context, we consider that the mean squared error MSE of an individual $f(x)$ of the population is given by:

$$\mathrm{MSE}\,(f(\mathbf{x})) = \frac{1}{n_T}\sum_{l=1}^{n_T}(y_l - f(\mathbf{x}_l))^2 \tag{8}$$

where the y_l are the predicted values, whereas the fitness function $A(f(\mathbf{x}))$ is defined by means of a strictly decreasing transformation of the MSE:

$$A\,(f(\mathbf{x})) = \frac{1}{1 + \dfrac{1}{n_T}\displaystyle\sum_{l=1}^{n_T}(y_l - f(\mathbf{x}_l))^2} \tag{9}$$

where $0<A(f(\mathbf{x}))<1$.

Parametric mutation consists of a simulated annealing algorithm [39] based on a strategy that proceeds by proposing jumps from the current model according to some user-designed mechanism. Structural mutation is more complex because it implies a modification of the structure of the function. Structural mutation allows the exploration of different regions in the search space and helps to keep up the diversity of the population. Five mutations are applied sequentially to each network. The first four are similar to the mutations of the GNARL model reported by Angeline *et al.* [36], namely: *Node addition* (NA), *Node deletion* (ND), *Connection addition* (CA) and *Connection deletion* (CD); to these, *Node fusion* (NF) is added. For each mutation (except NF), there is a minimum value of 1 for adding or removing nodes and connections, and a maximum value of 2 or 6 for adding or removing nodes and connections, respectively.

In this hybrid implementation of the basis functions, when a new node should be added to the networks, it is necessary to estimate the probability for adding SU or PU. In this work, we have considered with equal probability the addition of a sigmoidal or product unit hidden node, thus the probability is 0.5. These probabilities have been determined by means of trial and error, and they are supported along the whole evolutionary process. Finally, the stop criterion is reached whenever one of the following two conditions is fulfilled: (i) the algorithm achieves a given number of generations; (ii) there is no improvement for a number of generations either in the average performance of the best 20% of the population or in the fitness of the best individual. For more details about the evolutionary algorithm, see references [27] and [28].

3. EXPERIMENTAL

The whole procedure followed in this work is schematically shown in Figure 2. To obtain analytical data, 55 electropherograms provided by samples containing amikacin (15–300 μg/L), paromomycin (10–200 μg/L) and mixtures of both antibiotics with uniformly distributed concentrations of them (30–300 and 20–200 μg/L, respectively) were prepared in duplicate, as described elsewhere [34], by using 35 mM sodium borate at pH 9.0 as background electrolyte.

The Levenberg–Marquardt algorithm was used to estimate the four-parameter Weibull function ($\hat{S}_m$, $\hat{B}$, $\hat{C}$ and $\hat{t}_m$) associated with the profile of the electrophoretic peaks (convergence of the iterative process was achieved with a tolerance of 0.0001 and a maximum number of 100 iterations), whereas LDA was used to classify these electrophoregrams in three categories: G_1 and G_2 for samples containing only amikacin

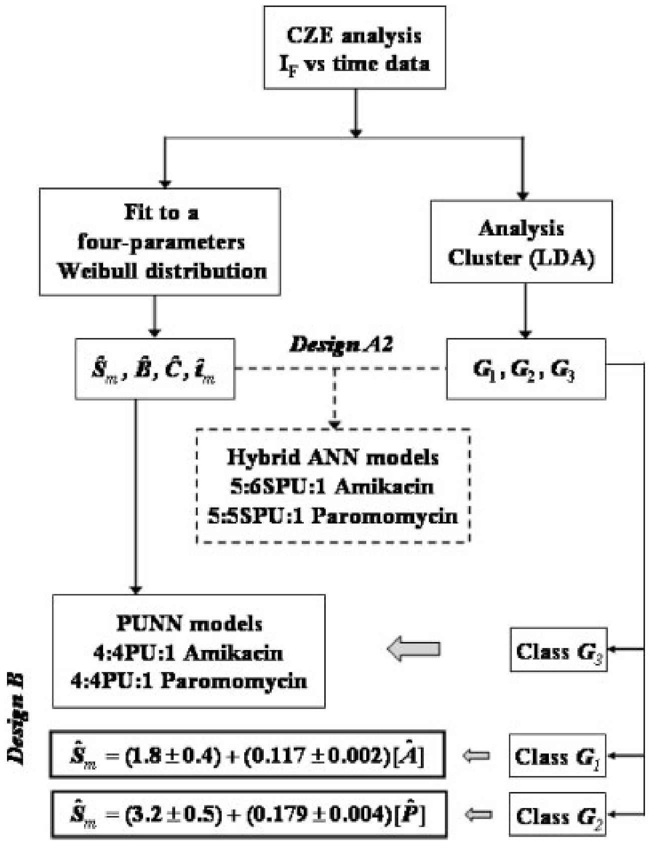


Figure 2. Flow diagram representing the whole analytical protocol.

or paromomycin, respectively and G_3 for samples containing mixtures of both antibiotics, which were generically denoted by $\hat{G}$. Prior to use these parameters as inputs for the ANN models, and in order to avoid saturation problems in the sigmoidal basis functions (preventing driving the weights to infinity) as well as to improve the learning process, each of the input (parameters of the Weibull curve and the category) and output (concentration of the antibiotics in the sample) were scaled in the range [0.1, 0.9]; this process was also useful for product basis functions because the lower bound is chosen to avoid inputs values near to zero that can produce very large values of the outputs for negative exponents, whereas the upper bound is chosen to avoid dramatic changes in the outputs of the network when there are weights with large values (especially in the exponents). Thus, the new scaled variables were expressed as follows:$\hat{S}_m^*$, $\hat{B}^*$, $\hat{C}^*$, $\hat{t}_m^*$ and $\hat{G}^*$ for the input variables and $[\hat{A}]^*$ and $[\hat{P}]^*$ for the output variables. For example $[\hat{A}]^*$ is calculated as follows:

$$[\hat{A}]^* = \frac{[A] - [A_{\min}]}{[A_{\max}] - [A_{\min}]} \times 0.8 + 0.1 \tag{10}$$

where $[A]$ is the original concentration of amikacin in the sample, $[A_{\min}]$ and $[A_{\max}]$ are the minimum and maximum values and $[\hat{A}]^*$ is the scaled concentration. After optimizing the network models, estimations should be de-scaled according to the same equation.

The experimental design was conducted using a holdout cross-validation procedure where the size of the training set was approximately $3n/4$ and $n/4$ for the generalization set, where $n = 110$ is the size of the full data set. The algorithm software for SU, PU and SPU neural networks was designed in Java using the JCLEC library [40] and was run on a portable PC Pentium IV compatible computer. The accuracy of each model was assessed in terms of the SEP for the results obtained for both data sets, that is SEP_T for the training set, and SEP_G for the generalization set. In this way, the SEP was calculated as:

$$\text{SEP} = \frac{100}{\overline{C}_i}\sqrt{\frac{\sum_{i=1}^{n}(C_i - \hat{C}_i)^2}{n}} \tag{11}$$

where C_i and $\hat{C}_i$ are the experimental and expected values for the antibiotic concentration in the mixture, $\overline{C}_i$ is the mean of the experimental values of the training set, or of the generalization set and n is the number of patterns used (n_T for the training set and n_G for the generalization set). Finally, the partial least squares (PLS) multivariate calibration algorithm was provided by the Pirouette 3.11 software from Infometrix, Inc.

4. RESULTS AND DISCUSSION

In recent years, CZE is increasingly being considered as an alternative to HPLC for the determination of a great variety of compounds because it affords such significant benefits as higher resolution, smaller sample requirements and shorter analysis time. Despite this better resolution, CZE separation is sometimes not accomplished and therefore chemometric resolution is the suitable choice. Thus, it is understandable that the quantitative resolution of overlapping electrophoretic peaks is an area of growing interest for analytical chemists. ANNs have scarcely been used for this purpose despite their higher discrimination power regarding other current chemometric tools, even when using a smaller amount of chemical information [16,17]. The goal of this work was to evaluate different ANN models supported on the use of three different transfer functions, namely SU, PU and the hybrid model: SPU, for the quantification of unresolved CZE peaks with a high degree of overlapping by using analytical data provided by a single detector (LIF detection), that is in the absence of spectral discrimination. The approach was tested on the determination of amikacin (A) and paromomycin (P), two aminoglycoside antibiotics that cannot be resolved by CZE. After labeling the antibiotics with sulfoindocyanine succinimidyl ester (Cy5), electrophoretic data were obtained by monitoring the single LIF signal with a diode-laser detector. In order to extend the scope of the proposed methodology to more practical situations, in addition to the typical approach based on samples containing mixtures of both antibiotics, samples with only one antibiotic (A or P) have also been considered. It is noteworthy that the CZE peak showed a similar profile in all cases.

4.1. Selection of data and inputs for ANN models

One important issue to be addressed in order to obtain the best generalization capability of ANN models is the composition of the data set used for training and generalization. Two different data sets were tested from all the data (110 samples) obtained by preparing 55 individual and mixture synthetic samples in duplicate. Design A was constituted by the whole group of the samples (individual and mixtures) and design B only contains the samples with mixtures of antibiotics. In both cases, the size of the training and generalization sets was 3/4 and 1/4, respectively, of the whole set: 110 and 72 samples for design A and B, respectively. The so-called grid approach [41] was used for sampling mixtures including those with higher and smaller antibiotic ratios in the training set for assuring interpolation in the generalization process. This state was also considered for individual samples.

The selection of the multivariate input data for ANN models is also of great relevance because they may cause over-fitting of the trained networks. A small-sized neural network will be less prone to overtraining noise or the structure of the data in the training set, thus increasing its generalization capacity over a new data set and allowing users to gain knowledge from the trained neural network in order to achieve a better understanding of how the network solves the problem. As an alternative to principal component analysis (PCA), the most common choice used for reducing the data set, we have recently reported a useful approach based on the modeling of the chromatographic profile to a pre-determined function [16,17].

The electropherograms of pure amikacin, paromomycin and a mixture of equal concentrations of both antibiotics are shown in Figure 3A. As can be seen, the profiles of the CZE peaks were very similar, which makes clear the high overlapping grade of the bands of both antibiotics in mixtures samples. Although the precision (expressed as relative standard deviation) for the migration times in these electropherograms was within 1% (similar to that reported by us in other works using CZE with LIF detection) in order to improve their reproducibility, migration times were corrected using as reference one peak of the excess of Cy5 label.

The non-symmetric shape of these profiles suggests that they can be modeled with a pre-determined function, being the four-parameter Weibull distribution the best choice because it provides the best regression coefficients in the fitting. As stated above, the equation corresponding to this function is defined by the following parameters: $\hat{S}_m$ (peak height), $\hat{B}$ (dispersion of S_{t_i} values from S_m), $\hat{C}$ (associated to the inflection points of the curve) and $\hat{t}_m$ (residence time). Figure 3B shows the fit provided by the single four-parameter Weibull distribution on the CZE peak profiles. From these plots, it can be inferred that it was impossible to fix a common residence time interval to select the portion of the CZE profile subjected to fit for all samples. However, the fit can be easily achieved by using the following process: (1) a residence time interval of 10 data around the maximum was selected to start the fit and (2) this interval was successively increased by two data, one to every side of the interval. The number of data (the portion of the CZE profile) that provided the best fit, in terms of regression coefficient, was used to estimate the four parameters of the Weibull distribution. Upon examining the curves and from the estimated statistical parameters shown in Table I, it can be concluded that the four-parameter Weibull function is a fine tool for modeling these CZE data.

In addition, a cluster analysis was also carried out using a cross-validation holdout method (72 patterns for the training set and 38 for generalization set) adopting these four-parameters as variables for LDA. As can be seen in Table II, the confusion matrixes provide a total correct classification rate (CCR) value of 100 for both training and generalization sets; this table also includes the standardized coefficients for the canonical discriminant functions. From these results, the following labels were assigned to the different classes: $\hat{G}_1 = 1$

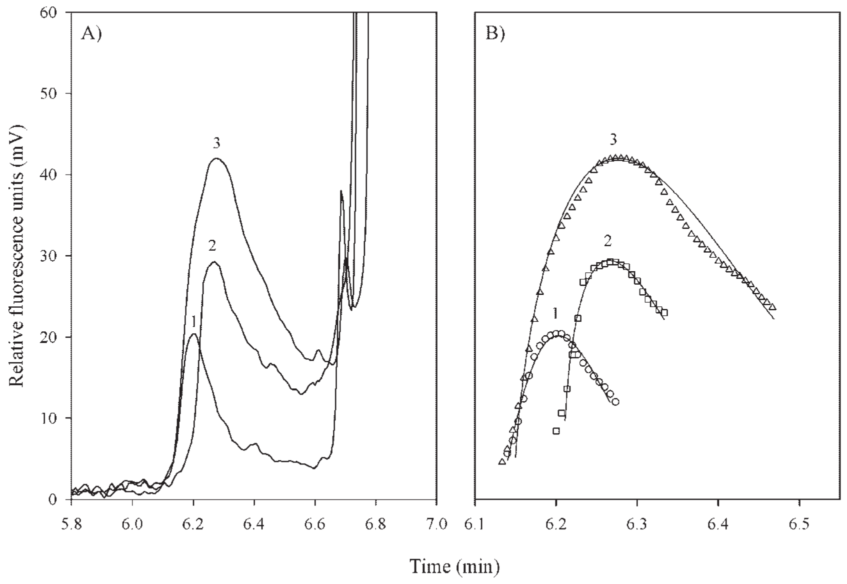


Figure 3. (A) Typical electropherograms corresponding to pure and mixture solutions of the assayed antibiotics: (1) 120 μg/L of amikacin; (2) 180 μg/L of paromomycin and (3) mixture containing 120 μg/L of amikacin and 180 μg/L of paromomycin. (B) CZE responses fitted to a four-parameter Weibull distribution (o) Experimental data and (–) Weibull curve.

Table I. Estimated parameters obtained in the modelling process of the electrophoretic peaks shown in Figure 3 by means of a Weibull distribution

	Parameters[a]				
Sample	$\hat{S}_m$	$\hat{B}$	$\hat{C}$	$\hat{t}_m$	r
Pure amikacin	20.1 ± 0.6	0.106 ± 0.004	1.9 ± 0.1	6.203 ± 0.002	0.9889
Pure paromomycin	28.8 ± 1.6	0.15 ± 0.02	1.5 ± 0.2	6.275 ± 0.008	0.9540
Mixture of antibiotics	42.0 ± 0.8	0.245 ± 0.009	1.57 ± 0.08	6.277 ± 0.006	0.9855

[a] Mean $\pm 1.96 \times$ SD; SD = standard deviation; r = regression coefficient.

and $\hat{G}_2 = 2$ for pure amikacin and paromomycin samples, respectively and $\hat{G}_3 = 3$ for samples containing mixtures of the antibiotics. These labels were also used in some cases as inputs in addition to the Weibull parameters.

4.2. Evaluation of the ANN models

To compare the predictive ability of ANN models in terms of topology, number of connections, homogeneity (confidence interval) and accuracy (SEP_G), network models with a single node in the output layer were designed (concentration of antibiotic to be determined in the individual or mixture sample). When using experimental design A, 4:5:1 and 5:6:1 architectures were chosen to start the model selection processes, whereas a 4:4:1 topology was used for design B, in all cases over 30 runs.

As can be seen in Table III, all models provided satisfactory results in terms of accuracy (SEP_G) and homogeneity (confidence interval) for determining the concentration of each antibiotic in the samples assayed. On comparing the two designs, higher SEP values were achieved by ANN models based on design A1 that uses all samples for training and generalization sets without discrimination among them through cluster analysis. In this case, only the four-parameters of the Weibull curve were used as inputs and the computational cost was clearly lower. When the cluster analysis was carried out, two approaches have been considered (see Figure 2): using the class label as input in addition to the Weibull parameters (design A2) or employing this information to create a new data set (design B) containing only the mixture samples. Thus in the design B, only the Weibull parameters were used as inputs, and the determination of the concentrations of the antibiotic in the individual samples was supported on the construction of the classical calibration plot, such as $\hat{S}_m$ versus [antibiotic] expressed in μg/L. From experimental results, the following calibration plots can be drawn:

$$\hat{S}_m = (1.8 \pm 0.4) + (0.117 \pm 0.002)\,[A], \qquad r = 0.9965 \quad (12)$$

$$\hat{S}_m = (3.2 \pm 0.5) + (0.179 \pm 0.004)[P], \qquad r = 0.9951 \quad (13)$$

As can be seen in Table III, ANN models based on design B provided lower SEP_G values, perhaps due to the biggest homogeneity of the data sets because only mixture samples were used for both training and generalization sets. However, the use of design A2 cannot be discarded a priori because it can be an acceptable choice depending on the analytical problem addressed (higher SPG_G values vs. lower computational cost).

Regarding the type of ANN models tested, those based on PU and SPU as transfer functions provided similar results for the design A2, whereas models of PUNN provided lower SPE_G values for design B, especially for the determination of P. This behavior is closely related to the composition of the data set. In fact, the design A2 composed by individual (linear) samples together with mixtures (nonlinear) is better modeled by ANN using hybrid transfer functions, whereas in the case of the design B (in the absence of individual samples), PU transfer functions yielded a better modeling of the chemical information.

In order to compare the quality achieved in each ANN model for the resolution of mixtures of these antibiotics, Table IV shows the results (in terms of concentration) obtained applying the models to the synthetic mixtures included in the generalization set of design B (see Figure 2). In addition, this table also shows the results achieved when the data were analyzed by using a classical standard reference method such as PLS regression. For the construction of the different PLS models, the cross-validation method was used and the number of significant factors in each case were chosen as the lower number whose root mean standard error (RMSE) of prediction by cross-validation was not significantly different from the lowest RMSE value:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}\left(C_i - \hat{C}_i\right)^2}{n}} \quad (14)$$

Based on the absolute error of the prediction $\left|C_i - \hat{C}_i\right|$ obtained using the data reported by the ANN and PLS

Table II. Summary of the results obtained in the LDA of the whole group of the samples (individual and mixtures) by using the estimated four-parameters of the Weibull function as variables

	Training				Generalization			
TR/PR	$G_1=1$	$G_2=2$	$G_3=3$	FC %	$G_1=1$	$G_2=2$	$G_3=3$	FC %
A. Rate of the number of cases that were classified correctly								
$G_1=1$	12	0	0	100	6	0	0	100
$G_2=2$	0	12	0	100	0	6	0	100
$G_3=3$	0	0	48	100	0	0	24	100
CCR				100				100

Variable	$\hat{S}_m$	$\hat{B}$	$\hat{C}$	$\hat{t}_m$
B. Standardized coefficients for the canonical discriminant functions				
Coefficients of function 1	−0.374	0.600	2.414	2.437
Coefficients of function 2	0.810	−0.387	0.246	0.503

PR = Predicted response; TR = Target response; FC = False correct.

Table III. Accuracy (mean ± 1.96 × SD) and statistical results of the algorithm used with different experimental designs and transfer functions (over 30 runs)

		Connections	SEP_T			SEP_G		
Analyte	Starting topology	Mean ± 1.96 × SD	Mean ± 1.96 × SD	Best	Worst	Mean ± 1.96 × SD	Best	Worst
Design A1								
A	4:5SU:1	26.5 ± 3.4	16.7 ± 9.9	10.1	31.7	19.4 ± 11.0	10.2	36.8
	4:5PU:1	18.3 ± 3.0	14.1 ± 3.6	11.0	18.3	22.0 ± 14.1	13.2	38.8
P	4:5SU:1	26.1 ± 3.7	21.7 ± 3.7	17.6	25.1	18.0 ± 5.0	14.2	24.1
	4:5PU:1	18.3 ± 2.9	20.4 ± 4.1	16.4	24.9	18.6 ± 6.6	13.5	27.5
Design A2								
A	5(4 + 1):6SU:1	33.6 ± 4.2	12.7 ± 5.3	8.2	17.8	16.0 ± 6.8	10.0	23.6
	5(4 + 1):6PU:1	23.8 ± 7.3	11.8 ± 3.9	8.9	16.1	20.4 ± 10.6	9.7	32.9
	5(4 + 1):6SPU:1	33.3 ± 4.7	12.9 ± 4.3	10.1	18.6	19.7 ± 11.8	9.1	38.9
P	5(4 + 1):6SU:1	34.8 ± 5.4	16.7 ± 5.0	12.1	21.4	16.1 ± 3.6	13.2	20.1
	5(4 + 1):6PU:1	23.2 ± 5.9	15.6 ± 4.8	10.7	20.4	18.2 ± 7.1	12.3	27.1
	5(4 + 1):6SPU:1	32.5 ± 6.1	13.6 ± 4.2	9.4	18.1	16.6 ± 7.2	10.9	27.7
Design B								
A	4:4SU:1	20.1 ± 3.8	7.2 ± 1.6	6.1	10.0	10.5 ± 3.2	8.4	15.3
	4:4PU:1	13.8 ± 3.1	7.0 ± 1.3	5.8	8.5	10.5 ± 2.2	8.2	13.1
	4:4SPU:1	19.2 ± 4.3	7.2 ± 1.9	5.4	8.9	10.2 ± 2.4	8.4	13.0
P	4:4SU:1	19.2 ± 4.9	8.3 ± 2.5	5.8	11.4	13.0 ± 6.7	6.5	22.2
	4:4PU:1	13.6 ± 4.4	8.8 ± 3.1	5.8	11.4	12.5 ± 8.8	5.6	22.6
	4:4SPU:1	19.2 ± 3.8	8.4 ± 3.2	6.0	12.9	12.3 ± 7.5	6.4	20.1

models in Table IV, the analysis of variance (ANOVA) technique was carried out to ascertain the statistical significance of observed differences among the four corresponding means, assuming that the absolute error values obtained have a normal distribution. A Kolmogorov–Smirnov test for normality was reached with *p*-values of 0.383, 0.389, 0.866 and 0.143 for amikacin and 0.538, 0.426, 0.235 and 0.885 for paramomycin, for SUNN, PUNN, SPUNN and PLS models, respectively. The ANOVA involved a linear regression model in which $\left|C_i - \hat{C}_i\right|$ was

Table IV. Comparison of the quality achieved for the quantification of amikacin and paromomycin mixtures using the ANN models and the PLS methodology

	Amikacin					Paromomycin				
		Found (μg/L)					Found (μg/L)			
$[\hat{A}]/[\hat{P}]$	Added (μg/L)	SUNN	PUNN	SPUNN	PLS	Added (μg/L)	SUNN	PUNN	SPUNN	PLS
8:1	240.0	231.9	244.8	241.2	238.2	30.0	16.2	26.1	18.9	10.2
8:1	240.0	240.9	253.8	250.5	248.1	30.0	17.7	27.6	21.6	9.3
1:2	30.0	23.1	27.9	25.8	7.8	60.0	75.9	66.6	77.4	82.2
1:2	30.0	31.5	33.6	35.7	30.0	60.0	66.0	61.2	71.7	73.8
4:2	120.0	102.6	103.5	104.1	118.5	60.0	71.1	72.3	75.0	87.6
4:2	120.0	108.3	110.4	108.0	122.4	60.0	71.4	71.7	72.9	94.2
6:2	180.0	173.4	176.7	171.9	190.5	60.0	61.5	64.8	60.9	88.8
6:2	180.0	170.4	174.9	167.1	185.4	60.0	64.5	66.9	62.4	98.4
8:2	240.0	225.6	226.5	222.6	241.8	60.0	55.8	61.8	57.3	91.5
8:2	240.0	234.3	235.2	231.9	249.9	60.0	57.6	63.3	59.7	85.5
1:4	30.0	40.2	47.7	39.3	32.1	120.0	111.9	102.0	116.7	112.2
1:4	30.0	38.7	46.2	42.0	39.0	120.0	118.2	109.2	127.8	119.4
6:4	180.0	195.9	197.4	201.3	213.9	120.0	103.5	105.9	102.6	111.3
6:4	180.0	196.2	196.8	200.7	212.4	120.0	112.2	115.8	110.7	113.4
4:6	120.0	115.5	119.1	115.5	122.4	180.0	178.5	177.6	178.2	185.7
4:6	120.0	114.6	117.3	121.5	129.0	180.0	182.7	180.6	185.1	192.9
6:6	180.0	166.8	163.5	179.7	186.3	180.0	176.7	179.4	177.9	182.7
6:6	180.0	174.0	169.8	192.3	198.9	180.0	167.1	167.7	168.9	174.6
4:8	120.0	115.5	115.8	118.5	124.8	240.0	244.8	237.0	243.3	255.3
4:8	120.0	112.8	112.2	117.6	125.4	240.0	251.7	241.5	250.5	263.1
10:8	300.0	264.9	273.0	268.8	252.6	240.0	254.4	239.7	251.1	270.6
10:8	300.0	270.3	275.4	277.8	260.1	240.0	246.3	237.3	243.0	260.4
4:10	120.0	130.2	116.4	120.9	137.1	300.0	296.4	310.2	300.0	273.0
4:10	120.0	124.2	111.0	117.0	134.1	300.0	297.9	307.5	301.5	277.2

Table V. Quantification equations and accuracy provided by the optimized 4:4:1 PUNN network topologies as applied to the determination of amikacin and paramomycin

	Amikacin	Paramomycin
Quantification equations	$[\hat{A}]^* = 0.52 + 2.29\hat{h}_1 - 1.28\hat{h}_2 + 0.36\hat{h}_3 + 2.80\hat{h}_4$	$[\hat{P}]^* = 0.25 - 1.97\hat{h}_1 + 3.40\hat{h}_2 - 2.07\hat{h}_3 - 0.06\hat{h}_4$
Transfer functions	$\hat{h}_1 = (\hat{S}_m^*)^{0.39}(\hat{B}^*)^{0.50}$	$\hat{h}_1 = (\hat{S}_m^*)^{0.42}(\hat{B}^*)^{0.46}(\hat{C}^*)^{0.46}(\hat{t}_m^*)^{0.06}$
	$\hat{h}_2 = (\hat{S}_m^*)^{0.92}(\hat{B}^*)^{0.52}(\hat{C}^*)^{0.86}$	$\hat{h}_2 = (\hat{S}_m^*)^{1.14}(\hat{C}^*)^{0.46}$
	$\hat{h}_3 = (\hat{S}_m^*)^{2.11}(\hat{t}_m^*)^{0.45}$	$\hat{h}_3 = (\hat{C}^*)^{1.63}(\hat{t}_m^*)^{4.91}$
	$\hat{h}_4 = (\hat{B}^*)^{3.87}(\hat{C}^*)^{1.49}$	$\hat{h}_4 = (\hat{B}^*)^{3.13}$
Effective links	14	14
SEP_T	5.8%	5.8%
SEP_G	8.2%	5.6%

the dependent variable and the independent variable was the type of model used for prediction. The comparisons were made in terms of a critical level for Snedecor's F; if the significance level, α, was higher than this critical level, p, the hypothesis of identical means was rejected. In the case of amikacin, this hypothesis was not accepted because the p-value is 0.749, higher than a standard $\alpha = 0.05$, whereas in the case of paramomycin, it was accepted because the p-value was 0.000. Based on these results, a test of multiple comparisons of Tamhane was carried out for paramomycin. Significant differences were found between PLS and ANN models according to the p-values provided by the test: 0.000 in all comparisons. In view of these results, we selected the PUNN model on the basis of its lower SEP_G values and simpler architecture (smaller effective links) than SUNN and SPUNN models.

The simplicity of the proposed ANN models permits us to derive straightforward quantitative equations for the determination of the concentration of each antibiotic using: (a) the parameters estimated by the Weibull regression of the peak and the class label provided by the cluster analysis; (b) the optimized network weights and (c) the transfer functions involved in the models. Table V shows the quantitative equations corresponding to the proposed PUNN models for both antibiotics.

4.3. Chemical interpretation of ANN models

Taking into account that the proposed PUNN models provided simple quantitative equations for the direct determination of the contribution of each antibiotic to the overlapping CZE peaks, quality chemical information could be derived in order to explain the relationship between the profile of the CZE peak (defined by the Weibull parameters) and the determination of the antibiotic in the mixture by using the proposed PUNN models. In this way, the value of each PU transfer function involved that was affected by its coefficient in the model was calculated over the scaled range studied for the input variables, $\hat{S}_m^*$, $\hat{B}^*$, $\hat{C}^*$ and $\hat{t}_m^*$, and then plotted one against another (see Figure 4).

4.3.1. Model for amikacin

From the quantification equation shown in Table V and from the plots in Figure 4A, it follows that the value of $[\hat{A}]^*$ depends mainly on the contribution of the PU functions $\hat{h}_1$ and $\hat{h}_2$ with opposite effects and also on the other PU

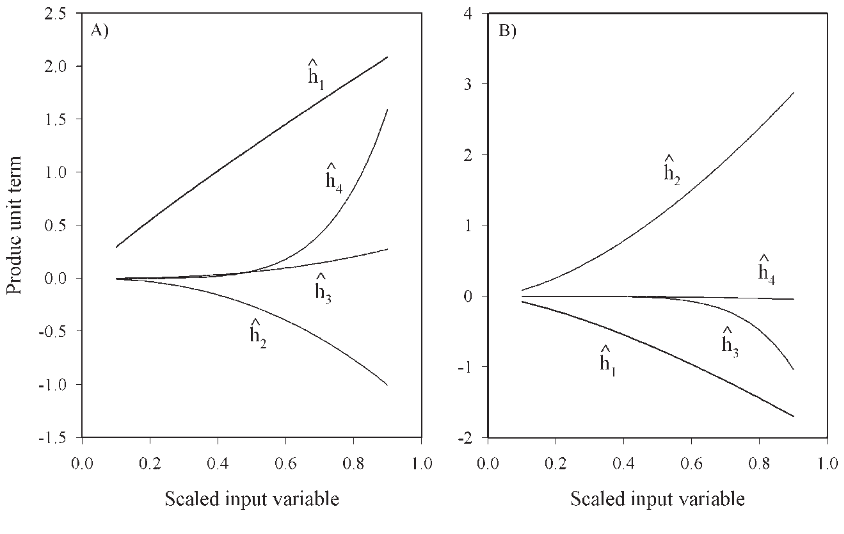


Figure 4. Relative contribution of the product unit terms used for the quantitative determination of the antibiotics provided by PUNN models. (A) amikacin and (B) paromomycin.

functions but only at higher values of the scaled input variable: for example the response of the function $\hat{h}_4$ at higher values of $\hat{B}^*$ and $\hat{C}^*$ is assigned to mixtures with higher values of the $[A]$ to $[P]$ ratio. In short, this function made the model provide more accurate results when modeling this type of samples. It is also noteworthy that the contribution of $\hat{t}_m^*$ was practically negligible because it was only involved in the function $\hat{h}_3$ which, on the other hand, is the one that least contributes to the value of $[\hat{A}]^*$. From these results, the direct dependencies of $\hat{S}_m^*$ and $\hat{B}^*$ parameters on $[\hat{A}]^*$ is clear, that is the value of the peak height and the degree of dispersion of the analytical signal values from it, and both taking into account their respective exponents in $\hat{h}_1$ to a similar extent.

4.3.2. Model for paromomycin

By using the PUNN model for the determination of P in the mixture, and from the quantification equation shown in Table V and plots in Figure 4B, it follows that the value of $[\hat{P}]^*$ depends basically on the $\hat{S}_m^*$ and $\hat{C}^*$ parameters, considering the dependencies showed by the $\hat{h}_1$ and $\hat{h}_2$ functions in Figure 4B. According to the relative values of the exponents in the $\hat{h}_1$ (basis function), a higher dependence can be ascribed to $\hat{C}^*$ parameter, which is associated to the inflection points of the curve and defines the concavity and convexity regions of the CZE peaks. Regarding the other PU functions, only $\hat{h}_3$ exerts a slight contribution at higher values of the scaled input variable, which is related to mixtures with high $[P]$ to $[A]$ ratios; as in the case of A, this function is of great relevance for achieving a better modeling of these mixtures by the proposed PUNN model.

5. CONCLUSIONS

As shown in this work, quantitative analysis in CZE is possible even in the case of overlapped peaks by means of ANNs designed by an evolutionary algorithm. Three ANN strategies were evaluated for modeling the CZE data provided from a set of samples containing single and mixtures of analyte concentrations. The four parameters of the Weibull curve fitted to the profile of the CZE peaks were suitable as inputs for the three types of ANN, although the predicted results can be improved with a prior LDA analysis and using in some case the class label associated to each sample as additional input. The calculated results indicated that ANN models with SU, PU and SPU units as transfer functions were a promising tool to resolve overlapped CZE peaks with acceptable errors. The designed PUNN models provided better accurate results, smaller network architectures and more robust models and they were quite simple and easier to interpret from a chemical point of view. In summary, ANN is a powerful tool for resolving CZE overlapped peaks while also allowing an important reduction in analysis time because it avoids time consumption by finding optimal conditions for the suitable CZE resolution.

Acknowledgements

The authors gratefully acknowledge the subsidy provided by the Spanish Inter-Ministerial Commission of Science and Technology of the Ministry of Education and Science under the CTQ2007-63962 (Department of Analytical Chemistry, University of Cordoba) and TIN2005-08386-C05-02 (Department of Computer Science, University of Cordoba) Projects. FEDER also provided additional funding. The research of Pedro Antonio Gutierrez has been supported by FPU-Predoctoral Program (Spanish Ministry of Education and Science).

REFERENCES

1. van-Zomeren PV, Metting HJ, Coenegracht PMJ, de-Jong GJ. Simultaneous resolution of overlapping peaks in high-performance liquid chromatography and micellar electrokinetic chromatography with diode array detection using augmented iterative target transformation factor analysis. *J. Chromatogr. A* 2005; **1096**: 165–176.
2. Vivo-Truyols G, Torres-Lapasio JR, van-Nederkassel AM, vander-Heyden Y, Massart DL. Automatic program for peak detection and deconvolution of multi-overlapped chromatographic signals. Part II: Peak model and deconvolution algorithms. *J. Chromatogr. A* 2005; **1096**: 146–155.
3. Rodriguez-Cuesta MJ, Boque R, Rius FX, Vidal JLM, Frenich AG. Development and validation of a method for determining pesticides in groundwater from complex overlapping HPLC signals and multivariate curve resolution. *Chemometr. Intell. Lab. Syst.* 2005; **77**: 251–260.
4. Wiberg K, Jacobsson SP. Parallel factor analysis of HPLC-DAD [diode-array detection] data for binary mixtures of lidocaine and prilocaine with different levels of chromatographic separation. *Anal. Chim. Acta* 2004; **514**: 203–209.
5. Comas E, Gimeno RA, Ferre J, Marce RM, Borrull F, Rius FX. Quantification from highly drifted and overlapped chromatographic peaks using second-order calibration methods. *J. Chromatogr. A* 2004; **1035**: 195–202.
6. van-Zomeren PV, Darwinkel H, Coenegracht PMJ, de Jong GJ. Comparison of several curve resolution methods for drug impurity profiling using high-performance liquid chromatography with diode-array detection. *Anal. Chim. Acta* 2003; **487**: 155–170.
7. Gross GM, Prazen BJ, Synovec RE. Parallel column liquid chromatography with a single multi-wavelength absorbance detector for enhanced selectivity using chemometric analysis. *Anal. Chim. Acta* 2003; **490**: 197–210.
8. Fraga CG, Bruckner CA, Synovec RE. Increasing the number of analyzable peaks in comprehensive two-dimensional separations through chemometrics. *Anal. Chem.* 2001; **73**: 675–683.
9. Sanchez FC, Rutan SC, Garcia MDG, Massart DL. Resolution of multicomponent overlapped peaks by the orthogonal projection approach, evolving factor analysis and window factor analysis. *Chemometr. Intell. Lab. Syst.* 1997; **36**: 153–164.
10. Zhang F, Li H. Resolution of overlapping capillary electrophoresis peaks by using chemometric analysis: improved quantification by using internal standard. *Chemometr. Intell. Lab. Syst.* 2006; **82**: 184–192.
11. Zhang F, Li H. Resolution of overlapping capillary electrophoresis peaks by using chemometric analysis: quantification of the components in compound reserpine tablets. *Electrophoresis* 2005; **26**: 1692–1702.
12. Li H, Zhang F, Havel J. Quantification of analytes in overlapping peaks from capillary electrophoresis using multivariate curve resolution-alternating least squares methods. *Electrophoresis* 2003; **24**: 3107–3115.
13. Olazabal V, Prasad L, Stark P, Olivares JA. Application of wavelet transforms and an approximate deconvolution

method for the resolution of noisy overlapped peaks in DNA capillary electrophoresis. *Analyst* 2004; **129**: 73–81.
14. Chen AJ, Li CH, Gao WH, Hu ZD, Chen XG. Separation and determination of active components in *Schisandra chinensis* Baill. and its medicinal preparations by non-aqueous capillary electrophoresis. *Biomed. Chromatogr.* 2005; **19**: 481–487.
15. Chen AJ, Li CH, Gao WH, Hu ZD, Chen XG. Application of non-aqueous micellar electrokinetic chromatography to the analysis of active components in radix *Salviae miltiorrhizae* and its medicinal preparations. *J. Pharm. Biomed. Anal.* 2005; **37**: 811–816.
16. Hervás C, Silva M, Serrano JM, Orejuela E. Heuristic extraction of rules in pruned artificial neural networks models used for quantifying highly overlapping chromatographic peaks. *J. Chem. Inf. Comput. Sci.* 2004; **44**: 1576–1584.
17. Hervás C, Martínez AC, Silva M, Serrano JM. Improving the quantification of highly overlapping chromatographic peaks by using product unit neural networks modeled by an evolutionary algorithm. *J. Chem. Inf. Model.* 2005; **45**: 894–903.
18. Garrido-Frenich A, Martinez-Galera M, Gil-Garcia MD, Martinez-Vidal JL, Catasus M, Marti L, Mederos MV. Resolution of HPLC-DAD highly overlapping analytical signals for quantitation of pesticide mixtures in groundwater and soil using multicomponent analysis and neural networks. *J. Liq. Chromatogr. Relat. Technol.* 2001; **24**: 651–668.
19. Li YB, Huang XY, Sha M, Meng XS. Resolution of overlapping chromatographic peaks by radial basis function neural network. *Sepu* 2001; **19**: 112–115.
20. Galeano-Diaz T, Guiberteau A, Ortiz JM, Lopez MD, Salinas F. Use of neural networks and diode-array detection to develop an isocratic HPLC method for the analysis of nitrophenol pesticides and related compounds. *Chromatographia* 2001; **53**: 40–46.
21. Zhang YX, Li H, Hou AX, Havel J. Artificial neural networks based on principal component analysis input selection for quantification in overlapped capillary electrophoresis peaks. *Chemometr. Intell. Lab. Syst.* 2006; **82**: 165–175.
22. Zhang F, Li H. Resolution of overlapping capillary electrophoresis peaks by using chemometric analysis: quantification of the components in compound reserpine tablets. *Electrophoresis* 2005; **26**: 1692–1702.
23. Zhang YX, Li H, Hou AX, Havel J. Artificial neural networks based on genetic input selection for quantification in overlapped capillary electrophoresis peaks. *Talanta* 2005; **65**: 118–128.
24. Sentellas S, Saurina J, Hernandez-Cassou S, Galceran MT, Puignou L. Quantitation in multianalyte overlapping peaks from capillary electrophoresis runs using artificial neural networks. *J. Chromatogr. Sci.* 2003; **41**: 145–150.
25. Bocaz-Beneventi G, Latorre R, Farkova M, Havel J. Artificial neural networks for quantification in unresolved capillary electrophoresis peaks. *Anal. Chim. Acta* 2002; **452**: 47–63.
26. Schmitt M. On the complexity of computing and learning with multiplicative neural networks. *Neural Comput.* 2001; **14**: 241–301.
27. Martínez-Estudillo AC, Martínez-Estudillo FJ, Hervás-Martínez C, García-Pedrajas N. Evolutionary product unit based neural networks for regression. *Neural Netw.* 2006; **15**: 477–486.
28. Martínez-Estudillo AC, Hervás-Martínez C, Martínez-Estudillo FJ, García-Pedrajas N. Hybridation of evolutionary algorithms and local search by means of a clustering method. *IEEE Trans. Syst. Man Cybern. B Cybern.* 2006; **36**: 534–546.
29. Duch W, Jankowsky N. *Transfer functions: hidden possibilities for better neural networks*, in 9th European Symposium on Artificial Neural Networks, (ESANN), Brugge (Belgium), 2001, pp. 81–94.
30. Cohen S, Intrator N. *Forward and backward selection in regression hybrid network*, in Third International Workshop on Multiplier Classifier Systems, 2002.
31. Cohen S, Intrator N. A hybrid projection-based and radial basis function architecture: initial values and global optimization. *Pattern Anal. Appl.* 2002; **5**: 113–120.
32. Donoho D. Projection based in approximation and a duality with kernel methods. *Ann. Statist.* 1989; **17**: 58–106.
33. Friedman J. Multivariate adaptive regression splines (with discussion). *Ann. Statist.* 1991; **19**: 1–141.
34. Serrano JM, Silva M. Trace analysis of aminoglycoside antibiotics in bovine milk by micellar electrokinetic chromatography with laser induced fluorescence detection. *Electrophoresis* 2006; **27**: 4703–4710.
35. Duda RO, Hart PE, Stork DG. *Pattern Classification* (2nd edn). Wiley-Interscience: New York, USA, 2001.
36. Angeline PJ, Saunders GM, Pollack JB. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Netw.* 1994; **5**: 54–65.
37. Yao X, Liu Y. A new evolutionary system for evolving artificial neural networks. *IEEE Trans. Neural Netw.* 1997; **8**: 694–713.
38. García N, Hervás C, Muñoz J. Covnet: cooperative coevolution of neural networks. *IEEE Trans. Neural Netw.* 2003; **14**: 575–596.
39. Kirkpatrick S Jr, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science,* 1983; **220**: 671–680.
40. Ventura S, Romero C, Zafra A, Delgado JA, Hervás C. JCLEC: A Java framework for evolutionary computation. *Soft. Comput. D 2007. Available in http://dx.doi.org/10.1007/s00500-007-0172-0*
41. U.S. Environmental Protection Agency. *Guidance on Choosing a Sampling Design for Environmental Data Collection*. EPA: Washington, USA, 2002.

Capítulo 3

Trabajos Publicados, Aceptados y Sometidos sobre Regresión Logística Generalizada utilizando Funciones de Base

3.1. Regresión Logística utilizando Funciones de Base de tipo Unidad Producto

Las publicaciones asociadas a esta parte son:

3.1.1. Simplificación Estructural de Modelos Híbridos Neuro-Logísticos para el Análisis Multi-espectral de Datos obtenidos mediante Sensores Remotos - *Structural Simplification of Hybrid Neuro-Logistic Regression Models in Multispectral Analysis of Remote Sensed Data*

- P.A. Gutiérrez, C. Hervás-Martínez, J.C. Fernández, M. Jurado-Expósito, J.M. Peña-Barragán, F. López-Granados. Structural simplification of hybrid neuro-logistic regression models in multispectral analysis of remote sensed data. Neural Network World 19:1 (2009) 3-20.
  - Estado: Publicado.
  - Índice de Impacto (JCR 2007): 0.280.
  - Área de conocimiento: *Computer Science, Artificial Intelligence*. Ranking 85/93. Cuarto Cuartil.
  - Área de conocimiento: *Neurosciences*. Ranking 202/211. Cuarto Cuartil.

Structural Simplification of Hybrid Neuro-Logistic Regression Models in Multispectral Analysis of Remote Sensed Data

P.A. Gutiérrez[*], *C. Hervás*[*], *J.C. Fernández*[*], *M. Jurado-Expósito*[†], *J.M. Peña-Barragán*[†] *and F. López-Granados*[†]

Abstract: Logistic Regression (LR) has become a widely used and accepted method to analyze binary or multiclass outcome variables, since it is a flexible tool that can predict the probability for the state of a dichotomous variable. A recently proposed LR method is based on the hybridization of a linear model and Evolutionary Product-Unit Neural Network (EPUNN) models for binary classification. This produces a high number of coefficients, so two different methods for simplifying the structure of the final model by reducing the number of initial or PU covariates are presented in this paper, both being based on the Wald test. The first method is a Backtracking Backward Search (BBS) method and the second is similar but based on the standard Simulated Annealing process for the decision steps (SABBS). In this study, we used aerial imagery taken in mid-May to evaluate the potential of two different combinations of LR and EPUNN (LR using PUs (LRPU), as well as LR using Initial covariates and PUs (LRIPU)) and the two presented methods for structural simplification of the final models (BBS and SABBS) for discriminating *Ridolfia segetum* patches (one of the most dominant, competitive and persistent weed in sunflower crops) in one naturally infested field of southern Spain. Then, we compared the performance of these methods to six commonly used classification algorithms, our proposals obtaining a competitive performance and a lower number of coefficients.

Key words: *Logistic regression; structural simplification; product-unit neural networks; evolutionary algorithms*

Received: ??
Revised and accepted: ??

1. Introduction

Classification problems attempt to solve the task of deciding the class membership y of an unknown data item $\mathbf{x}$ based on a data set $D = \{(\mathbf{x}_i, y_i)\}$ $i = 1, ..., n$ of

[*]Dept. of Computer Science and Numerical Analysis, University of Córdoba, 14071, Córdoba, Spain, {`i02gupep@uco.es,chervas@uco.es,fernandezcaballero@gmail.com`}
[†]Institute for Sustainable Agriculture, CSIC, 14080, Córdoba, Spain, {`montse.jurado@ias.csic.es,pa2pebaj@uco.es,flgranados@ias.csic.es`}

data items $\mathbf{x}_i$ with known class membership. The $\mathbf{x}_i$ are usually k-dimensional feature vectors, whose components are called covariates or independent variables. In most problem domains, there is no functional relationship between y and $\mathbf{x}$. In this case the relationship has to be described more generally by a probability distribution $P(\mathbf{x}; y)$; one then assumes that the data set D contains independent samples from P. From statistical decision theory, it is well known that the optimal class membership decision is to choose the class label y that maximizes posteriori distribution $P(y/\mathbf{x})$. Therefore there are different approaches to data classification: one which considers only one distinction between the classes previously defined and assigns a class label to an unknown data item, and another which attempts to model $P(y/\mathbf{x})$. This latter attempt yields not only a class label for a data item, but also a probability of class membership. Logistic Regression (LR), artificial neural networks (ANNs), and decision trees are all members of the second class, although they vary considerably in building an approximation to $P(y/\mathbf{x})$ from data. However, in spite of the great number of techniques developed to solve classification problems, there is no optimum methodology or technique to solve specific problems. This point has encouraged the comparison and combination of different types of classification [1, 2].

A recently proposed LR method is based on the hybridization of a linear model and Evolutionary Product-Unit Neural Network (EPUNN) models for binary [3] and multi-class [4] classification problems. The estimation of the model coefficients is carried out in two phases. First, the number of PU basis functions and the exponents' vector are determined by means of an evolutionary neural network algorithm. Secondly, a standard maximum likelihood optimization method determines the rest of the coefficients in the new space given by the initial variables and the PU basis functions previously estimated. This model allows the generation of non-linear classification surfaces and the identification of possible strong interactions that may exist between the covariates that define the classification problem. These models are less complex (number of new covariates or number of exponents in these covariates) than the alternative higher order polynomial models. However, the models result in a high number of coefficients, so two different methods for simplifying the structure of the final model by reducing the number of initial or PU covariates are presented in this paper, both being based on the Wald test. The first method is a Backtracking Backward Search (BBS) method, that starts with the full model with all the covariates, initial and PUs, pruning variables to the model sequentially and successively, until no further pruning can be made to improve the fit. At each step, the least significant covariate is selected in the discriminant function. The selected covariate is deleted if this does not reduce the fit. If it does, the second least significant covariate is considered. The second method is similar but based on the standard Simulated Annealing process for the decision steps (SABBS).

In order to analyze the performance and robustness of the proposed methodology, it is applied in this to a real agronomical problem that involves the discrimination of *Ridolfia segetum* patches in sunflower fields, using multispectral imagery. Sunflower (*Helianthus annuus* L.) is one of the most abundant crops in Andalusia, Southern Spain, with more than 320,000 ha sown annually [5]. Sunflower sowing and harvesting times are February-March and July-August, respectively, mainly grown under dry land conditions. *R. segetum* Moris (corn caraway) is a very frequent annual, umbelliferous weed that is abundant in clay soils in Andalusia. Its life cycle coincides with that of the sunflower, which enhances its competitive ability and results in an average crop yield reduction of about 32% when infestation is two *R. segetum* plants m^{-2} [6]. This weed is hard to control because it is not controlled by the pre-emergence and pre-plant incorporated herbicides used in sunflower. Consequently, post-emergence strategies such as tillage or hand weeding are commonly used, otherwise weed obstructs the harvester due to the fact that it still has partly green steam during the sunflower harvesting. This is a serious drawback if the harvester is equipped with yield monitor as habitually happens in precision agriculture management. Patchy distribution of broadleaf weeds in sunflower fields is well documented [7]. However, herbicides or other control strategies are not addressed to the infested zones, but are instead applied over the entire fields. The potential for overuse or application and the corresponding eco-environmental problems are evident. To overcome the possibility of minimizing the impact of inappropriate control strategy, the idea of Site-Specific Weed Management (SSWM) has been developed in the context of precision agriculture [8]. A key component of SSWM is that accurate and appropriate weed maps are required to take full advantage of site-specific herbicide applications. Mapping weed patches based on ground survey techniques on field scale is time consuming, expensive and unapproachable in field areas with difficult access. Remote sensing of weed canopies may be more efficient and suitable than field surveys and the majority of studies on discriminating weeds in cultivated systems have involved discrete broadband remote sensing (multispectral sensors) [9]. Approaches based on EPUNNs have been previously applied to remotely sensed images for agronomical objectives [10, 11].

Thus, the goal of this work is to assess the potential of two different combinations of LR and EPUNNs (LR using PUs, LRPU, and LR using Initial covariates and PUs, LRIPU) and the two presented methods for simplifying the structure of the final models (BBS and SABBS) for discriminating *R. segetum* patches in two different naturally infested fields. The results indicate that, with fewer restrictive assumptions, the models proposed are able to reduce the number of coefficients substantially without any significant decrease in classification accuracy.

The rest of the paper is organized as follows. Section 2 is devoted to the description of the standard binary LR model. Section 3 describes PUNNs and their evolution. Sections 4 includes the details of the LRPU and LRIPU models and Section 5 presents the different algorithms for structural simplification proposed in this work. Finally, the experiments and the comparison test carried out are included in Section 6 and Section 7 summarizes the conclusions of our work.

2. Binary Logistic Regression

The binary Logistic Regression (LR) technique considers a binary outcome variable y that is observed together with a vector $\mathbf{x}_i = (1, x_{i1}, x_{i2}, ..., x_{ik})$ of covariates for each of the n_{T} training samples (assuming that the vector of inputs includes the constant term 1 to accommodate the intercept). The two-class is coded via a 1/0 response y_i, associated with the first class. LR [12] is a widely used statistical modeling technique in which the conditional probability p of the dichotomous outcome event is related to a set of explanatory variables $\mathbf{x}$ in the form:

$$\mathrm{logit}(p) = \log\left(\frac{p}{1-p}\right) = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{x} \tag{1}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_k)$ is the vector of coefficients of the model, $\boldsymbol{\beta}^{\mathrm{T}}$ is the transposed vector and the odd of the event is $p/(1-p)$. A simple calculation in (1) shows that the probability of occurrence of an event as a function of the covariates is nonlinear and is given by:

$$p(\mathbf{x}, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}}}{1 + e^{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}}} = \frac{e^{f_{\mathrm{LR}}(\mathbf{x},\boldsymbol{\beta})}}{1 + e^{f_{\mathrm{LR}}(\mathbf{x},\boldsymbol{\beta})}}$$

The most commonly used method for obtaining the vector of coefficients $\boldsymbol{\beta}$ is the Iteratively Re-weighted Least Squares (IRLS), which is a nonlinear optimization algorithm that uses a series of weighted least squares subproblems to find LR model maximum-likelihood coefficient estimation. The implementation of IRLS applied in this work is based on that provided in [13], using the conjugate gradient method for solving the associated matricial equation.

3. Evolutionary Product Unit Neural Networks (EPUNNs)

The models we are testing are LR models based on the hybridization of the standard linear model and nonlinear terms constructed with basis functions obtained from EPUNNs. In this way, this section describes the specific details of the PUNN models and the Evolutionary Algorithm used for obtaining the PU coefficients.

3.1 Product Unit Neural Networks

PUNNs are an alternative to Multilayer Perceptrons (MLPs) and are based on multiplicative nodes instead of additive ones [14]. A multiplicative node is given by:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{k} x_i^{w_{ji}}$$

where k is the number of inputs and $\mathbf{w}_j = (w_{j1}, w_{j2}, ..., w_{jk})$. PUNNs have several advantages, including increased information capacity and the ability to express strong interactions between input variables. Furthermore, it is possible to obtain upper bounds of the Vapnik-Chervonenkis (VC) dimension of PUNNs similar to those obtained for MLPs [15]. Despite these advantages, PUNNs have a major handicap: they have more local minima and more probability of becoming trapped in them [16]. The activation function of the PUNN considered in this work is given by:

$$f_{\mathrm{PUNN}}(\mathbf{x}, \boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{m} \beta_j B_j(\mathbf{x}, \mathbf{w}_j)$$

```
1: Evolutionary Algorithm:
2: Generate a random population of size 1,000
3: repeat
4:     Calculate the fitness of every individual in the population
5:     Rank the individuals with respect to their fitness
6:     The best individual is copied into the new population
7:     The best 10% of population individuals are replicated and they substitute
       the worst 10% of individuals
8:     Apply parametric mutation to the best 10% of individuals
9:     Apply structural mutation to the remaining 90% of individuals
10: until the stopping criterion is fulfilled
```

Fig. 1 *Evolutionary Algorithm (EA) framework*

with $\boldsymbol{\theta} = (\boldsymbol{\beta}, \mathbf{w}_1, ..., \mathbf{w}_m)$. The outputs of the PUNNs are interpreted from the probability point of view, so the softmax transformation is used. The softmax activation function is given by:

$$g(\mathbf{x}, \boldsymbol{\theta}) = \frac{e^{f_{\text{PUNN}}(\mathbf{x},\boldsymbol{\beta})}}{1 + e^{f_{\text{PUNN}}(\mathbf{x},\boldsymbol{\beta})}}$$

3.2 Evolutionary Algorithm

In the past decade, Evolutionary Algorithms (EAs) and ANNs have been combined as a key research area, providing an interesting platform for simultaneously optimizing both the weights and architecture of connectionist models [17, 18, 19] while avoiding the shortcomings of traditional BackPropagation [20]. In this way, an Evolutionary Algorithm (EA) has been selected to estimate the coefficients and the structure of the PUNNs that minimize the classification error function. The complexity of the error surface of the proposed model justifies the use of an EA as part of the process of estimation of the model coefficients. Among the different metaheuristics, the EA selected in this work has proved excellent results when evolving PUNNs [21].

The basic framework of the evolutionary algorithm is the following: the search begins with an initial population of neural networks and, in each iteration, the population is updated using a population-update algorithm which evolves both its structure and weights. The population is subject to the operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving ANNs [22, 23].

The algorithm evolves architectures and connection weights simultaneously, each individual being a fully specified PUNN. The PUNNs are represented using an object-oriented approach and the algorithm deals directly with the PUNN phenotype. Each connection is specified by a binary value indicating if the connection exists and a real value representing its weight. As the crossover is not considered, this object-oriented representation does not assume a fixed order between the different hidden nodes. The general structure of the EA has been included in Fig. 1.

The fitness function $A(\boldsymbol{\theta})$ is a strictly decreasing transformation of the cross-entropy error function $E(\boldsymbol{\theta})$ [24] given by $A(\boldsymbol{\theta}) = 1/(1 + E(\boldsymbol{\theta}))$ where $\boldsymbol{\theta}$ are the parameters of the individual and $E(\boldsymbol{\theta})$ is:

$$E(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log g(\mathbf{x}_i, \boldsymbol{\theta}) + (1 - y_i)\log(1 - g(\mathbf{x}_i, \boldsymbol{\theta}))\right]$$

where $g(\mathbf{x}, \boldsymbol{\theta})$ is the softmax activation function.

The severity of both structural and parametric mutations depends on the temperature $T(\boldsymbol{\theta})$ of the PUNN model, defined by:

$$T(\boldsymbol{\theta}) = 1 - A(\boldsymbol{\theta}), \ \ 0 \leq T(\boldsymbol{\theta}) \leq 1$$

where $A(\boldsymbol{\theta})$ is the fitness of the individual with parameters $\boldsymbol{\theta}$. Parametric mutation (Fig. 1, step 8) is accomplished for each weight of the model β_j or w_{ji} adding Gaussian noise:

$$\begin{aligned} w_{ji}(t+1) = &\ \ w_{ji}(t) + \xi_1(t) \\ \beta_j(t+1) = &\ \ \beta_j(t) + \xi_2(t) \end{aligned}$$

where $\xi_i(t)$ represents a one dimensional normally distributed random variable, $N(0, \alpha_i(t) \cdot T(g))$. The $\alpha_i(t)$ value is updated throughout the evolutionary process, applying the simplest heuristic 1/5 success rule of Rechenberg [25]. This allows an initial coarse-grained search, and a progressively finer-grained search, as a model approaches a solution to the problem. The modifications on the coefficients β_j should be higher than the modifications on the exponents w_{ji}, what is achieved using a higher initial $\alpha_2(t)$ value, i.e. $\alpha_2(0) >> \alpha_1(0)$. The weights are sequentially mutated, hidden node after hidden node, and a standard Simulated Annealing process is applied to accept or reject the modifications in each node.

On the other hand, there are five different structural mutations (Fig. 1, step 9): node deletion, connection deletion, node addition, connection addition and node fusion. These five mutations are applied sequentially to each network. The first four are identical to the mutations in the generalized acquisition of recurrent links (GNARL) model [20]. The node fusion mutation operates randomly selecting two hidden nodes of the neural net and substituting them by a new node that is a combination of both. All the mutations are made sequentially in the given order, with probability $T(\boldsymbol{\theta})$, in the same generation on the same network. If the probability does not select any mutation, one of the mutations is chosen at random and applied to the network.

For further details about parametric and structural mutations and other characteristics of the algorithm see [3, 21, 26].

4. Hybrid Neuro-Logistic models

As previously stated, the Neuro-Logistic Regression models used in this paper include Logistic Regression using Product Units (LRPU) and Logistic Regression using Initial covariates and Product Units (LRIPU).

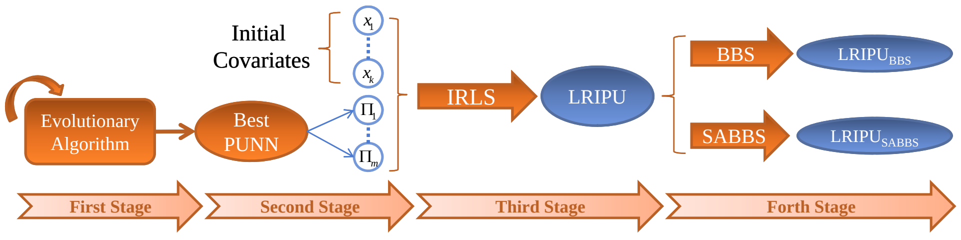


Fig. 2 *Scheme of the LRIPU coefficient optimization process*

4.1 Logistic Regression using Product Units (LRPU)

LRPU is a hybrid method that considers the EA presented in the previous section in order to obtain an EPUNN structure and hidden neuron weights that are accurate enough. When these are obtained, it applies the IRLS mechanism over the PU basis functions of the EPUNN selected. So the LRPU model composed only of PU basis function is given by:

$$f_{\text{LRPU}}(\mathbf{x}, \boldsymbol{\theta}) = \alpha_0 + \sum_{j=1}^{m} \alpha_j B_j(\mathbf{x}, \mathbf{w}_j)$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \mathbf{W})$, $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, ..., \alpha_m)$ and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m)$, with $\mathbf{w}_j = (w_{j1}, w_{j2}, ..., w_{jk})$. The coefficients $\mathbf{W}$ are given by the EA, and are not adjusted by the IRLS method. The IRLS method only optimizes the linear part of the model, i.e., the $\boldsymbol{\alpha}$ coefficients.

4.2 Logistic Regression using Initial covariates and Product Units (LRIPU)

The LRIPU model used is a hybridization of the LR model and the EPUNNs previously presented. The model extends LRPU, considering the initial covariates $\mathbf{x}$ of the problem. Its expression is given by:

$$f_{\text{LRIPU}}(\mathbf{x}, \boldsymbol{\theta}) = \alpha_0 + \sum_{j=1}^{m} \alpha_j B_j(\mathbf{x}, \mathbf{w}) + \sum_{j=1}^{k} \alpha_{(m+j)} x_j$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \mathbf{W})$, $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, ..., \alpha_m, \alpha_{m+1}, ..., \alpha_{m+k})$ and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m)$. The values adjusted with IRLS correspond to the $\boldsymbol{\alpha}$ vector, the coefficients $\mathbf{W}$ again being given by the EA.

An scheme of the different steps necessary for obtaining the model coefficients is given in Fig. 2. In the first stage, the EA is applied and, in the second stage, the PUs from the hidden layer of the best PUNN are extracted and appended to the original covariates input space. Then, the third stage consists of applying the IRLS method in order to obtain the coefficients of the LR model. The fourth stage consists of simplifying the structure of the final model and it will be presented in the next section.

```
Backtracking Backward Search Algorithm:
Apply IRLS over $V$, obtaining the $\boldsymbol{\alpha}$ coefficients and the associated $CCR_{\mathrm{T}}$
$exit \leftarrow$ false
repeat
  for all $v_i$ in $V$ do
    Obtain Wald statistic of the variable $v_i$
    $p_i \leftarrow p$-value of the Wald test with $H_0 \equiv \alpha_i = 0$
  end for
  $v_{\mathrm{1st}} \leftarrow$ variable with maximum $p_i$
  $V^{'} \leftarrow V - v_{\mathrm{1st}}$
  Apply IRLS over $V^{'}$, obtaining the $\boldsymbol{\alpha}^{'}$ coefficients and the associated $CCR^{'}_{\mathrm{T}}$
  if $CCR_{\mathrm{T}} > CCR^{'}_{\mathrm{T}}$ then
    $v_{\mathrm{2nd}} \leftarrow$ variable with second maximum $p_i$
    $V^{'} \leftarrow V - v_{\mathrm{2nd}}$
    Apply IRLS over $V^{'}$, obtaining the $\boldsymbol{\alpha}^{'}$ coefficients and the associated $CCR^{'}_{\mathrm{T}}$
    if $CCR_{\mathrm{T}} > CCR^{'}_{\mathrm{T}}$ then
      $exit \leftarrow$ true
    else
      $V \leftarrow V^{'}$
    end if
  else
    $V \leftarrow V^{'}$
  end if
until $exit$=true
```

Fig. 3 *BBS structural simplification algorithm*

5. Structural Simplification of the LRPU and LRIPU models

In order to reduce the size of LRPU and LRIPU models, we propose a forth stage of structural simplification (see Fig. 2). Two different algorithms are presented: a Backtracking Backward Search (BBS) and a Simulated Annealing Backtracking Backward Search (SABBS). Both methods make use of the Wald statistic, which is a score function commonly considered in LR. The Wald test is a statistical test, used to check whether the effect of a covariate exists or not in the odd of an event. In other words, it tests whether an independent covariate has a statistically significant effect over the dependent variable. As a result, a critical value (p-value) is obtained for each variable, where the associated coefficient equal to zero is the null hypothesis (H_0) to be contrasted. We consider both initial and PU covariates of the LRPU and LRIPU models and apply this test in order to simplify their structure.

5.1 Structural Simplification by a Backtracking Backward Search (BBS)

The first method presented starts with the full model with all the covariates, initial and PUs, pruning variables to the model sequentially and successively, until no further pruning can be made to improve the fit. It uses the Wald statistic for sorting the covariates (PU transformations or initial variables) and tries the elimination of a covariate in each step by a backtracking procedure. First the least significant covariate is selected in the discriminant function. The selected covariate is deleted if this does not reduce the fit. If it does, the second least significant covariate is considered. In this way, the algorithm is a Backtracking Backward method.

The procedure ends when none of the two chosen covariates is deleted. The pseudo-code associated with this algorithm is presented in Fig. 3, where V is the current set of covariates (initial or PUs) and CCR_{T} is the Correct Classification Rate or accuracy in the training set, which is defined by:

$$CCR = \frac{1}{N}\sum_{i=1}^{n} I(C(\mathbf{x}_n) = y_i), \qquad (2)$$

where $I(\bullet)$ is the zero-one loss function, $C(\mathbf{x}_n)$ is the class predicted by the model, y_i is the expected class value and n is the number of observations.

A scheme of the BBS algorithm is given in Fig. 4.

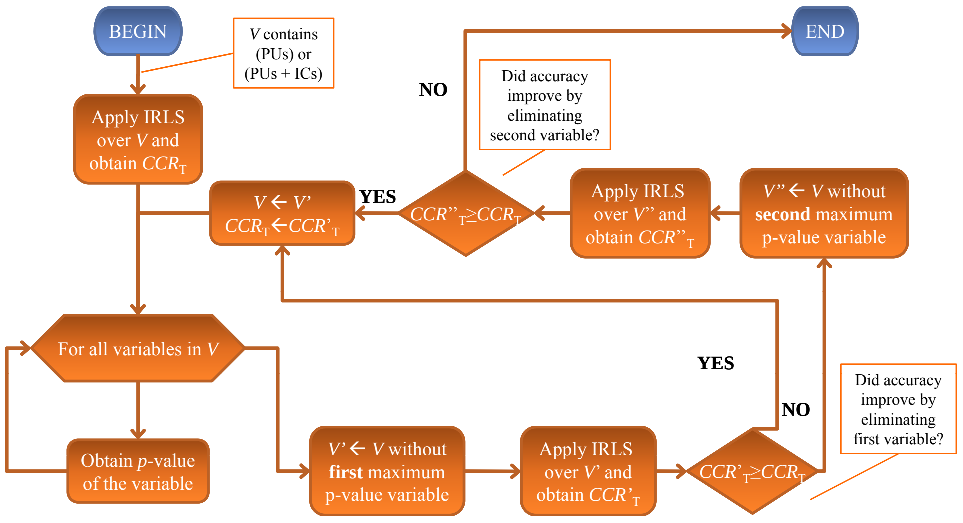

BEGIN
V contains (PUs) or (PUs + ICs)
Apply IRLS over V and obtain CCR_T
NO
END
Did accuracy improve by eliminating second variable?
YES
V ← V'
CCR_T ← CCR'_T
CCR''_T ≥ CCR_T
Apply IRLS over V'' and obtain CCR''_T
V'' ← V without second maximum p-value variable
For all variables in V
YES
NO
Did accuracy improve by eliminating first variable?
Obtain p-value of the variable
V' ← V without first maximum p-value variable
Apply IRLS over V' and obtain CCR'_T
CCR'_T ≥ CCR_T

Fig. 4 *Scheme of the Backtracking Backward Search (BBS) algorithm*

5.2 Structural Simplification by a Simulated Annealing Backtracking Backward Search (SABBS)

The second method is based on the standard SA heuristic [27]. The algorithm is very similar to that presented in the previous subsection but, when the elimina-

```
1: Simulated Annealing Backtracking Backward Search:
2: Apply IRLS over $V$, obtaining the $\boldsymbol{\alpha}$ coefficients and the associated $CCR_{\mathrm{T}}$
3: $exit \leftarrow$ false; $T \leftarrow 0.01\cdot$ Number variables
4: repeat
5:   for all $v_i$ in $V$ do
6:     Obtain Wald statistic of the variable $v_i$
7:     $p_i \leftarrow p$-value of the Wald test with $H_0 \equiv \alpha_i = 0$
8:   end for
9:   $v_{\mathrm{1st}} \leftarrow$ variable with maximum $p_i$
10:  $V^{'} \leftarrow V - v_{\mathrm{1st}}$
11:  Apply IRLS over $V^{'}$, obtaining the $\boldsymbol{\alpha}^{'}$ coefficients and the associated $CCR_{\mathrm{T}}^{'}$
12:  $dif \leftarrow (CCR_{\mathrm{T}}^{'} - CCR_{\mathrm{T}})$
13:  if $dif < 0$ and $U(0,1) > e^{dif/T}$ then
14:    $v_{\mathrm{2nd}} \leftarrow$ variable with second maximum $p_i$
15:    $V^{'} \leftarrow V - v_{\mathrm{2nd}}$
16:    Apply IRLS over $V^{'}$, obtaining the $\boldsymbol{\alpha}^{'}$ coefficients and the associated
       $CCR_{\mathrm{T}}^{'}$
17:    $dif \leftarrow (CCR_{\mathrm{T}}^{'} - CCR_{\mathrm{T}})$
18:    if $dif < 0$ and $U(0,1) > e^{dif/T}$ then
19:      $exit \leftarrow$ true
20:    else
21:      $V \leftarrow V^{'}$
22:    end if
23:  else
24:    $V \leftarrow V^{'}$
25:  end if
26:  $T \leftarrow 0.2T$
27: until $exit$=true
```

Fig. 5 *SABBS structural simplification algorithm*

tion of a variable results in a lower training CCR (CCR_{T}), it is accepted with a probability $e^{dif/T}$, where dif is the CCR_{T} difference between the model obtained using the variable and not using it, $dif = (CCR_{\mathrm{T}}^{'} - CCR_{\mathrm{T}})$, and T is the current temperature. The initial value for the temperature is $T = 0.01N$, where N is the number of initial covariates and PUs of the model. In each iteration, the temperature is updated with a $r = 0.2$ freezing factor. The pseudo-code associated with this algorithm is presented in Fig. 5, where $U(0,1)$ is a random uniform variable in the interval $[0,1]$. Finally, a scheme of the SABBS algorithm is given in Fig. 6. Those steps different from the BBS algorithm are marked in dark grey.

6. Experiments

We have tested the described methodology in a real agronomical problem of precision farming, consisting of mapping weed patches in crop fields, through remote sensed data.

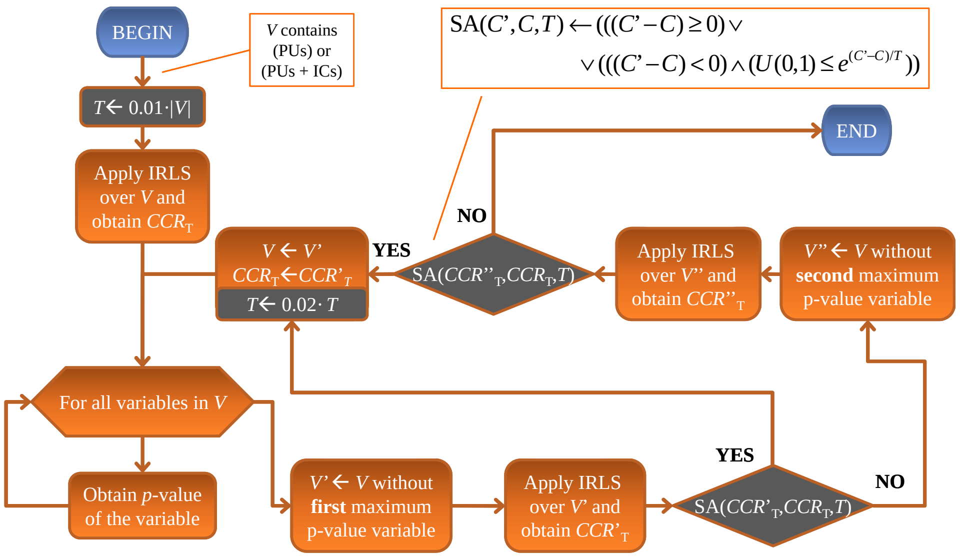

BEGIN
V contains (PUs) or (PUs + ICs)
$T \leftarrow 0.01 \cdot |V|$
Apply IRLS over V and obtain CCR_T
$SA(C', C, T) \leftarrow (((C' - C) \geq 0) \vee (((C' - C) < 0) \wedge (U(0,1) \leq e^{(C'-C)/T}))$
END
NO
YES
$V \leftarrow V'$ $CCR_T \leftarrow CCR'_T$ $T \leftarrow 0.02 \cdot T$
$SA(CCR''_T, CCR_T, T)$
Apply IRLS over V'' and obtain CCR''_T
V'' ← V without second maximum p-value variable
For all variables in V
Obtain p-value of the variable
V' ← V without first maximum p-value variable
Apply IRLS over V' and obtain CCR'_T
YES
NO
$SA(CCR'_T, CCR_T, T)$

Fig. 6 *Simulated Annealing Backtracking Backward Search (SABBS) algorithm*

6.1 Study sites, materials and experimental design

The study was conducted at two fields in Andalusia, southern Spain: at Matabueyes, 42 ha (coordinates 37°8'N, 4°8'W, WGS84), and at Santa Cruz, 28 ha (coordinates 37°8'N, 4°6'W, WGS84), in 2003 and 2004, respectively. Both fields were naturally infested by *R. segetum*. Conventional-Colour (CC, 400-700 nm) and Colour-near InfraRed (CIR, 500-900 nm) aerial photographs of the field were taken in mid-May. The four input variables included the digital values of all bands in each available image, that is: CC images responded to Blue (B, 400-500 nm), Green (G, 500-600 nm), and Red (R, 600-700 nm) broad bands of the electromagnetic spectrum, and CIR images to G, R and Near-InfraRed (NIR, 700-900 nm) bands. Further information about acquisition of aerial photographs, digitization and orto-rectification is given in [10] and [28].

To train and validate the classification models, a random ground sampling procedure was carried out, ensuring that all parts of the field area had an equal chance of being sampled with no operator bias [29]. For each image, we obtained 2,400 pixels as ground-truth pixels and georeferenced a total of 1,600 pixels in each phenological stage, where 800 pixels corresponded to *R. segetum* class, 400 pixels corresponded to the bare soil class and 400 corresponded to that of sunflower. The objective is the differentiation between *R. segetum* and weed-free (bare soil and sunflower) pixels. The experimental design was conducted using a stratified 10-fold cross-validation procedure with ten runs for each fold. This paper extends the results presented in [10] (where the combination of LR and EPUNNs was applied to the same problem) by studying the influence of the structural simplification methods herein proposed (BBS and SABBS) in the final results. Furthermore, the experimental design followed in [10] consisted of a holdout cross-validation and only one execution, the results not being comparable to those presented in this paper.

The models compared in the different experiments are the following: firstly, the application of the IRLS algorithm only over the PU basis functions extracted from EPUNN model of the EA (LRPU) and over the same basic functions together with initial covariates (LRIPU). Secondly, the two different structural simplification algorithms (BBS and SABBS) are applied over both LRPU and LRIPU models ($\mathrm{LRPU_{BBS}}$, $\mathrm{LRPU_{SABBS}}$, $\mathrm{LRIPU_{BBS}}$ and $\mathrm{LRIPU_{SABBS}}$).

Afterwards, all these models are compared to six machine learning classifiers: LR with attribute selection (SimpleLogistic), LR with a full logistic model (MultiLogistic), Logistic Model Trees (LMT), the C4.5 classification tree inducer, the naive Bayes tree learning algorithm (NBTree) and the AdaBoost.M1 algorithm with 100 maximum number of iterations (AdaBoost100) and using C4.5 as the base classifier. The description of these algorithms can be found in [30].

The EA was implemented using the Evolutionary Computation framework JCLEC [31] (`http://jclec.sourceforge.net`) and it is available in the non commercial JAVA tool named KEEL [32] (`http://www.keel.es`). The parameters used in the EA are common for both datasets. The PUNN models have the following structure: one input layer with four input nodes (corresponding to R, G, B and NIR digital values), one hidden layer with at least one hidden node and a maximum of six nodes (the number of hidden nodes is modified by the structural mutation) and one output layer with one output node, corresponding to the probability of *R. segetum* presence. The number of nodes that can be added or removed in a structural mutation is within the $[1, 2]$ interval. The number of connections that can be added or removed in a structural mutation is within the $[1, c]$ interval, where c is the integer part of a third of the number of connections in the model. The stop criterion is reached when the following condition is fulfilled: for 20 generations there is improvement neither in the average performance of the best 10% of the population nor in the fitness of the best individual. Regarding the parameters of BBS and SABBS methods (i.e. the number of covariates analyzed in each step, 2, the value for the initial temperature, $0.01 \cdot N$ and the freezing factor, $r = 0.2$), they have been obtained as the best result of a preliminary experimental design.

The other algorithms are available as part of the WEKA machine learning workbench [33] and we applied them to the Matabueyes and Santa Cruz datasets, selecting their default parameter values.

6.2 Evaluation of the Structural Simplification methods

Performance of each model has been evaluated using the CCR in the generalization set (CCR_{G}). In Table I, we show the mean and the standard deviation of this CCR_{G} for a total of 100 executions, and the mean and the standard deviation of the number of coefficients of the corresponding models (including α_j or w_{ji} coefficients). From the analysis of the LRPU model results, it can be concluded that the structural simplification methods considerably reduce the number of coefficients, this difference being higher for the $\mathrm{LRPU_{SABBS}}$ method. The generalization accuracy of the LRPU models is similar or better after simplifying their structure. A very similar behaviour is observed with respect to the LRIPU model: the accuracy is similar or better when using structural simplification (especially when using the $\mathrm{LRIPU_{BBS}}$ method) and the number of coefficients is significantly reduced (espe-

	Matabueyes		Santa Cruz	
	CCR_{G}	#Coef.	CCR_{G}	#Coef.
Method	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD
LRPU	69.53 ± 3.58	19.66 ± 2.19	75.88 ± 2.79	25.99 ± 3.52
LRPU$_{\mathrm{BBS}}$	70.07 ± 3.47	17.21 ± 3.14	76.22 ± 2.77	23.96 ± 3.14
LRPU$_{\mathrm{SABBS}}$	69.49 ± 3.45	13.69 ± 2.04	75.45 ± 3.03	21.70 ± 2.83
LRIPU	69.84 ± 3.57	23.08 ± 2.46	76.10 ± 2.67	27.53 ± 3.53
LRIPU$_{\mathrm{BBS}}$	70.32 ± 3.58	20.20 ± 3.45	76.26 ± 2.73	25.68 ± 3.19
LRIPU$_{\mathrm{SABBS}}$	70.10 ± 3.58	15.56 ± 2.91	75.85 ± 2.78	22.80 ± 3.08

Tab. I *Statistical results (Mean and Standard Deviation, SD) of the $CCR_{\mathbf{G}}$ and the number of coefficients (#Coef.) obtained using the different methods proposed*

cially when using the LRIPU$_{\mathrm{SABBS}}$ method). When analyzing both LRPU and LRIPU models and their structural simplification variants, the best CCR_{G} results are obtained by the LRIPU$_{\mathrm{BBS}}$ method and a similar accuracy is obtained by the LRIPU$_{\mathrm{SABBS}}$ methodology but with a lower number of connections.

In order to ascertain the statistical significance of the observed differences between the mean CCR_{G} and the mean #Coef. of the best models obtained for each methodology, we have applied the ANalysis Of VAriance (ANOVA) technique [34, 35, 36]. First of all, a non-parametric Kolmogorov-Smirnov test (KS-test) with a signification level $\alpha = 0.05$ was used to evaluate if the CCR_{G} and #Coef. values follow a normal distribution. As a result, a normal distribution can be assumed because all the p-values were higher than 0.05.

	Matabueyes	
	CCR_{G}	#Coef.
F-Test	0.487	0.000($*$)
Ranking of means	$\mu_{\mathrm{LRIPU_B}} \geq \mu_{\mathrm{LRIPU_S}} \geq \mu_{\mathrm{LRPU_B}} \geq$ $\geq \mu_{\mathrm{LRIPU}} \geq \mu_{\mathrm{LRPU}} \geq \mu_{\mathrm{LRPU_S}}$	$\mu_{\mathrm{LRPU_S}} < \mu_{\mathrm{LRIPU_S}} < \mu_{\mathrm{LRPU_B}} <$ $< \mu_{\mathrm{LRPU}} \leq \mu_{\mathrm{LRIPU_B}} < \mu_{\mathrm{LRIPU}}$
	Santa Cruz	
	CCR_{G}	#Coef.
F-Test	0.332	0.000($*$)
Ranking of means	$\mu_{\mathrm{LRIPU_B}} \geq \mu_{\mathrm{LRPU_B}} \geq \mu_{\mathrm{LRIPU}} \geq$ $\geq \mu_{\mathrm{LRPU}} \geq \mu_{\mathrm{LRIPU_S}} \geq \mu_{\mathrm{LRPU_S}}$	$\mu_{\mathrm{LRPU_S}} \leq \mu_{\mathrm{LRIPU_S}} \leq \mu_{\mathrm{LRPU_B}}$; $\mu_{\mathrm{LRPU_S}} < \mu_{\mathrm{LRPU_B}}$; $\mu_{\mathrm{LRPU_B}} < \mu_{\mathrm{LRIPU_B}} \leq \mu_{\mathrm{LRPU}} <$ $< \mu_{\mathrm{LRIPU}}$

($*$): Statistical significant different with $p-$value < 0.05
B: Backtracking Backward Search (BBS)
S: Simulated Annealing Backtracking Backward Search (SABBS)

Tab. II $p-$values *of the Snedecor's F ANOVA I test and ranking of means of the Tukey statistical multiple comparison tests for the CCR_{G} and #Coef. using the six different methodologies*

The ANOVA test involves a linear regression model in which CCR_{G} or #Coef. are the dependent variables and the independent variable is the type of methodology or model used for classification. The comparison was made in terms of a critical level of the Snedecor's F value. If the significance level, p, was higher than this critical value, α, we rejected the hypothesis of identical mean CCR_{G} or #Coef. In our case, this hypothesis is accepted for mean CCR_{G} values in both locations, because the $p-$values were 0.487 and 0.332 (see Table II), they being higher than a standard significance coefficient $\alpha = 0.05$. Consequently, we can conclude that there are not significant differences in mean CCR_{G}. The same hypothesis is not accepted for #Coef., because the $p-$values are equal to 0.000, lower than $\alpha = 0.05$.

Based on these last results and accepting the hypothesis that the variances for the different levels of the #Coef. are equal, we perform a Tukey test [35] for ranking the averages of each level of the factor. Our aim is to find the level of the factor whose mean #Coef. was significantly lower than the average of the rest of the levels of the factor. Table II shows the results obtained following a post-hoc Tukey's multiple comparison test, and the ranking of the different methodologies based on these results. In these rankings, $\mu_{\mathrm{a}} \geq \mu_{\mathrm{b}}$ is used to express that, although the mean CCR_{G} or #Coef. of the "a" methodology is higher than that of "b", the differences are not significant and $\mu_{\mathrm{a}} > \mu_{\mathrm{b}}$ is used to express that the mean results from "a" methodology are significantly higher that those from "b".

From the analysis of the statistical test results, we propose $\mathrm{LRPU}_{\mathrm{SABBS}}$ methodology for both locations, because it results in very similar CCR_{G} levels but with a significantly lower number of coefficients.

6.3 Comparison to other machine learning algorithms

In Table III, the most accurate results ($\mathrm{LRIPU}_{\mathrm{BBS}}$ and $\mathrm{LRIPU}_{\mathrm{SABBS}}$) and the most interpretable results ($\mathrm{LRPU}_{\mathrm{SABBS}}$) have been included, together with the results obtained by using the different WEKA algorithms. As the algorithms considered are all deterministic, we performed ten runs of a ten-fold stratified cross-validation (using the same splits into training/test set for every method), which also gives a hundred data points. In order to obtain the number of coefficients used in each WEKA model, we had to modify the source code of the algorithms for including these values into their outputs, taking into account the special characteristics of each model. These values have also been included in Table III.

Following the same methodology than in the previous subsection, statistical tests have been applied in order to ascertain the statistical significance of the observed differences between the mean CCR_{G} and the results are included in Table IV. For the mean #Coef., the differences are very high and obvious and, in our opinion, a statistical is not needed.

In Matabueyes, the accuracy of the models proposed in this paper is significantly higher than that obtained by NBTree, MultiLogistic and SimpleLogistic and similar to that obtained by the rest of algorithms. In Santa Cruz, a significantly higher accuracy is obtained using LMT, C4.5 and AdaBoost methods. In any case, the number of coefficients of the proposed models (specially that of the $\mathrm{LRPU}_{\mathrm{SABBS}}$ model) is lower than that of those models which obtain a similar or better accuracy (LMT, C4.5 and AdaBoost100), resulting in more interpretable and efficient models.

	Matabueyes		Santa Cruz	
	CCR_{G}	#Links	CCR_{G}	#Links
Method	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
SLogistic	65.86 ± 3.30	4.46 ± 0.50	65.22 ± 3.89	3.54 ± 0.63
MLogistic	66.14 ± 3.27	5.00	65.23 ± 4.20	5.00
LMT	71.68 ± 3.61	181.04 ± 91.23	80.36 ± 3.05	263.51 ± 76.59
C4.5	70.74 ± 3.54	33.63 ± 8.39	79.44 ± 3.29	44.68 ± 6.90
NBTree	66.78 ± 4.01	20.72 ± 9.08	75.96 ± 4.30	43.06 ± 16.56
Ada100	70.89 ± 3.56	108.42 ± 29.96	80.03 ± 2.96	277.66 ± 122.22
$\mathrm{LRIPU}_{\mathrm{BBS}}$	70.32 ± 3.58	20.20 ± 3.45	76.26 ± 2.73	25.68 ± 3.19
$\mathrm{LRIPU}_{\mathrm{SABBS}}$	70.10 ± 3.58	15.56 ± 2.91	75.85 ± 2.78	22.80 ± 3.08
$\mathrm{LRPU}_{\mathrm{SABBS}}$	69.49 ± 3.45	13.69 ± 2.04	75.45 ± 3.03	21.70 ± 2.83

Tab. III *Statistical results obtained with the proposed methodologies compared to other machine learning algorithms*

	Matabueyes
	CCR_{G}
F-Test	0.000(∗)
Ranking of means	$\mu_{\mathrm{LMT}} \geq \mu_{\mathrm{Ada100}} \geq \mu_{\mathrm{C4.5}} \geq \mu_{\mathrm{LRIPU_B}} \geq \mu_{\mathrm{LRIPU_S}} \geq \mu_{\mathrm{LRPU_S}}$; $\mu_{\mathrm{LMT}} > \mu_{\mathrm{LRIPU_S}}$; $\mu_{\mathrm{LRPU_S}} > \mu_{\mathrm{NBTree}} \geq \mu_{\mathrm{MLogistic}} \geq \mu_{\mathrm{SLogistic}}$
	Santa Cruz
	CCR_{G}
F-Test	0.000(∗)
Ranking of means	$\mu_{\mathrm{LMT}} \geq \mu_{\mathrm{Ada100}} \geq \mu_{\mathrm{C4.5}} > \mu_{\mathrm{LRIPU_B}} \geq \mu_{\mathrm{NBTree}} \geq \mu_{\mathrm{LRIPU_S}} \geq \mu_{\mathrm{LRPU_S}}$; $\mu_{\mathrm{LRPU_S}} > \mu_{\mathrm{MLogistic}} \geq \mu_{\mathrm{SLogistic}}$

(∗): Statistical significant different with $p-$value < 0.05
B: Backtracking Backward Search (BBS)
S: Simulated Annealing Backtracking Backward Search (SABBS)

Tab. IV $p-$values *of the Snedecor's F ANOVA I test and ranking of means of the Tukey statistical multiple comparison tests for the* CCR_{G} *using the different methodologies proposed and the different machine learning algorithms*

7. Conclusions

The structural simplification methods (BBS and SABBS) presented in this paper have demonstrated an important coefficient reduction for both the hybrid neuro-logistic models proposed in [3] (LRPU and LRIPU), yielding to a better or similar accuracy. In this way, more interpretable models that can lead to a better understanding of the classification problem tackled have been obtained. These models can provide information to program the suitable wavelengths of further Compact Airborne Spectral Imager (CASI) images for Site-Specific Weed Management (SSWM). The next investigation could address the acquisition of specific satellite imagery for mapping *R. Segetum* in larger areas, using only the more relevant channels and reducing the cost of the images. Moreover, the comparison of these models to six different commonly used machine learning methods has shown the LRIPU$_{\mathrm{BBS}}$ and LRIPU$_{\mathrm{SABBS}}$ methods as very competitive models with a lower number of coefficients and has demonstrated their capability to analyze multispectral imagery for predicting *R. segetum* presence probability in the field of study, providing a useful tool for early SSWM.

Acknowledgements

This work has been partially financed by the CICYT-TIN2005-08386-C05-02 and CICYT-AGL-2008-04689-CO3-03 projects (Spanish Inter-Ministerial Commission of Science and Technology), FEDER funds and the P08-TIC-3745 project (Junta de Andalucía). The research done by P.A. Gutiérrez and J.C. Fernández has been financed respectively by the FPU (grant reference AP-2006-01746) and the FPI (grant reference BES-2006-12543) Predoctoral Programs (Spanish Ministry of Education and Science).

References

[1] R. L. Major and C. T. Ragsdale, "Aggregating expert predictions in a networked environment," *Computers & Operations Research*, vol. 28, no. 12, pp. 1231–1244, 2001.

[2] R. Beghdad, "Applying fisher's filter to select kdd connections' features and using neural networks to classify and detect attacks," *Neural Network World*, vol. 17, no. 1, pp. 1–16, 2007.

[3] C. Hervas-Martinez and F. Martinez-Estudillo, "Logistic regression using covariates obtained by product-unit neural network models," *Pattern Recognition*, vol. 40, no. 1, pp. 52–64, 2007.

[4] C. Hervás-Martínez, F. J. Martínez-Estudillo, and M. Carbonero-Ruz, "Multilogistic regression by means of evolutionary product-unit neural networks," *Neural Networks*, vol. 21, no. 7, pp. 951–961, 2008.

[5] "MAPA," Spanish Ministry of Agriculture, Fisheries and Food, 2007. [Online]. Available: http://www.mapa.es/es/estadistica/infoestad.html

[6] P. Carranza-Cañadas, M. Saavedra, and L. García-Torres, "Competition of ridolfia segetum and sunflower," *Weed Research*, vol. 35, no. 5, pp. 369–375, 1995.

[7] M. Jurado-Expósito, F. López-Granados, J. L. González-Andújar, and L. García-Torres, "Characterizing population growth rate of convolvulus arvensis in no-tillage systems," *Crop Science*, vol. 45, no. 21, pp. 2106–2112, 2005.

[8] A. Srinivasan, *Handbook of Precision Agriculture.* Haworth Press Inc, 2006.

[9] R. B. Brown and S. D. Noble, "Site-specific weed management: sensing requirements - what do we need to see?" *Weed Science*, vol. 53, no. 2, pp. 252–258, 2005.

[10] P. A. Gutiérrez, F. López-Granados, J. M. Peña-Barragán, M. Jurado-Expósito, and C. Hervás-Martínez, "Logistic regression product-unit neural networks for mapping *Ridolfia segetum* infestations in sunflower crop using multitemporal remote sensed data," *Computers and Electronics in Agriculture*, vol. 62, no. 2, pp. 293–306, 2008.

[11] P. A. Gutiérrez, F. López-Granados, J. M. Peña-Barragán, M. Jurado-Expósito, M. T. Gómez-Casero, and C. Hervás, "Mapping sunflower yield as affected by *Ridolfia segetum* patches and elevation by applying evolutionary product unit neural networks to remote sensed data," *Computers and Electronics in Agriculture*, vol. 60, no. 2, pp. 122–132, 2008.

[12] D. W. Hosmer and S. Lemeshow, *Applied logistic regression (Wiley Series in probability and statistics).* New York; Chichester: Wiley Interscience, September 2000.

[13] P. Komarek and A. W. Moore, "Making logistic regression a core data mining tool with tr-irls," in *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 685–688.

[14] R. Durbin and D. Rumelhart, "Products units: A computationally powerful and biologically plausible extension to backpropagation networks," *Neural Computation*, vol. 1, no. 1, pp. 133–142, 1989.

[15] M. Schmitt, "On the complexity of computing and learning with multiplicative neural networks," *Neural Computation*, vol. 14, pp. 241–301, 2002.

[16] A. Ismail and A. P. Engelbrecht, "Pruning product unit neural networks," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2002, pp. 257–262.

[17] M. Köppen and K. Yoshida, "Many-objective training of a multi-layer perceptron," *Neural Network World*, vol. 17, no. 6, pp. 627–637, 2007.

[18] S. Michal and K. Stefan, "Automatic generation of neural network structures using genetic algorithm," *Neural Network World*, vol. 15, no. 5, pp. 381–394, 2005.

[19] J. T. Tsai, J. H. Chou, and T. K. Liu, "Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69–80, 2006.

[20] P. J. Angeline, G. M. Sauders, and J. B. Pollack, "An evolutionary algorithm that constructs recurren neural networks," *IEEE Transactions on Neural Networks*, vol. 5, pp. 54–65, 1994.

[21] F. J. Martínez-Estudillo, C. Hervás-Martínez, P. A. Gutiérrez, and A. C. Martínez-Estudillo, "Evolutionary product-unit neural networks classifiers," *Neurocomputing*, vol. 72, no. 1-2, pp. 548–561, 2008.

[22] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[23] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.

[24] C. M. Bishop, *Neural networks for pattern recognition.* Oxford, UK: Oxford University Press, 1996.

[25] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Stuttgart: Frommann-Holzboog, 1973.

[26] A. C. Martínez-Estudillo, F. J. Martínez-Estudillo, C. Hervás-Martínez, and N. García, "Evolutionary product unit based neural networks for regression," *Neural Networks*, vol. 19, no. 4, pp. 477–486, 2006.

[27] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[28] J. M. Peña-Barragán, F. López-Granados, M. Jurado-Expósito, and L. García-Torres, "Mapping *Ridolfia segetum* moris patches in sunflower (Helianthus annuus L.) crop using remote sensing," *Weed Research*, vol. 47, no. 2, pp. 164–172, 2007.

[29] R. M. McCoy, *Field Methods in Remote Sensing.* The Guilford Press, 2004.

[30] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1-2, pp. 161–205, 2005.

[31] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervas, "JCLEC: a Java framework for evolutionary computation," *Soft Computing*, vol. 12, no. 4, pp. 381–392, 2008.

[32] J. Alcala-Fdez, L. Sánchez, S. García, M. J. del Jesús, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: A software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2008.

[33] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., ser. Data Management Systems. Morgan Kaufmann (Elsevier), 2005.

[34] R. M. Mickey, O. J. Dunn, and V. A. Clark, *Applied Statistics: Analysis of Variance and Regression, 3rd Edition*, 3rd ed. Willey, 2004.

[35] R. G. Miller, *Simultaneous Statistical Inference*, 2nd ed. New York, USA: Wiley, 1981.

[36] G. W. Snedecor and W. G. Cochran, *Statistical Methods*, 6th ed. Iowa State University Press, 1989.

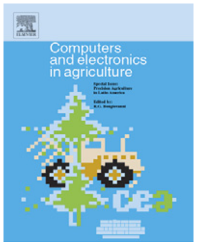
3.1.2. Regresión Logística mediante Redes Neuronales de Unidad Producto para la Realización de Mapas de Infestación de *Ridolfia segetum* en Campos de Girasol utilizando Datos Multi-Temporales obtenidos mediante Sensores Remotos - *Logistic regression product-unit neural networks for mapping Ridolfia segetum infestations in sunflower crop using multitemporal remote sensed data*

- P.A. Gutiérrez, F. López-Granados, J.M. Peña-Barragán, M. Jurado-Expósito, C. Hervás-Martínez. Logistic regression product-unit neural networks for mapping *Ridolfia segetum* infestations in sunflower crop using multitemporal remote sensed data. Computers and Electronics in Agriculture 64:2 (2008) 293-306.
    - Estado: Publicado.
    - Índice de Impacto (JCR 2007): 1.242.
    - Área de conocimiento: *Agriculture, Multidisciplinary*. Ranking 7/35. Primer Cuartil.
    - Área de conocimiento: *Computer Science, Interdisciplinary Applications*. Ranking 30/92. Segundo Cuartil.

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/compag



Logistic regression product-unit neural networks for mapping *Ridolfia segetum* infestations in sunflower crop using multitemporal remote sensed data

P.A. Gutiérrez[a,*], F. López-Granados[b], J.M. Peña-Barragán[b], M. Jurado-Expósito[b], C. Hervás-Martínez[a]

[a] *Department of Computer Science and Numerical Analysis, University of Cordoba, Campus de Rabanales, 14071 Cordoba, Spain*
[b] *Institute for Sustainable Agriculture, CSIC, Apdo. 4084, 14080 Cordoba, Spain*

ARTICLE INFO

Article history:
Received 25 March 2008
Received in revised form
23 June 2008
Accepted 30 June 2008

Keywords:
Artificial neural networks
Evolutionary algorithms
Multi-spectral imagery
Precision agriculture
Remote sensing
Weed patch classification

ABSTRACT

Remote sensing (RS), geographic information systems (GIS), and global positioning systems (GPS) may provide the technologies needed for farmers to maximize the economic and environmental benefits of precision farming. Site-specific weed management (SSWM) is able to minimize the impact of herbicide on environmental quality and arises the necessity of more precise approaches for weed patches determination. *Ridolfia segetum* is one of the most dominant, competitive and persistent weed in sunflower crops in southern Spain. In this paper, we used aerial imagery taken in mid-May, mid-June and mid-July according to different phenological stages of *R. segetum* and sunflower to evaluate the potential of evolutionary product-unit neural networks (EPUNNs), logistic regression (LR) and two different combinations of both (logistic regression using product units (LRPU) and logistic regression using initial covariates and product units (LRIPU)) for discriminating *R. segetum* patches and mapping *R. segetum* probabilities in sunflower crops on two naturally infested fields. Afterwards, we compared the performance of these methods in every date to two recent classification models (support vector machines (SVM) and logistic model trees (LMT)). The results obtained present the models proposed as powerful tools for weed discrimination, the best performing model (LRIPU) obtaining generalization accuracies of 99.2% and 98.7% in mid-June. Our results suggest that a strategy to implement SSWM is feasible with minima omission and commission errors, and therefore, with a very low probability of not detecting *R. segetum* patches. The paper proposes the application of a new methodology that, to the best of our knowledge, has not been previously applied in RS, and which obtains better accuracy than more traditional RS classification techniques, such as vegetation indices or spectral angle mapper.

1. Introduction

Remote sensing, geographic information systems (GIS), and global positioning systems (GPS) may provide the technologies needed for farmers to maximize the economic and environmental benefits of precision farming (Seelan et al., 2003). Patchy distribution of grass (monocotyledonous) and broadleaf (dicotyledonous) weeds in fields is well documented

* *Corresponding author at*: Department of Computer Science and Numerical Analysis, University of Cordoba, Campus de Rabanales, Albert Einstein Building 3rd Floor, 14071 Cordoba, Spain. Tel.: +34 957 218 349; fax: +34 957 218 630.
E-mail address: i02gupep@uco.es (P.A. Gutiérrez).

0168-1699/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.compag.2008.06.001

(Barroso et al., 2004a; Jurado-Expósito et al., 2003, 2005a). However, herbicides are not applied to the infested zones, but they are usually broadcast over entire fields. The potential for over application and the corresponding economical and environmental problems is evident. One aspect of overcoming the possibility of minimizing the impact of herbicide on environmental quality is the development of more efficient approaches for crop production determination and for site-specific weed management (SSWM). Srinivasan (2006) reported an overall overview of the current status of site-specific weed, nutrient, crop diseases and water management. Timmermann et al. (2003) evaluated the economical and ecological benefits of SSWM in annual crops in a 4 years study, concluding that costs savings were 90% and 60% for broadleaf and grass weeds herbicides, respectively.

A key component of SSWM is that accurate and appropriate weed maps are required to take full advantage of site-specific herbicide applications. Mapping weed patches based on ground surveys techniques on field scale is time consuming, expensive and unapproachable in field areas with difficult access. The importance of remote sensing in site-specific agriculture has been widely reviewed (Plant, 2001). Remote sensing of weed canopies may be more efficient and suitable than field surveys, and the majority of studies on discriminating weeds in cultivated systems have involved discrete broadband remote sensing (multispectral sensors) (Brown and Noble, 2005; Thorp and Tian, 2004). In this sense, the analysis of the accuracy of weed maps is essential to make possible the implementation of models with a high performance, leading to a reduction of the overall herbicide application. Detection of late-season weed infestation has demonstrated to have tremendous possibilities when spectral differences between crops and weeds prevail (Koger et al., 2003; López-Granados et al., 2006).

Sunflower (*Helianthus annuus* L.) is one of the most abundant crops in Andalusia, Southern Spain, with more than 320,000 ha sown annually (MAPA, 2007). Sunflower sowing and harvesting times are February–March and July–August, respectively, being mainly grown under dry land conditions. *Ridolfia segetum* Moris (corn caraway) is an annual, umbelliferous weed very frequently found and abundant in clay soils in Andalusia. Its life cycle coincides with that of the sunflower, which enhances the competitive ability and results in an average crop yield reduction of about 32% when infestation is two *R. segetum* plants per m^2 (Carranza-Cañadas et al., 1995). This weed is not controlled by pre-emergence and pre-plant incorporated herbicides, as it is in sunflower, and consequently post-emergence strategies such as tillage or hand weeding are commonly used. Otherwise weed obstructs the harvester since it still has partly green stem during the sunflower harvesting. This is a serious inconvenience if the harvester is equipped with a yield monitor, as it is frequent in precision agriculture management.

Logistic regression (LR) has become a widely used and accepted method of analysis of binary or multiclass outcome variables as it is more flexible and it can predict the probability for the state of a dichotomous variable (e.g., red-attack/non-attack for mountain pine beetle red-attack damage (Wulder et al., 2006)) based on the predictor variables (e.g., vegetation indices, slope, solar radiation, etc.) and has been widely applied in forestry to estimate tree and stand survival under competition (Monserud and Sterba, 1999; Vanclay, 1995). In the field of remote sensing, LR has been used with different aims, for example, for land-cover change detection (Fraser et al., 2005), and for mapping insect tree defoliation (Magnussen et al., 2004). LR and multiple linear regression techniques have also been used for identifying spectrally sensitive regions in order to detect nitrogen deficiency in corn (Cetin et al., 2005).

Approaches based on artificial neural networks (ANNs) have been considered in agricultural remote sensing (Goel et al., 2003; Gutiérrez et al., 2008; Uno et al., 2005; Yang et al., 2000a) to associate complicated spectral information with target attributes without any constraints for sample distribution (Mather, 2000). ANNs have long been applied for the land-cover classification in remotely sensed images (Atkinson and Tatnall, 1997; Kanellopoulos and Wilkinson, 1997; Zhai et al., 2006). In ANNs, the hidden neurons are the functional units and can be considered as generators of function spaces. Most existing neuron models are based on the summing operation of the inputs, and, more particularly, on sigmoidal unit functions, resulting in what is known as the multilayer perceptron (MLP).

Product-unit neural network (PUNN) models are an alternative to MLPs and are based on multiplicative neurons instead of additive ones. They correspond to a special class of feed-forward neural network introduced by Durbin and Rumelhart (1989). While MLP network models have been very successful, networks that make use of product units (PUs) have the added advantage of increased information capacity (Durbin and Rumelhart, 1989). That is, smaller PUNNs architectures can be used than those used with MLPs (Ismail and Engelbrecht, 2002). They aim to overcome the nonlinear effects of variables by means of nonlinear basis functions, constructed with the product of the inputs raised to arbitrary powers. These basis functions express the possible strong interactions between the variables, where the exponents may even take on real values and are suitable for automatic adjustment.

Classical neural networks training algorithms assume a fixed architecture difficult to establish beforehand. Evolutionary algorithms (EAs), which are stochastic search algorithms that execute a global search in the input space preventing the fall to local optimum (Angeline et al., 1994), have demonstrated great accuracy in designing a near optimal architecture. This fact, together with the complexity of the error surface associated with a PUNN, justifies the use of an EA to design the structure and to adjust the weights of these models (Martínez-Estudillo et al., 2006). Moreover, genetic algorithms (GAs) (the most widely applied EAs) have been independently used in several remote sensing applications (Mertens et al., 2003; Zhan et al., 2003).

Evolutionary artificial neural networks (EANNs) (Yao, 1999) are the combination of ANNs and EAs. Combining these two methods, each technique complements the disadvantages of the other (Yao, 1999). For example, a contribution by ANNs is the flexibility of nonlinear function approximation, which cannot be easily implemented with prototype EAs. Furthermore, neural networks play a significant role in remote sensing since they can handle massive, complex and incomplete data sets efficiently. As such, they are candidates to produce better results than some of the traditional classifiers, requiring in some cases less training data (Moody et al., 1996).

On the other hand, EAs have freed ANNs from simple gradient descent approaches of optimization. In fact, traditional ANNs based on backpropagation algorithms have some limitations. At first, the architecture of the artificial neural networks is fixed and a designer needs much knowledge to determine it. Besides, the error function of the learning algorithm must have a derivative. Finally, the algorithm frequently gets stuck in local optima because it is based on gradient-based search without stochastic property. The combination of EAs and ANNs can overcome these shortcomings and is particularly useful when the activation function of the neurons is non-differentiable and traditional gradient-based training algorithms cannot be used. Because an EA can treat no-differentiable and multimodal spaces, which are the typical case in the classification of remotely sensed imagery, there must be a great motivation to apply EANN to classification of remotely sensed imagery.

A very recent approach (Hervás-Martínez and Martínez-Estudillo, 2007) is based on the hybridization of the LR model and evolutionary PUNNs (EPUNNs), in order to obtain binary classifiers. In a first step, an EA is used to determine the basic structure of the product-unit model. That step can be seen as a global search in the space of the model coefficients. Once the basis functions have been determined by the EA, a transformation of the input space is considered. This transformation is performed by adding the nonlinear transformations of the input variables given by the PU basis functions obtained by the EA. The final model is linear in these new variables together with the initial covariates. This hybrid model outperforms the linear part and the nonlinear part obtaining a good compromise between them and performing well compared to several other learning classification techniques. This methodology has been extended to multiclass problems (Hervás-Martínez et al., 2008).

The WEKA machine learning workbench provides a general-purpose environment for automatic classification, regression, clustering and feature selection. It contains an extensive collection of machine learning algorithms and data pre-processing methods complemented by graphical user interfaces for data exploration and the experimental comparison of different machine learning techniques on the same problem. For comparison purposes, we consider in this work two of most widely accepted WEKA methods in machine learning community: the support vector machines (SVMs) and the logistic model trees (LMTs). SVMs are a recent classification method based on the statistical learning theory of Vapnik (1995) and they have been successfully applied to very large highly nonlinear problems such as character recognition. LMT is an algorithm that combines a decision tree structure with LR models, using LR at the leaves of the tree and resulting in a single tree. LMTs have been shown to be very accurate and compact classifiers (Landwehr et al., 2005).

One of the objectives of this work was to generate a weed probability map by using logistic models approaches, obtaining in this way probability maps, capable of indicating a range of weed presence likelihood, rather than a binary indication of weed or weed-free. On the other hand, we aimed to assess the potential of different classification models based on PUNNs and LR for reaching the best approach to map *R. segetum* patches in sunflower crop at a field scale, using remote sensed information and assessing the best date for discriminating this weed. Four were the models tested: (a) the LR statistical model, (b) the EPUNN model, and two combinations of both, (c) LR using product units (LRPU), i.e., LR with PU basis functions obtained from EPUNNs, and (d) LR using initial covariates and product units (LRIPU), i.e., LRPU extending the model with the initial covariates of the problem. Moreover these models are compared with the previously described machine learning classifiers: (e) SVM, and (f) LMT.

2. Materials and methods

2.1. Study sites, materials and experimental design

The study was conducted at two fields in Andalusia, southern Spain: at Matabueyes, 42 ha (coordinates 37°8′N, 4°8′W, WGS84), and at Santa Cruz, 28 ha (coordinates 37°8′N, 4°6′W, WGS84) in 2003 and 2004, respectively. Both fields were naturally infested by *R. segetum*. Soil at both locations was classified as Typic Chromoxerert (USDA-NRCS, 1998), with approximately 60% clay. Sunflower cv. Jalisco was seeded in mid-March at 4 kg ha^{-1} in rows 0.7 m apart and harvested in mid-August using the farm's MF34 combine equipped with a calibrated Fieldstar® yield monitor and a differentially corrected global positioning system (DGPS) receiver (Massey Fergusson®, AGCO Corporation, Duluth, GA, USA). The field sites were managed using shallow tillage production methods. Glyphosate (Roundup, isopropylamine salt, 360 g a.i. L^{-1}, Montsanto) was applied pre-emergence at 180 g a.i. L^{-1} for the control of annual weed seedlings in sunflower. At this rate, this herbicide has no effect on *R. segetum* emergence or development.

The sunflower phenological stages considered were adapted to the study conditions from Peña-Barragán et al. (2006), the vegetative phase (from the emergence stage to the early reproductive stage) in mid-May, the flowering phase (from the head flowering stage to the initial desiccation stage of lower leaves) in mid-June, and the senescent phase (from the stage in which the reproductive head is partly desiccated and browning to the stage in which the plant becomes completely desiccated and darkish/black) in mid-July. The *R. segetum* phenological stages were also based on previous studies as follows: in mid-May, the vegetative phase (weed growth from seedling to the vegetative stage without the floral stem and the inflorescences (or umbellas) still closed), in mid-June, the flowering phase (inflorescences are yellowing and the plant is at its largest size), and in mid-July, the senescent phase (weed desiccates and turns brown).

Conventional-colour (CC) and colour-infrared (CIR) aerial imagery of the studied field were taken in mid-May, mid-June and mid-July (except CIR images in mid-June at Matabueyes, due to technical problems).

The photographs were taken by a turboprop twin-engine plane CESSNA 402. The photographs were taken on cloudless days between 12 and 14 h standard time and the average flight height was 1525 m to obtain photographs at a scale 1:10,000. An automatic pilot was used for managing both photographic equipment and GPS and the camera was a RMK TOP 15, with a Zeiss objective, and a focal distance of 153.76 mm. Kodak Aerocolor III 2444 and Kodak Aerochrome S0734 film was used for CC and CIR photographs, respectively. Then, the photographs were digitalized with an AGFA Horizon A3 scanner, considering a resolution of 635 dots per inch (dpi), brightness and contrast not being adjusted on the digitalized images. The next step was to orthorectify the digitised images, using the fiducial marks of the aerial calibration certificate, 40 ground control points taken with a differential GPS TRIMBLE PRO-XRS equipped with a TDC-1 unit (centimetre precision) and a 10-m resolution raster DEM. Finally, images were resampled to a pixel size representing 40 cm × 40 cm ground area.

Input variables included the digital values of all bands in each available image, that is: CC images responded to blue (B, 400–500 nm), green (G, 500–600 nm), and red (R, 600–700 nm) broad bands of the electromagnetic spectrum, and CIR images to G, R and near-infrared (NIR, 700–900 nm) bands. The scanner produced a RGB digital image with 8-bit true colour, so pixels of the image showed digital counts within the range of 0–255 values. These digital values are considered as being directly proportional to the total light reflected from the scene (Flowers et al., 2001). All spatial and spectral data from images were grouped and saved to a unique multiband file taking into account two previous requirements: (1) the georeference error between images was less than one pixel, so similar pixels had the same coordinate, and (2) the NIR digital values of CIR images were corrected to the digital values of CC images, using the differences between the G and R bands of both original images.

To train and validate the classification models, a random ground sampling procedure was carried out at the time when aerial images were taken, ensuring that all parts of the field area had an equal chance of being sampled with no operator bias (McCoy, 2005). We georeferenced a total of 1600 pixels in each phenological stage, where 800 pixels corresponded to *R. segetum* class, 400 pixels corresponded to bare soil class and 400 corresponded to sunflower. In this way, the number of weed-free pixels was the same to the number of *R. segetum* pixels. The objective is the differentiation between *R. segetum* and all other pixels, as distinguishing between *soil* and *sunflower* is not needed for site-specific herbicide application.

The experimental design was conducted using a stratified holdout cross-validation procedure, where the size of the training set was approximately $3n/4$ and $n/4$ for the generalization set, n being the size of the full dataset. Consequently, each dataset mentioned above was randomly split in two datasets. A 1120 instances dataset was used for model training and the remaining 480 instances formed the generalization dataset. The supervised classification process is composed of three steps: one is the classifier training which aims at creating a reliable input–output relationship between remotely sensed data and land-cover class membership; then the models obtained are evaluated through the classification of previously unseen data whose land-cover class membership is known, in order to assess their generalization capability; the final step is image classification which applies the relationship established in the training process to the whole image.

2.2. *Methods*

Different methods have been applied for training the classifiers. These include from classical statistical approaches to the hybrid approaches proposed by Hervás-Martínez and Martínez-Estudillo (2007).

2.2.1. *Statistical methods: binary logistic regression*

Typically, in supervised image classification, a set of n_T training samples or pixels $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_{n_T}, \mathbf{y}_{n_T})$ is given. The inputs $\mathbf{x}_i$ (i.e., spectral bands) form a feature space X, and the output $\mathbf{y}_i$ (i.e., the target class) has a class label c, which belongs to a finite set C. A classification rule is designed based on the training data, so that, given a new input $\mathbf{x}_i$ of a pixel, a class c from C with the smallest probability of error is assigned to it.

In this paper the situation considered is the following: a binary outcome variable y (weed presence or weed-free) is observed together with a vector $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik},)$ of covariates for each of the n_T pixels (assuming that the vector of inputs includes the constant term 1 to accommodate the intercept). The two-class is coded via a 0/1 response y_i, where $y_i = 1$ for weed presence and $y_i = 0$ for weed-free pixels. Let p be the conditional probability associated with the first class. Logistic regression (Hosmer and Lemeshow, 1989) is a widely used statistical modeling technique in which the probability p of the dichotomous outcome event is related to a set of explanatory variables $\mathbf{x}$ in the form:

$$\log \mathrm{it}(p) = \ln\left(\frac{p}{1-p}\right) = f_{\mathrm{LR}}(\mathbf{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{x} \tag{1}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_k)$ is the vector of the coefficients of the model and $\boldsymbol{\beta}^{\mathrm{T}}$ the transpose vector and $f_{\mathrm{LR}}(\mathbf{x}, \boldsymbol{\beta})$ is the LR model. We refer to $p/(1-p)$ as odds-ratio and to the expression (1) as the log-odds or logit transformation. A simple calculation in Eq. (1) shows that the probability of occurrence of an event as a function of the covariates is nonlinear and is given by

$$p(\mathbf{x}; \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}}}{1 + e^{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}}} \tag{2}$$

The complementary event probability can therefore be obtained as $(1 - p(\mathbf{x};\boldsymbol{\beta}))$. Once the conditional probability function defined in (2) is known, the Bayesian (optimal) decision rule can be constructed:

$$r(\mathbf{x}) = \mathrm{sign}\left\{\ln\left(\frac{p(\mathbf{x}; \boldsymbol{\beta})}{1 - p(\mathbf{x}; \boldsymbol{\beta})}\right)\right\} \tag{3}$$

Given any test pixel $\mathbf{x}$, the probability p that the pixel belongs to the first class can be determined from (2). Similar to the maximum-likelihood classification, these class probabilities for each pixel may be outputted to reflect the actual proportion of classes within a pixel, thereby producing a soft, fuzzy or subpixel classification. The results from this paper advocate the utility of the LR as a potential approach for the soft classification similar to other recent approaches such as the MLP neural networks (Foody and Arora, 1996) or the decision tree regression (Xu et al., 2005). A hard classification can be produced by assigning the class having a maximum probability (in our case, as a binary outcome variable is considered, we can simply check if the probability p is greater or lower than the value 0.5). Observe that LR not only constructs a decision rule but it also finds a function that for any input vector defines the probability p that the vector $\mathbf{x}$ belongs to the first class, in our case *R. segetum* presence.

Let $D = \{(\mathbf{x}_l, y_l); 1 \le l \le n_T\}$ be the training data set, where the number of samples is n_T. Here it is assumed that the training sample is a realization of a set of independent and identically distributed random variables. The unknown regression coefficients β_i, which have to be estimated from the data, are directly interpretable as log-odds ratios or, in term of $\exp(\beta_i)$, as odds ratios. That log-likelihood used as the error function is

$$\begin{aligned} l(\boldsymbol{\beta}) &= \sum_{l=1}^{n_T} y_l \log p(\mathbf{x}_l;\boldsymbol{\beta}) + (1-y_l)\log(1-p(\mathbf{x}_l;\boldsymbol{\beta})) \\ &= \sum_{l=1}^{n_T} y_l \boldsymbol{\beta}^T\mathbf{x} - \log(1+e^{\boldsymbol{\beta}^T\mathbf{x}}) \end{aligned} \tag{4}$$

The estimation of the coefficient $\boldsymbol{\beta}$ is usually carried out by means of an iterative procedure like the Newton–Raphson algorithm or the iteratively reweighted least squares (IRLS) (Hastie et al., 2001). Typically the algorithm converges, since the log-likelihood is concave, but overshooting can occur. In the rare cases where log-likelihood decreases, step size halving will guarantee convergence. The conditions under which a global maximum exists and the maximum likelihood estimators which do not diverge are discussed by McLachlan (1992) and references therein.

2.2.2. *Statistical methods: support vector machines and logistic model trees*

SVM is now a very popular tool in machine learning, which explores the kernel techniques. Reasons for this popularity include geometric exploration and an accurate performance in many classification applications. SVM is basically a binary classifier, which finds the maximal margin (hyperplane) between two classes. SVM can classify nonlinearly separable data sets by plotting the data into a high-dimensional feature space using kernels. For further details and recent development of the SVMs, we refer readers to Cristianini and Shawe-Taylor (2000) and Vapnik (1995).

Tree induction methods and linear models are popular techniques for the prediction of nominal classes and numeric values in supervised learning tasks. For predicting numeric quantities, some work has been carried out on combining these two schemes into ‘model trees’, i.e., trees that contain linear regression functions at the leaves. Model trees, like ordinary regression trees, predict a numeric value for an instance that is defined over a fixed set of numeric or nominal attributes. Unlike ordinary regression trees, model trees construct a piecewise linear (instead of a piecewise constant) approximation to the target function. The final model tree consists of a tree with linear regression functions at the leaves, and the prediction for an instance is obtained by sorting it down to a leaf and using the prediction of the linear model associated with that leaf.

LMT is an algorithm that adapts this idea for classification problems, using LR instead of linear regression. In fact, LMT can be regarded as a new scheme for selecting the attributes to be included in the logistic regression models, and introduces a way of building the logistic models at the leaves by refining logistic models that have been trained at higher levels in the tree, i.e. on larger subsets of the training data. This algorithm has been showed as producing very accurate and compact classifiers (Landwehr et al., 2005).

2.2.3. *Logistic regression using covariates obtained by product-unit neural network models*

The models we are testing are LR models based on the hybridization of the standard linear model and nonlinear terms constructed with basis functions obtained from evolutionary product-unit neural networks.

2.2.3.1. *Product-unit neural networks.* PUNNs are an alternative to MLPs, and are based on multiplicative neurons instead of additive ones. A multiplicative neuron is given by $\prod_{i=1}^{k} x_i^{w_{ji}}$, where k is the number of the inputs. When the exponents are {0,1} a higher order unit is obtained, namely the sigma–pi unit (Lenze, 1994). In contrast to the sigma–pi unit, in the product-unit the exponents are not fixed and may even take real values.

Product-unit based neural networks have several advantages, including increased information capacity and the ability to express strong interactions between input variables. Furthermore, it is possible to obtain upper bounds of the Vapnik–Chervonenkis (VC) dimension (Vapnik, 1995) of product-unit neural networks similar to those obtained for MLP (Schmitt, 2002).

Despite these advantages, PUNNs have a major handicap: they have more local minima and more probability of becoming trapped in them (Ismail and Engelbrecht, 2002). The main reason for this difficulty is that small changes in the exponents can cause large changes in the total error surface and therefore their training is more difficult than the training of standard MLPs. Several efforts have been made to carry out learning methods for product-units (Ismail and Engelbrecht, 2002; Janson and Frenzel, 1993). The back propagation algorithm, which is the most common algorithm for training multilayer neural networks, does not work very well with the product-units because of its complex error surface.

The structure of the neural network considered is described in Fig. 1: an input layer with k neurons, a neuron for every input variable, a hidden layer with m neurons and an output layer with one neuron.

There are no connections between the neurons of a layer and none between the input and output layers either. The activation function of the j-th neuron in the hidden layer is given by $\prod_j(\mathbf{x}, \mathbf{w}_j) = \prod_{i=1}^{k} x_i^{w_{ji}}$, where w_{ji} is the weight of the connection between input neuron i and hidden neuron j and $\mathbf{w}_j = (w_{j1}, \ldots, w_{jk})$ is the weight vector. The activation function

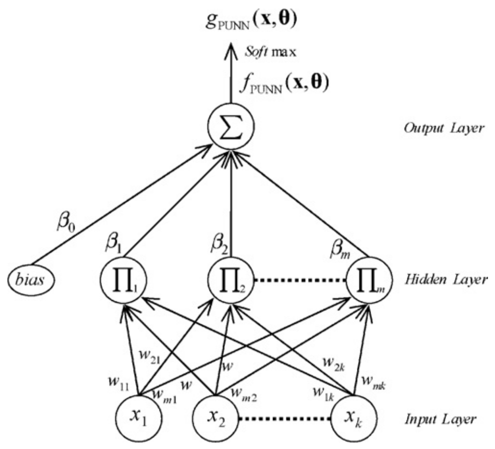


Fig. 1 – Model of a product-unit neural network.

of the output neuron is given by

$$f_{\text{PUNN}}(\mathbf{x},\boldsymbol{\theta}) = \beta_0 + \sum_{j=1}^{m} \beta_j \prod\nolimits_j (\mathbf{x}, \mathbf{w}_j) \tag{5}$$

where β_j is the weight of the connection between the hidden neuron j and the output neuron. The transfer function of all hidden and output neurons is the identity function.

We consider the softmax activation function (Bishop, 1995) given by

$$g_{\text{PUNN}}(\mathbf{x},\boldsymbol{\theta}) = \frac{\exp\left(f_{\text{PUNN}}(\mathbf{x},\boldsymbol{\theta})\right)}{1+\exp(f_{\text{PUNN}}(\mathbf{x},\boldsymbol{\theta}))} \tag{6}$$

where $f_{\text{PUNN}}(\mathbf{x},\ \boldsymbol{\theta})$ is the output of the output neuron for pattern $\mathbf{x}$ and $g_{\text{PUNN}}(\mathbf{x},\ \boldsymbol{\theta})$ is the probability that a pattern $\mathbf{x}$ has of belonging to the "weed presence" class. With this model, the cross-entropy error function is defined by the same expression than in (4), substituting $\boldsymbol{\beta}^{\text{T}}\mathbf{x}$ with $f_{\text{PUNN}}(\mathbf{x},\ \boldsymbol{\theta})$.

2.2.3.2. *Evolutionary product-unit neural networks.* In order to estimate the parameters and the structure of the PUNNs that minimizes the classification error function, an Evolutionary algorithm has been considered. The algorithm is similar to the one proposed by Martínez-Estudillo et al. (in press). The population-based evolutionary algorithm for architectural design and the estimation of real-coefficients have points in common with other evolutionary algorithms in the bibliography (Angeline et al., 1994; García-Pedrajas et al., 2002; Yao and Liu, 1997). The search begins with an initial population. This population is updated in each generation using a population-update algorithm, and is subject to the evolutionary operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving artificial networks (Angeline et al., 1994). For this reason, this EA belongs to the evolutionary programming (EP) paradigm. The general structure of the EA is detailed next:

```
Evolutionary programming algorithm
  (1) Generate a random population of size $N_P$
  (2) Repeat until the maximum number of generations
    (2a) Apply parametric mutation to the best 10% of individuals. Apply structural mutation to the remaining 90% of individuals
    (2b) Calculate the fitness of every individual in the population
    (2c) Add best fitness individual of the last generation (elitist algorithm)
    (2d) Rank the individuals with respect to their fitness
    (2e) Best 10% of population individuals are replicated and substitute the worst 10% of individuals
  (3) Select the best individual of the population in the last generation and return it as the final solution
```

First, the initial population is generated: the algorithm begins with the random generation of a larger number of networks than the number of neurons used during the evolutionary process. $10N_P$ networks are generated, where N_P is the number of individuals of the population to be trained during the evolutionary process. We consider $l(\boldsymbol{\theta})$ as the error function of an individual $f_{\text{PUNN}}(\mathbf{x},\ \boldsymbol{\theta})$ of the population, g being a PUNN; and then, the fitness measure is a decreasing strictly transformation of the error function $l(\boldsymbol{\theta})$ given by $A(g) = 1/(1 + l(\boldsymbol{\theta}))$, where $0 < A(g) \leq 1$.

The adjustment of both weights and structure of the PUNNs is performed by the complementary action of two mutation operators: parametric and structural mutation. Parametric mutation implies a modification in the coefficients (β_j) and the exponents (w_{ji}) of the model, using a self-adaptive simulated annealing algorithm (Kirkpatrick et al., 1983). Structural mutation modifies the topology of the neural nets, helping the algorithm to avoid local minima and increasing the diversity of the trained individuals. Five structural mutations are applied sequentially to each network: neuron deletion, connection deletion, neuron addition, connection addition and neuron fusion. In order to define the topology of the neural networks generated in the evolution process, three parameters are considered: m, M_E and M_I. They correspond to the minimum and the maximum number of hidden neurons in the whole evolutionary process and the maximum number of hidden neurons in the initialization process respectively. In order to obtain an initial population formed by models simpler than the most complex models possible, parameters must fulfil the condition $m \leq M_I \leq M_E$.

More details about the EA can be found in Martínez-Estudillo et al. (2006, in press).

2.2.3.3. *Logistic regression using product units.* Logistic regression using product units is a hybrid method that considers the EA presented in the previous section in order to obtain an EPUNN structure and hidden neuron weights accurate enough. When these are obtained, it applies the IRLS mechanism over the product unit basis functions of the EPUNN selected. So the LRPU composed only of PU basis function is given by

$$f_{\mathrm{LRPU}}(\mathbf{x}, \boldsymbol{\theta}) = \alpha_0 + \sum_{j=1}^{m} \beta_j \prod_{i=1}^{k} x_i^{w_{ji}} \tag{7}$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \mathbf{W})$, $\boldsymbol{\alpha} = (\alpha_0, \beta_1, \ldots, \beta_m)$ and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$, with $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jk})$, $w_{ji} \in R$. The coefficients $\mathbf{W}$ are given by the EA, they not being adjusted by the IRLS method. The IRLS method only optimizes the linear part of the model, i.e., the $\boldsymbol{\alpha}$ coefficients.

2.2.3.4. Logistic regression using initial covariates and product units. The LRIPU model used is a hybridization of the LR model and the EPUNNs previously presented. The model extends LRPU, considering the initial covariates $\mathbf{x}$ of the problem. Its expression is given by

$$f_{\mathrm{LRIPU}}(\mathbf{x}, \boldsymbol{\theta}) = \alpha_0 + \sum_{i=1}^{k} \alpha_i x_i + \sum_{j=1}^{m} \beta_j \prod_{i=1}^{k} x_i^{w_{ji}} \tag{8}$$

where $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \mathbf{W})$, $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_m)$ and $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$, with $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jk})$, $w_{ji} \in R$. The values adjusted with IRLS correspond to the $\boldsymbol{\alpha}$ vector, the coefficients $\mathbf{W}$ being given by the EA.

Finally, in order to reduce the high amount of variables of these LRIPU models, a backward stepwise procedure is used. The method starts with the full model with all the covariates, initial and PU, pruning variables to the model sequentially and successively, until no further pruning can be made to improve the fit. At each step, the least significant covariate is selected in the discriminant function, i.e., the one which shows the greatest critical value (p-value) in the hypothesis test, where the associated coefficient equal to zero is the hypothesis to be contrasted. The selected covariate is deleted if this does not reduce the fit. If none of the covariates is deleted, the second least significant covariate is considered following the procedure previously described. The procedure ends when all the tests for each covariate provide p-values smaller than the fixed significance level, or when none of the two chosen covariates is deleted.

2.2.4. Model development and evaluation

To start processing data, each of the input variables was scaled in the rank [0.1, 0.9]. These bounds in the PUNN models were chosen to avoid inputs values close to 0 that can result in very large values of the function for negative exponents and to avoid dramatic changes in the outputs of the network when there are weights with large values (especially in the exponents). The new scaled variables were named R^*, G^*, B^* and NIR*. For example, R^* is calculated as follows:

$$R^* = \frac{R - R_{\min}}{R_{\max} - R_{\min}} + 0.1 \tag{9}$$

where R is the red digital value and $R_{\min}$ and $R_{\max}$ are the minimum and maximum values in the whole dataset, respectively.

All the models were evaluated using their prediction error, defined by the correctly classified rate (CCR), over the training ($\mathrm{CCR_T}$) and the generalization ($\mathrm{CCR_G}$) datasets, which are given by the following expression:

$$\mathrm{CCR_D} = \frac{1}{N} \sum_{n=1}^{n_{\mathrm{T}}} I(C(\mathbf{x}_n) = y_n) \tag{10}$$

where $I(\cdot)$ is the zero-one loss function, n_{T} the number of pattern of the evaluated dataset D, y the binary outcome variable (weed presence or weed-free) and $\mathbf{x}_i$ is the vector of covariates for each i-th pixel. A good classifier tries to achieve the highest possible CCR in a given problem.

Furthermore, in order to obtain a more complete evaluation of the classification task performed by the different models, the confusion matrix associated with each dataset and model was derived. A confusion matrix is a visualization tool typically used in supervised learning (Provost and Kohavi, 1998). The output is compared to the observed outcome (event or non-event, i.e., weed or weed-free pixel), and assigned one of the four possible situations: (i) true negative (TN) when negative cases are correctly identified by the classifier; (ii) false positive (FP) when the classifier incorrectly identifies a non-event case as an event; (iii) false negative (FN) when the classifier incorrectly identifies an event case as a non-event; and (iv) true positive (TP) when the positives cases are correctly identified. For a given number of cases (n_{T}) in a dataset D, these indexes are inserted into a 2 × 2 confusion or contingency matrix ($M(g)$) as

$$M(g) = \begin{pmatrix} n_{\mathrm{TN}} & n_{\mathrm{FP}} \\ n_{\mathrm{FN}} & n_{\mathrm{TP}} \end{pmatrix} \tag{11}$$

where $n_{\mathrm{T}} = n_{\mathrm{TN}} + n_{\mathrm{FP}} + n_{\mathrm{FN}} + n_{\mathrm{TP}}$. The diagonal corresponds to the correctly classified patterns and the off-diagonal to the mistakes in the classification task. In this way, the $\mathrm{CCR_D}$ can also be expressed as $\mathrm{CCR_D} = (n_{\mathrm{TP}} + n_{\mathrm{TN}})/n_{\mathrm{T}}$. Two other additional measures are commonly derived from a confusion matrix for evaluating the classifier: the FP rate (or Type I error) and the FN rate (or Type II error). The FP rate is the proportion of negative instances that were erroneously reported as being positive and the false negative rate is the proportion of positive instances that were erroneously reported as negative. Their expressions are the following:

$$\mathrm{FP} = \frac{n_{\mathrm{FP}}}{n_{\mathrm{TN}} + n_{\mathrm{FP}}}, \qquad \mathrm{FN} = \frac{n_{\mathrm{FN}}}{n_{\mathrm{FN}} + n_{\mathrm{TP}}} \tag{12}$$

In this way, the FP and FN rates can also be expressed as $\mathrm{FP} = 1 - \mathrm{CCR_N}$ and $\mathrm{FN} = 1 - \mathrm{CCR_P}$, where $\mathrm{CCR_N}$ and $\mathrm{CCR_P}$ are the CCR values obtained when considering only negative and positive cases of dataset D, respectively, that is:

$$\mathrm{CCR_N} = \frac{n_{\mathrm{TN}}}{n_{\mathrm{TN}} + n_{\mathrm{FP}}}, \qquad \mathrm{CCR_P} = \frac{n_{\mathrm{TP}}}{n_{\mathrm{FN}} + n_{\mathrm{TP}}} \tag{13}$$

SPSS 13.0 software for Windows (SPSS 13.0, SPSS Inc. Chicago, IL) was used for applying the IRLS algorithm in LR, LRPU and LRIPU methodologies. The different EPUNN experiments were conducted using a software package developed in JAVA by the authors, as an extension of the JCLEC framework (http://jclec.sourceforge.net/) (Ventura et al., 2008). This software package is available in the non-commercial JAVA tool

Table 1 – Non-common parameters values of the evolutionary product-unit neural network (EPUNN) algorithm for each dataset, including number of generations, and topology defining parameters

Location	Date	#Gen.	*m*	M_I	M_E
Matabueyes	(mid-May)	450	2	3	4
	(mid-June)	600	2	2	3
	(mid-July)	550	3	3	4
Santa Cruz	(mid-May)	400	2	2	3
	(mid-June)	300	2	2	3
	(mid-July)	400	2	2	3

#Gen., maximum number of generations; *m*, minimum number of hidden neurons; M_I, maximum number of hidden neurons during the initialization process; M_E, maximum number of hidden neurons.

named KEEL (http://www.keel.es) (Alcala-Fdez et al., in press). Weights are assigned using a uniform distribution defined throughout two intervals, [−5, 5] for connections between the input layer and hidden layer and, for all kinds of neurons, [−10, 10] for connections between the hidden layer and the output layer. The values of non-common parameters adjusted through a trial-and-error process are shown in Table 1.

The sequential minimal optimization (SMO) algorithm is a java implementation of the SVM methodology which was used in our experiments. This algorithm, together with an implementation of LMT, are available as part of the WEKA machine learning workbench (Witten and Frank, 2005). We considered WEKA release 3.4.0 and both methods were applied to the datasets evaluated in this paper, using their default parameter values.

3. Results

The models obtained with the different evolutionary and statistical methodologies were evaluated using their prediction error over the training and generalization sets. First of all, the EPUNN methodology was run and the corresponding statistical results are shown in Table 2. As EPUNN is an evolutionary methodology, it is based on randomly generated numbers and this makes it a non-deterministic method. For this reason, the method was run 30 times and the best individual of the final population in each execution was extracted. Table 2 includes the average, the standard deviation, the best and the worst values of the CCR over the training (CCR_T) and the generalization (CCR_G) sets of these 30 models, together with their number of connections. From the analysis of these results, we can conclude that the EPUNN methodology was quite stable in training and generalization accuracy terms, the standard deviation not being very high in any date or location ($S.D._T < 1.86\%$ and $S.D._G < 2.43\%$). The most stable generalization results were obtained in mid-June at Matabueyes ($S.D._G = 0.57\%$) and Santa Cruz ($S.D._G = 0.18\%$), together with the highest accuracy ($mean_G = 97.85\%$ and $mean_G = 98.37\%$). The number of connections of the models obtained was quite low ($mean_{\#Conn.} = 14.5$) compared to the number of connections obtained using other ANNs multi-spectral imagery analysis approaches. For example, Gutiérrez et al. (2008) obtained a PUNN model with 28 connections when predicting sunflower crop from multi-spectral imagery. This fact assures not only more interpretable models but also a better generalization capability.

Once the evolution process was applied, we selected the best training individual of the 30 runs in each location/date and we constructed the corresponding LRPU and LRIPU models, using SPSS statistical software. For comparison purposes, we also applied standard LR, and the confusion matrixes for training and generalization sets corresponding to each of the four models are shown in Table 3. The number of pixels of each target response that are predicted as belonging to a specified response are represented in each location/date, and the CCR is calculated independently for each class (CCR_N and CCR_P), in order to better evaluate the performance of the classifiers. An example of interpretation of the different confusion matrixes is the following: if we consider the generalization set in mid-June at Matabueyes, the LR model correctly classifies 226 ground-truth *R. segetum*-absence ($Y = 0$) pixels and misclassifies the remaining 14 ground-truth *R. segetum*-absence pixels assigning to them the *R. segetum*-presence label ($Y = 1$), and obtaining an independent $CCR_N = 94.2\%$ accuracy in the *R. segetum*-absence class. At the same location/date, the LR model correctly classifies 228 ground-truth *R. segetum*-presence pixels ($Y = 1$) and misclassifies 12 ground-truth *R. segetum*-presence pixels assigning to them the *R. segetum*-absence label ($Y = 0$), and obtaining an independent $CCR_P = 95.0\%$ accuracy in the *R. segetum*-presence class. The LR model finally results in a global $CCR_G = 94.6\%$.

Table 2 – Accuracy and number of connection statistical results over 30 runs of the evolutionary product-unit neural network (EPUNN) methodology

Location	Date	CCR_T (%)				CCR_G (%)				#Conn.	
		Mean	S.D.	Best	Worst	Mean	S.D.	Best	Worst	Mean	S.D.
Matabueyes	(mid-May)	70.81	0.91	72.41	69.20	69.86	0.70	71.67	68.33	13.77	1.50
	(mid-June)	97.67	0.36	98.21	96.96	97.85	0.57	98.75	96.67	9.23	0.97
	(mid-July)	77.96	1.71	80.18	74.46	77.28	2.43	80.83	70.83	14.50	1.36
Santa Cruz	(mid-May)	73.96	1.86	77.48	70.74	75.51	1.37	78.43	72.81	10.63	1.30
	(mid-June)	99.15	0.11	99.33	98.94	98.37	0.18	98.88	97.98	11.33	0.96
	(mid-July)	83.96	0.47	84.79	83.06	83.18	0.54	84.04	82.02	9.67	1.03

CCR_T, correctly classified rate in the training set; CCR_G, correctly classified rate in the generalization set; #Conn., number of connections of the models; S.D., standard deviation.

Table 3 – Confussion matrix obtained by the different models at each location/date, global accuracy and independent accuracy for each class (*Y* = 0, *R. segetum* absence, and Y = 1, *R. segetum* presence)

Location	Phen. stage (date)	Target response	Training			Generalization		
			Predicted response		CCR (%)	Predicted response		CCR (%)
			Y = 0	Y = 1		Y = 0	Y = 1	
Matabueyes	Vegetative (mid-May)	Y = 0	384 *352* (383)[394]	176 *208* (177)[166]	68.5 *62.9* (68.4)[70.4]	164 *148* (164)[168]	76 *92* (76) [72]	68.3 *61.7* (68.3) [70.0]
		Y = 1	133 *171* (136)[141]	427 *389* (424)[419]	76.2 *69.5* (75.7)[74.8]	65 *69* (67) [69]	175 *171* (173)[171]	72.9 *71.3* (72.1)[71.3]
		CCR (%)			72.4 *66.2* (72.1)[72.6]			**70.6** *66.5* (70.2)[**70.6**]
	Flowering (mid-June)	Y = 0	547 *529* (547)[552]	13 *31* (13) [8]	97.7 *94.5* (97.7)[98.6]	236 *226* (237)[238]	4 *14* (3)[2]	98.3 *94.2* (98.8)[99.2]
		Y = 1	7 *30* (9) [11]	553 *530* (551)[549]	98.8 *94.6* (98.4) [98.0]	2 *12* (2) [2]	238 *228* (238)[238]	99.2 *95.0* (99.2)[99.2]
		CCR (%)			98.2 *94.6* (98.0)[98.3]			98.7 *94.6* (99.0)[**99.2**]
	Senescence (mid-July)	Y = 0	443 *296* (443)[447]	117 *264* (117)[113]	79.1 *52.9* (79.1)[79.8]	195 *138* (425)[189]	45 *102* (55)[51]	81.2 *57.5* (88.5)[78.8]
		Y = 1	105 *131* (111)[117]	455 *429* (449)[443]	81.2 *76.6* (80.2)[79.1]	52 *60* (53)[50]	188 *180* (187)[190]	78.3 *75.0* (77.9)[79.2]
		CCR (%)			80.1 *64.7* (79.6)[79.5]			**79.8** *66.3* (79.6) [79.0]
Santa Cruz	Vegetative (mid-May)	Y = 0	362 *353* (362)[361]	158 *167* (158)[159]	69.6 *67.9* (69.6)[69.4]	159 *143* (159)[157]	63 *79* (63)[65]	71.6 *64.4* (71.6)[70.7]
		Y = 1	76 *186* (73)[77]	443 *333* (446)[442]	85.4 *64.2* (85.9)[85.2]	33 *80* (33)[35]	190 *143* (190)[188]	85.2 *64.1* (85.2)[84.3]
		CCR (%)			77.5 *66.0* (77.8)[77.3]			**78.4** *64.3* (**78.4**)[77.5]
	Flowering (mid-June)	Y = 0	515 *514* (515)[514]	5 *6* (5)[6]	99.0 *98.8* (99.0) [98.8]	219 *219* (219)[219]	3 *3* (3)[3]	98.6 *98.6* (98.6)[98.6]
		Y = 1	2 *4* (1)[2]	517 *515* (518)[517]	99.6 *99.2* (99.8)[99.6]	4 *4* (4)[3]	219 *219* (219)[220]	98.2 *98.2* (98.2)[98.7]
		CCR (%)			99.3 *99.0* (99.4)[99.2]			98.4 *98.4* (98.4)[**98.7**]
	Senescence (mid-July)	Y = 0	390 *368* (392)[391]	130 *152* (128)[129]	75.0 *70.8* (75.4)[75.2]	163 *150* (165)[166]	59 *72* (57)[56]	73.4 *67.6* (74.3)[74.8]
		Y = 1	28 *87* (29)[30]	491 *432* (490)[489]	94.6 *83.2* (94.4)[94.2]	16 *37* (16)[14]	207 *186* (207)[209]	92.8 *83.4* (92.8)[93.7]
		CCR (%)			84.8 *77.0* (84.9)[84.7]			83.1 *75.5* (83.6)[**84.3**]

EPUNN results are presented in regular font, LR results in italic font, LRPU between parentheses and LRIPU between square brackets. The best methodology is presented in bold face.
Phen., phenological; EPUNN, evolutionary product-unit neural networks; LR, logistic regression; LRPU, logistic regression using product units; LRIPU, logistic regression using initial covariates and product units.

Table 4 – Expression of the probability equation associated to the different models

Location	Method	#Param.	Best models
Matabueyes	EPUNN	8	$P = 1/(1 + \exp(-(-0.424 + 75.419(G^{4.633}) + 0.322(R^{-1.888}) + 14.990(B^{3.496}\ G^{-3.415}))))$
	LR	4	$P = 1/(1 + \exp(-(-0.694 + 8.282(B) - 63.342(G) - 11.402(R))))$
	LRPU	8	$P = 1/(1 + \exp(-(-17.227 + 143.012(G^{4.633}) + 0.636(R^{-1.888}) + 23.021(B^{3.496}\ G^{-3.415}))))$
	LRIPU	9	$P = 1/(1 + \exp(-(18.027 + 130.674(B) - 133.662(G) - 29.346(R) + 353.147(G^{4.633}) - 3.396(B^{3.496}\ G^{-3.415}))))$
Santa Cruz	EPUNN	9	$P = 1/(1 + \exp(-(6.114 - 1.505(R^{-1.246}) - 25(G^{3.722}\ R^{1.867}) - 0.311(B^{2.665}\ N^{-3.875}))))$
	LR	4	$P = 1/(1 + \exp(-(-3.240 - 5.1(B) + 8.623(R) + 3.429(N))))$
	LRPU	9	$P = 1/(1 + \exp(-(6.803 - 1.682(R^{-1.246}) - 30.537(G^{3.722}\ R^{1.867}) - 0.317(B^{2.665}\ N^{-3.875}))))$
	LRIPU	11	$P = 1/(1 + \exp(-(1.436 + 5.427(G) + 3.169(N) - 1.268(R^{-1.246}) - 41.646(G^{3.722}\ R^{1.867}) - 0.239(B^{2.665}N^{-3.875}))))$

EPUNN, evolutionary product-unit neural networks; LR, logistic regression; LRPU, logistic regression using product units; LRIPU, logistic regression using initial covariates and product units; #Param., number of parameters of the models; *P*: probability of *R. segetum* presence; R^*, G^*, B^* and NIR*: digital values of red (R), green (G), blue (B) and near infrared (NIR) bands. Scaled variables R^*, G^*, B^* and NIR$^* \in [0.1, 0.9]$.

As shown in Table 3, LRIPU and EPUNN models were the better performing ones, achieving a very high accuracy in the generalization set in mid-June with 99.2% and 98.7% for the LRIPU model in Matabueyes and Santa Cruz, respectively. Moreover, LRIPU Types I and II errors are very low with values of $FP = (1 - CCR_N) = 0.08\%$ and $FN = (1 - CCR_P) = 0.08\%$ for Matabueyes and $FP = (1 - CCR_N) = 0.14\%$ and $FN = (1 - CCR_P) = 0.13\%$ for Santa Cruz. The LR model is not able to reflect the nonlinear relationships between input variables, necessary for performing a realistic classification task. The mathematical expressions of the different models are presented in Table 4, all of them being relatively simple, especially if we compare these expressions with the expressions that could be obtained using more traditional MLP Sigmoidal Units.

In Table 5, we show the performance obtained by the models developed in this work as compared to the performance obtained by SVM and LMT methodologies, two of the more recent and accurate classifier proposals in the machine learning community. As we can see in these results, SVM did not achieve a good performance in the different datasets resulting in very low generalization ability, and LMT obtains an accuracy very close to that obtained by the methodologies presented in this paper. However, EPUNN and LRIPU outperformed both SVM and LMT in five out of the six databases considered. Moreover, it is important to note that SVM produces a dichotomous classifier, which only gives a binary prediction in each pixel. The rest of models (including LMT) predict a probability, differentiating in this way pixels with a similar prediction (*R. segetum* presence or absence).

Finally, the classification task was generalized to the complete fields of study and the weed probabilities predicted by the best performing model (LRIPU) and the worst performing model (LR) were assigned to the corresponding pixels of a new map using ENVI (ENVI 4.0 software, Research Systems Inc.). We selected mid-June at Matabueyes because the best *R. segetum* discrimination results were obtained using LRIPU in this date. In order to visualize the differences in accuracy on *R. segetum* discrimination at Santa Cruz, we selected mid-July because the performances of both LR and LRIPU methods in mid-June were very similar to each other (98.4% and 98.7%, respectively, Table 5). In Figs. 2 and 3, the corresponding maps are represented. We can conclude from the analysis of the maps that LRIPU model was more precise in the prediction of the probability of *R. segetum* presence, discriminating more clearly the different patches zones of the study fields.

4. Discussion

Classification accuracy of *R. segetum* patches in the sunflower crop was consistently affected by the dates when aerial images were taken, LRIPU being the most accurate method in both locations. All the algorithms for classification studied provided higher accuracies in images taken in this order:

Table 5 – Comparative performance of the probability equations for the different approaches proposed and for two other recent state-of-the-art methodologies: logistic model trees and support vector machines

Location	Date	CCR_T (%)						CCR_G (%)					
		EPUNN	LR	LRPU	LRIPU	SVM	LMT	EPUNN	LR	LRPU	LRIPU	SVM	LMT
Matabueyes	(mid-May)	72.4	66.2	72.1	72.6	67.2	76.4	70.6	66.5	70.2	70.6	65.6	70.1
	(mid-June)	98.2	94.6	98.0	98.3	93.6	98.8	98.7	94.6	99.0	99.2	93.7	98.7
	(mid-July)	80.1	64.7	79.6	79.5	59.8	91.3	79.8	66.3	79.6	79.0	59.0	82.0
Santa Cruz	(mid-May)	77.5	66.0	77.8	77.3	67.0	89.4	78.4	64.3	78.4	77.5	65.2	77.3
	(mid-June)	99.3	99.0	99.4	99.2	97.6	99.2	98.4	98.4	98.4	98.7	97.0	98.2
	(mid-July)	84.8	77.0	84.9	84.7	78.4	86.2	83.1	75.5	83.6	84.3	77.5	83.8

CCR_T, correctly classified rate in the training set; CCR_G, correctly classified rate in the generalization set; EPUNN, evolutionary product-unit neural networks; LR, logistic regression; LRPU, logistic regression using product units; LRIPU, logistic regression using initial covariates and product units; SVM, support vector machines; LMT, logistic model trees.

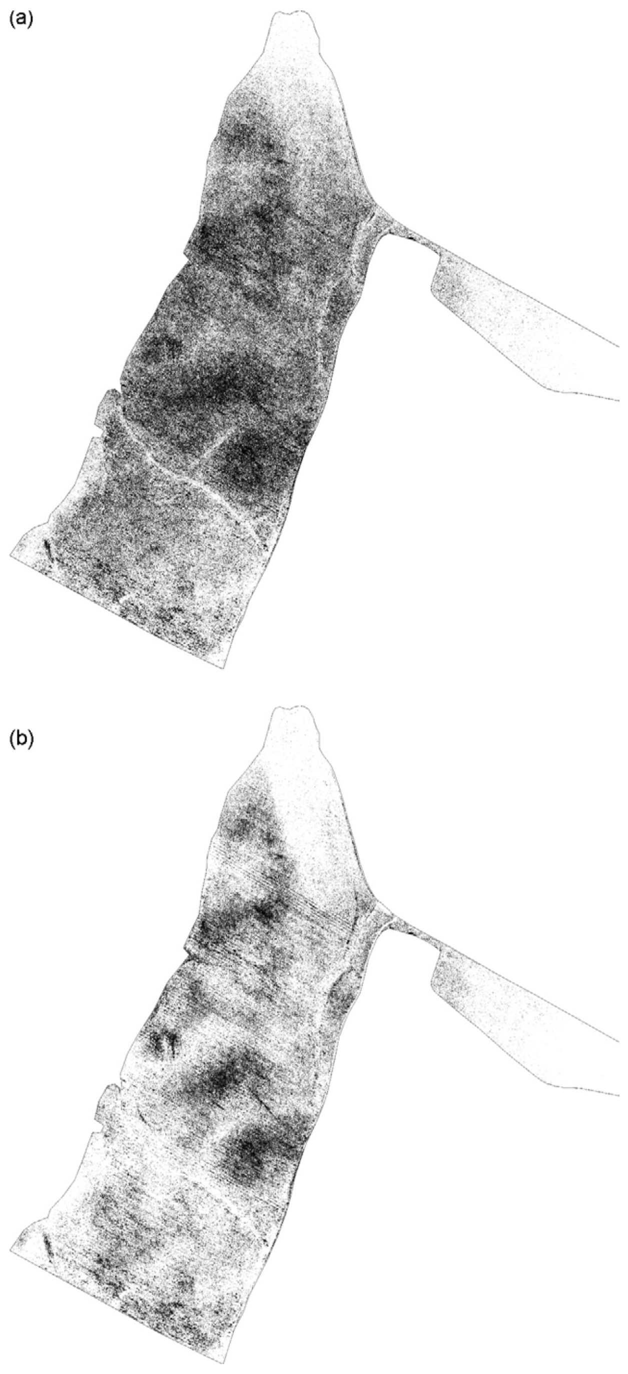


Fig. 2 – *R. segetum* probability maps for Matabueyes in mid-June. The probability ranges from $p = 0$ (*R. segetum* free pixels), represented in white colour, to $p = 1$ (*R. segetum* presence pixels), represented in dark green. (a) LR: *R. segetum* probability map. (b) LRIPU: *R. segetum* probability map.

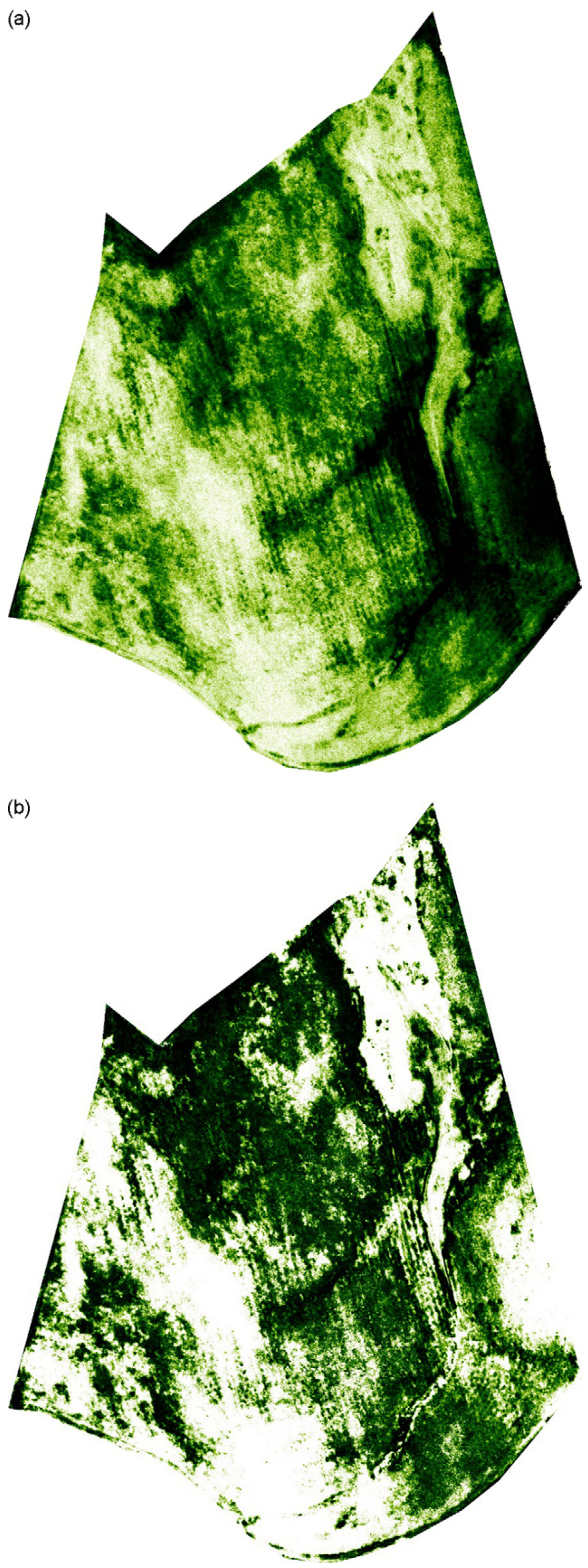


Fig. 3 – *R. segetum* probability maps for Santa Cruz in mid-July. The probability ranges from $p = 0$ (*R. segetum* free pixels, represented in white colour) to $p = 1$ (*R. segetum* presence pixels, represented in dark green). (a) LR: *R. segetum* probability map. (b) LRIPU: *R. segetum* probability map.

mid-June (corresponding to the flowering phase) > mid-July (corresponding to the senescent phase) > mid-May (corresponding to the vegetative phase). None of the classification methodologies used over the mid-May and mid-July imagery yielded accuracies higher than 84.3%, and, therefore it should be recommended not taking any images in the corresponding phenological stages to discriminate *R. segetum* patches in sunflower. By contrast, all the classification algorithms studied in mid-June discriminated the weed with accuracies higher than 98%, except for LR with 94.5% of accuracy, being LRIPU the most accurate. This differential accuracy is in agreement with previous results of Peña-Barragán et al. (2007), where they already discussed the influence of *R. segetum* and sunflower phenological stages in these results.

One of the most interesting results from this study is that our methods minimized the false detection of presence (Type I error) or absence (Type II error) of weed in both locations, finding that the LRIPU method selected as optimal provided classification accuracies that were very high and similar in both locations. By contrast, pixel-based image classification using traditional vegetation indices approaches, previously reported by Peña-Barragán et al. (2007), produced maximum accuracies of 85% and 99% with R/B index at Matabueyes and Santa Cruz, respectively. The low classification accuracy obtained at Matabueyes by using conventional vegetation indices approaches, or those herein obtained in mid-May and in mid-July for image classification, may both underestimate the presence of *R. segetum* ($Y = 1$) patches and thus misclassify the total surface of weed infestation (omission error or Type II error), or overestimate the presence of *R. segetum* and therefore provide a conservative estimation of weed infestation (commission error or Type I error). From an agronomic point of view, this has important implications because any control strategy should not assume the risk of applying an unnecessary herbicide on soil or crop or of allowing *R. segetum* to go untreated. In this last case, two *R. segetum* uncontrolled plants per m^2 would result in an average sunflower yield reduction of about 32% as previously reported by Carranza-Cañadas et al. (1995).

Every algorithm, excepting SVM, estimates the probability of occurrence of *R. segetum*. This is beneficial for control strategies since the models not only predict a dichotomous outcome, i.e. presence or absence of weed, but also the probability associated with the occurrence of this event. In our case, we simply checked when this probability was greater or lower than a probability of 0.5. We applied this general assumption so that pixels ranging from 0.5 to 1 were classified as presence of *R. segetum*, and the remaining ones as absence of weed. However, when considering a wider probability threshold for presence of *R. segetum*, e.g. from 0.3 to 1, a higher amount of pixels would be classified as presence of weed and a more conservative weed map would be achieved. So, if the probability range associated to the binary outcome is changed or adjusted according to spatial distribution of *R. segetum*, a harder classification could be obtained in dates with worst classification results. For example, spatial pattern of *R. segetum* patches have shown positive correlation with field elevation, which means that this weed is dominant in areas with high elevation values (Jurado-Expósito et al., 2005b). Taking into account this information, the probability threshold associated to the first class (i.e., *R. segetum* presence) could be adjusted by decreasing it (<0.5) in higher parts of the fields and increasing it (>0.5) in the lower ones. Consequently, this more adjusted probability interval than the one selected in our study could improve our results in mid-May and mid-July and possibly, these dates could also be recommended for generating weed maps.

A key component of population dynamics of weeds is that grass and broadleaf weed infestations are often persistent and relatively stable in location year to year (Barroso et al., 2004b; Jurado-Expósito et al., 2004). Thus, late-season weed detection maps can be used to design site-specific control in subsequent years. However, to take full advantage of our results, next investigations could explore the potential of high resolution satellite imagery such as QuickBird and the coming images of WorldView II for mapping *R. segetum* patches in sunflower in larger areas (of at least over 64 km^2). Jurado-Expósito et al. (2005b), applying geostatistical techniques, demonstrated that the extension of *R. segetum* patches in sunflower was at least 9 m. According to these results, the classification accuracy herein presented, and taking into consideration the QuickBird imagery provides four channels (B, G, R and NIR) of multispectral wavebands with 2.4 m or 2.8 m of spatial resolution, it would be possible to successfully map this weed on a large surface, provided that QuickBird has been proved to be a useful data source for mapping invasive plant species (Tsai and Chou, 2006).

5. Conclusions

This study demonstrated the capability of LR and PUNN combination models to analyze multispectral imagery for predicting *R. segetum* presence probability and mapping *R. segetum* patches in the different fields of study. LRIPU and EPUNN models provided better accuracy than linear LR models both in training sets and generalization sets. Excellent generalization accuracies were obtained through the application of the best performing model (LRIPU) in mid-June at both locations. Our study corroborated that the phenological stages/dates when aerial images were taken significantly affect the accuracy in discriminating *R. segetum* patches in sunflower crop, decreasing in efficiency in the two fields considered in the following order: mid-June (corresponding to the flowering phase) > mid-July (corresponding to the senescent phase) > mid-May (corresponding to the vegetative phase). Therefore, reliable mapping of *R. segetum* in sunflower should be generated using images around mid-June, in order to apply a more efficient site-specific control in subsequent years. The maps generated using LRIPU models were more precise than those generated by the linear LR model. Moreover, two advanced methodologies (SVM and LMT) were compared to the methodologies herein presented, resulting in lower accuracy for SVM and LMT, in the six locations/dates evaluated and in five datasets, respectively. Granted that, computational requirements for EPUNN were much higher than for LR, SVM, LTM or traditional vegetation indices approaches, those necessary for LRIPU were nearly insignificant. Thus, considering that precision agriculture management requires a great accuracy, we suggest that the criteria for selecting our algorithms or vegetation indices classification should not be based on decreasing computational requirements or complexity, but on the accuracy of discrimination.

Acknowledgements

This work has been partially financed by TIN2005-08386-C05-02 and AGL-2008-04670-CO3-03, projects of the Spanish Inter-

Ministerial Commission of Science and Technology and FEDER funds. The research of P.A. Gutiérrez has been subsidized by the FPU Predoctoral Program (Spanish Ministry of Education and Science), grant reference AP2006-01746.

REFERENCES

Alcala-Fdez, J., Sánchez, L., García, S., del Jesús, M.J., Ventura, S., Garrell, J.M., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernández, J.C., Herrera, F., in press. KEEL: a software tool to assess evolutionary algorithms for data mining problems. Soft Computing: A Fusion of Foundations, Methodologies and Applications, doi:10.1007/s00500-008-0323-y.

Angeline, P.J., Saunders, G.M., Pollack, J.B., 1994. An evolutionary algorithm that constructs recurrent neural networks. IEEE Transaction on Neural Networks 5 (1), 54–65.

Atkinson, P.M., Tatnall, A.R.L., 1997. Neural networks in remote sensing. International Journal of Remote Sensing 18 (4), 699–709.

Barroso, J., Fernández-Quintanilla, C., Maxwell, B.D., Rew, L., 2004a. Simulating the effects of spatial weed pattern and resolution of mapping and spraying on economics of site-specific management. Weed Research 44 (3), 413–488.

Barroso, J., Fernández-Quintanilla, C., Ruiz, D., Hernaiz, P., Rew, L., 2004b. Spatial stability of *Avena sterilis* spp. ludoviciana under annual applications of low rates of imazamethabenz. Weed Research 44 (3), 178–186.

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press, UK.

Brown, R.B., Noble, S.D., 2005. Site-specific weed management: sensing requirements-what do we need to see? Weed Science 53 (2), 252–258.

Carranza-Cañadas, P., Saavedra, M., García-Torres, L., 1995. Competition of *Ridolfia segetum* and sunflower. Weed Research 35 (5), 369–375.

Cetin, H., Pafford, J.T., Mueller, T.G., 2005. Precision agriculture using hyperspectral remote sensing and GIS. In: Proceedings of the 2nd International Conference on Recent Advances in Space Technologies (RAST2005), pp. 70–77.

Cristianini, N., Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, UK.

Durbin, R., Rumelhart, D., 1989. Products units: a computationally powerful and biologically plausible extension to backpropagation networks. Neural Computation 1 (1), 133–142.

Flowers, M., Weisz, R., Heiniger, R., 2001. Remote sensing of winter wheat tiller density for early nitrogen application decisions. Agronomy Journal 93 (4), 783–789.

Foody, M., Arora, M.K., 1996. Incorporating mixed pixel in the training, allocation and testing stages of supervised classification. Pattern Recognition Letters 17 (13), 1389–1398.

Fraser, R.H., Abuelgasim, T.A., Latifovic, R., 2005. A method for detecting large-scale forest cover change using coarse spatial resolution imagery. Remote Sensing of Environment 95 (4), 414–427.

García-Pedrajas, N., Hervás-Martínez, C., Muñoz-Pérez, J., 2002. Multi-objective cooperative coevolution of artificial neural networks. Neural Networks 15 (10), 1255–1274.

Goel, P.K., Prasher, S.O., Patel, R.M., Landry, J.A., Bonnell, R.B., Viau, A.A., 2003. Classification of hyperspectral data by decision trees and artificial neural networks to identify weed stress and nitrogen status of corn. Computers and Electronics in Agriculture 39 (2), 67–93.

Gutiérrez, P.A., López-Granados, F., Peña-Barragán, J.M., Jurado-Expósito, M., Gómez-Casero, M.T., Hervás-Martínez, C., 2008. Mapping sunflower yield as affected by *Ridolfia segetum* patches and elevation by applying evolutionary product unit neural networks to remote sensed data. Computers and Electronics in Agriculture 60 (2), 122–132.

Hastie, T., Tibshirani, R.J., Friedman, J., 2001. The elements of statistical learning. In: Data Mining, Inference and Prediction. Springer, Berlin.

Hervás-Martínez, C., Martínez-Estudillo, F.J., 2007. Logistic regression using covariates obtained by product-unit neural network models. Pattern Recognition 40 (1), 52–64.

Hervás-Martínez, C., Martínez-Estudillo, F.J., Carbonero-Ruz, M., 2008. Multilogistic regression by means of evolutionary product-unit neural networks. Neural Networks 21, 951–961.

Hosmer, D.W., Lemeshow, S., 1989. Applied Logistic Regression. Wiley, New York.

Ismail, A., Engelbrecht, A.P., 2002. Pruning product unit neural networks. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN'2002), Honolulu, USA, pp. 257–262.

Janson, D.J., Frenzel, J.F., 1993. Training product unit neural networks with genetic algorithms. IEEE Expert 8 (5), 26–33.

Jurado-Expósito, M., López-Granados, F., García-Torres, L., García-Ferrer, A., Sánchez de la Orden, M., Atenciano, S., 2003. Multi-species weed spatial variability and site-specific management maps in cultivated sunflower. Weed Science 51 (3), 319–328.

Jurado-Expósito, M., López-Granados, F., González-Andújar, J.L., García-Torres, L., 2004. Spatial and temporal analysis of *Convolvulus arvensis* L. populations over four growing seasons. European Journal of Agronomy 21 (3), 287–296.

Jurado-Expósito, M., López-Granados, F., González-Andújar, J.L., García-Torres, L., 2005a. Characterizing population growth rate of *Convolvulus arvensis* in wheat-sunflower no tillage systems. Crop Science 45, 2106–2112.

Jurado-Expósito, M., López-Granados, F., Peña-Barragán, J.M., García-Torres, L., 2005b. Weed density predicition with secondary input of DEM information. In: Stafford, J. (Ed.), Proceedings of the Fifth European Conference on Precision Agriculture. Uppsala, Sweden, pp. 115–122.

Kanellopoulos, I., Wilkinson, G.G., 1997. Strategies and best practice for neural network image classification. International Journal of Remote Sensing 18 (4), 711–725.

Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Koger, H.K., Sahw, R.R., Watson, C.E., Reddy, K.N., 2003. Detecting late-season weed infestation in soybean (Glycine max). Weed Technology 17 (4), 696–704.

Landwehr, N., Hall, M., Frank, E., 2005. Logistic model trees. Machine Learning 59 (1/2), 161–205.

Lenze, B., 1994. How to make sigma–pi neural networks perform perfectly on regular training sets. Neural Networks 7 (8), 1285–1293.

López-Granados, F., Peña-Barragán, J.M., Jurado-Expósito, M., García-Torres, L., 2006. Using remote sensing for identification of late-season grassy weeds patches in wheat (*Triticum durum*) for precision agriculture. Weed Science 54 (2), 346–353.

Magnussen, S., Boudewyn, P., Alfaro, R., 2004. Spatial prediction of the onset of spruce budworm defoliation. The Forestry Chronicle 80 (4), 485–494.

MAPA, 2007. Spanish Ministry of Agriculture, Fisheries and Food. www.mapa.es/es/estadistica/infoestad.html.

Martínez-Estudillo, A.C., Mártínez-Estudillo, F.J., Hervás-Martínez, C., García-Pedrajas, N., 2006. Evolutionary product unit based neural networks for regression. Neural Networks 19 (4), 477–486.

Martínez-Estudillo, F.J., Hervás-Martínez, C., Gutiérrez, P.A., Mártínez-Estudillo, A.C., in press. Evolutionary product-unit

neural networks classifiers. Neurocomputing, available online January 15, 2008, doi:10.1016/j.neucom.2007.11.019.
Mather, P.M., 2000. Computer Processing of Remotely-sensed Images: An Introduction. Wiley, UK.
McCoy, R.M., 2005. Fields Methods in Remote Sensing. The Guilford Press, New York.
McLachlan, G., 1992. Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York.
Mertens, K.C., Verbeke, L.P.C., Ducheyne, E.I., De Wulf, R.R., 2003. Using genetic algorithms in sub-pixel mapping. International Journal of Remote Sensing 24 (21), 4241–4247.
Monserud, R.A., Sterba, H., 1999. Modeling individual tree mortality for Austrian forest species. Forest Ecology and Management 113 (2/3), 109–123.
Moody, A., Gopal, S., Strahler, A.H., 1996. Artificial neural network response to mixed pixels in coarse-resolution satellite data. Remote Sensing of Environment 58 (3), 329–343.
Peña-Barragán, J.M., López-Granados, F., Jurado-Expósito, M., García-Torres, L., 2006. Spectral discrimination of *Ridolfia segetum* and sunflower as affected by phenological stage. Weed Research 46 (1), 10–21.
Peña-Barragán, J.M., López-Granados, F., Jurado-Expósito, M., García-Torres, L., 2007. Mapping *Ridolfia segetum* Moris patches in sunflower (*Helianthus annuus* L.) crop using remote sensing. Weed Research 47 (2), 164–172.
Plant, R.E., 2001. Site-specific management: the application of information technology to crop production. Computers and Electronics in Agriculture 30 (1), 9–29.
Provost, F., Kohavi, R., 1998. On applied research in machine learning. Machine Learning 30 (2/3), 127–132.
Schmitt, M., 2002. On the complexity of computing and learning with multiplicative neural networks. Neural Computation 14 (2), 241–301.
Seelan, S.K., Laguette, S., Casady, G.M., Seielstad, G.A., 2003. Remote sensing applications for precision agriculture: a learning community approach. Remote Sensing of Environment 88 (1/2), 157–169.
Srinivasan, A., 2006. Handbook of Precision Agriculture: Principles and Applications. The Haworth Press, New York.
Thorp, K.R., Tian, L.F., 2004. A review of remote sensing of weeds in agriculture. Precision Agriculture 5 (5), 477–508.
Timmermann, C., Gerhards, R., Kuehbauch, W., 2003. The economic impact of the site specific weed control. Precision Agriculture 4 (3), 249–260.
Tsai, F., Chou, M.J., 2006. Texture augmented analysis of high resolution satellite imagery in detecting invasive plants species. Journal of Chinese Institute of Engineering 29 (4), 581–592.
Uno, Y., Prasher, S.O., Lacroix, R., Goel, P.K., Karimi, Y., Viau, A., Patel, R.M., 2005. Artificial neural networks to predict corn yield from compact airborne spectrographic imager data. Computers and Electronics in Agriculture 47 (2), 149–161.
USDA-NRCS, 1998. Keys to Soil Taxonomy, 8th ed. USDA-NRCS, Washington, DC, USA.
Vanclay, J.K., 1995. Growth models for tropical forests: a synthesis of models and methods. Forest Science 41 (1), 7–42.
Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer, New York.
Ventura, S., Romero, C., Zafra, A., Osuna, J.A., Hervás-Martínez, C., 2008. JCLEC: a JAVA framework for evolutionary computation. Soft Computing: A Fusion of Foundations, Methodologies and Applications 12 (4), 381–392.
Witten, I.H., Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed. Morgan Kaufmann, San Francisco.
Wulder, M.A., White, J.C., Bentz, B., Alvarez, M.F., Coops, N.C., 2006. Estimating the probability of mountain pine beetle red-attack damage. Remote Sensing of Environment 101 (2), 150–166.
Xu, M., Watanachaturaporn, P., Varshney, P.K., Arora, M.K., 2005. Decision tree regression for soft classification of remote sensing data. Remote Sensing of Environment 97 (3), 322–336.
Yang, C., Prasher, S.O., Landry, J.A., Ramaswamy, H.S., Ditommaso, A., 2000a. Application of artificial neural networks in image recognition and classification of crop and weeds. Canadian Agriculture Engineering 42 (3), 147–152.
Yao, X., 1999. Evolving artificial neural networks. Proceedings of the IEEE 87 (9), 1423–1447.
Yao, X., Liu, Y., 1997. A new evolutionary system for evolving artificial neural networks. IEEE Transaction on Neural Networks 8 (3), 694–713.
Zhai, Y., Thomasson, J.A., Boggess III, J.E., Sui, R., 2006. Soil texture classification with artificial neural networks operating on remote sensing data. Computers and Electronics in Agriculture 54 (2), 53–68.
Zhan, H.G., Lee, Z.P., Shi, P., Chen, C.Q., Carder, K.L., 2003. Retrieval of water optical properties for optically deep waters using genetic algorithms. IEEE Transactions on Geoscience and Remote Sensing 41 (5), 1123–1128.

3.1.3. Regresión Logística Multi-clase mediante Redes Neuronales Evolutivas como una Herramienta de Clasificación para Discriminar Señales Altamente Solapadas: Investigación Cualitativa de Compuestos Volátiles Orgánicos en Aguas Contaminadas mediante el Análisis de Datos de Cromatografía de Gases acoplada con Espectrometría de Masas - *Multilogistic regression by evolutionary neural network as a classification tool to discriminate highly overlapping signals: Qualitative investigation of volatile organic compounds in polluted waters by using headspace-mass spectrometric analysis*

- C. Hervás-Martínez, M. Silva, P.A. Gutiérrez, A. Serrano. Multilogistic regression by evolutionary neural network as a classification tool to discriminate highly overlapping signals: Qualitative investigation of volatile organic compounds in polluted waters by using headspace-mass spectrometric analysis. Chemometrics and Intelligent Laboratory Systems 92:2 (2008) 179-185.
  - Estado: Publicado.
  - Índice de Impacto (JCR 2007): 2.063.
  - Área de conocimiento: *Automation & Control Systems.* Ranking 5/52. Primer Cuartil.
  - Área de conocimiento: *Chemistry, Analytical.* Ranking 27/70. Segundo Cuartil.
  - Área de conocimiento: *Computer Science, Artificial Intelligence.* Ranking 16/93. Primer Cuartil.
  - Área de conocimiento: *Instruments & Instrumentation.* Ranking 6/55. Primer Cuartil.
  - Área de conocimiento: *Mathematics, Interdisciplinary Applications.* Ranking 5/74. Primer Cuartil.
  - Área de conocimiento: *Statistics & Probability.* Ranking 8/91. Primer Cuartil.

Chemometrics and Intelligent Laboratory Systems 92 (2008) 179–185

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemolab

Multilogistic regression by evolutionary neural network as a classification tool to discriminate highly overlapping signals: Qualitative investigation of volatile organic compounds in polluted waters by using headspace-mass spectrometric analysis

César Hervás [a], Manuel Silva [b,*], Pedro Antonio Gutiérrez [a], Antonio Serrano [b]

[a] Department of Computer Science, Albert Einstein Building, University of Cordoba, E-14071 Cordoba, Spain
[b] Department of Analytical Chemistry, Marie-Curie Building (Annex), University of Cordoba, E-14071 Cordoba, Spain

ARTICLE INFO

Article history:
Received 26 November 2007
Received in revised form 12 March 2008
Accepted 13 March 2008
Available online 27 March 2008

Keywords:
Logistic regression
Product unit neural networks
Multi-class pattern recognition
Volatile organic compounds
Headspace-mass spectrometric analysis

ABSTRACT

This work investigates the ability of multilogistic regression models including nonlinear effects of the covariates as a multi-class pattern recognition technique to discriminate highly overlapping analytical signals using a very short number of input covariates. For this purpose, three methodologies recently reported by us were applied based on the combination of linear and nonlinear terms which are transformations of the linear ones by using evolutionary product unit neural networks. To test this approach, drinking water samples contaminated with volatile organic compounds such as benzene, toluene, xylene and their mixtures were classified in seven classes through the very close data provided by their headspace-mass spectrometric analysis. Instead of using the total ion current profile provided by the MS detector as input covariates, the three-parameter Gaussian curve associated to it was used as linear covariates for the standard multilogistic regression model, whereas the product unit basic functions or their combination with the linear covariates were used for the nonlinear models. The hybrid nonlinear model, pruned by a backward stepwise method, provided the best classification results with a correctly classified rate for the training and generalization sets of 100% and 76.2%, respectively. The reduced dimensions of the proposed model: only three terms, namely one initial covariate and two basis product units, enabled to infer interesting interpretations from a chemical point of view.

1. Introduction

Qualitative analysis is increasingly being viewed as an emerging branch of analytical chemistry [1] due to the combination of powerful instrumental techniques, such as chromatography with mass spectrometric or diode-array detector, Fourier-transform infrared spectroscopy, chemical microsensors array, etc. with chemometric methods which expands the possibilities of the identification. “Classification according to specific criteria” is the general definition for qualitative testing [2], which also includes the well-known screening methods used for finding out if a sample contains one or more specific analytes based on a binary response. So, in a broad sense, qualitative analysis is really a simple classification methodology. Several chemometric methods have been used in Analytical Chemistry for qualitative analysis as supervised or unsupervised classification tools. Although a detailed review of these methods is out of the scope of this paper, some choices are factor analysis, cluster analysis (CA), K-Nearest Neighbours (KNN), linear and quadratic discriminant analysis (LDA & QDA), partial least squares discriminant analysis (PLS-DA), and soft independent modelling of class analogy (SIMCA), among others [3,4]. In addition, artificial neural networks (ANNs), with their pattern recognition and modelling capabilities, have currently become powerful classification tools in qualitative chemical analysis [5,6].

Multilogistic regression is a special case of generalized linear model methodology where the assumptions of normality and constant variance of the residuals are not satisfied [7,8]. Multilogistic regression models have demonstrated their accuracy in many classification frameworks often providing classifiers easily interpretable that can be used for the appropriate selection of the adequate model in real supervised learning situations [9–11]. In the last few years, multilogistic regression models have shown their potential as classification and screening tools particularly in clinical analysis, where they have been applied for predicting the probability of suffering a certain disease in terms of the levels of different biomarkers [12–15] or for providing a positive or negative diagnostic over a form of cancer on the basis of different indicators such as serum proteins profile [16,17], concentrations of certain trace elements in bodily fluids [18] and plasma levels of organochlorines [19], among others. In addition, other classification problems have been successfully addressed with multilogistic regression such as the characterization of honey from its mineral content [20], the characterization of pharmaceuticals based on

* Department of Analytical Chemistry, Marie-Curie Building (Annex), Rabanales Campus, University of Cordoba, E-14071 Cordoba, Spain. Tel.: +34 957 212099; fax: +34 957 218614.
E-mail address: qa1sirom@uco.es (M. Silva).

0169-7439/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.chemolab.2008.03.005

the determination of polymorphic purity [21] and the authentication of virgin olive oils of very close geographical origins by near infrared spectroscopy [22].

In this work, the potential of a methodology recently developed by our research group [23] is evaluated as an analytical classification tool for the discrimination between classes that provide highly overlapped signals using a very short number of initial covariates. Specifically, the methodology implies an enhancement of the standard multilogistic regression by including the nonlinear effects of the covariates on the basis of the hybridization of the initial and nonlinear transformed covariates. These nonlinear transformed covariates are constructed with product unit (PU) basis functions given by products of the inputs of a product unit neural network (PUNN) raised up to real powers, which capture the possible strong interactions between the variables. PU functions correspond to a special class of feed-forward neural networks, namely PUNN, introduced by Durbin and Rumelhart [24] and subsequently developed by other authors [25–30]. In this way, standard and product unit basis functions covariates multilogistic regression models were tested [23], including the standard multilogistic regression model (MR) based on the initial covariates and two other MR models using product unit basis functions: the first constructed only on the PU basis functions of the PUNNs (MRPU) and the second with both PUs and initial covariates (MRIPU).

The classification efficiency on the training and the generalization data sets of these models are compared among themselves and also with those provided by classical statistical algorithms such as LDA and QDA. Moreover, recent common classification methods of artificial intelligence, such as support vector machine (SVM) and decision trees, are applied and compared to these MR models. SVM [31,32] is a very popular tool in machine learning that explores the kernel techniques with good geometric explanation, and which usually performs very well in many classification applications; whereas the decision tree algorithm for classification constructs decision trees, where the leaves represent classifications and the branches represent conjunctions of features that lead to those classifications [33,34].

To test this classification tool, a complex analytical pattern recognition problem was investigated, namely the classification of drinking water samples contaminated by the volatile organic compounds (VOCs) benzene, toluene and xylene based on headspace-mass spectrometric (HS-MS) data. The seven classes of water samples contain one of these compounds as well as their binary or ternary mixtures. The complexity of this chemical system is related to highly overlapped MS signals provided by the classes in study. In addition, a data treatment methodology is proposed to extract useful chemical information from the volatile profile provided by the HS-MS based on the decreasing of the number of input parameters, and whose aim is to use as simple multilogistic regression models as possible. So, the number of covariates, used also as inputs to the PUNNs, was estimated by the Levenberg–Marquardt method in the form of a three-parameter Gaussian curve associated with the total ion current profile provided by the MS detector using a similar methodology to the one based on Gaussian and Weibull functions [35–39], previously reported by us. These compounds were also chosen because they belong to the group of most dangerous water pollutants. Their recognition is very important due to differences in toxicity of these compounds and their impact on human health and the environment.

Despite the relatively low number of patterns used for the training and generalization sets, the proposed approach provided good results for the qualitative investigation of these VOCs in polluted waters, and to our knowledge no study on the use of nonlinear transformed covariates multilogistic regression models has to date been reported in this context. In addition, it provides a more quality information than the classical screening methods based on the typical binary response, and it can be extended to others methodologies that insert analytes from a sample directly into a MS, such as membrane introduction MS, among others.

2. Classification method

In multiclassification problems, measurements x_i ($i=1,2,\ldots,k$) are made from a single individual (or object), and individuals are classified into one of J classes on the basis of these measurements. In this paper, the common technique of representing the class levels with a "1-of-J" encoding vector $\mathbf{y}=(y^{(1)},y^{(2)},\ldots y^{(J)})$ is used. Thus, from a training sample defined as $D=\{(\mathbf{x}_n,\mathbf{y}_n);n=1,2,\ldots,N\}$ in which $\mathbf{x}_n=(x_{1n},\ldots,x_{kn})$ is the vector of measurements taking values in $\Omega\subset R^k$ and $\mathbf{y}_n$ is the class level of the nth individual, the query is to find a decision function $C:\Omega\rightarrow\{1,2,\ldots J\}$ based on generalized linear regression models, with new covariates based on PU basis functions for classifying the individuals. A misclassification occurs when a decision rule C assigns an individual (based on measurements vector) to a class j when it is actually coming from a class $l\neq j$.

The logistic regression methods are common statistical tools for modelling discrete response variables; such as multiple, binary, categorical and ordinal responses. In the multiple and categorical cases, the conditional probability that $\mathbf{x}$ belongs to class l verifies:

$$p\left(y^{(l)}=1|\mathbf{x}\right)>0,\ l=1,2,\ldots,J,\ \mathbf{x}\in\Omega \tag{1}$$

and sets the function:

$$f_l(\mathbf{x},\boldsymbol{\theta}_l)=\log\frac{p\left(y^{(l)}=1|\mathbf{x}\right)}{p\left(y^{(J)}=1|\mathbf{x}\right)},\ l=1,2,\ldots,J,\ \mathbf{x}\in\Omega \tag{2}$$

where $\boldsymbol{\theta}_l$ is the weight vector corresponding to the class l and $f_J(\mathbf{x},\boldsymbol{\theta}_J)\equiv 0$ considering the J class as the base class. Under a multilogistic regression model, the probability that $\mathbf{x}$ belongs to class l is then given by

$$p\left(y^{(l)}=1|\mathbf{x},\boldsymbol{\theta}\right)=\frac{\exp f_l(\mathbf{x},\boldsymbol{\theta}_l)}{\sum_{j=1}^{J}\exp f_j(\mathbf{x},\boldsymbol{\theta}_j)},\ l=1,2,\ldots,J \tag{3}$$

where $\boldsymbol{\theta}=(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_{J-1})$.

An individual should be assigned to the class which has the maximum probability, given the vector measurement $\mathbf{x}$, that is: $C(\mathbf{x})=\hat{l}$, where $\hat{l}=\arg\max_l f_l(\mathbf{x},\hat{\boldsymbol{\theta}}_l)$, for $l=1,\ldots,J$. In this work, a multilogistic regression model developed recently by us based on the combination of linear and nonlinear terms [23] is applied to qualitative analysis. The nonlinear part of the function $f_l(\mathbf{x},\boldsymbol{\theta}_l)$ corresponds to a special class of feed-forward neural networks, namely PUNNs, an alternative to the standard sigmoidal neural networks, which are based on multiplicative nodes instead of additive ones, which have been mainly used for regression problems [24–30].

In a supervised learning process, the components of the weights vector $\boldsymbol{\theta}=(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_{J-1})$ are estimated from the training data set D. To achieve the maximum likelihood for the estimation of $\boldsymbol{\theta}$, one can minimize the negative log-likelihood function. The estimation of the vector parameter $\hat{\boldsymbol{\theta}}$ is carried out by means of a hybrid procedure. So, the methodology is based on the combination of an evolutionary algorithm (global explorer) and a local optimization procedure (local exploiter) carried out by the standard maximum likelihood optimization method. The process of estimation of the basis function coefficients is structured in three steps and more specific details about it and the corresponding optimization procedure can be seen in [23].

It is important to remark some characteristics of the first step of the methodology, an evolutionary programming (EP) algorithm that find the weights and the number of PU functions for the MRPU and MRIPU models. The population-based evolutionary algorithm for the architectural design and estimation of real-coefficients was selected on the basis that crossover is not used due to its potential disadvantages in evolving artificial networks [40], although it has common points with other evolutionary algorithms reported in the literature [40–45]. The search begins with an initial population selected randomly. On each generation of the population is updated using a population-update algorithm. The population is subject to the operations of replication and parametric and structural mutation. More details about the specific characteristics of this evolutionary algorithm are reported elsewhere [28,29,46].

3. Experimental

Experimental data matrix was obtained by using the HS-MS data provided by a set of 63 drinking water samples spiked with individual standards of benzene, toluene or xylene as well as with binary or ternary mixtures of them at concentrations between 5 and 30 μg/l. Table 1 shows the composition of the resulting seven classes. The samples were usually prepared in duplicated as described elsewhere [47] and the total ion current profile provided by the MS detector operated in full scan mode with a range from m/z 50 to 110 at 10.3 scans/s was used as the analytical signal. The experimental design is based on the random distribution of the water samples of each class using 2/3 and 1/3 for the training and generalization sets, respectively (see Table 1).

The Levenberg–Marquardt algorithm was used to estimate the three-parameter Gaussian function associated with the total ion current profile provided by the MS detector. These parameters were defined as follows: $\hat{I}_m$ (maximum abundance), $\hat{t}_m$ (time corresponding to the maximum abundance) and $\hat{B}$(dispersion of the abundance values from I_m). In order to use these parameters as inputs to the assayed PUNNs, they were scaled over the range 0.1 to 0.9. Thus, the new scaled variables were expressed as follows: $\hat{I}_m^*$, $\hat{t}_m^*$ and $\hat{B}^*$. After optimizing the network models, estimations should be de-scaled according to the same procedure.

Multilogistic regression models were fitted to the data obtained by means of a multilogistic regression modelling procedure included in SPSS 12.0 for Windows [48]. The multilogistic regression was performed with a backward conditional method, which selects the most significant covariates. To measure the classifier's performance, the output was compared to the observed outcome of the seven classes, and the correctly classified rate (CCR) for training and generalization sets were obtained, CCR_T and CCR_G, respectively; the CCR being defined as follows:

$$\text{CCR} = \frac{1}{N}\sum_{n=1}^{N} I(C(\mathbf{x}_n) = \mathbf{y}_n) \tag{4}$$

where $I(\cdot)$ is the zero-one loss function. A good classifier tries to achieve the highest generalization CCR value in a given problem.

The parameters used in the EP algorithm were the same in the two product unit basis functions multilogistic regression models. We consider the same parameter values than those reported in [49] due to their robustness. The exponents w_{ji} and the coefficients β_j^l were initialized in the [−5,5] interval. The size of the population is N=1000 and the maximum number of hidden nodes is m=4. The number of nodes that can be added or removed in a structural mutation is within the [1,2] interval, whereas the number of connections that can be added or removed in a structural mutation is within the [1,6] interval. The number of runs of the EP algorithm was 30 with 200 generations for each run.

Results are compared using LDA, QDA, SVM and the C4.5 tree inducer algorithm for classification. Regarding the classical statistical algorithms, it is well known that if the input variables have a Gaussian distribution and we assume that the variance–covariance matrices are equal, then LDA produces the best Bayes error over the training set, and if the variance–covariance matrices are different the same property is satisfied for QDA. The C4.5 classification tree inducer algorithm is run with the standard options: the confidence threshold for pruning is 0.25, the minimum number of instances per leaf is 2. For pruning, both subtree replacement and subtree rising are considered. The SMO and J48 algorithms are a java implementation of the SVM and C4.5 methodologies, which are part of the Weka machine learning workbench [50] release 3.4.0, using the default parameter values.

4. Results and discussion

Direct sampling MS methods are based on the insertion of the analytes from a sample into a MS using a simple interface with minimal sample preparation and no prior chromatographic separation [51–53]. The recent development of a methodology based on the direct coupling of a headspace sampler with a mass spectrometry detector (HS-MS) has enabled a drastic reduction in analysis time increasing the sample throughput [54]. Generally, data can be obtained by using the mass spectrum that represents the sum of intensities of all the ions detected during the data-acquisition time. Afterwards, it is necessary to extract the information contained in the profile signal and convert it into useful information, which requires the use of chemometric approaches. To date, most applications of HS-MS have focused on qualitative analysis related to quality control in foods [55–58] and environmental pollution [59,60] by applying different classification techniques such as LDA, CA, SIMCA and KNN among others.

As stated above, the goal of this work was to evaluate the potential of a multilogistic regression model recently reported by us [23] in

Table 1
Composition of the training and generalization sample sets

	Training sample set, μg/l			Generalization sample set, μg/l		
	Benzene	Toluene	Xylene	Benzene	Toluene	Xylene
Class 1	5			5		
	10			10		
	15			30		
	15					
	30					
	30					
Class 2		5			5	
		10			30	
		10			30	
		15				
		15				
		30				
Class 3			5			10
			5			15
			10			30
			15			
			30			
			30			
Class 4	5	10		5	5	
	5	30		5	30	
	10	30		30	30	
	15	15				
	15	15				
	30	30				
Class 5	5		10	5		5
	5		30	5		30
	10		30	30		30
	15		15			
	15		15			
	30		30			
Class 6		5	5		10	30
		5	10		30	30
		5	30		30	30
		5	30			
		15	15			
		15	15			
Class 7	5	10	5	5	5	5
	5	10	5	5	30	5
	5	30	5	10	30	10
	15	15	15			
	15	15	15			
	30	30	30			

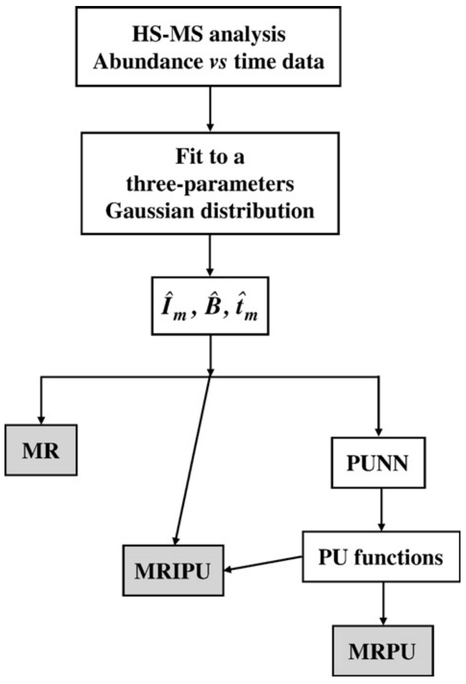

HS-MS analysis
Abundance *vs* time data
Fit to a three-parameters Gaussian distribution
$\hat{I}_m, \hat{B}, \hat{t}_m$
MR
PUNN
MRIPU
PU functions
MRPU

Fig. 1. Flow diagram of the whole chemometric protocol.

qualitative analysis. This methodology, which includes the nonlinear effects of the covariates, has only been applied to benchmark problems. To evaluate the response of the methodology in qualitative analysis, several strategies were merged in this work: 1) the selection of an analytical classification problem in which the classes involved provided analytical signals with a high degree of overlapping, such as the case of the HS-MS profiles obtained in the analysis of drinking waters contaminated by VOCs; 2) the use of a limited number of variables, such as those estimated by the Levenberg–Marquardt method in the form of a three-parameter Gaussian curve associated with the total ion current profile provided by the MS detector; and 3) the use of the PU basis functions as nonlinear terms for the multilogistic regression models. Fig. 1 shows a schematic diagram of the multilogistic regression approaches tested in this paper for the classification of assayed polluted drinking water samples.

4.1. Treatment of the HS-MS data

The volatile profiles provided by the HS-MS instrument corresponding to the seven classes of contaminated drinking waters under study are shown in Fig. 2A, in which a representative profile of each case is included. As can be seen, although the classes containing benzene and toluene give narrower bands (e.g. see peaks 1 and 4) and those containing xylene provide wider ones (e.g. see peaks 3 and 5), the profiles were very similar in general, which makes clear the high overlapping grade of the bands of the different classes. On the other hand, Fig. 2B shows the volatile profile of an un-contaminated and contaminated drinking water (sample 4 in Fig. 2A) in order to evaluate the contribution of the background signal. As can be seen, the signal provided by the analytes can be successfully differentiated from the background resulting in a net signal as shown in Fig. 2C.

The shape of the analyte volatile profile suggests that it should be modelled with a pre-determined function so that its definite parameters can be used as the initial covariates for the MR model, also obtaining the PU basis functions for the MRPU and MRIPU models. This situation provides simpler MR models as well as reduces PUNN complexity and learning time, which is of great practical interest. Taking into account that the resulting analyte volatile profile bands are practically symmetric, the three-parameter Gaussian function was found to be the best choice for modelling HS-MS data. As stated above, the equation corresponding to this function is defined by the following parameters: $\hat{I}_m$ (maximum abundance), $\hat{t}_m$ (time corresponding to the maximum abundance) and $\hat{B}$ (dispersion of the abundance values from $\hat{I}_m$). Fig. 2C shows the fit provided by the three-parameter Gaussian function on the volatile profile obtained in the HS-MS

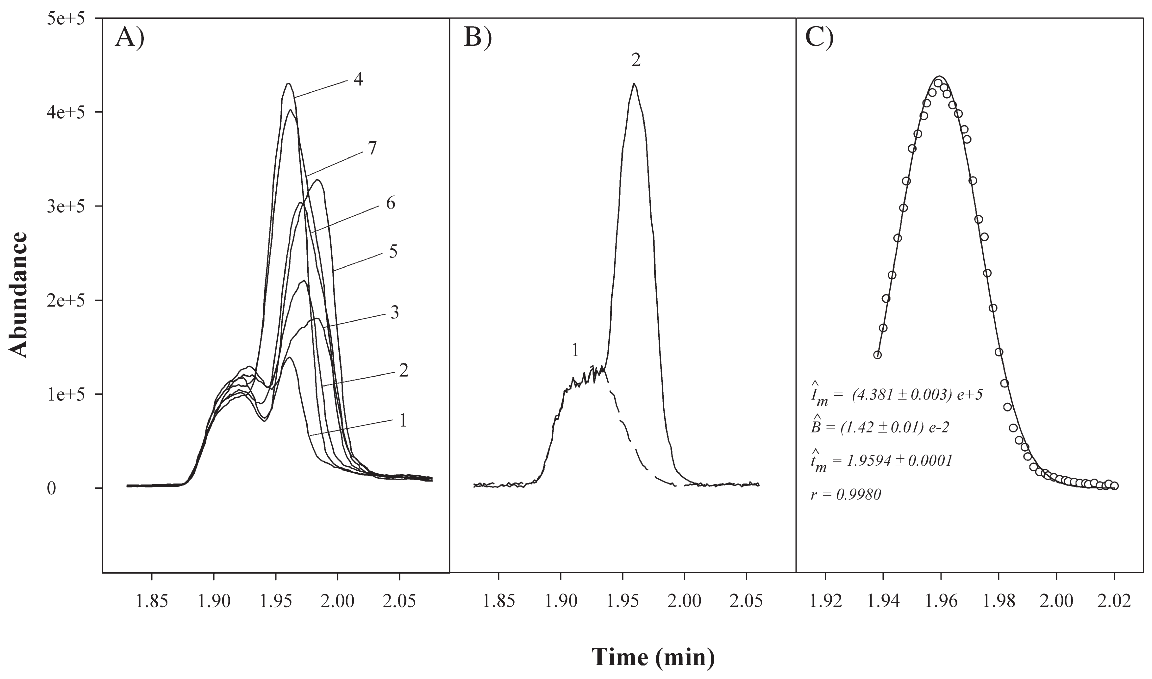


Fig. 2. A) Typical HS-MS profiles corresponding to drinking water samples containing: 1) 15 μg/l of benzene; 2) 15 μg/l of toluene; 3) 10 μg/l of xylene; 4) 10 μg/l of benzene and 30 μg/l of toluene; 5) 5 μg/l of benzene and 30 μg/l of xylene; 6) 15 μg/l of toluene and 15 μg/l of xylene; and 7) 15 μg/l of benzene, 15 μg/l of toluene and 15 μg/l of xylene. B) HS-MS profiles corresponding to: 1) un-contaminated and 2) polluted drinking water with a mixture containing 10 μg/l of benzene and 30 μg/l of toluene. C) HS-MS response fitted to a three-parameter Gaussian distribution. (o) Experimental data and (–) Gaussian curve.

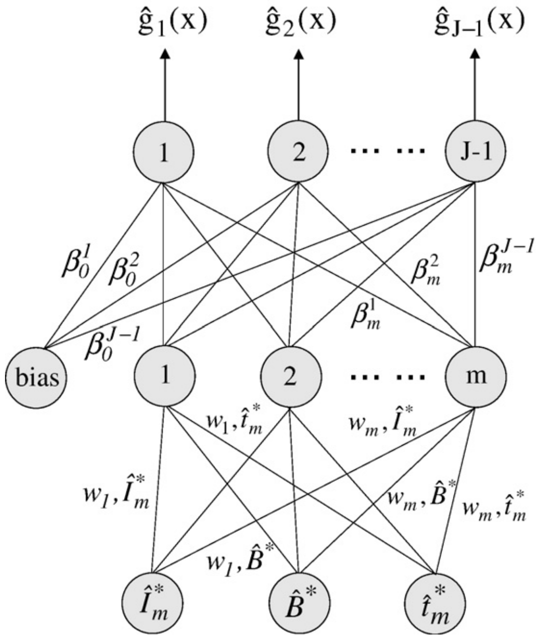


Fig. 3. Functional scheme of the neural network based on PU basis functions. $w_1,\ldots,w_m$ and $\beta_0,\ldots,\beta_m$ are the regression coefficients of the model. Other symbols are defined in the text.

analysis of the drinking water spiked with a binary mixture of benzene and toluene (sample 4 in Fig. 2A). Upon examining both curves and from the estimated statistical parameters, it can be concluded that the three-parameter Gaussian distribution is a fine tool for modelling this kind of HS-MS data.

4.2. Estimation of PU basis functions

Product unit basis functions covariates multilogistic regression models were designed exchanging the classical initial covariates for PU basis functions (MRPU), which are nonlinear themselves, and combining both PU basis functions and the initial covariates (MRIPU). Instead of selecting standard sigmoidal functions, which are based on additive functions, we selected PU basis functions taking into account that PUNNs have an increased information capacity and the ability to form higher-order combinations of inputs [24].

In order to estimate the PU basis functions, the classification ability of different PUNN models, (the general scheme of a PUNN model is shown in Fig. 3) was compared in terms of topology, number of connections, homogeneity (confidence interval) and CCR by using network models with three nodes in the input layer (the three-parameter of the Gaussian function), and six nodes in the output layer (one less than the number of classes as stated in Section 2); thus, 3:5:6 and 3:6:6 architectures were tested.

As shown in Table 2, the classification of the polluted drinking water samples using the best PUNN model, which included five PU basis functions in its hidden layer, is not very satisfactory, and therefore the use of another chemometric approaches such as MR models can be an interesting choice in order to improve the CCR_G values. These PU basis functions also provided useful information from a chemical point of view. In fact, the more relevant PU basis functions (see the discriminant functions, DFs, in Table 2) are $\hat{PU}_2$, $\hat{PU}_4$ and $\hat{PU}_5$ which depend on the initial variables $\hat{t}_m^*$and $\hat{B}^*$, that is, the time corresponding to the maximum abundance of the total ion current profile and the dispersion of the abundance values from it. From these results it is clear that the $\hat{I}_m^*$ values are not necessary for the classification of the contaminated water samples when using PUNNs.

4.3. Evaluation of the multilogistic regression models

As stated above, three multilogistic regression models were tested: MR, MRPU and MRIPU and their features are compared in Tables 3–5; in particular, Table 3 shows the DFs for each model, which depend on the following covariates: $\hat{I}_m^*$, $\hat{B}^*$ and $\hat{t}_m^*$ for the MR; $\hat{PU}_1$ to $\hat{PU}_5$ for MRPU; and $\hat{t}_m^*$, $\hat{PU}_1$ and $\hat{PU}_4$, for MRIPU model, and Table 4 shows the rate of the number of cases that were correctly classified (confusion matrix) for each model. Regarding classification ability (see Table 5), all models provided CCR_T values of 100%; however, the best value for the generalization set (CCR_G) was achieved by using the MRIPU model. Table 5 also shows the results given by other classification algorithms for comparison purposes, such as the classical LDA and QDA algorithms as well as SMO and J48 algorithms implemented for the SVM and C4.5 classification methods. These results justify the use of the MR models proposed in this work to solve the addressed analytical problem. In view of the CCR_T values in Table 5, it can be considered that there is overfitting in MR models; however, if such high percentages of good classification are not achieved in the training process, the CCR_G values are not higher than 66.7%.

By analyzing these results, some chemical interpretations can be drawn from the characteristics of the DFs and the confusion matrix. From the sign and value of the coefficients of the DFs (these functions are related to the discrimination power of one class with respect to the other classes) it can be derived that in general $\hat{B}^*$and $\hat{t}_m^*$ are the most significant covariates for the addressed classification problem. Regarding the proposed MRIPU model, the DFs only depend on the

Table 2
Accuracy, statistical results and PU basis and discriminant functions for the best model obtained by using evolutionary PUNNs (over 30 runs)

PUNN features							
	Connection	CCR_T			CCR_G		
Range topology	Mean ± 1.96 × SD	Mean ± 1.96 × SD	Best	Worst	Mean ± 1.96 × SD	Best	Worst
3:5:6–3:6:6	37.9 ± 8.8	87.9 ± 9.2	97.6	76.2	63.5 ± 12.7	71.4	52.4
PU basis functions							
$\hat{PU}_1$: $(\hat{B}^*)^{-0.40}$			$\hat{PU}_2$: $(\hat{B}^*)^{4.04}(\hat{t}_m^*)^{-1.06}$				$\hat{PU}_3$: $(\hat{I}_m)^{1.91}$
$\hat{PU}_4$: $(\hat{B}^*)^{-1.63}$ $(\hat{t}_m^*)^{2.47}$			$\hat{PU}_5$: $(\hat{B}^*)^{-3.05}$ $(\hat{t}_m^*)^{2.75}$				
Discriminant functions (DF)							
DF_1: $4.64 - 11.4\ \hat{PU}_1 - 2.44\ \hat{PU}_2 + 2.29\ \hat{PU}_3 - 3.61\ \hat{PU}_4 + 7.30\ \hat{PU}_5$							
DF_2: $-3.29 - 4.04\ \hat{PU}_2 + 4.09\ \hat{PU}_4 + 6.12\ \hat{PU}_5$							
DF_3: $-5.89 - 1.80\ \hat{PU}_2 + 12.9\ \hat{PU}_4 + 3.20\ \hat{PU}_5$							
DF_4: $-3.38 + 0.52\ \hat{PU}_1 - 9.25\ \hat{PU}_2 + 0.97\ \hat{PU}_5$							
DF_5: $-5.28 + 5.29\ \hat{PU}_1 - 1.28\ \hat{PU}_2 + 7.69\ \hat{PU}_4$							
DF_6: $-2.18 - 1.57\ \hat{PU}_2 + 6.15\ \hat{PU}_4 + 4.09\ \hat{PU}_5$							

Table 3
Discriminant functions (DF) for the MR, MRPU and MRIPU models

MR model

$f_l = \hat{\alpha}_0 + \hat{\alpha}_{l,\hat{I}^*_m}\hat{I}^*_m + \hat{\alpha}_{l,\hat{B}^*}\hat{B}^* + \hat{\alpha}_{l,\hat{t}^*_m}\hat{t}^*_m$ $l = 1,2,\ldots,6$

DF	$\hat{\alpha}_0$	$\hat{\alpha}_{l,\hat{I}^*_m}$	$\hat{\alpha}_{l,\hat{B}^*}$	$\hat{\alpha}_{l,\hat{t}^*_m}$
1	323.3	−65.5	−1074.6	875.9
2	223.1	−37.6	−1188.6	1188.2
3	5.6	210.9	−1269.2	1470.3
4	302.3	49.0	−929.9	661.9
5	57.9	44.9	−258.3	280.3
6	172.0	−41.5	−1041.6	1143.7

MRPU model

$f_1 = \hat{\beta}_0 + \hat{\beta}_{l,\hat{PU}_1}\hat{PU}_l + \hat{\beta}_{l,\hat{PU}_2}\hat{PU}_2 + \hat{\beta}_{l,\hat{PU}_3}\hat{PU}_3 + \hat{\beta}_{l,\hat{PU}_4}\hat{PU}_4 + \hat{\beta}_{l,\hat{PU}_5}\hat{PU}_5$ $l = 1,2,\ldots,6$

DF	$\hat{\beta}_0$	$\hat{\beta}_{l,\hat{PU}_1}$	$\hat{\beta}_{l,\hat{PU}_2}$	$\hat{\beta}_{l,\hat{PU}_3}$	$\hat{\beta}_{l,\hat{PU}_4}$	$\hat{\beta}_{l,\hat{PU}_5}$
1	67.1	−447.3	0.7	−78.9	783.1	−312.6
2	−283.2	−550.0	1.2	−116.9	252.4	543.5
3	129.3	−563.3	−0.4	521.1	1108.2	−625.9
4	109.1	−181.3	−0.1	93.0	415.1	−294.0
5	−78.8	−63.9	−3.9	71.6	−19.2	200.4
6	−896.7	−503.6	2.0	−157.0	−246.9	1695.4

MRIPU model

$f_1 = \hat{\alpha}'_{l,\hat{t}^*_m}\hat{t}^*_m + \hat{\beta}'_{l,\hat{PU}_1}\hat{PU}_1 + \hat{\beta}'_{l,\hat{PU}_4}\hat{PU}_4$ $l = 1,2,\ldots,6$

DF	$\hat{\alpha}_{l,\hat{t}^*_m}$	$\hat{\beta}_{l',\hat{PU}_1}$	$\hat{\beta}_{l,\hat{PU}_4}$
1	−394.0	−970.9	1359.2
2	−1238.6	−1235.2	2522.6
3	−2615.3	−1115.7	4164.4
4	666.8	−386.9	−445.4
5	−632.4	−140.2	995.0
6	−2710.6	−864.5	4149.9

covariates $\hat{t}^*_m$ and $\hat{B}^*$ involved in three terms: one linear in $\hat{t}^*_m$ and two nonlinear in $\hat{PU}_1$ and $\hat{PU}_4$, such as $(\hat{B}^*)^{0.40}$ and $(\hat{B}^*)^{-1.63}(\hat{t}^*)^{2.47}$, respectively. Taking into account the sign and value of the respective coefficients of these DFs, it can be inferred that the $\hat{PU}_4$ covariate is the one that exerts a more significant effect in the classification process, except for the class 4 where $\hat{t}^*_m$ is the most relevant. In other words, the interaction between the initial covariates $\hat{B}^*$ and $\hat{t}^*_m$ is the key for the classification of the polluted drinking waters because both are involved in the most relevant term of the DFs and $\hat{B}^*$contributed in a greater extend due to its negative exponent in the PU basis function $\hat{PU}_4 = (\hat{B}^*)^{-1.63}(\hat{t}^*_m)^{2.47}$.

As stated above, Table 4 shows the confusion matrix obtained in the classification of drinking waters by using the three models; it is clear the better performance of the proposed MRIPU model for the classification of these samples. Despite the better results provided by the proposed model, only one sample of the class 2 is correctly classified (the other models also misclassifying these samples). This behaviour can be ascribed to the similar chemical composition of the two unclassified samples in the class 2 (30 μg/l toluene) with other two included in the class 4 (30 μg/l toluene+5 and 30 μg/l benzene, respectively). Taking into account that benzene provides narrower total ion current profile than toluene, its contribution to $\hat{t}^*_m$ and $\hat{B}^*$ in the mixture is less significant than the contribution of toluene, and therefore a slight experimental error in the preparation and/or HS-MS detection of these samples can originate this problem in the correct classification of samples of the class 2.

5. Conclusions

As it has been shown throughout this study, a multilogistic regression model recently reported by us, composed by original covariates and their nonlinear transformations designed by using evolutionary product unit neural networks has demonstrated to be a powerful tool for multi-class pattern recognition in qualitative analysis. Several chemical and chemometric conclusions can be inferred from the results: (i) the improving of the standard MR model by considering the nonlinear effects of the covariates. The ensuing hybrid MR model provided better accurate results for the classification of polluted drinking waters than the other MR alternatives tested and those achieved by the classical discriminant analysis

Table 4
Rate of the number of cases in the generalization set that were classified correctly for the models assayed: (MR, MRPU, MRIPU)

Class predicted/ target	$l=1$	$l=2$	$l=3$	$l=4$	$l=5$	$l=6$	$l=7$
$l=1$	(3, 3, 3)	–	–	–	–	–	–
$l=2$	–	(1, 1, 1)	–	(2, 2, 2)	–	–	–
$l=3$	–	–	(2, 1, 2)	–	–	(1, 2, 1)	–
$l=4$	(1, 0, 0)	–	–	(2, 3, 3)	–	–	–
$l=5$	–	–	–	–	(2, 2, 2)	(1, 1, 1)	–
$l=6$	–	–	(2, 3, 0)	–	–	(1, 0, 3)	–
$l=7$	–	–	–	(0, 0, 1)	(1, 1, 0)	–	(2, 2, 2)

Table 5
Comparison of the quality achieved for the classification of polluted drinking waters using discriminant analysis (LDA and QDA), support vector machine (SVM), model tree algorithm (C4.5), standard multilogistic regression (MR) and multilogistic regression using product unit basis functions methodologies (MRPU and MRIPU)

Algorithm	CCR_T	CCR_G
LDA	90.5	66.7
QDA	100.0	66.7
SVM	86.3	61.9
C4.5	85.7	66.7
MR	100.0	61.9
MRPU	100.0	57.1
MRIPU	100.0	76.2

methodologies such as LDA and QDA and other common classification techniques such as SVM and C4.5. (ii) The good results achieved in spite of the complexity of the analytical problem selected for the evaluation of the models: the classification of seven drinking waters contaminated with benzene, toluene, xylene or their mixtures at μg/l levels using highly overlapping HS-MS data and with a minimum number of patterns in the generalization test. (iii) The simplification of the models by using as initial covariates the three-parameter Gaussian curve associated with the total ion current profile provided by the MS detector, and (iv) the use of an evolutionary algorithm for obtaining the proposed hybrid MR model. Thus, relationships can be inferred from the initial covariates or their PU transformations on the classification process in order to establish the relative influence of each one.

Acknowledgments

The authors gratefully acknowledge the subsidy provided by the Spanish Inter-Ministerial Commission of Science and Technology of the Ministry of Education and Science under the CTQ2007-63962 (Department of Analytical Chemistry, University of Cordoba) and TIN2005-08386-C05-02 (Department of Computer Science, University of Cordoba) Projects. FEDER also provided additional funding. The research of P.A. Gutiérrez has been subsidised by the FPU Predoctoral Program (Spanish Ministry of Education and Science) grant reference AP2006-01746.

References

[1] M. Valcárcel, S. Cárdenas (Eds.), Modern Qualitative Analysis, Trends Anal. Chem., vol. 24, 2005, issue 6.
[2] S.L.R. Ellison, W.A. Hardcastle, Expression of Uncertainty in Qualitative Testing, LGC report LGC/VAM/2002/021, LGC, Teddington, Middlesex, UK, 2002 Available from: www.vam.org.uk.
[3] R.G. Brereton, Chemometrics: Data Analysis for the Laboratory and Chemical Plant, John Wiley & Sons, New York, 2003.
[4] B.K. Lavine, Anal. Chem. 72 (2000) 91R–97R.
[5] G. Dreyfus, Neural Networks: Methodology and Applications, Springer, Heidelberg, 2005.
[6] H.F. Wang, D.Z. Chen, Y.Q. Chen, Chemom. Intell. Lab. Syst. 70 (2004) 23–31.
[7] J.A. Nelder, R.W.M. Wedderburn, J. Roy, Stat. Soc. A 135 (1972) 370–384.
[8] P.Y. McCullagh, J.A. Nelder, Generalized Linear Models, 2nd edn. Chapman & Hall, New York, 1989.
[9] A.M. Aguilera, M. Escabias, M.J. Valderrama, Comput. Stat. Data Anal. 50 (2006) 1905–1924.
[10] S. Dreiseitl, L. Ohno-Machado, J. Biomed. Inform. 35 (2002) 352–359.
[11] D. Hosmer, S. Lemeshow, Applied Logistic Regression, 2nd ed. Willey, New York, 2000.
[12] C.F. Qiu, S. Saito, M. Sakai, K. Ogawa, K. Nagata, M.A. Williams, Clin. Biochem. 39 (2006) 1016–1021.
[13] A. Berdeli, G. Emingil, A. Gurkan, G. Atilla, T. Kose, Clin. Biochem. 39 (2006) 357–362.
[14] Y. Hasui, Y. Hamanaka, N. Okayama, Y. Suehiro, F. Shinozaki, Y. Ueyama, Y. Hinoda, J. Clin. Lab. Anal. 20 (2006) 47–51.
[15] B.L. Mitchell, Y. Yasui, J.W. Lampe, P.R. Gafken, P.D. Lampe, Proteomics 5 (2005) 2238–2246.
[16] M. Rasouli, A. Okhovatian, A. Enderami, Clin. Chem. Lab. Med. 43 (2005) 913–918.
[17] J.N. Li, Z. Zhang, J. Rosenzweig, Y.Y. Wang, D.W. Chan, Clin. Chem. 48 (2002) 1296–1304.
[18] E.A. Hernandez-Caraballo, F. Rivas, A.G. Pérez, L.M. Marco-Parra, Anal. Chim. Acta 533 (2005) 161–168.
[19] A.S. Furberg, T. Sandanger, I. Thune, I.C. Burkow, E.J. Lund, Environ. Monit. 4 (2002) 175–181.
[20] O.M. Hernández, J.M.G. Fraga, A.I. Jiménez, F. Jiménez, J.J. Arias, Food Chem. 93 (2005) 449–458.
[21] M. Blanco, J. Coello, H. Iturriaga, S. Maspoch, C. Pérez-Maseda, Anal. Chim. Acta 407 (2000) 247–254.
[22] E. Bertrán, M. Blanco, J. Coello, H. Iturriaga, S. Maspoch, I. Montoliu, J. Near Infrared Spectrosc. 8 (2000) 45–52.
[23] C. Hervás-Martínez, F.J. Martínez-Estudillo, M. Carbonero-Ruz, Neural Networks, , 2008 accepted, available online 20 January 2008.
[24] R. Durbin, D. Rumelhart, Neural Comput. 1 (1989) 133–142.
[25] D.J. Janson, J.F. Frenzel, IEEE Expert 8 (1993) 26–33.
[26] L.R. Leerink, C.L. Giles, B.G. Horne, M.A. Jabri, Adv. Neural Inf. Process. Syst. 7 (1995) 537–544.
[27] A.P. Engelbrecht, A. Ismail, International Conference on Neural Networks 2002, Proceedings of the Conference, Honolulu, Hawai, 2002.
[28] A.C. Martínez-Estudillo, C. Hervás-Martínez, F.J. Martínez-Estudillo, N. García-Pedrajas, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 36 (2006) 534–545.
[29] A.C. Martínez-Estudillo, F.J. Martínez-Estudillo, C. Hervás-Martínez, N. García-Pedrajas, Neural Netw. 19 (2006) 477–486.
[30] M. Schmitt, Neural Comput. 14 (2001) 241–301.
[31] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge, 2000.
[32] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[33] J.R. Quinlan, in: A. Adams, L. Sterling (Eds.), 5th Australian Joint Conference on Artificial Intelligence. Proceedings of the Conference, Singapore, World Scientific, 1995, pp. 343–348.
[34] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
[35] C. Hervás-Martínez, R. Toledo, M. Silva, J. Chem. Inf. Comput. Sci. 41 (2001) 1083–1092.
[36] C. Hervás-Martínez, M. Silva, J.M. Serrano, E. Orejuela, J. Chem. Inf. Comput. Sci. 44 (2004) 1576–1584.
[37] C. Hervás-Martínez, A.C. Martínez-Estudillo, M. Silva, J.M. Serrano, J. Chem. Inf. Model. 45 (2005) 894–903.
[38] C. Hervás-Martínez, M. Silva, Chemom. Intell. Lab. Syst. 85 (2007) 232–242.
[39] C. Hervás-Martínez, P.A. Gutiérrez, M. Silva, J.M. Serrano, J. Chemom. 21 (2007) 567–577.
[40] P.J. Angeline, G.M. Saunders, J.B. Pollack, IEEE Trans. Neural Netw. 5 (1994) 54–65.
[41] G.F. Miller, P.M. Todd, S.U. Hedge, 3er International Conference on Genetic Algorithms and Their Applications, Proceedings of the Conference, San Mateo, CA, 1989.
[42] X. Yao, Y. Liu, IEEE Trans. Neural Netw. 8 (1997) 694–713.
[43] D.B. Fogel, International Conference on Neural Networks, Proceedings of the Conference, San Francisco, CA, 1993.
[44] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, Neural Netw. 15 (2002) 1255–1274.
[45] X. Yao, Proceeding of the IEEE 9 (1999) 1423–1447.
[46] C. Hervás-Martínez, F.J. Martínez-Estudillo, Pattern Recogn. 40 (2007) 52–64.
[47] A. Serrano, M. Gallego, M. Silva, Anal. Chem. 79 (2007) 2997–3002.
[48] SPSS, Advanced Models. Copyright 12.0 SPSS Inc., 2003, Chicago, IL.
[49] F.J. Martínez-Estudillo, C. Hervás-Martínez, P.A. Gutiérrez-Peña, A.C. Martínez-Estudillo, S. Ventura, Evolutionary Product-Unit Neural Networks for Classification in Intelligent Data and Automated Learning (IDEAL 2006), Springer, Berlin, 2006, pp. 1320–1328.
[50] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann, San Francisco, USA, 2000.
[51] M.B. Wise, M.R. Guerin, Anal. Chem. 69 (1997) 26A–32A.
[52] S. Bauer, Trends Anal. Chem. 14 (1995) 202–213.
[53] R.T. Short, S.K. Toler, G.P.G. Kibelka, D.T. Rueda Roa, R.J. Bell, R.H. Byrne, Trends Anal. Chem. 25 (2006) 637–646.
[54] J.L. Pérez-Pavón, M. del Nogal-Sánchez, C. García-Pinto, M.E. Fernández-Laespada, B. Moreno-Cordero, A. Guerrero-Peña, Trends Anal. Chem. 25 (2006) 257–266.
[55] F. Peña, S. Cárdenas, M. Gallego, M. Valcárcel, J. Chromatogr. A 1074 (2005) 215–221.
[56] F. Peña, S. Cárdenas, M. Gallego, M. Valcárcel, Anal. Chim. Acta 526 (2004) 77–82.
[57] M.P. Martí, J. Pino, R. Boque, O. Busto, J. Guasch, Anal. Bioanal. Chem. 382 (2005) 440–443.
[58] M.P. Martí, O. Busto, J. Guasch, J. Chromatogr. A 1057 (2004) 211–217.
[59] M. del Nogal-Sánchez, J.L. Pérez-Pavón, C. García-Pinto, M.E. Fernández-Laespada, B. Moreno-Cordero, Anal. Bioanal. Chem. 382 (2005) 372–380.
[60] A. Serrano, M. .Gallego, J. Chromatogr. A 1045 (2004) 181–188.

3.2. Regresión Logística utilizando Funciones de Base de tipo Radial - *Logistic Regression by means of Evolutionary Radial Basis Function Neural Networks*

Las publicaciones asociadas a esta parte son:

- P.A. Gutiérrez, C. Hervás-Martínez, F.J. Martínez-Estudillo. Logistic Regression by means of Evolutionary Radial Basis Function Neural Networks. Enviado a IEEE Transactions on Neural Networks.
    - Estado: Enviado.
    - Índice de Impacto (JCR 2007): 2.769.
    - Área de conocimiento: *Computer Science, Artificial Intelligence*. Ranking 7/93. Primer Cuartil.
    - Área de conocimiento: *Computer Science, Hardware & Architecture*. Ranking 4/45. Primer Cuartil.
    - Área de conocimiento: *Computer Science, Theory & Methods*. Ranking 3/79. Primer Cuartil.
    - Área de conocimiento: *Engineering, Electrical & Electronic*. Ranking 7/227. Primer Cuartil.

Logistic Regression by means of Evolutionary Radial Basis Function Neural Networks

Journal:	*Transactions on Neural Networks*
Manuscript ID:	TNN-2009-P-1139
Manuscript Type:	Paper
Date Submitted by the Author:	27-Jan-2009
Complete List of Authors:	Gutiérrez, Pedro Antonio; University of Cordoba, Department of Computer Science and Numerical Analysis Hervas, Cesar; University of Cordoba, Department of Computer Science and Numerical Analysis Martinez, Francisco J.; Faculty of Business Administration, Department of Management and Quantitative Methods
Keywords:	radial basis function neural networks, logistic regression, classification, evolutionary algorithms, artificial neural networks, evolutionary programming

Logistic Regression by means of Evolutionary Radial Basis Function Neural Networks

Pedro Antonio Gutiérrez, *Student Member, IEEE,* César Hervás-Martínez, *Member, IEEE,* and Francisco J. Martínez-Estudillo, *Member, IEEE,*

Abstract

This paper proposes a hybrid multilogistic methodology, named Logistic regression using Initial and Radial Basis Function covariates (LIRBF). The process for obtaining the coefficients is carried out in three steps. First, an Evolutionary Programming (EP) algorithm is applied, aimed to produce a RBF Neural Network (RBFNN) with a reduced number of RBF transformations and the simplest structure possible. Then, the initial covariates' space is transformed by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the last generation. Finally, a maximum likelihood optimization method determines the coefficients associated with a multilogistic regression model built on this augmented covariate space. In this final step, two different multilogistic regression algorithms are applied, one that considers all initial and RBF covariates (MultiLogistic Initial-RBF regression, MLIRBF) and another one that incrementally constructs the model and applies cross-validation, resulting in an automatic covariate selection (SimpleLogistic Initial-RBF regression, SLIRBF). The methodology proposed is tested using eighteen benchmark classification problems from well-known machine learning problems and two real agronomical problems. The results are compared to the corresponding multilogistic regression methods applied over the initial covariate space, to the RBFNNs obtained by the EP algorithm (RBFEP) and to other competitive machine learning methods. The SLIRBF models are found to be better than the corresponding multilogistic regression methods and the RBFEP method for almost all datasets, obtaining the highest mean accuracy rank when compared to the rest of methods in all datasets.

Index Terms

Radial basis function neural networks, logistic regression, classification, evolutionary algorithms, artificial neural networks, evolutionary programming.

This work has been partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). The research of P.A. Gutiérrez has been funded by the FPU Predoctoral Program (Spanish Ministry of Education and Science), grant reference AP2006-01746.

P.A. Gutiérrez and C. Hervás-Martínez are with the Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, C2 building, 14004 - Córdoba, Spain, e-mail: i02gupep@uco.es, chervas@uco.es.

F.J. Martínez-Estudillo is with the Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14004 - Córdoba, Spain, e-mail: fjmestud@etea.com.

The authors would like to thank to the Precision Farming and Remote Sensing Unit of the Department of Crop Protection, Institute of Sustainable Agriculture (CSIC, Spain), for providing us the datasets corresponding to the real agronomical problem of Section V-B.

I. Introduction

The pace at which pattern classification algorithms have been successfully applied to new problems has increased exponentially over the past years [1]. Examples of application come from all branches of science, technology and medicine (e.g. medical diagnosis, handwritten character recognition, dynamic signature verification or satellite image analysis), and in many regards this is the most important statistical problem around [2]. A taxonomy of pattern classifiers (proposed by Lippmann [3]) considers five groups of classifiers: probability density functions, *global* posterior probability classifiers (based on computing elements which have high nonzero outputs over a large region of the input space), *local* posterior probability classifiers (based on computing elements which have high nonzero outputs over only a localized region of the input space), nearest neighbour methods and rule forming approaches.

The traditional statistical approach to pattern recognition is a natural application of Bayesian decision theory and the linear logistic regression model is one of its main representatives. As suggested by Hastie and Tibshirani [2], an obvious way to generalize the linear logistic regression model is to replace the linear predictors by a nonparametric version of them, such as an arbitrary regression surface, a more structured high dimensional regression surface (estimated using procedures as Friedman's MARS [4]) or even more structured non-parametric models such as an additive model of basis functions.

This paper is aimed to develop this last idea, presenting a competitive study in multi-class neural learning which combines different statistical and soft computing elements such as multilogistic regression, Radial Basis Function Neural Networks (RBFNNs) and Evolutionary Algorithms (EAs). We extend the ideas introduced in a recently proposed combination of neural networks and logistic regression [5], [6]. This proposal was based on the hybridization of a linear multilogistic regression model and a nonlinear Product-Unit Neural Network model for binary and multi-class classification problems. Product Units, which are *global* approximators, were combined with the initial covariates, and a multilogistic regression model was built on both. The methodology allowed the identification of possible strong interactions that may exist between the covariates which define the classification problem.

In this work, a new aspect of this methodology is presented, since we combine a linear model with a RBFNN nonlinear model and then we estimate the coefficients using logistic regression. The approach, named Logistic regression using Initial and Radial Basis Function covariates (LIRBF), consists of a multilogistic regression model built on the combination of the initial covariates and the RBFs of a RBFNN, which are *local* approximators. RBFNNs, as an alternative to multilayer perceptrons, have been found to be very helpful to many engineering problems since: (1) they are universal approximators [7]; (2) they have more compact topology than other neural networks [8]; and (3) their learning speed is fast because of their locally tuned neurons [9].

Although logistic regression is a simple and useful procedure, it poses problems when applied to real classification problems, where we cannot frequently make the stringent assumption of additive and purely linear effects of the covariates [10]. In this way, our technique overcomes these difficulties by augmenting the input vector with new RBF variables. From the opposite point of view, adding linear terms to a RBFNN yields simpler and easier to interpret models. Specifically, the linear terms reduce the variance associated with the overall modeling procedure and the likelihood of ending up with unnecessary RBFs. Finally, if a covariate only appears linearly in the final model, then the interpretation of this covariate if easier than the interpretation of a covariate which appears in a RBF transformation.

Logistic regression models are usually fit by maximum likelihood, where the Newton-Raphson algorithm is the traditional way to estimate the maximum a posteriori parameters. Usually, the algorithm converges since the log-likelihood is concave. However, in our approach, the nonlinearity of the Gaussian RBFs with respect to the centres and radii implies that the corresponding Hessian matrix is generally indefinite and the likelihood could have local maxima. These reasons justify, in our opinion, the use of an alternative heuristic procedure as an EA to estimate the parameters of the model.

The estimation of the coefficients is carried out in three steps. In a first step, an Evolutionary Programming (EP) algorithm determines the number of Gaussian RBFs in the model and their corresponding centres and radii. The algorithm is aimed to produce a reduced number of RBF transformations with the simplest structure possible, i.e. trying to select the most important input variables for the estimation of the RBFs. This step can be seen as a heuristic search in the coefficients' model space. Once the basis functions have been determined by the EP algorithm, we consider a transformation of the covariate space by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the last generation. The final model is linear in the set of variables formed by the RBFs and the initial covariates. Now, the Hessian matrix is definite and fitting proceeds with a maximum likelihood optimization method. In this final step, two different multilogistic regression algorithms are applied: (1) MultiLogistic, which considers all initial and RBF covariates and (2) SimpleLogistic, which incrementally constructs the model and applies cross-validation, resulting in an automatic covariate selection. This results in two different models: MultiLogistic Initial-RBF regression (MLIRBF) and SimpleLogistic Initial-RBF regression (SLIRBF).

We evaluate the performance of our methodology on eighteen datasets taken from the UCI repository [11] and two datasets corresponding to a real agronomical problem of precision farming. The results are compared to the corresponding multilogistic regression methods applied over the initial input space (SimpleLogistic, SLogistic, and MultiLogistic, MLogistic), to the RBFNNs obtained by the EP algorithm (RBFEP) and to other competitive machine learning techniques. The SLIRBF method is found to be better than pure SLogistic and the RBFEP method in almost all datasets, and obtains the significantly highest mean accuracy rank when compared to the rest of methods in all datasets.

This paper is organized as follows: a brief analysis of some works related with the models proposed is given in Section II; Section III is devoted to a description of the LIRBF models; Section IV describes the LIRBF learning algorithm; Section V explains the experiments carried out; and finally, Section VI summarizes the conclusions of our work.

II. Related Works

In this section, we establish a brief overview of the mainly related methods to the methodology proposed in this paper: RBFNNs, Support Vector Machines (SVM) and Multilogistic Regression Linear Product-Unit (MRLPU) neural networks.

RBFNNs can be considered a local approximation procedure, and the improvement in both its approximation ability as well as in the construction of its architecture has been note-worthy [12]. RBFNNs have been used in the most varied domains, from function approximation to pattern classification, time series prediction, data mining, signals processing, and nonlinear system modelling and control [13]. RBFNNs use, in general, hyper-ellipsoids to split the pattern space. This is different from Multilayer Perceptrons which build their classifications on hyper-planes, defined by a weighted sum.

In RBFNNs, the main problem is how to determine the hidden centres' number and their locations. If all the training samples are selected as hidden centres, the generalization capability of the network will become very poor so that many noised or deformed samples will be not able to be recognized, although the network is guaranteed to converge to some satisfying solution. Alternatively, there are many approaches to determine the hidden centres. For instance, the number and position of the RBFs may be fixed and defined a priori [14]; they may be determined by input clustering (k-means clustering, fuzzy k-means clustering, hierarchical clustering and Self Organizing Map neural networks) [15] and input-output clustering [16]. An interesting alternative is to evolve RBFNNs using Evolutionary Algorithms (EAs). In [17] a very complete state of the art of the different approaches and characteristics of a wide range of EA and RBFNN combinations is given. For example, RBFNN design has been approached by using evolutionary clustering techniques with local search [18], hybrid algorithms [19], [20], multi-objective algorithms [21], or by evolving only the basis functions [22], [23] or space-filling curves to generate the RBF centres [24].

From a structural point of view, RBFNNs and the LIRBF models proposed in this paper are closely related to direct kernel methods [25] and Support Vector Machines (SVM) with Gaussian kernel functions [26]. The SVM works well in two-class classification, that is, $y \in \{-1, 1\}$, but its appropriate extension to the multi-class case is still an ongoing research issue [27], [28]. Another property of the SVM is that it only estimates $\mathrm{sign}(p(\mathbf{x}) - 1/2)$ [29], where $p(\mathbf{x})$ is the conditional probability that $\mathbf{x}$ belongs to the corresponding class. However, the probability $p(\mathbf{x})$ is often of interest itself [30]. The LIRBF models proposed in this paper are naturally defined for the multi-class case and provide an estimation of $p(\mathbf{x})$, since they are based on the multilogistic regression model.

The SVM is related to the regularized function estimation in the Reproducing Kernel Hilbert Spaces (RKHS) [30]. In this way, the solution to the minimization problem is given by a linear combination of Mercer kernels [31] evaluated over each of the training patterns. It often happens that a sizeable fraction of the coefficients can be zero (the rest of points being the support points). An extension of the SVM in order to offer a natural estimate of $p(\mathbf{x})$ and to be generalized to the multi-class case is the Kernel Logistic Regression (KLR) [32], i.e. replacing the loss function of the SVM by the Negative Log-Likelihood (NLL) function. This functional model is somewhat equivalent to the

LIRBF structure, although, MLIRBF and SLIRBF also considers a linear term in the input covariates. Furthermore, because KLR compromises the hinge loss function of the SVM, it no longer has the support points property; in other words, all the training patterns are included in the solution. This causes the number of coefficients of these models to be higher than that of the proposed LIRBF models.

The predictor function is typically linear for logistic regression models. However, Hastie and Tibshirani [2] suggested the generalization of logistic regression models by using additive basis function structures. From this point of view, a first proposal of a combination of neural networks and logistic regression is given in two recent works [5], [6]. The model is based on the hybridization of a linear multilogistic regression model and a nonlinear Product-Unit Neural Network (PUNN) model for binary and multi-class classification problems. This methodology (Multilogistic Regression Linear Product-Unit neural networks, MRLPU) allows the generation of hybrid linear/nonlinear classification surfaces and the identification of possible strong interactions that may exist between the covariates which define the classification problem. The main differences of these works and the present paper are the following:

- First of all, in this paper we are using RBFs (which are *local* approximators) for the nonlinear part of the model, while the MRLPU method is based on PUs (which are *global* approximators).
- The EA presented in this paper is based on the determination of the RBFNN models, where the population initialization (radii and centres of the Gaussian functions) is more complex than that of the PUNN models.
- The SimpleLogistic algorithm used in this paper (see Section IV-A) is more advanced than the Iteratively Reweighted Least Squares method used for MRLPU, and performs an automatic structural simplification of the model. This results in more robust maximum likelihood estimations and avoids over-fitting.
- A more exhaustive ten fold experimental design with ten repetitions per each fold has been performed, since the whole process has been automated.

III. LIRBF MODELS

In the classification problem, measurements x_i, $i = 1, 2, ..., k$, are taken on a single individual (or object), and the individuals are to be classified into one of J classes on the basis of these measurements. It is assumed that J is finite, and the measurements x_i are random observations from these classes. Let $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, ..., N\}$ be a training dataset, where $\mathbf{x}_n = (x_{1n}, ..., x_{kn})$ is the vector of measurements taking values in $\mathbf{\Omega} \subset \mathbb{R}^k$, and $\mathbf{y}_n$ is the class level of the n-th individual. The common technique of representing the class levels using a "1-of-J" encoding vector is adopted, $\mathbf{y} = \left(y^{(1)}, y^{(2)}, ..., y^{(J)}\right)$, such as $y^{(l)} = 1$ if $\mathbf{x}$ corresponds to an example belonging to class l and $y^{(l)} = 0$ otherwise. Based on the training samples, we wish to find a decision function $F : \mathbf{\Omega} \to \{1, 2, ..., J\}$ for classifying the individuals. In other words, F provides a partition, say $D_1, D_2, ..., D_J$, of $\mathbf{\Omega}$, where D_l corresponds to the l-th class, $l = 1, 2, ..., J$, and measurements belonging to D_l will be classified as coming from the l-th class. A misclassification occurs when the decision rule F assigns an individual (based on the measurement vector) to a class j when it is actually coming from a class $l \neq j$.

To evaluate the performance of the classifiers the correct classification rate (CCR or C) is defined by

$$C = \frac{1}{N} \sum_{n=1}^{N} I(F(\mathbf{x}_n) = \mathbf{y}_n), \tag{1}$$

where $I(\bullet)$ is the zero-one loss function. A good classifier tries to achieve the highest possible C for a given problem. It is usually assumed that the training data are independent and identically distributed samples from an unknown probability distribution. Suppose that the conditional probability that $\mathbf{x}$ belongs to class l verifies: $p\left(y^{(l)} = 1 \middle| \mathbf{x}\right) > 0$, $l = 1, 2, ..., J$, $\mathbf{x} \in \mathbf{\Omega}$, and sets the function:

$$\log \frac{p\left(y^{(l)} = 1 \middle| \mathbf{x}\right)}{p\left(y^{(J)} = 1 \middle| \mathbf{x}\right)} = f_l(\mathbf{x}, \boldsymbol{\theta}_l),$$

where $\boldsymbol{\theta}_l$ is the weight vector corresponding to class l, the right-hand side of the equation is the predictor function, and $f_J(\mathbf{x}, \boldsymbol{\theta}_J) = 0$. In this way, the probability for one of the classes (the last one, in our case) does not need be estimated. Under a multinomial logistic regression, the probability that $\mathbf{x}$ belongs to class l is then given by:

$$p\left(y^{(l)} = 1 \middle| \mathbf{x}, \boldsymbol{\theta}\right) = \frac{\exp f_l\left(\mathbf{x}, \boldsymbol{\theta}_l\right)}{\sum_{j=1}^{J} \exp f_j\left(\mathbf{x}, \boldsymbol{\theta}_j\right)},\ l = 1, 2, ..., J,$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{J-1})$.

The classification rule coincides with the optimal Bayes' rule. In other words, an individual should be assigned to the class which has the maximum probability, given the measurement vector $\mathbf{x}$:

$$F(\mathbf{x}) = \hat{l}, \text{where } \hat{l} = \arg\max_{l} p\left(y^{(l)} = 1 \middle| \mathbf{x}, \boldsymbol{\theta}\right),\ l = 1, ..., J.$$

The predictor function of our logistic regression model proposal is based on the combination of the standard linear model on the initial covariates and a nonlinear model constructed with RBF transformed covariates, which captures well defined locations in the covariate space. The general expression of the predictor function is given by:

$$f_l(\mathbf{x}, \boldsymbol{\theta}_l) = \alpha_0^l + \sum_{i=1}^{k} \alpha_i^l x_i + \sum_{j=1}^{m} \beta_j^l \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{r_j^2}\right), \tag{2}$$

where $l = 1, 2, ..., J-1$, $\boldsymbol{\theta}_l = (\boldsymbol{\alpha}^l, \boldsymbol{\beta}^l, \mathbf{W})$ is the vector of parameters for each discriminant function, $\boldsymbol{\alpha}^l = (\alpha_0^l, \alpha_1^l, ..., \alpha_k^l)$ and $\boldsymbol{\beta}^l = \left(\beta_1^l, ..., \beta_m^l\right)$ are the coefficients of the multilogistic regression model, $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ...,$ $\mathbf{w}_m)$ are the parameters of the RBF nonlinear transformations, $\mathbf{w}_j = (\mathbf{c}_j, r_j)$, $\mathbf{c}_j = (c_{j1}, c_{j2}, ..., c_{jk})$ is the centre or average of the j-th Gaussian RBF transformation, r_j is the corresponding radius or standard deviation and $c_{ji}, r_j \in \mathbb{R}$.

IV. LIRBF LEARNING ALGORITHM

In the supervised learning context, the components of the weight vectors $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{J-1})$ are estimated from the training dataset D. To perform the maximum likelihood estimation of $\boldsymbol{\theta}$, one can minimize the negative log-likelihood function:

$$\begin{aligned} L(\boldsymbol{\theta}) &= -\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \log p\left(\mathbf{y}_n \middle| \mathbf{x}_n, \boldsymbol{\theta}\right) = \\ &= \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \Big[-\textstyle\sum_{l=1}^{J-1} y_n^{(l)} f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) + \\ &\quad + \log \textstyle\sum_{l=1}^{J-1} \exp f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) \Big], \end{aligned} \tag{3}$$

where $\theta_l = (\alpha^l, \beta^l, \mathbf{W})$ and $f_l(\mathbf{x}_n, \theta_l)$ corresponds to the LIRBF model defined in (2).

The nonlinearity of the model with respect to the parameters $\mathbf{c}_j$ and r_j of $\mathbf{W}$, and the indefinite character of the associated Hessian matrix of $L(\theta)$ do not recommend the use of gradient-based methods to maximize the log-likelihood function. Moreover, the optimal number of basis functions of the model (i.e. the number of RBF transformations) is unknown. Thus, the estimation of the vector parameter $\hat{\theta}$ is carried out by means of a hybrid procedure described below.

The methodology proposed is based on the combination of an Evolutionary Programming (EP) algorithm (global explorer) and a local optimization procedure (local exploiter) carried out by the standard maximum likelihood optimization method. In a first step, the EP algorithm is applied to design the structure and training the weights of a RBFNN. The evolutionary process determines the number m of RBFs in the model, and the corresponding matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_m)$, where $\mathbf{w}_j = (\mathbf{c}_j, r_j)$ is given by the centre vector and the radius of the j-th RBF. Once the basis functions have been determined by the EP algorithm, we consider a transformation of the input space by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the final generation of the EP algorithm.

The model is now linear in these new variables and the initial covariates. The remaining coefficient vector α and β are calculated by the maximum likelihood optimization method: choose the parameters that maximize the probability of the observed data points. For the multilogistic regression model, there are no closed-form solutions for these estimates. Instead, numeric optimization algorithms that approach the maximum likelihood solution iteratively and reach it in the limit have to be used. In the next subsection, two algorithms for obtaining this maximum likelihood solution are introduced. Then, the different steps of the LIRBF learning algorithm are described and, in the last subsection, the details of the EP algorithm are given.

A. Algorithms for Multilogistic Regression Maximum Likelihood Optimization

In this paper, two different algorithms have been considered for obtaining the maximum likelihood solution for the multilogistic regression model, both available in the WEKA machine learning workbench [33]:

1) MultiLogistic: It is an algorithm for building a multinomial logistic regression model with a ridge estimator to guard against overfitting by penalizing large coefficients, based on the work by le Cessie and van Houwelingen [34].

In order to find the coefficient matrices α and β for which $L(\theta)$ in (3) is minimized, a Quasi-Newton Method is used. Specifically, the method used is the active-sets' method with Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [35].

2) SimpleLogistic: This algorithm builds multinomial logistic regression models fitting them by using the LogitBoost algorithm [36], which was proposed by Friedman et al. for fitting *additive logistic regression models* by maximum likelihood. These models are a generalization of the (linear) logistic regression models described above.

LogitBoost is a boosting algorithm that performs forward stagewise fitting: in every iteration i, it computes 'response variables' that encode the error of the currently fit model on the training examples (in terms of probability estimates), and then tries to improve the model by adding a new function $f_{ij}(\mathbf{x})$ to the committee of functions $F_j(\mathbf{x})$, being $1 \leq j \leq J$ and J the number of classes. As shown by Friedman et al. [36], this amounts to performing a quasi Newton step in every iteration, where the Hessian matrix is approximated by its diagonal. In the special case that the $f_{ij}(\mathbf{x})$ and so the $F_j(\mathbf{x})$ are linear functions of the input variables, the additive logistic regression model is equivalent to the linear logistic model. Assuming that $F_j(\mathbf{x}) = \boldsymbol{\alpha}_j^T\mathbf{x}$, the equivalence of the two models is established by setting $\boldsymbol{\alpha}_j = \boldsymbol{\beta}_j - \boldsymbol{\beta}_J$ for $j = 1, ..., J-1$ and $\boldsymbol{\alpha}_J = \boldsymbol{\beta}_J$. This means that LogitBoost can be used for learning multilogistic regression models by fitting a standard least-squares regression function as the f_{ij}. However, it is also possible to use even simpler functions for the f_{ij}: simple regression functions that perform a regression on only one covariate present in the training data. In this way, fitting simple regression by least-squared error means fitting a simple regression function to each covariate in the data using least-squares as the error criterion, and then selecting the covariate that gives the smallest squared error. The final model found by LogitBoost will be the same because Quasi-Newton stepping is guaranteed to actually find the maximum likelihood solution if the likelihood function is convex. Using simple regression instead of multiple ones will basically slow down the process, but, if it is stopped before it converges, this will result in automatic covariate selection, because the model will only include the most relevant covariates present in data.

SimpleLogistic algorithm is based on applying LogitBoost with simple regression functions and determining the optimum number of iterations by a five fold cross-validation: the data is equally splitted five times into training and test, LogitBoost is run on every training set up to a maximum number of iterations (500) and the classification error on the respective test set is logged. Afterwards, LogitBoost is run again on all data using the number of iterations that gave the smallest error on the test set averaged over the five folds. Further details about the algorithm can be found in [37].

B. Estimation of the model coefficients

In this subsection, we describe the steps of the LIRBF learning algorithm in detail. The process is structured in three steps.

Step 1. For determining the best RBFNN model, we apply an EP algorithm to find the basis functions:

$$\mathbf{B}(\mathbf{x}, \mathbf{W}) = \{B_1(\mathbf{x}, \mathbf{w}_1), B_2(\mathbf{x}, \mathbf{w}_2), ..., B_m(\mathbf{x}, \mathbf{w}_m)\},$$

corresponding to the nonlinear part of $f_l(\mathbf{x}, \boldsymbol{\theta}_l)$ in (2). We have to determine the number of basis functions m and the weight matrix $\mathbf{W}$. To apply evolutionary neural network techniques, we consider a RBFNN with softmax outputs and the standard structure: an input layer with a node for every input variable; a hidden layer with several RBFs; and an output layer with $J-1$ nodes, where J is the number of classes. There are no connections between the nodes of a layer and none between the input and output layers either. A scheme of these models is given in Fig. 1.

The activation function of the j-th node in the hidden layer is given by:

$$B_j(\mathbf{x}, \mathbf{w}_j) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{r_j^2}\right),$$

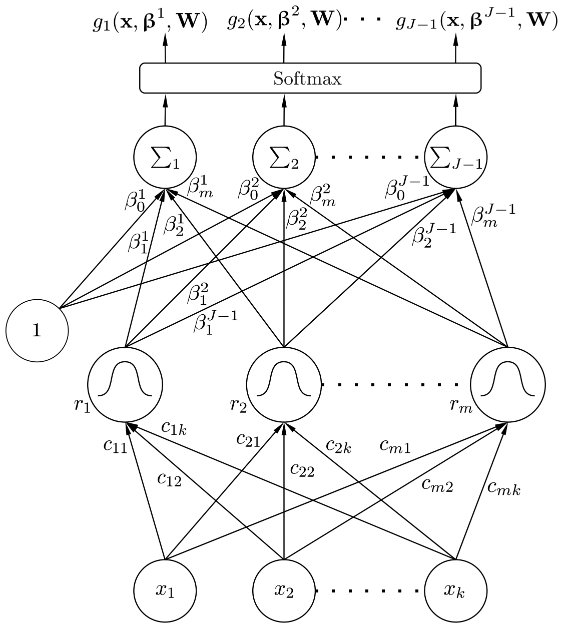

$g_1(\mathbf{x},\boldsymbol{\beta}^1,\mathbf{W})$
$g_2(\mathbf{x},\boldsymbol{\beta}^2,\mathbf{W})$
$g_{J-1}(\mathbf{x},\boldsymbol{\beta}^{J-1},\mathbf{W})$
Softmax
Σ_1
Σ_2
Σ_{J-1}
β_0^1 β_1^1 β_2^1 β_m^1
β_0^2 β_1^2 β_2^2 β_m^2
β_0^{J-1} β_1^{J-1} β_2^{J-1} β_m^{J-1}
1
r_1 r_2 r_m
c_{11} c_{12} c_{1k} c_{21} c_{22} c_{2k} c_{m1} c_{m2} c_{mk}
x_1 x_2 x_k

Fig. 1. Structure of Radial Basis Function Neural Networks: an input layer with k input variables, a hidden layer with m RBFs and an output layer with $J-1$ nodes

where $\mathbf{c}_j = (c_{j1}, ..., c_{jk})$ and c_{ji} is the weight of the connection between the i-th input node and the j-th RBF. The activation function of the l-th output node is given by:

$$g_l(\mathbf{x},\boldsymbol{\beta}^l,\mathbf{W}) = \beta_0^l + \sum_{j=1}^{m} \beta_j^l B_j(\mathbf{x},\mathbf{w}_j), \tag{4}$$

where β_j^l is the weight of the connection between the j-th RBF and the l-th output node and β_0^l is the bias of the l-th output node. The transfer function of all output nodes is the identity function.

The weight matrix $\mathbf{W}$ is estimated by means of an evolutionary neural network algorithm (detailed in Section IV-C) that optimizes the error function given by the negative log-likelihood for N observations associated with the RBFNN model:

$$\begin{aligned} L^*(\boldsymbol{\beta},\mathbf{W}) = & \tfrac{1}{N}\textstyle\sum_{n=1}^{N}\Big[-\sum_{l=1}^{J-1} y_n^{(l)} g_l(\mathbf{x}_n,\boldsymbol{\beta}^l,\mathbf{W}) + \\ & + \log\textstyle\sum_{l=1}^{J-1}\exp g_l(\mathbf{x}_n,\boldsymbol{\beta}^l,\mathbf{W})\Big]. \end{aligned} \tag{5}$$

Although in this step the evolutionary process obtains a concrete value for the $\boldsymbol{\beta}$ vector, we only consider the estimated weight vector $\hat{\mathbf{W}} = (\hat{\mathbf{w}}_1, \hat{\mathbf{w}}_2, ..., \hat{\mathbf{w}}_m)$, which builds the basis functions. The values for the $\boldsymbol{\beta}$ vector will be determined in step 3 together with those of the $\boldsymbol{\alpha}$ coefficient vector.

Step 2. We consider the following transformation of the input space by including the nonlinear basis functions obtained by the EP algorithm in Step 1 for the best model of the last generation:

$$\begin{aligned} & H : \mathbb{R}^k \to \mathbb{R}^{k+m}, \\ & (x_1, x_2, ..., x_k) \to (x_1, x_2, ..., x_k, z_1, z_2, ..., z_m), \end{aligned}$$

where $z_1 = B_1(\mathbf{x}, \hat{\mathbf{w}}_1), z_2 = B_2(\mathbf{x}, \hat{\mathbf{w}}_2), ..., z_m = B_m(\mathbf{x}, \hat{\mathbf{w}}_m)$.

Step 3. Using the matrix notation, the model of (2) can be expressed as:

$$f_l(\mathbf{x}, \boldsymbol{\theta}_l) = \boldsymbol{\alpha}^l \mathbf{x} + \boldsymbol{\beta}^l \mathbf{z},\ l = 1, 2, ..., J-1,$$

where $\mathbf{x} = (x_1, x_2, ..., x_k)$ and $\mathbf{z} = z_1, z_2, ..., z_m$.

In the third step, we minimize the negative log-likelihood function of (3) with respect to the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ of $\boldsymbol{\theta}$:

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \ & \tfrac{1}{N} \textstyle\sum_{n=1}^{N} \Big[-\sum_{l=1}^{J-1} y_n^{(l)} (\boldsymbol{\alpha}^l \mathbf{x}_n + \boldsymbol{\beta}^l \mathbf{z}_n) + \\ & + \log \textstyle\sum_{l=1}^{J-1} \exp(\boldsymbol{\alpha}^l \mathbf{x}_n + \boldsymbol{\beta}^l \mathbf{z}_n) \Big], \end{aligned}$$

where $\mathbf{x}_n = (1, x_{1n}, ..., x_{kn})$. Now, the Hessian matrix of the negative log-likelihood in the new variables $x_1, x_2, ...,$ $x_k, z_1, z_2, ..., z_m$ is semi-definite positive.

In this final step, both algorithms presented in Section IV-A have been applied for obtaining the parameter matrix $\boldsymbol{\theta}$. This results in two different models, one with all $x_1, x_2, ..., x_k, z_1, z_2, ..., z_m$ covariates present in the model (*MultiLogistic* algorithm) and the other with only those variables selected by the *SimpleLogistic* algorithm (see Section IV-A). These two approaches will be called MultiLogistic Initial-RBF regression (MLIRBF) and SimpleLogistic Initial-RBF regression (SLIRBF), respectively.

For comparison purposes, we have also considered the multilogistic regression models that are obtained with these two algorithms but constructed only from the nonlinear transformations given by the RBFNN of the EP algorithm, i.e. $z_1, z_2, ..., z_m$. This results on two other approaches which we will call MultiLogistic RBF regression (MLRBF) and SimpleLogistic RBF regression (SLRBF).

Finally, in Fig. 2, a schematic view of the steps of the methodology and the different methods associated is presented. The methods are represented in a double squared box. The best RBFNN individual obtained in Step 1 by the EP algorithm is also evaluated (RBFEP method).

C. Evolutionary acquisition of the RBF nonlinear transformations

In this subsection, the evolutionary algorithm used for obtaining the RBF nonlinear transformations is presented. The algorithm is aimed to produce a reduced number of RBF transformations with the simplest structure possible, i.e. trying to select the most important input variables for the estimation of the RBFs.

Among the different paradigms of Evolutionary Computation, we have chosen Evolutionary Programming (EP) due to the fact that we are evolving artificial neural networks. Therefore, crossover is not used due to its potential disadvantages in evolving artificial networks [38]. The population-based evolutionary algorithm for architectural design and the estimation of the real coefficients have points in common with other evolutionary algorithms in the bibliography [38]–[41]. The search begins with an initial population, and, with each iteration, the population is updated using a population-update algorithm, applying parametric and structural mutation to the best 10% and the best 90% of the population, respectively. The algorithm is elitist with respect to best individual, and the best 10% of individuals replace the worst 10% of individuals in each generation. The stop condition is defined as follows: the

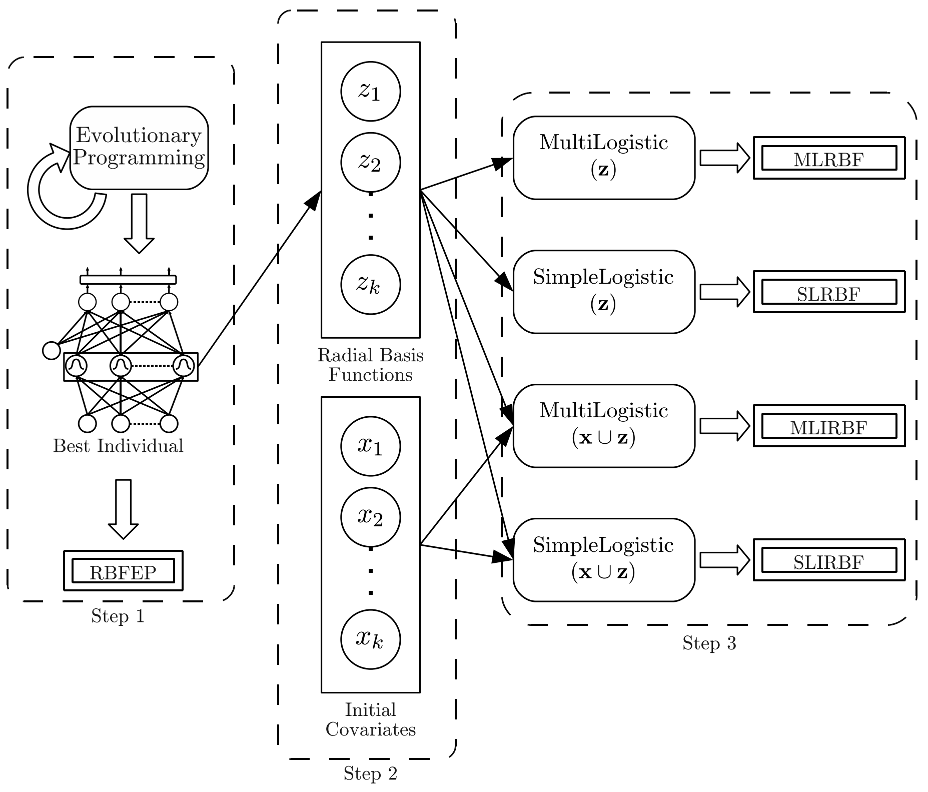


Fig. 2. Different steps of the LIRBF methodology. The different methods associated to this methodology are presented in a double squared box.

variance of the fitness of the best 10% of the population is less than a user-defined value, or a maximum number of generations is reached.

The RBFEP algorithm is detailed in Fig. 3, where p^{B} is the best optimized RBFNN returned by the algorithm. The main characteristics of the algorithm are the following:

1) Representation of the Individuals: The algorithm evolves architectures and connection weights simultaneously, each individual being a fully specified RBFNN. The neural networks are represented using an object-oriented approach and the algorithm deals directly with the RBFNN phenotype. Each connection is specified by a binary value indicating if the connection exists, and a real value representing its weight. As the crossover is not considered, this object-oriented representation does not assume a fixed order between the different RBFs.

In order to define the topology of the neural networks, two parameters are considered: $M_{\min}$ and $M_{\max}$. They correspond, respectively, to the minimum and maximum number of RBFs in the whole evolutionary process.

2) Error and Fitness Functions: We consider $L^*(\boldsymbol{\beta}, \mathbf{W})$ defined in (5) as the error function of an individual $g(\bullet, \boldsymbol{\beta}, \mathbf{W})$ of the population. Observe that g is a RBFNN and can be seen as a multi-valuated function:

$$g(\mathbf{x}, \boldsymbol{\beta}, \mathbf{W}) = \left(g_1(\mathbf{x}, \boldsymbol{\beta}^1, \mathbf{W}), ..., g_{J-1}(\mathbf{x}, \boldsymbol{\beta}^{J-1}, \mathbf{W})\right),$$

```
RBFEP Algorithm:
Input: Training dataset ($D$)
Output: Best optimized RBFNN ($p^{\mathrm{B}}$)
 1: $P^{\mathrm{I}} \leftarrow \{p^{\mathrm{I}}_1, ..., p^{\mathrm{I}}_{5000}\}$ // $p^{\mathrm{I}}_i$ is a randomly generated RBFNN
 2: $\forall p^{\mathrm{I}}_i \in P^{\mathrm{I}}$, $f^{\mathrm{I}}_i \leftarrow A(p^{\mathrm{I}}_i)$ // Evaluate fitness
 3: $P \leftarrow \{p_{(1)}, ..., p_{(5000)}\}, \quad (p_{(i)} \prec p_{(j)}) \iff (f^{\mathrm{I}}_i > f^{\mathrm{I}}_j)$ // Sort individuals in $P^{\mathrm{I}}$ by increasing $f^{\mathrm{I}}_i$
 4: $P \leftarrow \{p_{(1)}, ..., p_{(500)}\}$ // Retain the best 500 RBFNNs
 5: $\forall p_{(i)} \in P$, $p_{(i)} \leftarrow k\text{-means}(p_{(i)})$ // Improve individuals' centres
 6: while not Stop Condition do
 7:     $\forall p_i \in P$, $f_i \leftarrow A(p_i)$ // Evaluate fitness
 8:     $P \leftarrow \{p_{(1)}, ..., p_{(500)}\}, \quad (p_{(i)} \prec p_{(j)}) \iff (f_i > f_j)$ // Sort individuals in $P$ by increasing $f_i$
 9:     $p^{\mathrm{B}} \leftarrow p_{(1)}$ // Store Best Individual
10:     $P^{\mathrm{P}} \leftarrow \{p_{(1)}, ..., p_{(50)}\}$ // Parametric mutation parents (best $10\%$ of individuals)
11:     $P^{\mathrm{S}} \leftarrow \{p_{(1)}, ..., p_{(449)}\}$ // Structural mutation parents (best $90\%$ of individuals minus one)
12:     $\forall p^{\mathrm{P}}_{(i)} \in P^{\mathrm{P}}$, $p^{\mathrm{P}}_{(i)} \leftarrow \mathrm{parametricMutation}(p^{\mathrm{P}}_{(i)})$ // Apply parametric mutation
13:     $\forall p^{\mathrm{S}}_{(i)} \in P^{\mathrm{S}}$, $p^{\mathrm{S}}_{(i)} \leftarrow \mathrm{structuralMutation}(p^{\mathrm{S}}_{(i)})$ // Apply structural mutation
14:     $P \leftarrow P^{\mathrm{P}} \cup P^{\mathrm{S}} \cup \{p^{\mathrm{B}}\}$ // Offspring including the elite
15: end while
16: $\forall p_i \in P$, $f_i \leftarrow A(p_i)$ // Evaluate fitness
17: $P \leftarrow \{p_{(1)}, ..., p_{(500)}\}, \quad (p_{(i)} \prec p_{(j)}) \iff (f_i > f_j)$ // Sort individuals in $P$ by increasing $f_i$
18: $p^{\mathrm{B}} \leftarrow p_{(1)}$
19: return $p^{\mathrm{B}}$
```

Fig. 3. EP training algorithm framework

where $g_l(\mathbf{x}, \boldsymbol{\beta}^l, \mathbf{W})$ is defined in (4).

The fitness measure needed for evaluating the individuals (Fig. 3, steps 2, 7 and 16) is a strictly decreasing transformation of the error function $L^*(\boldsymbol{\beta}, \mathbf{W})$ given by $A(g) = \frac{1}{1+L^*(\boldsymbol{\beta}, \mathbf{W})}$, where $0 < A(g) \leq 1$.

3) Initialization of the population: The initial population is generated trying to obtain RBFNNs with the maximum possible fitness. First, $5,000$ random RBFNNs are generated (Fig. 3, step 1), where the number of RBFs m is a random value in the interval $[M_{\min}, M_{\max}]$. The number of connections between all RBFs of an individual and the input layer is a random value in the interval $[1, k]$ and all of them are connected with the same randomly chosen input variables. In this way, all the RBFs of each individual are initialized in the same random subspace of the input variables. A random value in the $[-I, I]$ interval is assigned for the weights between the input layer and the hidden layer and in the $[-O, O]$ interval for those between the hidden layer and the output layer. The obtained individuals are evaluated using the fitness function and the initial population is finally obtained by selecting the best 500 RBFNNs (Fig. 3, steps 2-4).

In order to improve the randomly generated centres, the standard k-means clustering algorithm [42] is applied using these random centres as the initial centroids for the algorithm and a maximum number of iterations of 100

(Fig. 3, step 5).

4) Parametric Mutation: Parametric mutation (Fig. 3, step 10) alters the value of the coefficients of the model. It is accomplished for each coefficient $w \in (\boldsymbol{\beta}, \mathbf{W})$ of the $g(\bullet, \boldsymbol{\beta}, \mathbf{W})$ individual adding a Gaussian noise, $w(t+1) = w(t) + \xi(t)$, where $\xi(t) \in N(0, \alpha(t))$ and $N(0, \alpha(t))$ represents a one-dimensional normally distributed random variable with mean 0 and variance $\alpha(t)$. The weights are sequentially mutated, hidden node after hidden node, and a standard simulated annealing process [43] is applied to accept or reject the modifications in each RBF. Thus, if ΔA is the difference in the fitness function before and after the random step, the criterion is: if $\Delta A \geq 0$, the step is accepted, and if $\Delta A < 0$, the step is accepted with a probability $\exp(\Delta A/T(g))$, where the temperature $T(g)$ of an individual g is given by $T(g) = 1 - A(g),\ \ 0 \leq T(g) < 1$.

The variance $\alpha(t)$ is updated throughout the evolution. There are different methods to update the variance. We use one of the simplest methods: the $1/5$ success rule of Rechenberg [44]. This rule states that the ratio of successful mutations should be $1/5$. Therefore, if the ratio of successful mutations is greater than $1/5$, the mutation deviation should increase; otherwise, the deviation should decrease. Thus:

$$\alpha(t+s) = \begin{cases} (1+\lambda)\alpha(t), & \text{if} \ \ s_g > 1/5, \\ (1-\lambda)\alpha(t), & \text{if} \ \ s_g < 1/5, \\ \alpha(t) & \text{if} \ \ s_g = 1/5, \end{cases}$$

where s_g is the frequency of successful mutations over s generations and $\lambda = 0.1$. The adaptation tries to avoid being trapped in local minima and to speed up the evolutionary process when the searching conditions are suitable.

5) Structural Mutation: Structural mutation (Fig. 3, step 11) implies a modification in the structure of the RBFNNs and allows the exploration of different regions in the search space, helping to keep the diversity of the population. There are five different structural mutations: node addition, node deletion, connection addition, connection deletion, and node fusion. These five mutations are applied sequentially to each network, each one with a specific probability. The node mutations are performed as follows:

- Node addition. One or more RBFs are added to the hidden layer. The origin nodes of the connections from the input layer are chosen randomly and have a random value in the interval $[-I, I]$. The origin nodes of the connections to the output layer are chosen randomly and its values are also random values in the interval $[-O, O]$.
- Node deletion. One or more RBFs, together with their connections, are randomly selected and deleted.
- Node fusion. Two randomly selected RBFs, a and b, are replaced by a new node c, which is a combination of both. The connections common to both basis functions are kept, with a weight given by:

$$c_{ci} = \frac{r_a}{r_a+r_b} c_{ai} + \frac{r_b}{r_a+r_b} c_{bi} \quad \beta_c^i = \frac{(\beta_a^i + \beta_b^i)}{2}$$
$$r_c = \frac{(r_a+r_b)}{2}$$

  Those connections not shared by the basis functions are inherited by c with probability 0.5 and their weight is unchanged.

The number of nodes added or deleted in node addition, node deletion and node fusion mutations is calculated as $\Delta_{\min} + uT(g)[\Delta_{\max} - \Delta_{\min}]$, u being a random uniform variable in the interval $[0, 1]$, $T(g) = 1 - A(g)$ the temperature of the neural net, and $\Delta_{\min}$ and $\Delta_{\max}$ a minimum and maximum number of nodes specified as parameters.

The connection structural mutations are performed as follows:

- Connection addition. Connection addition mutations are first performed in the hidden layer and then in the output layer. When adding a connection from the input layer to the hidden layer, a node from each layer is selected randomly, and then the connection is added with a random weight. A similar procedure is performed from the hidden layer to the output layer.
- Connection deletion. In the same way, connection deletion mutation is first performed in the hidden layer and then in the output layer, choosing randomly the origin node from the previous layer and the target node from the mutated layer.

We apply connection mutations sequentially for each mutated neural net, first, adding (or deleting) $1 + u[\Delta_{\mathrm{o}} n_{\mathrm{o}}]$ connections from the hidden layer to the output layer and then, adding (or deleting) $1 + u[\Delta_{\mathrm{h}} n_{\mathrm{h}}]$ connections from the input layer to the hidden layer, u being a random uniform variable in the interval $[0, 1]$, Δ_{o} and Δ_{h} previously defined ratios of number of connections in the hidden and the output layer, and n_{o} and n_{h} the current number of connections in the output and the hidden layers.

Parsimony is also encouraged in evolved networks by attempting the five structural mutations sequentially, where node or connection deletion and node fusion are always attempted before addition. Moreover, the deletion and fusion operations are made with higher probability ($T(g)$ for deletion and fusion mutations and $T^2(g)$ for addition ones). If a deletion or fusion mutation is successful, no other mutation will be made. If the probability does not select any mutation, one of the mutations is chosen at random and applied.

6) Parameters of the Algorithm: All the parameters used in the evolutionary algorithm except $M_{\min}$ and $M_{\max}$ have the same values in all the problems analyzed below. We have done a simple linear rescaling of the input variables in the interval $[-2, 2]$, X_i^* being the transformed variables. The centres c_{ji} are initialized in this interval (i.e. $[-I, I] = [-2, 2]$), and the coefficients β_j^l are initialized in the $[-5, 5]$ interval (i.e. $[-O, O] = [-5, 5]$). The initial value of the radii r_j is obtained as a random value in the interval $(0, d_{\max}]$, where $d_{\max}$ is the maximum distance between two training input examples.

The size of the population is $N = 500$. We have considered $\alpha(0) = 0.5$, $\lambda = 0.1$ and $s = 5$ for the parametric mutations. For the structural mutations, the number of RBFs that can be added or removed in a structural mutation is within the $[\Delta_{\min}, \Delta_{\max}] = [1, 2]$ interval. The ratio of the number of connections to add or delete in the hidden and the output layer during structural mutations is $\Delta_{\mathrm{o}} = 0.05$ and $\Delta_{\mathrm{h}} = 0.3$.

The stop criterion is reached whenever one of the following two conditions is fulfilled: the variance of the fitness of the best 10% of the population is less than 10^{-4} or 500 generations are completed.

V. Experiments

The proposed methodology is applied to eighteen datasets taken from the UCI repository [11], to test its overall performance when compared to other methods. Two additional datasets described in Section V-B, which correspond to a real agronomical problem of discriminating cover crops in olive orchards, have been included.

The first subsection defines the experimental design and the next one describes the characteristics of the real agronomical problems. Then, the comparison of the proposed models to the SimpleLogistic, MultiLogistic and RBFEP methods is presented. Finally, the last subsection is devoted to a comparison to other machine learning algorithms.

A. Experimental Design

The proposed methods (MLRBF, SLRBF, MLIRBF and SLIRBF) are compared to different algorithms:

- Multi-logistic regression methods, including the SimpleLogistic (SLogistic) and MultiLogistic (MLogistic) algorithms applied over the initial covariate space (see Section IV-A). As our models are logistic regression models, it is necessary to compare its performance to standard logistic regression algorithms.
- The RBFEP method. As our models are built from the RBFs of the best RBFNN obtained by the EP algorithm, it is necessary to compare its performance to the original RBFEP method.
- High performance machine learning algorithms:
  - The Logistic Model Tree (LMT) [37] classifier.
  - The C4.5 classification tree inducer [45].
  - The Naive Bayes Tree learning algorithm (NBTree) [46].
  - The AdaBoost.M1 algorithm [47], using C4.5 as the base learner and the maximum number of iterations set to 10 and 100 iterations (Ada10 and Ada100).
  - A Gaussian RBF Network (RBFNetwork) [48], deriving the centres and width of hidden units using k-means and combining the outputs obtained from the hidden layer using logistic regression. k-means is applied separately to each class to derive k clusters for each class. The structure and learning process of this RBFNetwork is very similar to the MLRBF method.
  - The $C-\mathrm{SVM}$ algorithm [30] with RBF kernels (SVM).

  These are the best performing methods from those presented in [37]. We have also included the RBFNetwork and SVM models, due to their similarities to the models proposed in this paper. The description and some previous results of these methods can be found in [30], [33] and [37].

The selected datasets present different numbers of instances, features and classes (see Table I). The minimum and maximum number of hidden nodes have been obtained as the best result of a preliminary experimental design, considering a small, medium and high value: $[M_{\min}, M_{\max}] \in \{[1,3],[4,6],[10,12]\}$. This value is also included in Table I.

For the RBFEP, MLRBF, SLRBF, MLIRBF and SLIRBF methods, the experimental design was conducted using a 10-fold cross-validation procedure, with 10 repetitions per each fold. For the other methods, the results have been

TABLE I
CHARACTERISTICS OF THE TWENTY DATASETS USED FOR THE EXPERIMENTS: NUMBER OF INSTANCES (Size), NUMBER OF REAL (R), BINARY (B) AND NOMINAL (N) INPUT VARIABLES, TOTAL NUMBER OF INPUTS (#In.), NUMBER OF CLASSES (#Out.), PER-CLASS DISTRIBUTION OF THE INSTANCES (Distribution) AND MINIMUM AND MAXIMUM NUMBER OF HIDDEN NODES USED FOR EACH DATASET ($[M_{\min}, M_{\max}]$)

Dataset	Size	R	B	N	#In.	#Out.	Distribution	$[M_{\min}, M_{\max}]$
Hepatitis	155	6	13	–	19	2	$(32, 123)$	$[1, 3]$
Glass(G2)	163	9	–	–	9	2	$(87, 76)$	$[10, 12]$
Sonar	208	60	–	–	60	2	$(97, 111)$	$[10, 12]$
Heart-c	302	6	3	4	26	2	$(164, 138)$	$[1, 3]$
Ionosphere	351	33	1	–	34	2	$(126, 225)$	$[10, 12]$
Vote	435	–	16	–	16	2	$(267, 168)$	$[4, 6]$
Australian	690	6	4	5	51	2	$(307, 383)$	$[4, 6]$
Breast-w	699	9	–	–	9	2	$(458, 241)$	$[4, 6]$
German	1000	6	3	11	61	2	$(700, 300)$	$[1, 3]$
Post-Op.	90	–	1	6	20	3	$(2, 24, 64)$	$[1, 3]$
Iris	150	4	–	–	4	3	$(50, 50, 50)$	$[1, 3]$
Newthyroid	215	5	–	–	5	3	$(150, 35, 30)$	$[4, 6]$
Balance	625	4	–	–	4	3	$(288, 49, 288)$	$[4, 6]$
Cortijo(Sp)	80	7	–	–	7	4	$(40, 20, 10, 10)$	$[1, 3]$
Cortijo(Su)	50	7	–	–	7	4	$(10, 20, 10, 10)$	$[1, 3]$
Lymph.	148	3	9	6	38	4	$(2, 81, 61, 4)$	$[3, 6]$
Anneal	898	6	14	18	59	5	$(8, 99, 684, 67, 40)$	$[4, 6]$
Glass	214	9	–	–	9	6	$(70, 76, 17, 13, 9, 29)$	$[10, 12]$
Zoo	101	1	15	–	16	7	$(41, 20, 5, 13, 4, 8, 10)$	$[4, 6]$
Audiology	226	–	61	8	96	24	$(1, 1, 57, 22, 1, 2, 20, 48, 2, 6, 2, 4, 2, 2, 9, 3, 1, 2, 22, 4, 1, 4, 8, 2)$	$[1, 3]$

All nominal variables are transformed to binary variables.
Heart-c: Heart disease (Cleveland); Lymph.: Lymphography; Post-Op.: Post-Operative.

obtained performing 10 times a 10-fold cross validation, because all are deterministic methods, i.e. they are not based in random values and return the same result for each execution. The results for the other methods have been taken from the paper of Landwehr et al. [37], except for the SVM and RBFNetwork methods and the Post-Op., Newthyroid, Cortijo(Sp) and Cortijo(Su) datasets, not included in Landwehr's work.

The RBFEP algorithm was implemented in JAVA using the Evolutionary Computation framework JCLEC [49] (http://jclec.sourceforge.net). For the MLRBF, SLRBF, MLIRBF and SLIRBF methods, we slightly modified the RBFEP algorithm, applying the SimpleLogistic and MultiLogistic algorithms from WEKA [33]. We also used "libsvm" [50] for obtaining the results of the SVM method and WEKA for obtaining the results of the RBFNetwork method and the Post-Op., Newthyroid, Cortijo(Sp) and Cortijo(Su) datasets.

The performance of each method has been evaluated using the correct classification rate or C in the generalization set (C_{G}, see (1)).

B. Description of the "Cortijo(Sp)" and "Cortijo(Su)" Datasets

Both datasets correspond to a real agronomical problem of precision farming, which consists of discriminating cover crops in olive orchards as affected by its phenological stage, using a high-resolution field spectroradiometer.

Olive is the main perennial Spanish crop and soil management in olive orchards is mainly based on intensive and tillage operations, which have a great relevancy in terms of the increase of atmospheric CO_2, desertification, erosion and land degradation. Due to this negative environmental impact, the European Union only subsidizes cropping systems which alter the natural soil as little as possible and protect it with cover crops. Current methods to estimate the cover crop soil coverage consist of sampling and ground visits to only 1% of the total olive orchards. Remotely sensed data may offer the ability to efficiently identify and map crops and cropping methods over large areas [51]. To detect and map olive trees and cover crops it is necessary that suitable differences exist in spectral reflectance among them and bare soil. Moreover, the spectral signature of any vegetation type is different depending on the phenological stage in which it is obtained [52]. This analysis will help to program the suitable wavelengths of airborne hyperspectral sensors. In this way, hyperspectral measurements obtained using an ASD Handheld FieldSpec Spectroradiometer (Analytical Spectral Devices, Inc., Boulder) were used to discriminate four classes: live cover crops, dead cover crops, bare soil and olive trees.

The study was conducted in Andalusia, southern Spain, in a location named “Cortijo del Rey” in early spring [Cortijo(Sp)] and early summer [Cortijo(Su)]. Forty spectral signatures of live cover crop, twenty of dead cover crops, ten of olive trees, and ten of bare soil were taken in spring in 2007. The same number of samples were acquired in the summer but only ten samples of live cover crop could be obtained. The hyperspectral range was between 400 and 900 nm. However, preliminary experiments suggested that using only seven wavelengths was enough for characterizing the original spectral curves and so the datasets only include these wavelengths.

C. Comparison to SLogistic, MLogistic and RBFEP

First of all, an analysis of the performance of all the proposed methods (MLRBF, SLRBF, MLIRBF and SLIRBF) when compared to the SLogistic, MLogistic and RBFEP methods is performed. This is essential because the proposed methods are based on different combinations of the RBFs of a RBFNN obtained using the RBFEP method and both SimpleLogistic and MultiLogistic logistic regression algorithms. Consequently, the different proposals must achieve a better performance than all these methods in order to justify the additional complexity of these combinations.

In Table II, the mean and the standard deviation of the correct classification rate in the generalization set (C_{G}) is shown for each dataset and a total of 100 executions. Based on the mean C_{G}, the ranking of each method in each dataset ($R = 1$ for the best performing method and $R = 7$ for the worst one) is obtained and the mean accuracy ($\overline{C_{\mathrm{G}}}$) and mean ranking ($\overline{R}$) are also included in Table II.

TABLE II

COMPARISON OF THE PROPOSED METHODS TO SLOGISTIC, MLOGISTIC AND RBFEP: MEAN AND STANDARD DEVIATION OF THE ACCURACY RESULTS (C_{G}) FROM 100 EXECUTIONS OF A 10-FOLD CROSS VALIDATION, MEAN ACCURACY ($\overline{C_{\mathrm{G}}}$) AND MEAN RANKING ($\overline{R}$)

	Method ($C_{\mathrm{G}}(\%)$)						
	SLogistic	MLogistic	RBFEP	MLRBF	SLRBF	MLIRBF	SLIRBF
Dataset	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD	Mean $\pm$ SD
Hepatitis	84.20 $\pm$ 8.20	83.89 $\pm$ 8.12	81.69 $\pm$ 6.30	*85.83 $\pm$ 1.84*	**86.01 $\pm$ 1.91**	82.07 $\pm$ 8.94	84.21 $\pm$ 6.61
Glass(G2)	76.32 $\pm$ 8.89	69.56 $\pm$ 10.77	80.05 $\pm$ 9.80	79.25 $\pm$ 9.58	**80.29 $\pm$ 9.53**	76.50 $\pm$ 10.35	*80.18 $\pm$ 9.76*
Sonar	75.93 $\pm$ 8.51	72.47 $\pm$ 8.90	*80.59 $\pm$ 8.00*	79.59 $\pm$ 9.03	**80.60 $\pm$ 8.53**	75.80 $\pm$ 9.42	80.56 $\pm$ 8.35
Heart-c	83.30 $\pm$ 6.35	83.70 $\pm$ 6.64	*84.14 $\pm$ 4.79*	83.72 $\pm$ 4.64	83.65 $\pm$ 5.25	82.81 $\pm$ 5.03	**84.77 $\pm$ 5.09**
Ionosphere	87.78 $\pm$ 4.99	87.72 $\pm$ 5.57	*93.83 $\pm$ 4.43*	92.92 $\pm$ 4.25	93.78 $\pm$ 4.38	89.28 $\pm$ 4.92	**94.06 $\pm$ 4.18**
Vote	*95.93 $\pm$ 2.58*	95.65 $\pm$ 3.12	95.65 $\pm$ 2.65	95.50 $\pm$ 2.79	95.56 $\pm$ 2.62	93.75 $\pm$ 3.59	**96.37 $\pm$ 2.85**
Australian	85.04 $\pm$ 3.97	85.33 $\pm$ 3.85	85.87 $\pm$ 3.26	**86.23 $\pm$ 3.44**	86.04 $\pm$ 3.28	85.16 $\pm$ 4.07	*86.07 $\pm$ 2.79*
Breast-w	96.21 $\pm$ 2.19	**96.50 $\pm$ 2.18**	96.13 $\pm$ 2.36	95.80 $\pm$ 2.42	96.04 $\pm$ 2.43	95.83 $\pm$ 2.36	*96.28 $\pm$ 2.33*
German	**75.34 $\pm$ 3.50**	*75.24 $\pm$ 3.54*	72.00 $\pm$ 2.19	72.06 $\pm$ 3.83	71.62 $\pm$ 7.98	74.91 $\pm$ 3.71	75.11 $\pm$ 3.48
Post-Op.	70.44 $\pm$ 6.74	59.67 $\pm$ 10.55	*70.67 $\pm$ 5.81*	70.00 $\pm$ 6.22	**70.78 $\pm$ 5.84**	62.44 $\pm$ 13.29	*70.67 $\pm$ 6.42*
Iris	95.93 $\pm$ 4.82	**97.07 $\pm$ 4.77**	96.20 $\pm$ 5.78	96.13 $\pm$ 5.85	96.33 $\pm$ 5.05	96.67 $\pm$ 4.59	*96.80 $\pm$ 4.88*
Newthyroid	*96.74 $\pm$ 3.85*	96.20 $\pm$ 4.86	95.48 $\pm$ 3.13	95.49 $\pm$ 4.28	96.65 $\pm$ 2.88	95.90 $\pm$ 2.96	**96.92 $\pm$ 2.57**
Balance	88.74 $\pm$ 2.91	89.44 $\pm$ 3.29	91.15 $\pm$ 1.31	*94.65 $\pm$ 2.93*	94.42 $\pm$ 2.83	**94.83 $\pm$ 2.69**	94.34 $\pm$ 2.69
Cortijo(Sp)	*95.25 $\pm$ 7.70*	91.63 $\pm$ 9.74	86.00 $\pm$ 5.11	88.63 $\pm$ 9.42	87.63 $\pm$ 8.79	92.13 $\pm$ 7.88	**96.13 $\pm$ 7.04**
Cortijo(Su)	91.00 $\pm$ 12.19	89.60 $\pm$ 14.06	86.60 $\pm$ 15.06	86.20 $\pm$ 14.13	87.80 $\pm$ 13.30	*91.40 $\pm$ 10.73*	**92.80 $\pm$ 10.45**
Lymph.	**84.37 $\pm$ 9.97**	77.58 $\pm$ 10.59	81.27 $\pm$ 10.52	81.06 $\pm$ 10.26	80.99 $\pm$ 10.30	75.82 $\pm$ 12.07	*84.30 $\pm$ 10.23*
Anneal	**99.48 $\pm$ 0.70**	99.24 $\pm$ 0.79	91.18 $\pm$ 1.65	93.23 $\pm$ 2.40	93.35 $\pm$ 2.35	99.28 $\pm$ 0.93	*99.31 $\pm$ 0.82*
Glass	65.29 $\pm$ 8.03	63.12 $\pm$ 9.16	65.94 $\pm$ 9.40	66.06 $\pm$ 9.51	66.44 $\pm$ 9.55	*66.65 $\pm$ 11.35*	**67.01 $\pm$ 9.39**
Zoo	94.69 $\pm$ 6.59	94.85 $\pm$ 6.34	93.95 $\pm$ 6.92	94.05 $\pm$ 7.36	94.06 $\pm$ 7.07	**95.84 $\pm$ 6.19**	*95.15 $\pm$ 5.90*
Audiology	**84.10 $\pm$ 7.35**	79.71 $\pm$ 8.36	36.84 $\pm$ 8.89	46.21 $\pm$ 12.33	46.03 $\pm$ 10.58	78.62 $\pm$ 10.12	*83.78 $\pm$ 8.53*
$\overline{C_{\mathrm{G}}}(\%)$	*86.30*	84.41	83.26	84.13	84.40	85.28	**87.74**
$\overline{R}$	3.90	4.55	4.73	4.60	*3.85*	4.45	**1.93**

The best result is in bold face and the second best result in italic.

TABLE III
CRITICAL DIFFERENCE VALUES, MEAN RANKING AND DIFFERENCES OF RANKINGS OF THE BONFERRONI-DUNN TEST, USING SLOGISTIC, SLRBF AND SLIRBF AS THE CONTROL METHODS

	Control Method		
$\overline{R}$	SLogistic	SLRBF	SLIRBF
$\overline{R}_{(1)} = 3.90$	$-$	$\lvert\overline{R}_{(1)} - \overline{R}_{(5)}\rvert = 0.05$	$\lvert\overline{R}_{(1)} - \overline{R}_{(7)}\rvert = 1.98^{+}_{\bullet}$
$\overline{R}_{(2)} = 4.55$	$\lvert\overline{R}_{(2)} - \overline{R}_{(1)}\rvert = 0.65$	$\lvert\overline{R}_{(2)} - \overline{R}_{(5)}\rvert = 0.70$	$\lvert\overline{R}_{(2)} - \overline{R}_{(7)}\rvert = 2.63^{+}_{\bullet}$
$\overline{R}_{(3)} = 4.73$	$\lvert\overline{R}_{(3)} - \overline{R}_{(1)}\rvert = 0.82$	$\lvert\overline{R}_{(3)} - \overline{R}_{(5)}\rvert = 0.88$	$\lvert\overline{R}_{(3)} - \overline{R}_{(7)}\rvert = 2.80^{+}_{\bullet}$
$\overline{R}_{(4)} = 4.60$	$\lvert\overline{R}_{(4)} - \overline{R}_{(1)}\rvert = 0.70$	$\lvert\overline{R}_{(4)} - \overline{R}_{(5)}\rvert = 0.75$	$\lvert\overline{R}_{(4)} - \overline{R}_{(7)}\rvert = 2.68^{+}_{\bullet}$
$\overline{R}_{(5)} = 3.85$	$\lvert\overline{R}_{(5)} - \overline{R}_{(1)}\rvert = 0.50$	$-$	$\lvert\overline{R}_{(5)} - \overline{R}_{(7)}\rvert = 2.52^{+}_{\bullet}$
$\overline{R}_{(6)} = 4.45$	$\lvert\overline{R}_{(6)} - \overline{R}_{(1)}\rvert = 0.55$	$\lvert\overline{R}_{(6)} - \overline{R}_{(5)}\rvert = 0.60$	$\lvert\overline{R}_{(6)} - \overline{R}_{(7)}\rvert = 1.93^{+}_{\bullet}$
$\overline{R}_{(7)} = 1.93$	$\lvert\overline{R}_{(7)} - \overline{R}_{(1)}\rvert = 1.98^{-}_{\bullet}$	$\lvert\overline{R}_{(7)} - \overline{R}_{(5)}\rvert = 1.93^{-}_{\bullet}$	$-$
$CD_{(\alpha=0.1)} = 1.64,\ CD_{(\alpha=0.05)} = 1.80$			

$\bullet$: Statistically significant difference with $\alpha = 0.05$.

$+$: The difference is in favour of the control method.

$-$: The difference is against the control method.

(1): SLogistic; (2): MLogistic; (3): RBFEP; (4): MLRBF; (5): SLRBF; (6): MLIRBF; (7): SLIRBF.

CD: Critical Difference.

From the analysis of the results, it can be concluded, from a purely descriptive point of view, that the SLIRBF method obtains the best result for seven datasets, the SLogistic and SLRBF methods yield the higher performance for four datasets, MLogistic and SLIRBF for two datasets and MLRBF only for one dataset. Furthermore, the SLIRBF method obtains the best mean ranking ($\overline{R} = 1.93$) followed by the SLRBF method ($\overline{R} = 3.85$) and reports the highest mean accuracy ($\overline{C_{\mathrm{G}}} = 87.74\%$) followed by the SLogistic method ($\overline{C_{\mathrm{G}}} = 86.30\%$). The mean accuracy and ranking obtained by the SLIRBF and SLRBF are higher than those obtained by MLIRBF and MLRBF, which confirms the necessity of selecting the most important covariates (initial covariates or RBFs) in the final model in order to avoid overfitting (e.g. Post-op and Newthyroid datasets).

To determine the statistical significance of the rank differences observed for each method in the different datasets, we have carried out a non-parametric Friedman test [53] with the ranking of C_{G} of the best models as the test variable (since a previous evaluation of the C_{G} values results in rejecting the normality and the equality of variances' hypothesis). The test shows that the effect of the method used for classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.18)$ and the F-distribution statistical value is $F^* = 4.89 \notin C_0$. Consequently, we reject the null-hypothesis stating that all algorithms perform equally in mean ranking.

On the basis of this rejection, three post-hoc non-parametric Bonferroni-Dunn tests [54], [55] were applied with the best performing algorithms (SLogistic, SLRBF and SLIRBF) as the control methods. The results of the Bonferroni-Dunn test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Table III using the corresponding critical values for the two-tailed Bonferroni-Dunn test.

The SLIRBF method obtains a significant higher ranking of C_{G} when compared to all the methods and none of the remaining methods obtain significant higher rankings when compared to each other. Consequently, the SLIRBF method obtains a significant higher performance than the two methods from which it is derived, SLogistic and RBFEP, what justifies the proposal.

D. Comparison to other machine learning algorithms

In this last subsection, a comparison of the best performing method of those proposed (SLIRBF) is presented. As previously stated, very competitive methods and other that share some characteristics with the different proposals have been selected.

The comparison is again performed using the correct classification rate or accuracy in the generalization set (C_{G}), the mean accuracy ($\overline{C_{\mathrm{G}}}$) and the mean ranking ($\overline{R}$), and the results are included in Table IV. A descriptive analysis of the results lead to the following remarks: the SLIRBF method obtains the best results for seven out of the twenty datasets, the second best results for other seven datasets and the best mean accuracy ($\overline{C_{\mathrm{G}}} = 87.74\%$); Ada100 obtains the best results for nine datasets, the second best results for other two datasets and the third best mean accuracy ($\overline{C_{\mathrm{G}}} = 86.14\%$); LMT achieves the best performance for only one dataset and the second best performance for other four datasets although it results in the second best mean accuracy ($\overline{C_{\mathrm{G}}} = 86.88\%$). These three methods result in the best mean rankings, the best one being obtained by SLIRBF ($\overline{R} = 2.33$), followed by Ada100 ($\overline{R} = 3.00$) and LMT ($\overline{R} = 3.50$).

It is necessary again to ascertain if there are significant differences in mean ranking of C_{G}, so a procedure similar to that used in the previous subsection has been applied. The non-parametric Friedman test also shows that the effect of the method used for classification is statistically significant at a significance level of 5%, as the confidence interval is $C_0 = (0, F_{0.05} = 2.08)$ and the F-distribution statistical value is $F^* = 9.26 \notin C_0$. Consequently, we reject the null-hypothesis stating that all algorithms of this second comparison perform equally in mean ranking.

On the basis of this rejection, three post-hoc non-parametric Bonferroni-Dunn tests were applied with the best performing algorithms (LMT, Ada100 and SLIRBF) as the control methods. The results of the Bonferroni-Dunn test for $\alpha = 0.1$ and $\alpha = 0.05$ can be seen in Table V using the corresponding critical values. The results show that LMT obtains better significant mean ranking of C_{G} than C4.5, RBFNetwork and SVM for $\alpha = 0.05$ and better than NBTree for $\alpha = 0.10$. Ada100 yields significant better results than the same four methods, but in this case for $\alpha = 0.05$ in all methods. Finally, SLIRBF obtains significant better results than C4.5, NBTree, Ada10, RBFNetwork and SVM for $\alpha = 0.05$. Although the mean ranking of SLIRBF is better than that of LMT and Ada100, these differences are not significant. However, it can observed in Table V that the differences of ranking obtained by SLIRBF take values higher than 3 for some methods, while those obtained by LMT and Ada100 are not higher than 2.88.

Therefore, we can conclude that the results obtained by SLIRBF make it a very competitive method when compared to the learning schemes previously mentioned.

TABLE IV
COMPARISON OF THE PROPOSED METHODS TO OTHER MACHINE LEARNING ALGORITHMS: MEAN AND STANDARD DEVIATION OF THE ACCURACY RESULTS (C_{G}) FROM 100 EXECUTIONS OF A 10-FOLD CROSS VALIDATION, MEAN ACCURACY ($\overline{C_{\mathrm{G}}}$) AND MEAN RANKING ($\overline{R}$)

	Method ($C_{\mathrm{G}}(\%)$)							
	LMT	C4.5	NBTree	Ada10	Ada100	RBFNetwork	SVM	SLIRBF
Dataset	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD	Mean ± SD
Hepatitis	*84.21 ± 8.21*	79.22 ± 9.57	81.36 ± 10.80	82.38 ± 8.01	**84.93 ± 7.79**	82.01 ± 9.71	79.38 ± 2.26	*84.21 ± 6.61*
Glass(G2)	77.28 ± 9.90	78.15 ± 8.50	80.75 ± 9.40	*85.17 ± 7.75*	**88.72 ± 6.42**	76.03 ± 11.03	77.03 ± 10.03	80.18 ± 9.76
Sonar	76.32 ± 9.58	73.61 ± 9.34	77.16 ± 10.46	79.22 ± 8.70	**85.14 ± 7.84**	71.99 ± 10.44	55.95 ± 9.66	*80.56 ± 8.35*
Heart-c	*83.51 ± 6.67*	76.94 ± 6.59	80.60 ± 6.29	78.76 ± 7.09	80.00 ± 6.55	81.92 ± 7.61	56.75 ± 3.71	**84.77 ± 5.09**
Ionosphere	92.99 ± 4.13	89.74 ± 4.38	89.49 ± 5.12	93.05 ± 3.92	*94.02 ± 3.83*	91.23 ± 5.47	93.00 ± 4.28	**94.06 ± 4.18**
Vote	95.90 ± 2.67	**96.57 ± 2.56**	95.03 ± 3.29	95.51 ± 3.05	95.26 ± 3.13	94.55 ± 3.56	95.98 ± 2.41	*96.37 ± 2.85*
Australian	85.04 ± 3.84	85.57 ± 3.96	85.07 ± 4.03	84.01 ± 4.36	**86.43 ± 3.98**	75.54 ± 5.37	85.51 ± 4.26	*86.07 ± 2.79*
Breast-w	96.18 ± 2.20	95.01 ± 2.73	*96.60 ± 2.04*	96.08 ± 2.16	**96.70 ± 2.18**	96.14 ± 2.01	96.05 ± 2.23	96.28 ± 2.33
German	**75.37 ± 3.53**	71.25 ± 3.17	74.07 ± 4.10	70.91 ± 3.60	74.53 ± 3.26	70.28 ± 3.58	68.92 ± 1.07	*75.11 ± 3.48*
Post-Op.	70.11 ± 7.01	69.67 ± 7.04	66.67 ± 10.36	55.33 ± 14.39	56.33 ± 13.97	66.78 ± 10.42	*70.44 ± 5.74*	**70.67 ± 6.42**
Iris	95.80 ± 4.89	94.73 ± 5.30	93.53 ± 5.64	94.33 ± 5.22	94.53 ± 5.05	95.80 ± 4.80	**97.47 ± 3.88**	*96.80 ± 4.88*
Newthyroid	96.74 ± 3.85	92.62 ± 5.60	92.60 ± 5.57	94.56 ± 4.19	94.98 ± 4.42	*96.79 ± 3.46*	75.43 ± 4.57	**96.92 ± 2.57**
Balance	89.71 ± 2.68	77.82 ± 3.42	75.83 ± 5.32	78.35 ± 3.78	76.11 ± 4.09	85.85 ± 3.83	*89.98 ± 1.90*	**94.34 ± 2.69**
Cortijo(Sp)	*95.25 ± 7.70*	87.50 ± 11.51	84.50 ± 11.53	91.00 ± 8.90	91.25 ± 8.79	82.38 ± 11.53	87.00 ± 9.55	**96.13 ± 7.04**
Cortijo(Su)	*91.40 ± 11.81*	77.40 ± 14.95	72.00 ± 12.39	80.00 ± 14.49	79.60 ± 14.77	88.00 ± 12.06	84.20 ± 13.72	**92.80 ± 10.45**
Lymph.	84.10 ± 10.00	75.84 ± 11.05	80.89 ± 8.77	80.87 ± 8.63	**84.72 ± 8.41**	76.27 ± 9.68	80.31 ± 8.44	*84.30 ± 10.23*
Anneal	99.52 ± 0.73	98.57 ± 1.04	98.53 ± 1.23	*99.59 ± 0.70*	**99.63 ± 0.65**	90.75 ± 2.18	85.74 ± 2.67	99.31 ± 0.82
Glass	69.15 ± 8.99	67.63 ± 9.31	70.16 ± 9.67	*75.15 ± 7.59*	**78.78 ± 7.80**	65.60 ± 10.80	67.61 ± 8.41	67.01 ± 9.39
Zoo	94.89 ± 6.44	92.61 ± 7.33	94.95 ± 6.24	*96.15 ± 6.11*	**96.35 ± 6.07**	94.05 ± 6.62	92.58 ± 6.14	95.15 ± 5.90
Audiology	84.19 ± 7.17	77.26 ± 7.47	76.82 ± 7.38	**84.84 ± 7.46**	*84.70 ± 7.57*	71.97 ± 8.53	67.41 ± 5.50	83.78 ± 8.53
$\overline{C_{\mathrm{G}}}(\%)$	*86.88*	82.89	83.33	84.76	86.14	82.70	80.34	**87.74**
$\overline{R}$	3.50	5.65	5.50	4.40	*3.00*	5.88	5.75	**2.33**

The best result is in bold face and the second best result in italic.

TABLE V
CRITICAL DIFFERENCE VALUES, MEAN RANKING AND DIFFERENCES OF RANKINGS OF THE BONFERRONI-DUNN TEST, USING LMT, ADA100 AND SLIRBF AS THE CONTROL METHODS

	Control Method		
$\overline{R}$	LMT	Ada100	SLIRBF
$\overline{R}_{(1)} = 3.50$	$-$	$\lvert\overline{R}_{(1)} - \overline{R}_{(5)}\rvert = 0.50$	$\lvert\overline{R}_{(1)} - \overline{R}_{(8)}\rvert = 1.18$
$\overline{R}_{(2)} = 5.65$	$\lvert\overline{R}_{(2)} - \overline{R}_{(1)}\rvert = 2.15^{+}_{\bullet}$	$\lvert\overline{R}_{(2)} - \overline{R}_{(5)}\rvert = 2.65^{+}_{\bullet}$	$\lvert\overline{R}_{(2)} - \overline{R}_{(8)}\rvert = 3.33^{+}_{\bullet}$
$\overline{R}_{(3)} = 5.50$	$\lvert\overline{R}_{(3)} - \overline{R}_{(1)}\rvert = 2.00^{+}_{\circ}$	$\lvert\overline{R}_{(3)} - \overline{R}_{(5)}\rvert = 2.50^{+}_{\bullet}$	$\lvert\overline{R}_{(3)} - \overline{R}_{(8)}\rvert = 3.18^{+}_{\bullet}$
$\overline{R}_{(4)} = 4.40$	$\lvert\overline{R}_{(4)} - \overline{R}_{(1)}\rvert = 0.90$	$\lvert\overline{R}_{(4)} - \overline{R}_{(5)}\rvert = 1.40$	$\lvert\overline{R}_{(4)} - \overline{R}_{(8)}\rvert = 2.08^{+}_{\bullet}$
$\overline{R}_{(5)} = 3.00$	$\lvert\overline{R}_{(5)} - \overline{R}_{(1)}\rvert = 0.50$	$-$	$\lvert\overline{R}_{(5)} - \overline{R}_{(8)}\rvert = 0.67$
$\overline{R}_{(6)} = 5.88$	$\lvert\overline{R}_{(6)} - \overline{R}_{(1)}\rvert = 2.38^{+}_{\bullet}$	$\lvert\overline{R}_{(6)} - \overline{R}_{(5)}\rvert = 2.88^{+}_{\bullet}$	$\lvert\overline{R}_{(6)} - \overline{R}_{(8)}\rvert = 3.55^{+}_{\bullet}$
$\overline{R}_{(7)} = 5.75$	$\lvert\overline{R}_{(7)} - \overline{R}_{(1)}\rvert = 2.25^{+}_{\bullet}$	$\lvert\overline{R}_{(7)} - \overline{R}_{(5)}\rvert = 2.75^{+}_{\bullet}$	$\lvert\overline{R}_{(7)} - \overline{R}_{(8)}\rvert = 3.43^{+}_{\bullet}$
$\overline{R}_{(8)} = 2.33$	$\lvert\overline{R}_{(8)} - \overline{R}_{(1)}\rvert = 1.18$	$\lvert\overline{R}_{(8)} - \overline{R}_{(5)}\rvert = 0.67$	$-$
$CD_{(\alpha=0.1)} = 1.90,\ CD_{(\alpha=0.05)} = 2.07$			

$\bullet, \circ$: Statistically significant differences with $\alpha = 0.05$ ($\bullet$) and $\alpha = 0.1$ ($\circ$).

$+$: The difference is in favour of the control method.

(1): LMT; (2): C4.5; (3): NBTree; (4): Ada10; (5): Ada100; (6): RBFNetwork; (7): SVM; (8): SLIRBF.

CD: Critical Difference.

VI. CONCLUSIONS

This paper combines three powerful techniques used in machine learning research: multilogistic regression, evolutionary algorithms and Radial Basis Function Neural Networks (RBFNNs). The approach carries out an adequate combination of the three elements to solve multiclass problems and follows the proposal of Hastie, Tibshirani and Friedman of a generalization of the linear logistic model using additive models of basis functions [2], [30]. Specifically, this new approach consists of a multilogistic regression model built on the combination of linear covariates and Gaussian RBFs for the predictor function. The process for obtaining the coefficients is carried out in three steps: (1) an EP algorithm aimed to produce a reduced number of RBF transformations with the simplest structure possible (i.e. trying to select the most important input variables for the estimation of these RBFs), (2) a transformation of the input space by adding the nonlinear transformations of the input variables given by the RBFs of the best individual in the last generation and (3) a maximum likelihood optimization method. In this final step, two different multilogistic regression algorithms are applied, one that considers all initial and RBF covariates (MLIRBF) and another one that incrementally constructs the model and applies cross-validation, resulting in an automatically covariate selection (SLIRBF).

From the analysis of the results obtained, several conclusions can be drawn. First of all, the covariate selection process incorporated in the SimpleLogistic algorithm of the the SLIRBF and SLRBF methods is necessary in some datasets for avoiding overfitting. Then, the results reported by the MLIRBF and SLIRBF methods (built on initial and RBF covariates) are better for almost all datasets than those reported by their equivalent standard multilogistic regression methods, MLogistic and SLogistic (built only on the initial covariates). Finally, SLIRBF is a very competitive method, obtaining high accuracy values when compared to other recent machine learning methods such as LMT or SVM. It obtains the statistically significant highest mean rank of C_{G} when compared to

SLogistic, MLogistic, RBFEP, C4.5, NBTree, Ada10, RBFNetwork and SVM, and a highest mean rank of C_{G} (but not statistically significant) when compared to LMT and Ada100.

The results obtained encourage us to perform a more extended experimental design, considering a higher number of datasets and problems with more classes. From another point of view, the methodology proposed is associated to a very specific type of RBF function (Gaussian function) and other RBF functions or kernel basis function could be used.

REFERENCES

[1] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[2] T. J. Hastie and R. J. Tibshirani, "Nonparametric regression and classification. Part II - nonparametric classification," in *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, ser. Computer and System Sciences, V. Cherkassky, J. H. Friedman, and H. Wechsler, Eds. Springer, 1996, vol. 136, pp. 70–82.

[3] R. P. Lippmann, "Neural networks, bayesian a posteriori probabilities, and pattern classification," in *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, ser. Computer and System Sciences, V. Cherkassky, J. H. Friedman, and H. Wechsler, Eds. Springer, 1996, vol. 136, pp. 83–104.

[4] J. Friedman, "Multivariate adaptive regression splines (with discussion)," *Annals of Statistics*, vol. 19, pp. 1–141, 1991.

[5] C. Hervas-Martinez and F. Martinez-Estudillo, "Logistic regression using covariates obtained by product-unit neural network models," *Pattern Recognition*, vol. 40, no. 1, pp. 52–64, 2007.

[6] C. Hervás-Martínez, F. J. Martínez-Estudillo, and M. Carbonero-Ruz, "Multilogistic regression by means of evolutionary product-unit neural networks," *Neural Networks*, vol. 21, no. 7, pp. 951–961, 2008.

[7] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[8] S. Lee and R. Kil, "A gaussian potential function network with hierarchical self-organising learning," *Neural Networks*, vol. 4, no. 2, pp. 207–224, 1991.

[9] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, August 2006.

[11] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[12] C. M. Bishop, "Improving the generalization properties of radial basis function neural networks," *Neural Computation*, vol. 3, no. 4, pp. 579–581, 1991.

[13] R. J. Howlett and L. C. Jain, *Radial Basis Function Networks 1: Recent Developments in Theory and Applications*, 1st ed., ser. Studies in Fuzziness and Soft Computing. Physica-Verlag Heidelberg, 2001.

[14] K. Hunt, D. Sbarbaro, R. Zbikowski, and P. Gawthrop, "Neural networks for control system - a survey," *Automatica*, vol. 28, pp. 1083–1112, 1992.

[15] Z. Uykan and C. Guzelis, "Input-output clustering for determining centers of radial basis function network," in *Proceedings of the 1997 European Conference on Circuit Theory and Design*, vol. 2, Technical University of Budapest. Hungary: European Circuit Society, 1997, pp. 435–439.

[16] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 306–314, 2000.

[17] O. Buchtala, M. Klimek, and B. Sick, "Evolutionary optimization of radial basis function classifiers for data mining applications," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 5, pp. 928–947, 2005.

[18] L. N. de Castro, E. R. Hruschka, and R. J. G. B. Campello, "An evolutionary clustering technique with local search to design RBF neural network classifiers," in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IJCNN'04)*, vol. 3, 2004, pp. 2083–2088.

[19] Z. Q. Zhao and D. S. Huang, "A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability," *Applied Mathematical Modelling*, vol. 31, pp. 1271–1281, 2007.

[20] S. Chen, Y. Wu, and B. L. Luk, "Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1239–1243, 1999.

[21] J. González, I. Rojas, J. Ortega, H. Pomares, F. J. Fernández, and A. F. Días, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Transactions Neural Networks*, vol. 14, no. 6, pp. 1478–1495, 2003.

[22] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 869–880, 1996.

[23] B. A. Whitehead, "Genetic evolution of radial basis function coverage usingorthogonal niches," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1525–1528, 1996.

[24] B. A. Whitehead and T. D. Choate, "Evolving space-filling curves to distribute radial basis functionsover an input space," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 15–23, 1994.

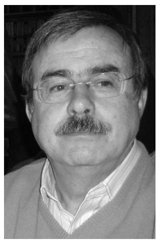
[25] M. J. Embrechts, B. Szymanski, and M. Sternickel, "Introduction to scientific data mining: Direct kernel methods and applications," in *Computationally Intelligent Hybrid Systems*, S. J. Ovaska, Ed. New York: Wiley Interscience, 2004, ch. 10, pp. 317–362.

[26] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, 1st ed. Cambridge, U.K.: Cambridge University Press, 2000.

[27] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1999.

[28] Y. Lee, Y. Lin, and G. Wahba, "Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 67–81, 2004.
[29] Y. Lin, "Support vector machines and the bayes rule in classification," *Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 259–275, 2003.
[30] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, August 2001.
[31] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philosophical Transactions of the Royal Society of London*, vol. A, pp. 415–446, 1909.
[32] S. S. Keerthi, K. B. Duan, S. K. Shevade, and A. N. Poo, "A fast dual algorithm for kernel logistic regression," *Machine Learning*, vol. 61, no. 1–3, pp. 151–165, 2005.
[33] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed., ser. Data Management Systems. Morgan Kaufmann (Elsevier), 2005.
[34] S. le Cessie and J. van Houwelingen, "Ridge estimators in logistic regression," *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.
[35] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, 1982.
[36] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The Annals of Statistics*, vol. 38, no. 2, pp. 337–374, 2000.
[37] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1-2, pp. 161–205, 2005.
[38] P. J. Angeline, G. M. Sauders, and J. B. Pollack, "An evolutionary algorithm that constructs recurren neural networks," *IEEE Transactions on Neural Networks*, vol. 5, pp. 54–65, 1994.
[39] X. Yao, "Global optimization by evolutionary algorithms," in *Proceedings of the 2nd Aizu International Symposium on Parallel Algorithms/Architecutre Synthesis, (pAs'97)*, Aizu-Wakamatsu, Japan, 1997, pp. 282–291.
[40] A. C. Martínez-Estudillo, C. Hervás-Martínez, F. J. Martínez-Estudillo, and N. García-Pedrajas, "Hybridization of evolutionary algorithms and local search by means of a clustering method," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 36, no. 3, pp. 534–545, 2006.
[41] F. J. Martínez-Estudillo, C. Hervás-Martínez, P. A. Gutiérrez, and A. C. Martínez-Estudillo, "Evolutionary product-unit neural networks classifiers," *Neurocomputing*, vol. 72, no. 1-2, pp. 548–561, 2008.
[42] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. Academic Press, 1999.
[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[44] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog, 1973.
[45] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
[46] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid," in *Proceedings of the Second International Conference on Knoledge Discovery and Data Mining (KDD96)*. Menlo Park, CA: AAAI Press, 1996, pp. 202–207.
[47] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann, 1996, pp. 148–156.
[48] I. T. Nabney, "Efficient training of rbf networks for classification," *International Journal of Neural Systems*, vol. 14, no. 3, pp. 201–208, 2004.
[49] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervas, "JCLEC: a Java framework for evolutionary computation," *Soft Computing*, vol. 12, no. 4, pp. 381–392, 2008.
[50] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[51] S. South, J. Qi, and D. P. Lusch, "Optimal classification methods for mapping agricultural tillage practices," *Remote Sensing of Environment*, vol. 91, no. 1, pp. 90–97, 2004.
[52] J. Peña-Barragán, F. López-Granados, M. Jurado-Expósito, and L. García-Torres, "Spectral discrimination of ridolfia segetum and sunflower as affected by phenological stage," *Weed Research*, vol. 46, no. 1, pp. 10–21, 2006.
[53] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.
[54] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–56, 1961.
[55] Y. Hochberg and A. Tamhane, *Multiple Comparison Procedures*. John Wiley & Sons, 1987.

Pedro A. Gutiérrez-Peña was born in Córdoba, Spain, in 1982. He received the B.S. degree in Computer Science from the University of Sevilla, Spain, in 2006. He is currently working toward the Ph.D. Degree at the Department of Computer Science and Numerical Analysis (University of Córdoba, Spain), in the area of Computer Science and Artificial Intelligence. His current interests include neural networks and their applications, evolutionary computation and hybrid algorithms.



César Hervás-Martínez was born in Cuenca, Spain. He received the B.S. degree in statistics and operating research from the Universidad Complutense, Madrid, Spain, in 1978 and the Ph.D. degree in mathematics from the University of Seville, Seville, Spain, in 1986. He is a Professor with the University of Córdoba in the Department of Computing and Numerical Analysis in the area of computer science and artificial intelligence and an Associate Professor in the Department of Quantitative Methods in the School of Economics. His current research interests include neural networks, evolutionary computation, and the modelling of natural systems.

Francisco J. Martínez-Estudillo was born in Villacarrillo, Jaén. He received the B.S. degree in mathematics in 1987 and the Ph. D. degree in Mathematics in 1991, speciality Differential Geometry, both from the University of Granada, Granada, Spain. From 1987 to 2002, he developed his research in non-Euclidean geometry, Lorentz spaces and maximal surfaces. He is currently a professor in the Department of Management and Quantitative Methods in ETEA, University of Córdoba, Spain. His current research interests include structure optimization of neural networks, evolutionary algorithms and multiobjective optimization.