UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Departamento de Ciencias de la Computación e Inteligencia Artificial

Programa Oficial de Postgrado "Diseño, Análisis y Aplicaciones de Sistemas Inteligentes"

Tesis Doctoral

REDES NEURONALES EVOLUTIVAS DE FUNCIONES DE BASE RADIAL GENERALIZADAS EN CLASIFICACIÓN: APLICACIONES

Adiel Castaño Méndez

Granada 2012

Editor: Editorial de la Universidad de Granada
Autor: Adiel Castaño Méndez
D.L.: GR 1918-2013
ISBN: 978-84-9028-604-3

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Programa Oficial de Postgrado "Diseño, Análisis y Aplicaciones de Sistemas Inteligentes"

REDES NEURONALES EVOLUTIVAS DE FUNCIONES DE BASE RADIAL GENERALIZADAS EN CLASIFICACIÓN: APLICACIONES

Memoria de Tesis que prensenta

Adiel Castaño Méndez

como requisito para optar al grado de
Doctor en Informática

Tutor de Tesis

César Hervás Martínez

Director de Tesis

César Hervás Martínez

Granada 2012

Universidad de Granada

REDES NEURONALES EVOLUTIVAS DE FUNCIONES DE BASE RADIAL GENERALIZADAS EN CLASIFICACIÓN: APLICACIONES

MEMORIA QUE PRESENTA

Adiel Castaño Méndez

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Octubre de 2012

DIRECTOR

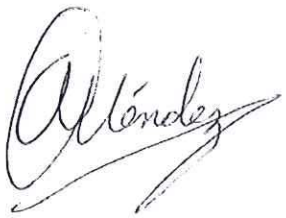
César Hervás Martínez

Departamento de Informática y Análisis Numérico (Universidad de Córdoba)

La memoria titulada, "Redes Neuronales Evolutivas de Funciones de Base Radial Generalizadas en Clasificación: Aplicaciones" que presenta D. Adiel Castaño Méndez para optar al grado de doctor, se ha realizado dentro del programa de doctorado Soft Computing y Sistemas Inteligentes del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, bajo la dirección del Catedrático de Universidad César Hervás Martínez, del Departamento de Informática y Análisis Numérico de la Universidad de Córdoba.

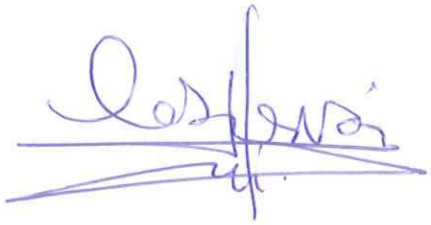
Códoba, Octubre de 2012

El doctorando



Fdo: Adiel Castaño Méndez

El director



Fdo: César Hervás Martínez

Esta Tesis Doctoral ha sido financiada en parte con cargo a los Proyectos TIN2008-06681-C06-03, TIN2011-22794 de la Comisión Interministerial de Ciencia y Tecnología (CICYT), al Proyecto de Excelencia P08-TIC-3745 de la Junta de Andalucía, con fondos FEDER y a la Asociación Universitaria Iberoamericana de Postgrado.

A mis abuelos que me "quieren" mucho y en especial a ese padre que fue y aún es, a tí Aye dedico esta tesis.

Agradecimientos

Esta investigación es uno de los retos más grandes a los cuales me he tenido que enfrentar.

Quisiera expresar mi más sincero agradecimiento a mi director, César Hervás Martínez, quien se comportó como un padre conmigo, guiándome hacia el conocimiento y dejándome dar mis propios pasos hacia la ilustración. Su dedicación, apoyo, trabajo y calidad humana me han motivado para la finalización de esta tesis. No menor agradecimiento tengo para un amigo y compañero de trabajo, Francisco Fernández Navarro, un científico que me mostró que solo el trabajo arduo es la vía segura al éxito.

También quisiera agradecer la acogida que me dieron todos los miembros del grupo de investigación AYRNA al empezar mi investigación como becario, inicialmente Pedro Antonio Gutiérrez Peña, y más adelante Manuel Cruz Ramírez y Javier Sánchez Monedero.

Gracias a mis compañeros del Departamento de Informática de la Universidad de Pinar del Río, sin cuya colaboración no sería posible el logro de esta investigación.

Y finalmente, a mi familia, que es mi mejor compañía, gracias a mi madre por educarme así, a mi esposa por entender y apoyar mis duros días de trabajo, a mi hermano Tonito por su cariño, a mi abuela y a Toni por su confianza en mí.

Sinceramente, muchas gracias a todos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Lista de abreviaturas y siglas

ABL Algoritmo de Búsqueda Local

AC Añadir Conexión

ACO Optimización Colonia de Hormigas o "Ant Colony Optimization"

AG Algoritmo Genético

AE Algoritmo Evolutivo

AH Algoritmo Híbrido

AM Algoritmo Memético

AN Añadir Nodos

AUC Área Bajo la Curva Operativa Característica

AYRNA Aprendizaje y Redes Neuronales Artificiales

BARS "Best Agglomerative Ranked Subset"

BD Base de Datos

BL Búsqueda Local

BP "BackPropagation"

C Precisión

CCR "Correct Classification Rate"

CE Computación Evolutiva

CP Clasificación de Patrones

E Entropía Cruzada

EE Estrategias Evolutivas

EC Eliminar Conexión

EN Eliminar Nodos

FCBF "Fast Correlation-Based Filter"

FLC Controladores de Lógica Difusa

FP Falsos Positivos

FPR Porcentaje de Falsos Positivos

FN Falsos Negativos

FV Función de Verosimilitud

GC Correlación Cuadrática Generalizada

GG Gaussiana Generalizada

GM Minkowski Generalizada

GNA Algoritmo Gauss-Newton

GRBF Función de Base Radial Generalizada

GRBFNN Redes Neuronales de Base Radial Generalizada

HEP "Hybrid Evolutionary Programming"

HEPC "Hybrid Evolutionary Programming with Clustering"

HVDM Métrica Heterogenénea de Diferencia de Valores

HEOM Métrica Heterogénea Euclídea-Solapada

IRLS "Iterative Re-weigthed Least Squared"

iRProp Retropropagación Elástica Mejorada ("Inproved Resilient Backpropagation")

iRProp+ Retropropagación Elástica Mejorada ("Inproved Resilient Backpropagation")

IVDM Métrica Interpolada de Diferencia de Valores

JCLEC Librería de Clases en Java para Computación Evolutiva

KEEL "Knowledge Extraction based on Evolutionary Learning"

LogRBF Función de Base Radial Logarítmica

LM Levenberg-Marquardt

LMT "Logistic Model Tree"

GMCRBF MacroMedia ("Macro-Average")

GMCRBF Función de Base Radial Multicuadrática Generalizada

GMCIRBF Función de Base Radial Multicuadrática Inversa Generalizada

MAVG Macro-Avera

MCRBF Multicuadráticas Generalizadas

MCIRBF Multicuadráticas Inversas Generalizadas

MG Media Geométrica

MLP Perceptrón Multicapa

MSE Error Cuadrático Medio

N Patrones con Clase Negativa

NNEP "Neural Net Evolutionary Programming"

OLS Mínimos Cuadrados Ortogonales

P Precisión Positiva

PCA Análisis de Componentes Principales

PE Programación Evolutiva

PG Programación Genética

PG Proporcional Integral Derivativo

PSO Optimización por Sistema de Partículas

PS Sistema de Partículas

QRBF Función de Base Radial Q-Gaussiana

RBF Función de Base Radial

RL Regresión Logística

ROC "Receiver Operating Characteristic" o Curva Operativa Característica

RNA Redes Neuronales Artificiales

RProp Retropropagación Elástica (Resilient Backpropagation)

RMSE Raíz del Error Cuadrático Medio

SRBF Función de Base Radial Estándar

SA Recocido Simulado o "Simmulated Annealing"

Sp Especificidad

SS Búsqueda Dispersa

SVD Descomposiciún en Valores Singulares

SVM Máquinas de Soporte Vectorial

TPR Sensibilidad

TS Búsqueda Tabú

TP Verdaderos Positivos

TN Verdaderos Negativos

TPS "Thin Plate Spline"

U.C. Universidad Central

UP Unidades Producto

UN Unir Nodos

UPRBF Unidades Producto-Función de Base Radial

US Unidades Sigmoide

URBF Unidades con Funciones de Base Radial

USRBF Unidad Sigmoide-RBF

VDM Métrica de Diferencia de Valores

VUS Volumen Bajo la Superficie

WVDM Métrica Enmarcada de Diferencia de Valores

Capítulo 1

INTRODUCCIÓN

1.1. Motivación

El problema de la Clasificación de Patrones ,CP, dentro del ámbito del Reconocimiento de Patrones, es una de las formas más comunes a la cual reducir problemas abordados por numerosas ramas de la ciencia. La comunidad científica posee en la CP una herramienta versátil utilizada en el campo de la *visión con ordenador* para la clasificación de entornos urbanos, agrícolas y marinos desde imágenes de satélite, reconocimiento de huellas dactilares, reconocimiento e interpreatación de objetos en escenas, identificación en partes en líneas de ensamblado, detección de defectos, detección de tumores, análisis de cromosomas, conteo de células en la sangre, reconocimiento de caracteres: lectura de etiquetas, procesado de cheques bancarios, lectura de textos, reconocimiento de matrículas, etc. Otras aplicaciones no menos importantes donde la clasificación de patrones es de gran utilidad son: la evaluación de ganado bovino, pronóstico de perspectividad de yacimientos minerales, diagnóstico diferencial de enfermedades, lectura diagnóstica de electroseñales, identificación de objetos mediante sonidos (aviones, vehículos) e identificación de objetos mediante rastros (balística), entre otras aplicaciones.

La CP tiene como objetivo fundamental establecer una técnica o un modelo que permita, a partir de datos provenientes de una muestra concreta (conjunto de entrenamiento), predecir el valor de una o todas las variables medidas sobre los datos que no intervienen en el proceso de entrenamiento (conjunto

de generalización). Los modelos o técnicas de clasificación obtenidos predicen la *clase* de los patrones a los que se aplican. A su vez los modelos creados permiten (no todos en igual medida) la interpretación de las interacciones entre las variables medidas(relación causa-efecto); resultando una fuente importante de conocimiento para los expertos en la ciencia de la cual provienen los datos sujetos a clasificación.

El problema de la Clasificación de Patrones ha sido abordado desde diferentes perspectivas, algunas ya clásicas. Dentro de las técnicas más conocidas se encuentran los modelos que definen una función de error optimizada mediante algoritmos basados en gradiente, quienes al igual que las Máquinas de Soporte Vectorial (SVM) optimizan una función objetivo. Otros algoritmos ya comunes son los algoritmos basados en incertidumbre quienes construyen modelos basados en incertidumbre probabilística o difusa modelando la pertenencia de un patrón a una determinada clase como un valor de probabilidad o de incertidumbre.

1.2. Objetivo

La presente tesis tiene como objetivo fundamental la **obtención de multiclasificadores basados en Funciones de Base Radial Generalizada**. Nuestra propuesta se centra en la utilización de Algoritmos Evolutivos para entrenar Modelos de Redes Neuronales Artificiales y Modelos Neuro-Logísticos basados en Funciones de Base Radial Generalizada. Dichos modelos serán estimados tanto estructural (número de funciones de base, número de coeficientes del modelo) como paramétricamente (valor de los coeficientes del modelo).

Para el logro de este objetivo fundamental se definieron los siguientes objetivos específicos:

- Realizar un estudio de las técnicas de entrenamiento de modelos de RNAs.
- Diseñar e implementar de un Algoritmo Híbrido para el entrenamiento de Redes Neuronales de Base Radial Generalizada.

- Diseñar e implementar de un Algoritmo Evolutivo para el entrenamiento de modelos Neuro-Logísticos de Base Radial Generalizada.

- Validar los algoritmos elaborados con bases de datos internacionales y problemas reales de clasificación.

1.3. Resumen

Dentro de los modelos más utilizados para la Clasificación de Patrones se encuentran las Redes Neuronales Artificiales quienes de forma resumida se conceptualizan como: una aplicación sobre el espacio de variables de entrada que produce como resultado un valor o etiqueta perteneciente al espacio de salida caracterizado por la codificación de la clase a predecir.

De la utilización de los AEs para la optimización de RNAs surgen las denominadas Redes Neuronales Artificiales Evolutivas [2]. Esta área de investigación ha atraído mucha atención en la última década, proporcionando una herramienta muy adecuada para la optimización tanto de la topología como de los coeficientes de los modelos de RNAs [3]. Si tenemos en cuenta la dificultad que implica el establecimiento de una arquitectura adecuada junto con los correspondientes coeficientes de la RNA, la aplicación de AEs para el diseño de la estructura y la optimización de los coeficientes de una RNA está ampliamente justificada.

Para la transformación del espacio de entrada las RNAs presentan las llamadas funciones de base quienes representan la transformación de los datos de entrada. La presente investigación propone la utilización de las Funciones de Base Radial Generalizadas GRBF en modelos de RNA para ser entrenadas mediante Algoritmos Híbridos combinando el poder de exploración de los Algoritmos Evolutivos junto a la capacidad explotadora de los Algoritmos de Búsqueda Local con el fin de lograr la convergencia del algoritmo a una solución factible.

La utilización de las Funciones de Base Radial Generalizadas se justifica debido a su capacidad de ajuste a los datos que se encuentran en el límite de

dos o más GRBFs. Este comportamiento se debe a la presencia de un parámetro que contrae o relaja la curvatura de la GRBF. Con el fin de aumentar la rapidez y la precisión del entrenamiento de las Redes Neuronales de Base Radial Generalizada GRBFNN la presente tesis propone la reformulación de la GRBF permitiendo tanto al Algoritmo de Búsqueda Local (ABL) como al Algoritmo Evolutivo presentes en el Algoritmo Híbrido propuesto en la tesis, una convergencia más rápida. Este *Algoritmo Híbrido para el Entrenamiento de Redes Neuronales de Base Radial Generalizada* constituye un resultado de la investigación. La eficacia del algoritmo ha sido probada sobre bases de datos internacionalmente conocidas, pertenecientes a la UCI Machine Learning Repository [4]. Tras la obtención de este resultado se procedió a hibridar las GRBFNNs con la Regresión Logística.

Dentro de los modelos que combinan las RNAs con otros modelos bien conocidos se encuentran los Modelos Neuro-Logísticos quienes crean modelos logísticos sobre las variables generadas por las funciones de base de las RNAs. Este enfoque permite crear modelos que representan la interacción de las variables de entrada más allá de la simple combinación lineal descrita por el modelo de Regresión Logística, RL. La introducción de transformaciones más complejas en la RL dificulta el entrenamiento mediante el método de máxima verosimilitud. Por ello, el modelado de la estructura y de los coeficientes de las transformaciones no lineales inducidas por las funciones de base constituye un área donde la utilización de Algoritmos Evolutivos, se justifica.

La presente tesis propone un *Algoritmo Evolutivo para la creación de Modelos Neuro-Logísticos* el cual realiza una Regresión Logística sobre las salidas de las Funciones de Base Radial Generalizadas modeladas por un AE. El Algoritmo Evolutivo maneja conceptos como la perfecta multicolinealidad presente en las covariables que surgen gracias a las GRBFs con baja cobertura; dicha propiedad provoca que la precisión de la RLs caiga abruptamente. Por ello el AE trata las GRBF durante el transcurso de la evolución.

La precisión de los modelos Neuro-Logísticos elaborados por el AE son validados en el cuerpo de la tesis sobre datos de *clasificación de incapacidad laboral* y de *clasificación de microarrays*, preprocesados los últimos con técnicas

de selección de características.

1.4. Estructura de la Tesis

La memoria está organizada según el esquema siguiente:

- En el capítulo 2, se introducen los conceptos básicos acerca de las Redes Neuronales Artificiales RNAs necesarios para el desarrollo de los capítulos posteriores. Haciendo énfasis en las Redes "FeedForward" se procede a describir las distintas funciones de base de los nodos de la capa oculta de la red. Se analiza de forma especial la estructura de las Funciones de Base Radial (RBF) las cuales resultarán de importancia en posteriores capítulos. Seguidamente se describe cómo son empleados los modelos de RNAs en problemas de clasificación centrándonos en las métricas de rendimiento tanto para clasificación binaria como multiclase. Tras mostrar la utilidad de las RNAs en problemas de clasificación se describen los métodos de entrenamiento para ajustar dichos modelos.

- En el capítulo 3, se describe el funcionamiento de los Algoritmos Evolutivos para el entrenamiento de RNAs. Para ello se describe la metodología general creada por Kalyanmoy Deb para estructurar el funcionamiento de los Algoritmos Evolutivos. En concordancia con la propuesta de Deb se describe el funcionamiento del algoritmo NNEP para entrenar Redes Neuronales con nodos RBF estándar. Dicho algoritmo sirve de base para los algoritmos propuestos en los Capítulos 7 y 8.

- En el capítulo 4 , se describen las principales arquitecturas de hibridación de Algoritmos Evolutivos, ilustrando las alternativas más utilizadas para lograr su combinación. Además se describen trabajos anteriores desarrollados por el grupo de investigación AYRNA en el área de las Redes Neuronales Evolutivas, específicamente en el desarrollo de Algoritmos Híbridos.

- En el capítulo 5, se realiza una descripción de los modelos Neuro-Logísticos: su arquitectura y principales conceptos. Para ello se comenta el funcionamiento de los modelos de RL: su arquitectura, algoritmo clásico de entrenamiento (IRLS) y conceptos como la regularización de los coeficientes del modelo de RL y la *perfecta multicolinealidad*.

- En el capítulo 7, se propone un nuevo algoritmo para el entrenamiento de Redes Neuronales de Base Radial Generalizada y se valida mediante la experimentación sobre datos pertenecientes a la UCI Machine Learning Repository.

- En el capítulo 8, se propone un nuevo algoritmo para el entrenamiento de modelos Neuro-Logísticos de Base Radial Generalizada.

- En el capítulo 9, se describen las aplicaciones de los algoritmos propuestos en los capítulos 7 y 8 sobre clasificación de Microarrays y un problema real de clasificación de discapacidad laboral.

- En el capítulo 10, se muestran las conclusiones de la presente investigación indicando cómo se cumplieron cada uno de los objetivos específicos trazados en la tesis. Además se sientan las bases de los trabajos futuros en el desarrollo de algoritmos de entrenamiento de RNAs.

De forma general la estructura de la tesis se resume en 6 capítulos donde se describe el marco teŕico de la investigación donde se hace hincapié en el algoritmo NNEP, el cual servirá de base para las propuestas descritas en el capítulo 7 y 8. Por uĺtimo se dan los resultados de la investigación y las conclusiónes en los capítulos 9 y 10 respectivamente.

Capítulo 2

REDES NEURONALES

2.1. Terminología de Redes Neuronales Artificiales

Una red neuronal consiste en un conjunto de unidades de procesamiento, conocidas como *nodos, funciones de base o unidades de base*, las cuales están conectadas entre sí. La conectividad de una red neuronal viene dada en términos de una arquitectura o topología, la cual es un grafo con conexiones entre los nodos o unidades. Aquellos nodos que no tienen conexiones de entrada de un nodo anterior se denominan nodos de entrada y los nodos de los que no sale ninguna conexión a un nodo posterior se denominan nodos de salida. El resto de nodos se denominan nodos ocultos. Los nodos de computación de una red son los nodos de salida y los nodos ocultos. Todos los nodos que se encuentran a la misma distancia en el grafo de los nodos de entrada forman una capa.

Las redes neuronales pueden dividirse en diferentes tipos en función de su conectividad, siendo las más comunes: las redes con conexiones hacia adelante (feedforward) y las redes recurrentes (backward) (véase la Figura 2.1). En el primer caso los nodos solo tienen conexión hacia nodos pertenecientes a capas posteriores no siendo así en el caso de los nodos pertenecientes a las redes recurrentes donde los nodos poseen conexiones hasta nodos pertenecientes a capas anteriores, posteriores e incluso a nodos que se encuentran en la misma capa. Los elementos que caracterizan un nodo de una RNA son:

- Conexiones ponderadas: hacen el papel de las conexiones sinápticas. La existencia de conexiones determina si es posible que un nodo influya sobre otro, mientras que el valor de los pesos y el signo de los mismos definen el tipo (excitatorio/inhibitorio) y la intensidad de la influencia.

- Función de activación: calcula el valor de base o entrada total al nodo, generalmente como simple suma ponderada de todas las variables de entrada recibidas, es decir, de las variables de entrada multiplicadas por el peso o valor de las conexiones. Llamaremos $\boldsymbol{f(x, w)}$ al valor obtenido, siendo $\boldsymbol{x}$ el vector de variables de entrada y $\boldsymbol{w}$ el vector de conexiones.

- Función de salida o de transferencia: calcula la salida del nodo en función de la activación de la misma. Se representa como $h(\cdot)$ y se usan diferentes tipos de funciones, desde simples funciones de umbral hasta funciones no lineales. El valor final de salida ϕ_j de un nodo oculto sería el siguiente:

$$\phi_j(\boldsymbol{x}, w_j) = h(f(\boldsymbol{x}, w_j)) \tag{2.1}$$

  donde j representa el índice del nodo.

En los modelos de Redes Neuronales las funciones de activación son, en general, diferentes según se trate de nodos de la capa de salida o de nodos pertenecientes a la capa oculta. Usualmente, las funciones de salida de los nodos de la capa oculta son de diferentes tipos (ver epígrafe 2.2.3.3). Existen numerosas referencias a diferentes modelos de redes neuronales entre las que caben destacar las de (Bishop, 1995 [5] y Haykin, 1994 [6]).

El presente trabajo aborda las redes neuronales con conexiones hacia adelante (feedforward) con uno o varios nodos de salida de acuerdo al tipo de clasificación (binaria o multiclase) que aborda. Los modelos de RNA tienen una única capa intermedia o capa oculta (Figura 2.1 (a)).

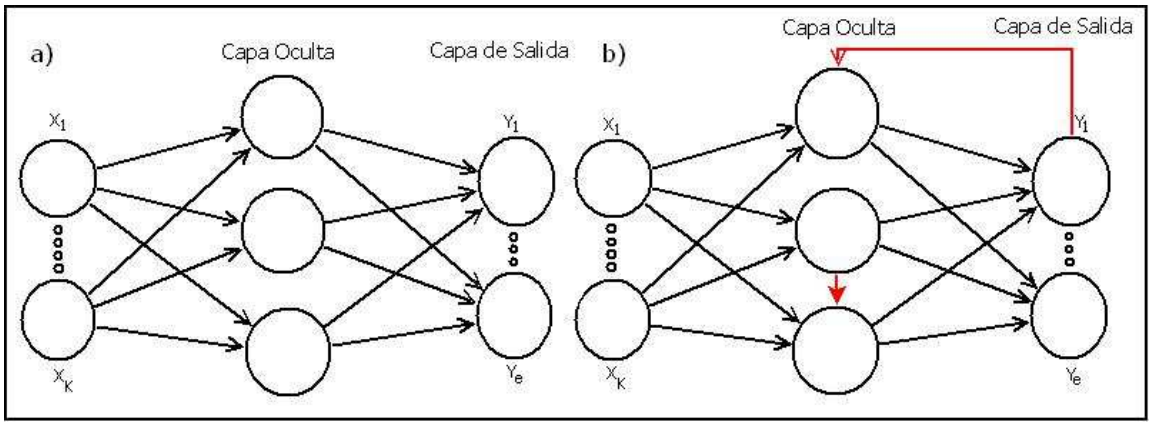


Figura 2.1: Topología de redes neuronales artificiales a) Red "FeedForward" b) Red Recurrente ("FeedBackward").

2.2. Modelos de Redes Neuronales Artificiales

El uso de distintos tipos de nodos en la capa oculta permite "particionar" el espacio de características mediante diferentes funciones discriminantes. Los nodos sigmoides, productos u otros de tipo proyección construyen un conjunto de hipersuperficies para delimitar las fronteras entre las clases. Al ser presentado un nuevo patrón o elemento del conjunto de entrenamiento a la red mediante las funciones de base, este se proyecta en el espacio en nuevas regiones de forma tal que se active el nodo o nodos de salida que más relación guarda con las funciones de base asociadas al patrón. Por otra parte en los nodos de tipo radial [7] si el patrón que se le presenta a la red neuronal se encuentra muy lejos de los nodos, no se activará ninguna salida.

En esta Tesis trataremos problemas de clasificación, pudiendo ser el número de nodos de salida superior a uno, en función del número de clases asociadas al problema. La arquitectura de estas RNAs se refleja en la Figura 2.2, donde $\boldsymbol{n}$ es el número de variables de entrada del problema, $\boldsymbol{M}$ es el número de nodos de la capa oculta y $\boldsymbol{L}$ es el número de salidas del problema. A cada nodo se le incorpora un sesgo o valor numérico que se incorpora al modelo. Si representamos con $\boldsymbol{\theta} = \{\theta_1, ..., \theta_j\}$ al conjunto de coeficientes asociados a la RNA, el modelo funcional asociado de cada uno de los nodos de salida es el

siguiente:

$$f_l(\boldsymbol{x}, \boldsymbol{\theta}_{il}) = \beta_{l0} + \sum_{j=1}^{M} \beta_{lj}\phi_j(\boldsymbol{x}, \boldsymbol{w}_j) \qquad l = 1, \ldots, L \tag{2.2}$$

siendo $1 \leq \mathrm{l} \leq \mathrm{L}$, $\boldsymbol{\theta}_l = \{\boldsymbol{\beta}_l, W\}$ el conjunto de coeficientes para el nodo l de salida, $\boldsymbol{\beta}_l = \{\beta_{l0}, \beta_{l1}, ..., \beta_{lM}\}$ el valor de las conexiones entre capa oculta y capa de salida para dicho nodo, $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_M\}$ el conjunto de coeficientes asociados a los nodos de la capa oculta, $\boldsymbol{w}_j = \{w_{j0}, w_{j1}, ..., w_{jn}\}$ el valor de las conexiones del nodo j de la capa oculta y $\phi_j(\boldsymbol{x}, \boldsymbol{w}_j)$ la salida del nodo j de la capa oculta.

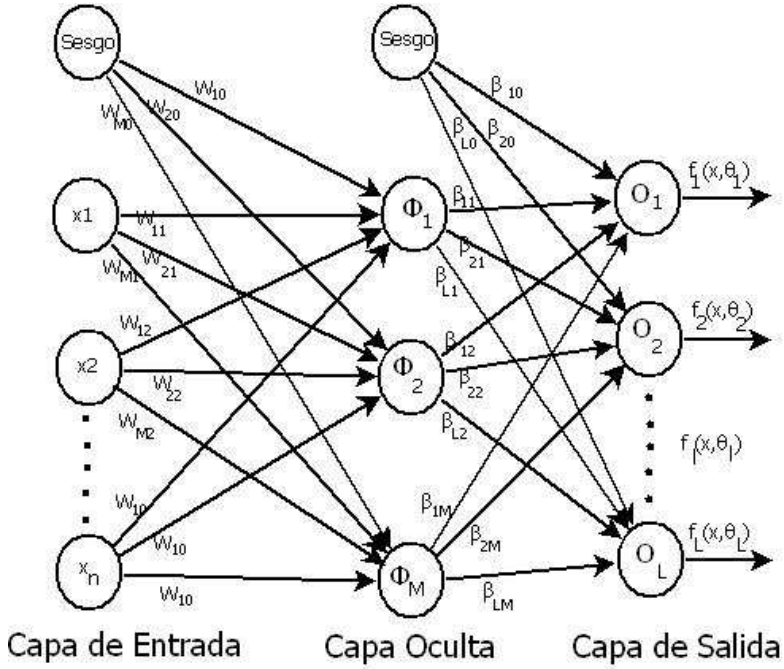


Figura 2.2: Taxonomía de una Red Neuronal.

2.2.1. Redes Neuronales de Nodos de Tipo Sigmoide (US)

Las redes de USs o Perceptrones Multicapa (MLPs) son aquellas que están formadas por nodos tipo sigmoide en su capa oculta, los cuales siguen un modelo aditivo de proyección con la función sigmoide como función de transferencia.

En general, una función sigmoide es una función real de variable real diferenciable, con una primera derivada $no-negativa$ o $no-positiva$ y con solo un punto de inflexión. Además posee dos asíntotas horizontales cuando $t \to \pm\infty$. El modelo más común de nodo sigmoide utiliza la función $f(x) = \frac{1}{1+e^{-x}}$ quedando expresada como se muestra a continuación:

$$\phi_j(\boldsymbol{x}, \boldsymbol{w}_j) = \frac{1}{1 + e^{-(w_{j0}+w_{j1}x_1+w_{j2}x_2+...+w_{jn}x_n)}} = \frac{1}{1 + e^{-(w_{j0}+\sum_{i=1}^{n} w_{ji}x_i)}} \quad (2.3)$$

Las redes US poseen una importante propiedad: la familia de funciones reales que representan pueden aproximar cualquier función continua con suficiente precisión, con tal de que el número de nodos de la capa oculta no esté acotado. Esta propiedad proporciona una sólida base de carácter teórico que ha sido esencial para el desarrollo, el estudio y las aplicaciones de estas redes [8, 9, 10]. Las MLP constituyen sin duda un modelo muy estudiado de RNA tanto del punto de vista teórico [11, 12], de su aplicación concreta en múltiples problemas [13, 14, 15] como de los distintos algoritmos para su entrenamiento [16].

2.2.2. Redes Neuronales de Nodos de Tipo Producto (UP)

Las redes de UP, introducidas por Durbin y Rumelhart en 1989 [17], son aquellas que están formadas por nodos de tipo producto en su capa oculta, los cuales siguen el modelo multiplicativo de proyección con la siguiente función de salida:

$$\phi_j(\boldsymbol{x}, \boldsymbol{w}_j) = x_1^{w_{j1}} x_2^{w_{j2}} ... x_n^{w_{jn}} = \prod_{i=1}^{n} x_i^{w_{ji}} \quad (2.4)$$

siendo en este caso $\boldsymbol{w}_j = \{w_{j1}, w_{j2}, ..., w_{jn}\}$, 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 [18]:

- Como consecuencia del Teorema de Stone-Weierstrass, se ha demostrado que las redes de UP son aproximadores universales [18], (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 y ratios entre las variables.

- Durbin y Rumelhart [17] demostraron que la capacidad de información de un único nodo 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 un nodo de tipo aditivo, siendo N el número de entradas del nodo.

- 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 [19].

- 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 [20, 21, 22, 23, 24, 25, 26]. De esta forma, cuando existen interacciones entre las variables que intervienen en un determinado fenómeno, las Redes Neuronales basadas en UPs permiten obtener modelos más simplificados que los MLPs, en cuanto a necesitar un número de funciones de base inferior.

- Junto a esto, es posible obtener cotas superiores de la dimensión VC (dimensión de Vapnik-Chervonenkis [27]) para redes basadas en UPs similares a las conocidas para las redes MLP [28], lo que supone que poseen una capacidad de generalización similar.

- A diferencia de lo que ocurre con las redes de US, las funciones derivadas parciales del modelo obtenido a partir de una red de UP 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.

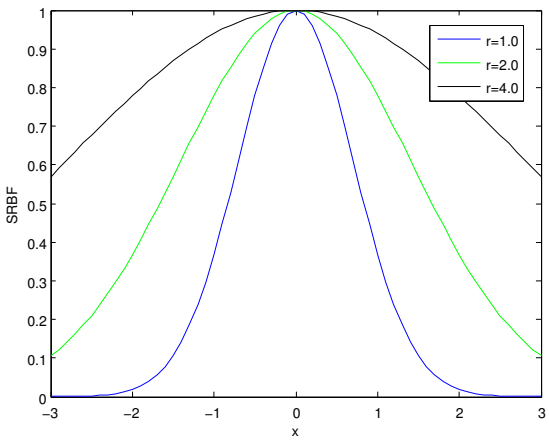
Como contrapartida, las redes de UP presentan un inconveniente importante, la superficie de error asociada es especialmente compleja con numerosos óptimos locales y regiones planas, y por tanto existe una mayor probabilidad de que los algoritmos de búsqueda queden atrapados en alguno de los óptimos locales [29]. 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 [28]. Así, los algoritmos de entrenamiento de redes basados en el gradiente quedan con frecuencia y, de forma especial, en 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 este tipo de redes [30]. Esta dificultad relacionada con el entrenamiento, es una de las razones por las que, a pesar de las ventajas anteriormente señaladas, la teoría de Redes Neuronales basadas en UPs ha tenido poco desarrollo y han sido menos utilizadas como herramientas para problemas de clasificación y de regresión que otros tipos de redes [31].

2.2.3. Redes Neuronales de Nodos de Tipo (RBF)

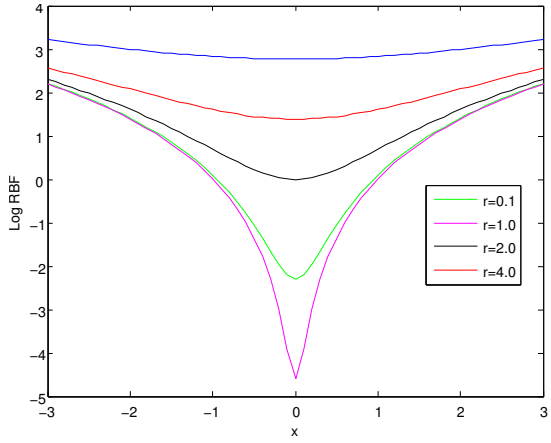
Los nodos ocultos de las Redes Neuronales con nodos RBFs presentan funciones de transferencia de tipo *kernel* [32, 33]. Cada uno de los nodos RBF realiza una aproximación local independiente del espacio de búsqueda, normalmente mediante una campana de Gauss. La capa de salida de tipo lineal unifica el efecto independiente de cada nodo, sumando cada valor obtenido. La intención es que cada nodo esté situado en un entorno del espacio de búsqueda formado por las variables de entrada y además con un radio determinado del entorno. El proceso de aprendizaje de la red neuronal de RBFs consistirá en ir *moviendo* dichos nodos a lo largo del espacio de búsqueda, modificando los centros y los radios para ajustar los nodos a los datos de entrenamiento. La función de transferencia será equivalente a una función de distancia (sección 2.2.3.1), generalmente la euclídea (tomando como centro de la RBF el vector $\boldsymbol{w}_j$) y la función de activación puede tener muchas formas (sección 2.2.3.2), algunas de las cuales se pueden ver en el Cuadro 2.1.

Un aspecto interesante de las redes neuronales con nodos RBF es su capacidad de construir aproximaciones globales [34, 35] a funciones mediante la combinación de nodos centrados en los vectores de pesos de cada nodo.

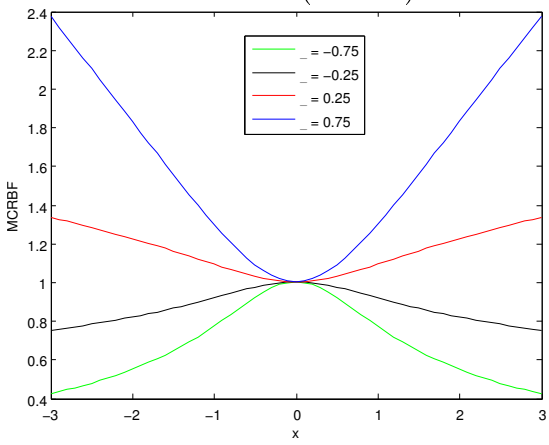
A continuación en la Figura 2.3 se representan en un sistema cartesiano algunas de las funciones de base radial más comunes.



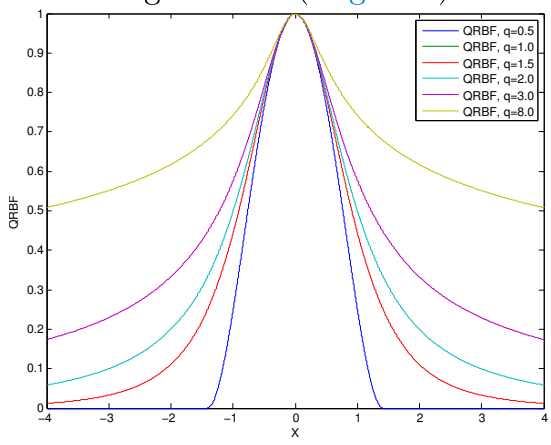
Gaussiana (SRBF)



Logarítmica (LogRBF)



Multicuadráticas Generalizadas (MCRBF)



q-Gaussiana (QRBF)

Figura 2.3: Representación de las diferentes funciones de base radial.

2.2.3.1. Funciones de Transferencia

Las funciones de transferencia utilizadas por los nodos RBF son medidas de similaridad entre un patrón y el vector de coeficientes de cada nodo definido

sobre las variables independientes. La medida más utilizada por los nodos RBF es la distancia euclídea; no obstante pueden ser utilizadas diversas medidas de similaridad como las que se muestran a continuación:

- Euclídea

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{2.5}$$

- Métrica de Diferencia de Valores → Value Difference Metric (VDM) [36]
- Métrica Interpolada de Diferencia de Valores → Interpolated Value Difference Metric (IVDM) [37]
- Métrica Enmarcada de Diferencia de Valores → Windowed Value Difference Metric (WVDM) [37]
- Métrica Heterogénea de Diferencia de Valores → Heterogeneous Value Difference Metric (HVDM) [37]
- Métrica Heterogénea Euclídea-Solapada → Heterogeneous Euclidean-Overlap Metric (HEOM) [37]
- Minkowsky [38]
- Minkowski Generalizada (GM) [39]
- Otras distancias como la distancia Manhattan, Camberra, Chebychev, Cuadrática, Mahalanobis, etc.

2.2.3.2. Funciones de Activación

Las funciones de activación de las RBFs poseen diversas formas. Algunas de las más conocidas se muestran a continuación donde r = $\boldsymbol{x}$ - $\boldsymbol{\mu}$ (es la distancia euclídea de x a μ, ver sección 2.2.3.1):

Nombre	Expresión	Parámetros

Gaussiana	$e^{-\frac{r^2}{2\sigma^2}}$	Con parámetro de normalización $\sigma > 0$
Gaussiana Generalizada	$e^{-(\frac{r}{\theta})^\tau}$	Con parámetro $\theta > 0$
Multicuadráticas	$(r^2 + \sigma^2)^{\frac{1}{2}}$	Con parámetro de normalización $\sigma > 0$
Multicuadráticas Generalizadas	$(r^2 + \sigma^2)^\beta$	Con parámetro de normalización $\sigma > 0$ y $1 < \beta < 0$
Multicuadráticas Inversas	$(r^2 + \sigma^2)^{-\frac{1}{2}}$	Con parámetro de normalización $\sigma > 0$
Multicuadráticas Inversas Generalizadas	$(r^2 + \sigma^2)^{-\beta}$	Con parámetro de normalización $\sigma > 0$ y $1 < \beta < 0$
Cúbicas	r^3	
Spline Radial(TPS)	$r^2 * log(r)$	
Spline Radial Generalizado(TPS)	$r^{(2*i)} \times \log r$	$i \in N$
Logarítmica	$\log(r^2 + \sigma^2)$	Con parámetro de normalización $\sigma > 0$
q-Gaussiana	$(1-(1-q)\frac{r^2}{\sigma^2})^{\frac{1}{1-q}}$	Con parámetro de normalización $\sigma > 0$

Tabla 2.1: Tabla de funciones de activación de nodos RBF.

Las redes RBF a la vista de los diferentes tipos de funciones de activación 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 solo algunos nodos ocultos 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 las redes RBF suele constar de dos etapas, siendo las funciones de base o nodos (función de transferencia + función de activación) aproximadas en primer lugar mediante técnicas de aprendizaje *no supervisado*, y en segundo lugar los pesos entre la capa oculta y la capa de salida son aproximados a través de métodos *supervisados* [40].

2.2.3.3. Redes Neuronales de Nodos de Tipo GRBF

Los nodos GRBF surgen de generalizar el exponente de la Gaussiana de los nodos SRBF introduciendo una mayor flexibilidad a los modelos de red y permitiendo una mayor capacidad de ajuste a los datos. Si bien, la función de activación de los nodos SRBFs es una Gaussiana como se formula a continuación:

$$f(x) = e^{-(\frac{x}{r})^2} \tag{2.6}$$

donde r es el radio del nodo SRBF y x es el resultado de aplicar la función de transferencia del nodo SRBF ante un patrón.
La generalización de los nodos SRBF puede estar sujeta a varios enfoques centrados fundamentalmente en el radio y en el exponente de la Gaussiana como proponen los nodos GRBF sustituyendo el exponente cuadrático por un nuevo parámetro τ como se presenta a continuación:

$$f(x) = e^{-(\frac{x}{r})^\tau} \tag{2.7}$$

La presencia del parámetro τ permite la contracción/relajación de la Gaussiana Generalizada como se representa en la Figura 2.4.

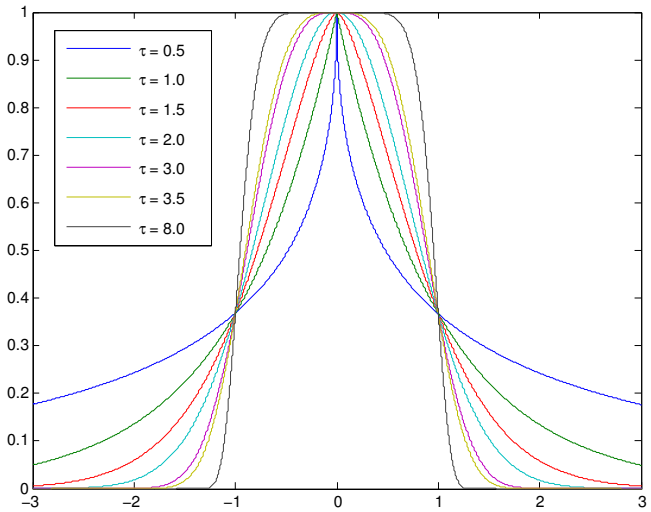


Figura 2.4: Curvaturas de la Gaussiana para distintos valores de τ con $r = 1$.

Una ventaja de los nodos GRBF es la posibilidad de simular comportamientos constantes en un intervalo mediante la asignación de altos valores al

parámetro τ. En la Figura 2.4 se representa la curvatura de una GG con $r = 1$ y $\tau = 8$ (equivalente en el eje positivo de las abscisas) a la función:

$$f(x) = \begin{cases} 1, & \text{si } x < 1 \\ 0, & \text{si } x \geq 1 \end{cases} \tag{2.8}$$

Por otra parte, la presencia de un nuevo parámetro respecto a los nodos SRBF dificulta el entrenamiento de los nodos GRBF pues variaciones en el valor de τ tanto para un mismo valor del radio como para radios distintos no produce cambios en igual magnitud sobre la curvatura de la GG. Esta característica que introduce el parámetro τ dificulta el entrenamiento de los modelos de RNA tanto mediante algoritmos basados en gradiente como mediante algoritmos estocásticos, pues retrasa su convergencia.

Analizando el comportamiento de la curvatura de la GG auxiliándonos de la Figura 2.4 se observa que para un mismo radio iguales cambios en la magnitud del parámetro τ produce cambios no "equivalentes" sobre la GG. Para ello es suficiente con comparar la diferencia entre las los pares de Gaussianas donde $\tau = 0.5$ y $\tau = 1$ respecto a las curvas donde $\tau = 3.0$ y $\tau = 3.5$. Es evidente que el primer par de curvas son muy diferentes entre sí mientras que el segundo par prácticamente se solapan de forma gráfica.

Similar comportamiento presenta la curvatura de la GG para distintos valores de radios. Con el aumento de la magnitud del radio, la curvatura se relaja como se observa en la Figura 2.5 donde las curvas representadas de color azul y verde respectivamente se diferencian en el valor del radio siendo la segunda curva menos contraída que la segunda. De esta forma para mantener la contracción/relajación de la GG (ver curvas roja y azul de la Figura 2.5) es necesario asignar altos valores al exponente. Un análisis funcional de la GG nos indica que *el radio produce una relajación de la misma en la dirección del eje de las abscisas. De este modo se puede concluir aseverando que la curvatura de la GG depende tanto de radio como del valor de τ, luego para calibrar la curvatura es necesario ajustar ambos parámetros.*

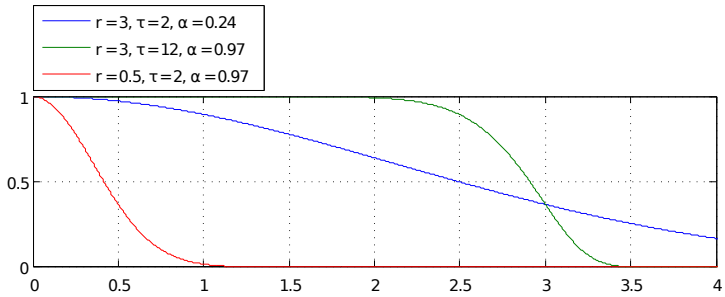


Figura 2.5: Representación de diferentes GRBFs con diferentes valores de exponente y radio.

Un resumen de las características de las GGs y por ende de las GRBFs nos muestra que:

1. Permite la contracción/relajación de la Gaussiana.
2. Permite a los nodos GRBF asignar altos valores (cercanos a uno) de salida a patrones que se encuentran distantes del centro del nodo.
3. La curvatura de la GG depende tanto de radio como del valor de τ luego para calibrar la curvatura es necesario ajustar ambos parámetros.

2.3. Redes Neuronales para Clasificación

En un problema de clasificación supervisada el objetivo es predecir la clase de pertenencia de los patrones del conjunto de generalización, a partir de los valores numéricos obtenidos para las variables y clases observadas del conjunto de entrenamiento. Las variables $x_i, i = 1, 2, ..., n$ están asociadas a un patrón que debe ser clasificado en una de las L clases basándonos en dichas medidas.

Las RNA son modelos que han sido utilizados ampliamente para abordar un problema de clasificación; para ello, en principio, se establece un número de nodos en la capa de salida igual al número de clases del problema a tratar. Se asume que L es finito y que las variables x_i son observaciones aleatorias. Definimos el conjunto de entrenamiento en la forma $D = \{(\boldsymbol{x}_p, \boldsymbol{y}_p); p = 1, 2, ..., N\}$,

donde $\boldsymbol{x}_p = (x_{1p}, ..., x_{np})$ es el vector con las variables de entrada que toma valores en $\Omega \subset \mathbb{R}^{\kappa}$ e y_p representa la clase del patrón p.

Adoptaremos la técnica común de codificación de las clases mediante un vector "1-de-L" para representar la clase del patrón. Estos vectores tienen la forma $\boldsymbol{y} = (y^{(1)}, y^{(2)}, ..., y^{(L)})$, siendo $y^{(j)}$ igual a 1 si el patrón pertenece a la clase j, e igual a 0 en caso contrario. Basándonos en la información aportada por el conjunto de entrenamiento, el objetivo es encontrar una estimación de los coeficientes de la RNA ($\hat{\boldsymbol{\theta}}$) que dado que clasifican bien a los patrones del conjunto de entrenamiento, clasifiquen bien los patrones del conjunto de generalización. Esto equivale a obtener una función de decisión C : $\Omega \rightarrow \{1, 2, ..., L\}$ para clasificar a los individuos. En otras palabras, la función hace una partición $\{D_1, D_1, \ldots, D_L\}$ de Ω, donde D_j es el conjunto de patrones que pertenece a la clase cuyo valor o etiqueta es j. Una clasificación incorrecta se produce si C asigna a un patrón la clase D_j cuando en realidad pertenece a la clase D_i con $D_i \neq D_j$.

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:

$$g_j(x_p, \hat{\theta}_j) = \frac{e^{f_j(x_p, \hat{\theta}_l)}}{\sum_{l=1}^{L} e^{f_l(x_p, \hat{\theta}_l)}} \quad para\ l = 1, \ldots, L \tag{2.9}$$

Siendo L el número de clases del problema, $f_j(\boldsymbol{x}_p, \hat{\boldsymbol{\theta}}_{\boldsymbol{l}})$ la función de salida del nodo j para el patrón $\boldsymbol{x}_p$ y $g_j(x_p, \hat{\theta}_j)$ la probabilidad de que el patrón $\boldsymbol{x}_p$ pertenezca a la clase D_j. La transformación softmax produce estimaciones probabilísticas positivas en todas las salidas, siendo la suma de estas probabilidades "a posteriori" igual a 1, lo que hace que puedan ser interpretadas como la probabilidad "a posteriori" de pertenencia de un patrón a la clase correspondiente. Teniendo en cuenta esta consideración, se puede comprobar que la clase predicha por la RNA $C(\boldsymbol{x}_{\boldsymbol{p}}, \hat{\boldsymbol{\theta}}_{\boldsymbol{j}})$ es la correspondiente al nodo de la capa de salida cuyo valor de salida es mayor. Expresado formalmente, esto quiere decir que:

$$C(x_p, \hat{\theta_j}) = \hat{l},\ donde\ \hat{l} = argmax_i\ g_j(x_p, \hat{\theta_j}) \quad (2.10)$$

$$= argmax_i\ f_j(x_p, \hat{\theta_j}),\ para\ 1 \leq i \leq L \quad (2.11)$$

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 $g_L(x_p, \hat{\theta_L})$ se puede obtener en función del resto como $1 - \sum_{i=1}^{L-1} g(x_p, \hat{\theta_j})$. 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, $f_L(x_p, \hat{\theta_L}) = 0$. Este nodo no será entrenado, para así reducir el número de coeficientes a estimar y la carga computacional del aprendizaje.

2.3.1. Métricas de rendimiento

La evaluación del rendimiento es algo decisivo para obtener una medida del rendimiento de un clasificador, en cuanto a los conjuntos de entrenamiento y generalización. En ocasiones el proceso de diseño u obtención de un clasificador conlleva una serie de etapas que implican un proceso iterativo, donde cada iteración, puede alterar considerablemente el clasificador que se está diseñando. Se requiere, por tanto, una reevaluación del clasificador en cada iteración para determinar cuál ha sido el impacto producido en su rendimiento, premiando, generalmente, aquellos cambios que lo han hecho mejorar.

En la literatura existen diversas medidas para determinar el rendimiento de un clasificador [41, 42, 43], pero antes de pasar a describir algunas de las más utilizadas, es conveniente definir lo que se denomina matriz de contingencia o matriz de confusión de un clasificador.

Dado un problema de clasificación multiclase con L clases, siendo $L \geq 2$, y N patrones de entrenamiento o generalización, la matriz de contingencia

$M(g)$, de dimensión LxL, de un clasificador g, está dada por:

$$\boldsymbol{M(g)} = \begin{array}{c} \text{Clase Predicha} \\ \left(\begin{array}{cccc} n_{11} & n_{12} & \dots & n_{1L} \\ n_{21} & n_{22} & \dots & n_{2L} \\ \dots & \dots & n_{ii} & \dots \\ n_{L1} & n_{L2} & \dots & n_{LL} \end{array} \right) \end{array} \text{Clase Real}$$

donde las filas indican la clase real de pertenencia y las columnas la pronosticada por el clasificador.

Formalmente, la matriz de confusión, $M(g)$, se puede definir como:

$$\boldsymbol{M(g)} = \left\{ n_{ij}; \sum_{i,j=1}^{L} n_{ij} = N \right\}$$

donde n_{ij} representa el número de patrones asignados a la clase i, cuando realmente pertenecen a la clase j. La diagonal corresponde a los patrones correctamente clasificados para las L clases del problema, y los $L(L-1)$ elementos fuera de la diagonal principal corresponden a los errores de clasificación. En consecuencia, los totales por fila indican el número de patrones pertenecientes a cada una de las clases, y la suma de estos totales es el tamaño de la muestra.

Si analizamos la matriz de confusión:

$$\left(\begin{array}{cccc} n_{11} & \dots & \dots & n_{1L} \\ \dots & \dots & n_{ij} & \dots \\ \dots & \dots & \dots & \dots \\ n_{L1} & \dots & \dots & n_{LL} \end{array} \right)$$

tenemos que:

$$\begin{aligned} n_{i\circ} &= \sum_{j=1}^{L} n_{ij} \quad i = 1, ..., L \\ n_{\circ j} &= \sum_{i=1}^{L} n_{ij} \quad j = 1, ..., L \end{aligned}$$

siendo $n_{i\circ}$ el total de patrones asociados a la clase i (suma por filas), y $n_{\circ j}$ el número de patrones predichos por el clasificador que se han clasificado como pertenecientes a la clase j (suma por columnas).

A partir de las expresiones anteriores, se puede deducir que $\sum_{i=1}^{L} n_{i\circ} = N$ y que $\sum_{j=1}^{L} n_{\circ j} = N$. Por tanto, las frecuencias relativas correspondientes al número de patrones de la clase i sobre el total de patrones viene dada por:

$$f_{i\circ} = \frac{n_{i\circ}}{N}$$

y la frecuencia relativa correspondiente al número de patrones que el clasificador ha clasificado como clase j, con respecto al total de patrones viene dada por:

$$f_{\circ j} = \frac{n_{\circ j}}{N}$$

En el caso de clasificación binaria (una clase positiva o éxito y una clase negativa o fracaso), es decir, con un valor de $L = 2$, la matriz de confusión está dada por:

$$\boldsymbol{M(g)} = \begin{pmatrix} TP & FN \\ FP & TN \end{pmatrix}$$

donde:

- TP significa "verdaderos positivos", o número de elementos que son de la clase positiva y que el clasificador ha clasificado como positivos.
- FN significa "falsos negativos", o número de elementos que son de la clase positiva y que el clasificador ha clasificado como negativos.
- FP significa "falsos positivos", o número de elementos de la clase negativa que son clasificados como positivos.
- TN significa "verdaderos negativos", o número de elementos de la clase negativa que son clasificados como negativos.

2.3.1.1. Métricas para problemas binarios

A continuación exponemos las métricas para problemas de clasificación binaria que más se utilizan a la hora de obtener el rendimiento de un clasificador:

Precisión, C: Efectividad global de un clasificador o porcentaje de patrones totales correctamente clasificados. Es una medida que suele proporcionarse en tanto porciento.

$$C = \frac{TP + TN}{TP + FN + FP + TN} = \frac{TP + TN}{N}$$

Precisión positiva, P: Porcentaje de patrones correctamente clasificados de la clase positiva con respecto a todos los elementos que el clasificador predijo como positivos. Dicho de otra forma sería el porcentaje de ejemplos que el clasificador ha predicho como positivos y que realmente son positivos.

$$P = \frac{TP}{TP + FP}$$

Sensibilidad, TPR: También se nombra por *Recall* o *True Positive Rate* (TPR). Es el porcentaje de patrones correctamente clasificados de la clase positiva con respecto al número total de elementos existentes de esa clase, o también se puede decir que es la efectividad del clasificador para identificar los elementos de la clase positiva.

$$TPR = \frac{TP}{TP + FN}$$

Especificidad, Sp: Porcentaje de patrones correctamente clasificados de la clase negativa con respecto al número total de elementos existentes de esa clase, o también se puede decir que es la efectividad del clasificador para identificar los elementos de la clase negativa.

$$Sp = \frac{TN}{FP + TN}$$

Porcentaje de falsos positivos, FPR: También se nombra por *False Positive Rate* (FPR), y se define como el porcentaje de patrones incorrectamente clasificados de la clase negativa con respecto al número total de elementos existentes de esa clase.

$$FPR = \frac{FP}{FP + TN} = 1 - Sp$$

***Fscore*:** Es una medida que se basa en P y en TPR, y se puede interpretar como una media ponderada de ambas. Es la relación entre los elementos positivos y aquellos dados por el clasificador.

$$Fscore = \frac{2}{\frac{1}{P}+\frac{1}{TPR}} = 2 \times \frac{P \times TPR}{P + TPR}$$

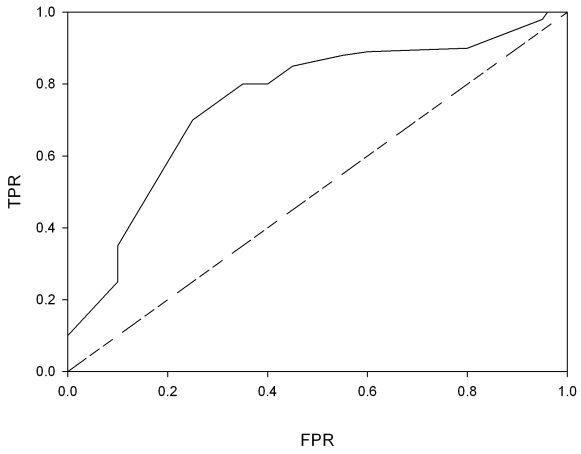
Media geométrica, GM La media geométrica intenta maximizar la precisión en las dos clases que componen un determinado problema de la forma más balanceada posible. Aunque esta medida se puede extender para problemas multiclases su utilización no es usual.

$$GM = \sqrt{TPR \cdot Sp}$$

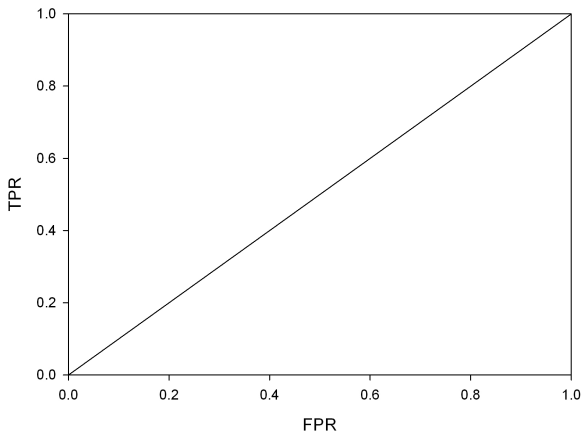
Área bajo la curva operativa característica, AUC: La curva operativa característica ROC [44] es una de las técnicas más usadas habitualmente para la comparación del rendimiento mostrado por dos o más clasificadores binarios. Dichas curvas son una alternativa al uso de la precisión y sus problemas derivados [45, 46], y es una buena técnica para comprobar si un clasificador es mejor que otro en términos de la clase minoritaria en problemas no balanceados donde hay una clase mayoritaria y una minoritaria. Las curvas ROC son gráficas en dos dimensiones, en las que se representan los errores de clasificación de la clase negativa o FPR en el eje horizontal, y la precisión de la clase positiva o TPR en el eje vertical. Las curvas ROC muestran toda la información relacionada con el rendimiento de un clasificador, y permiten una rápida visualización del tipo de relación entre los rendimientos de varios clasificadores. El análisis de la curva ROC, o simplemente análisis ROC, proporciona información para seleccionar los modelos posiblemente óptimos, y es también independiente de la distribución de las clases en la población. El análisis ROC se relaciona de forma directa con el análisis de coste-beneficio en la toma de decisiones.

Dentro de una gráfica ROC, un clasificador domina a otro mientras más arriba y más a la izquierda se encuentre (ver figura 2.6). El mejor método posible de predicción se situaría en un punto en la esquina superior

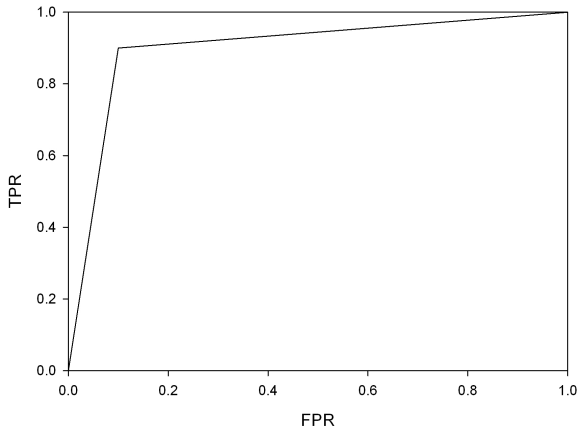
izquierda, o coordenada $(0, 1)$ del plano ROC, representando un 100 % de sensibilidad (ningún falso negativo) y un 100 % de especificidad (ningún falso positivo). Este punto $(0, 1)$ se denomina clasificación perfecta. Por el contrario, una clasificación totalmente aleatoria daría un punto a lo largo de la línea diagonal (línea de no-discriminación), desde el extremo inferior izquierdo hasta el extremo superior derecho.



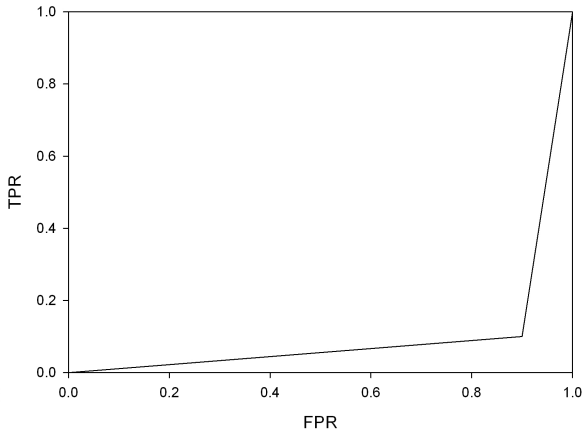
(a) Curva ROC



(b) Clasificador aleatorio



(c) Buen clasificador (alto *TPR* y bajo *FPR*)



(d) Mal clasificador, (bajo *TPR* y alto *FPR*)

Figura 2.6: Ejemplos de curvas ROC.

La extensión de las curvas ROC para dos clases a problemas multiclase es interesante, ya que conferiría los beneficios del análisis ROC a más problemas en reconocimiento de patrones. Se han realizado multitud de aproximaciones, aunque actualmente no hay ningún análisis sobre este

tema que esté completamente consolidado [47]. En [48], se considera un problema de optimización multi-objetivo donde el objetivo es minimizar simultáneamente los $Q(Q-1)$ errores de clasificación dados por los valores no pertenecientes a la diagonal principal de la matriz de confusión, donde el error se define por $\frac{n_{ij}}{n_{io}}$, para $i = 1, 2, ..., L$ y $j \neq i$. De esta forma, el coste computacional crece en función del número de clases. En [49], se propone un algoritmo que a partir de la matriz de confusión de un clasificador identifica las clases independientes y grupos de clases que interactúan entre sí, permitiendo la descomposición de la matriz en una curva ROC con un número bajo de grupos dimensionales. Esto reduce la complejidad computacional considerablemente. La hiper-superficie ROC descompuesta se puede tratar como en el caso ideal (dos clases), permitiendo realizar aproximaciones coste-beneficio y la aproximación de Neyman-Pearson [50], así como el volumen bajo la curva, AUC. Otra manera de trabajar con problemas multi-clase es mediante el uso del volumen sobre la superficie ROC (*Volume under Surface*, VUS) [51] con curvas en 3-D [52, 53]. Por ejemplo, en [54], el concepto de curvas ROC se extiende a cuestiones de diagnóstico médico con tres posibles alternativas; se dibuja una superficie ROC en tres dimensiones para una tarea de decisión tridimensional, añadiendo el VUS sobre la superficie ROC. De esta manera, el VUS resume la precisión global del modelo, similar al AUC de una curva ROC hecho sobre una tarea de clasificación con dos alternativas. La obtención de información en los puntos de la superficie se puede calcular de la misma forma que para curvas ROC bidimensionales; así se pueden comparar tres tipos de curvas ROC bidimensionales, en función de la información de cada superficie.

En resumen el AUC es la capacidad del clasificador para evitar una clasificación falsa, teniendo en cuenta tanto a la clase positiva como a la negativa. Esta medida es una aproximación del área bajo una curva ROC (Receiver Operating Characteristic), y se puede ver como una transformación lineal del índice de Youden [55]. Se puede calcular de manera más precisa mediante el "Algoritmo 2" mostrado en [44]. Concretamen-

te el AUC es una porción del área de un cuadrado de lado la unidad, estando su valor entre 0 y 1.

2.3.1.2. Métricas para problemas multiclase

Para este tipo de problemas, al aumentar la dimensión de la matriz de confusión, es necesario definir nuevas medidas asociadas al comportamiento concreto del clasificador para una determinada clase.

A partir de la matriz de confusión multiclase mostrada en la sección 2.3.1, la sensibilidad de una clase i en problemas multiclase, se define como el número de patrones correctamente predichos en esa clase con respecto al número total de patrones de dicha clase (probabilidad de predecir correctamente un ejemplo de la clase i), y la denotaremos por:

$$S_i = \frac{n_{ij}}{n_{i\circ}}, \qquad i = 1, ..., L.$$

La especificidad, Sp, de una clase i en problemas multiclase, se define como el número de patrones correctamente predichos en esa clase con respecto al número total de patrones predichos por el clasificador para esa clase. Hay que hacer notar que para problemas binarios la especificidad se refiere a la clase negativa, según está definida en la sección 2.3.1.1:

$$Sp_i = \frac{n_{ii}}{n_{\circ j}}, \qquad j = 1, ..., L.$$

Teniendo en cuenta dicha matriz y las definiciones anteriores, las métricas más utilizadas para problemas multiclase son las siguientes:

Precisión, C: Al igual que en el caso de problemas binarios es el porcentaje de patrones correctamente clasificados. Es una medida que suele darse en tanto por ciento:

$$C = \left(\frac{1}{N}\right) \sum_{j=1}^{L} n_{jj}$$

Equivalentemente, C puede expresarse como media ponderada de las sensibilidades:

$$C = \sum_{i=1}^{L} \frac{n_{i\circ}}{N} S_i, \qquad i = 1, ..., L, \tag{2.12}$$

donde los pesos corresponden a las probabilidades "a priori" de cada clase.

Error cuadrático medio, MSE: El error cuadrático medio es una medida que proporciona un promedio del error cometido por un clasificador entre los valores predichos y los reales u observados. Los valores que puede tomar esta medida están entre 0 e ∞. Normalmente se utiliza en problemas de regresión, aunque hay autores que también la usan para clasificación [56, 57].

$$MSE = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \left(\hat{Y}_i - Y_i\right)^2$$

siendo $\hat{Y}_i$ la etiqueta estimada para el patrón i, e Y_i la etiqueta real del patrón i.

En cuanto a los valores que puede tomar la variable $\hat{Y}_i$ y la variable Y_i, tanto en esta métrica como en las siguientes que se exponen dentro de esta sección, dependen de la interpretación que se haga de la(s) salida(s) que proporciona el clasificador que estemos evaluando, y de los valores tomados como verdaderos. Algunos ejemplos posibles de interpretación, aplicados en este caso a RNAs, podrían ser los siguientes: en primer lugar una red neuronal con varias salidas interpretadas de manera probabilística, donde el valor de cada salida indica la probabilidad de pertenencia a una clase determinada; en segundo lugar una red neuronal con una sola salida, comprendida entre 0 y 1, donde la pertenencia a una determinada clase se encuentra en un cierto rango de esa salida; y en tercer lugar una red de varias salidas, donde cada una de ellas indica 0 o 1, dependiendo de si un patrón pertenece o no a una clase determinada. Por tanto, habrá que adaptar los valores $\hat{Y}_i$ y los valores $\hat{Y}_i$, de forma que se pueda aplicar la correspondiente métrica.

Raíz del error cuadrático medio, RMSE: La raíz cuadrada del error cuadrático medio, también proporciona un promedio del error cometido por un clasificador. Sus valores también están entre 0 e ∞.

$$RMSE = \sqrt{\left(\frac{1}{N}\right)\sum_{i=1}^{N}\left(\hat{Y}_i - Y_i\right)^2}$$

Entropía cruzada, E: La entropía cruzada, es una función de error basada en las probabilidades de pertenencia de cada uno de los patrones de un conjunto de datos a cada una de las clases que componen un determinado problema. Los valores que puede tomar esta medida están entre 0 e ∞.

$$E = -\left(\frac{1}{N}\right)\sum_{n=1}^{N}\sum_{l=1}^{L} y_n^{(l)} \log g_l(\boldsymbol{x}_n) \tag{2.13}$$

donde $y_n^{(l)}$ es igual a 1 si el patrón n pertenece a la clase l, y 0 en caso contrario, y donde $g_l(\boldsymbol{x}_n)$ es la probabilidad de que el patrón n pertenezca a la clase l.

***Generalized Squared Correlation*, GC^2:** La correlación cuadrática generalizada se puede considerar como una generalización para problemas multiclase del coeficiente de correlación de Matthews [58] para dos clases.

$$GC^2 = \frac{1}{N\,(L-1)}\sum_{i,j=1}^{L}\frac{(n_{ij} - e_{ij})^2}{e_{ij}}$$

donde $e_{ij} = \dfrac{n_{i\circ}n_{\circ j}}{N}$ es el número esperado de patrones en la posición ij de la matriz de confusión, teniendo como hipótesis que las asignaciones y las predicciones son independientes.

***Macro-Average*, MAVG:** La macro-media se define como la media de las sensibilidades de cada clase, sin considerar la desviación típica. Da la misma importancia o peso a cada una de las clases de un problema.

$$MAVG = \frac{1}{L}\sum_{i=1}^{L} S_i$$

2.3.2. Tests estadísticos para la comparación de algoritmos

El test de Friedman [59] es un test no paramétrico equivalente a las "Repited-Measures" *ANOVA*. Este test elabora un "ranking" de algoritmos para cada base de datos de forma separada, al mejor algoritmo se le asigna el "ranking" 1, al segundo mejor algoritmo se le asigna el "ranking" 2 y así sucesivamente. En caso de empate se le asigna la media de los "rankings" (Ejemplo si dos algoritmos están empatados en el segundo lugar entonces se les asigna a ambos un valor de "ranking" igual a 2.5 producto de calcular el promedio de 2 y 3).

Sea r_i^j el "ranking" del j-ésimo algoritmo dentro un conjunto de k algoritmos, al ser aplicado sobre la i-ésima base de datos dentro de un conjunto de N bases de datos. El test de Friedman [59] compara el promedio de los rankings de los algoritmos, $R_j = \frac{1}{N}\sum_i r_i^j$ y establece la hipótesis nula indicando que todos los algoritmos son equivalentes y por ende sus "rankings" R_j deben ser iguales. El test de Friedman utiliza un estadístico que sigue una distribución χ_F^2 con k-1 grados de libertad; para cuando N $>$ 10 and k $>$ 5 el estadístico se calcula como:

$$\chi_F^2 = \frac{12n}{k(k+1)}[\sum_j R_j^2 - \frac{k(k+1)^2}{4}] \tag{2.14}$$

El estadístico de Friedman posee la característica no deseada de ser conservativa por lo que Iman y Davenport (1980) proponen un mejor estadístico en términos de χ_F^2:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{2.15}$$

el cual se distribuye siguiendo una F-distribución con k-1 y (k-1)(N-1) grados de libertad.

En caso de que la hipótesis nula se rechace, no se puede afirmar que existan diferencias significativas entre los algoritmos por lo que se procede efectuar la prueba de Nemenyi "post-hoc" [60]. Esta prueba compara cada par de algoritmos entre sí, definiendo que dos algoritmos son significativamente

diferentes si el promedio de sus rankings ante cada base de datos difieren en almenos en:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{2.16}$$

donde los valores críticos q_α se determinan a partir del estadítico de rango tipificado dividido por $\sqrt{2}$.

Cuando todos los clasificadores se comparan con un clasificador de control, podemos en lugar de la prueba Nemenyi [60] utilizar uno de los procedimientos generales para el control del error "family-wise" en las pruebas de hipótesis múltiple, tales como la corrección de Bonferroni o procedimientos similares. Aunque estos métodos son generalmente conservadores y pueden tener poco poder, ellos son en este caso, más poderosos que el test de Nemenyi [60]. El uso de la corrección de Bonferroni realiza de $k \times (k-1)/2$, mientras que cuando las comparaciones en comparación con un control (test de Nemenyi [60]) se realizan solo $k-1$ comparaciones. El test estadístico para comparar el $i-simo$ y el $j-simo$ clasificador utiliza el valor de z.

$$z = (R_i - R_j) \sqrt{\frac{k(k+1)}{6N}} \tag{2.17}$$

El valor de z se compara con la probabilidad correspondiente a la tabla de distribución normal, con un valor de α apropiado.

Para contrastar el procedimiento Bonferroni-Dunn paso a paso ("single-step") se establecen procedimientos que analizan el p-valor resultante de comparar cada algoritmo respecto a un algoritmo de control. En 1979 Holm [1] propone un algoritmo que analiza los p-valores ordenados ($p_1 \leq p_2 \leq \cdots \leq p_{k-1}$) comparando cada p_i con $\alpha/(k-i)$. En primer lugar, se comprueba que $p_1 \leq \alpha/(k-1)$ en caso positivo la hipótesis nula se rechaza y se concluye en que existen diferencias significativas entre el algoritmo cuyo p-valor es p_1 y así sucesivamente. En caso de que un p-valor p_m no cumpla la desigualdad $p_m \leq \alpha/(k-m)$ se acepta la hipótesis nula y se detiene el algoritmo conclu-

yendo que los algoritmos $p_m, p_{m+1}, \ldots, p_{k-1}$ no poseen diferencias significativas con el algoritmo de control.

Otra prueba estadística utilizada en el cuerpo de la tesis es el "test de Kruskal-Wallis" (de William Kruskal y W. Allen Wallis) es un método no paramétrico para probar si un grupo de datos proviene de la misma población. Intuitivamente, es idéntico al ANOVA con los datos reemplazados por categorías. Es una extensión del "Mann-Whitney U test" para 3 o más grupos.

Ya que es una prueba no paramétrica, la prueba de Kruskal-Wallis no asume normalidad en los datos, en oposición al tradicional ANOVA. Sí asume, bajo la hipótesis nula, que los datos vienen de la misma distribución. Una forma común en que se viola este supuesto es con datos heterocedásticos.

1. Se calcula el estadístico K: donde:

   - n_i es el número de observaciones en el grupo i.
   - r_{ij} es el rango (entre todas las observaciones) de la observación j en el grupo i.
   - N es el número total de observaciones entre todos los grupos.
   - $\overline{r}_i = \frac{\sum_{j=1}^{n_i} r_{ij}}{n_i}$ es el promedio del grupo i.
   - $\overline{r}$ es el promedio de todos los grupos.

     Note que el denominador de la expresión para K es exactamente $\frac{(N-1)N(N+1)}{12}$.

     Luego: $K = \frac{12}{N(N+1)} \sum_{i=1}^{g} n_i(\overline{r} - \overline{r}_i)^2$

2. Se puede realizar una corrección para los valores repetidos dividiendo K por $1 - \frac{\sum_{i=1}^{G} t_i^3 - t_i}{N^3 - N}$ donde G es el número de grupos de diferentes rangos repetidos, y t_i es el número de observaciones repetidas dentro del grupo i que tiene observaciones repetidas para un determinado valor. Esta corrección hace cambiar a K muy poco al menos que existan un gran número de observaciones repetidas.

3. Finalmente, el p-valor es aproximado por $Pr(\chi^2 \geq K)$ Si algún n_i es pequeño (< 5) la distribución de K puede ser distinta de la chi-cuadrado.

Otra prueba utilizada fue el "Wilcoxon signed-ranks test" [61], es un test no paramétrico alternativo al "t-test" para pares de muestras. Este test crea un "ranking" de diferencias (según sea la métrica) entre 2 clasificadores sobre cada conjunto de datos, ignorando los signos, y compara los rankings para calcular las diferencias positivas y negativas.

Sea d_i la diferencia entre "performance scores" de dos clasificadores sobre la i-ésima base de datos dentro de un conjunto de N. Posteriormente se ordenan las diferencias respecto a su valor absoluto y se les asigna un valor de "ranking" de acuerdo a su valor absoluto, en caso de empates se asigna el promedio de los "rankings" de los clasificadores que están empatados. Sea R^+ la suma de los rankings para las bases de datos en las cuales el segundo algoritmo sobrepasa al primero. Sea R^- la suma de los rankings de forma opuesta. Los rankings con $d_i = 0$ son divididos en 2 subgrupos, si la cantidad es impar, se ignora uno de ellos.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.18}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{2.19}$$

Sea T $= min(R^+, R^-)$, entonces el estadístico :

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{2.20}$$

se distribuye normalmente. Con $\alpha = 0.05$ la hipótesis nula se rechaza cuando $z < -1.86$.

$$z = \frac{T - \frac{1}{4}N(N+1)}{\frac{1}{24}N(N+1)(2N+1)} \tag{2.21}$$

2.4. Entrenamiento de Redes Neuronales

2.4.1. Métodos de Entrenamiento

A continuación, se expone brevemente una visión global de los métodos clásicos de optimización utilizados para el entrenamiento de RNAs, y de los métodos que incorporan algún tipo de heurística o técnica para mejorar el aprendizaje [62].

2.4.1.1. Métodos clásicos

La mayoría de los métodos clásicos utilizados para el entrenamiento de RNAs tienen el problema de que pueden llegar a estancarse en una solución que no sea lo suficientemente idónea para el problema que se esté intentando resolver. Los métodos más comunes son:

Métodos constructivos: Los métodos constructivos comienzan con una red mínima y van añadiendo nodos y conexiones hasta que ésta es capaz de resolver el problema planteado con una determinada precisión. Lo que se hace es adaptar el tamaño de la red al problema en cuestión, teniendo la ventaja de que no se necesita hacer una estimación "a priori" del tamaño de la misma, pero se necesita un experto con suficiente experiencia y un proceso de prueba y error [63, 64]. Este tipo de métodos son susceptibles de caer en mínimos locales [3], además de que solo pueden llevar a cabo la búsqueda en subconjuntos reducidos del espacio de las posibles topologías.

Métodos destructivos o de poda: Los métodos destructivos o de poda [65, 66] parten de una red suficientemente grande (inicialmente puede sobreentrenar y ser demasiado compleja), y sucesivamente se van eliminando nodos y conexiones hasta llegar a una topología en la que la eliminación de un nodo no mejore los resultados. La ventaja de estos métodos es que las redes que se obtienen son pequeñas y por lo tanto más fáciles

de implementar y entrenar. De nuevo, es primordial la experiencia del diseñador, y se necesita un proceso de prueba y error.

Métodos basados en gradiente: El algoritmo de retropropagación (*Back-Propagation*, BP) [67] y sus múltiples variantes [68, 69, 67] (ver siguiente sección) es uno de los más carismáticos y utilizados dentro de los métodos basados en gradiente. Este tipo de metodologías tiene como principal inconveniente el poder caer en óptimos locales y quedar estancado el proceso de aprendizaje, por lo que es necesario establecer de antemano una serie de parámetros, como por ejemplo la tasa de aprendizaje, la inicialización de los pesos (influye en la rapidez del aprendizaje y en el que se alcance o no el mínimo global de la función de error), el número de capas ocultas y el número de nodos en cada capa, entre otros.

Los algoritmos clásicos suelen presentar una serie de problemas [3, 2]:

1. Imposibilidad de calcular el gradiente cuando la función de activación no es derivable.

2. Incapacidad de encontrar un mínimo global si la función de error es multimodal y/o no diferenciable.

3. Ausencia de convergencia de los algoritmos clásicos de entrenamiento cuando el número de dígitos utilizados para representar la función de activación o los pesos de la red (precisión) no es suficientemente grande.

4. Tendencia, por parte de los algoritmos de entrenamiento, a obtener excesivas soluciones no óptimas en cada ejecución.

5. En general, los métodos constructivos y destructivos limitan las arquitecturas disponibles. En algunos de éstos métodos, una vez se ha explorado una arquitectura y se ha decidido que es insuficiente, se adopta una nueva arquitectura, de manera que las antiguas arquitecturas se convierten en inalcanzables. Además, los algoritmos constructivos y destructivos basan su estudio en subconjuntos topológicos, en lugar de basarse en la clase completa de las arquitecturas de red. En consecuencia, estos algoritmos

tienden a optimizar una clase de arquitectura en lugar de ajustar una arquitectura apropiada para un problema determinado.

6. Las deficiencias de los métodos constructivos y destructivos se derivan de métodos inadecuados para la asignación de los componentes estructurales de una red. Se asumen que las restricciones topológicas limitan la complejidad de las interacciones estructurales y paramétricas, e incrementan la probabilidad de encontrar una red suficiente buena para resolver el problema, pero lo ideal sería que esas restricciones proviniesen de la tarea o problema en sí, y no que estuvieran implícitas en el algoritmo.

7. Otros métodos usan una sola modificación estructural predefinida, como añadir un nodo completamente conectado en capa oculta, para generar topologías sucesivas. Tales métodos como la escalada en colina estructurada (*Structural Hill Climbing*), son susceptibles de quedar atrapados en óptimos locales estructurales, y hacen recaer la carga de la tarea de inducción principalmente en la identificación de los valores adecuados de los parámetros, en vez de distribuir la carga uniformemente.

2.4.1.2. Algoritmos Basados en Gradiente

Estos algoritmos utilizan una arquitectura fija y se calcula el valor de los coeficientes, de forma que se minimice una función de coste apropiada. Estos algoritmos son mucho más populares que los anteriormente citados, ya que, además de no producir modelos tan complejos, obtienen redes capaces de afrontar tareas que van más allá del reconocimiento de patrones. Su nombre viene del hecho de que en cada paso se realiza una modificación de los pesos sinápticos en la misma dirección del gradiente de la función de error. Sin embargo, al estar basados en derivadas, las funciones de transferencia deben ser cuidadosamente escogidas, siendo necesario que sean continuas y derivables. Por lo tanto, abandonan el uso del nodo de tipo escalón.

El Algoritmo de Retropropagación (BP)

El conocido algoritmo de retropropagación (BackPropagation, BP) es un ejemplo de este tipo de métodos. Se basa en la inicialización de los pesos con valores

aleatorios y en el cálculo iterativo de la estimación de los pesos, realizado en dos pasadas que recorren la red en direcciones opuestas [70] . El principal inconveniente de este algoritmo es que la minimización de la función de coste es un problema no lineal y, por lo tanto, es común que quede atrapado en mínimos locales.

Los parámetros más importantes del algoritmo básico hacen referencia, entre otras cuestiones, al criterio de finalización del algoritmo [71] , a la constante μ de velocidad de aprendizaje o al tipo de minimización (minimización "*batch*" del error producido por todos los patrones en cada iteración o minimización "*online*" de cada uno de los patrones por separado).

El algoritmo básico adolece de la dificultad común a todo método basado en gradiente: la convergencia de la función de coste es lenta. Para evitar este problema, se propone el uso de un *término de momento* que suaviza el comportamiento oscilatorio del algoritmo [72]. Desde sus inicios el Algoritmo de Retropropagación ha sido objeto de estudio para aumentar su poder de generalización [73] así como su aplicación a los distintos modelos de RNAs [74, 75, 76, 77].

El Algoritmo iRProp+

El algoritmo *iRprop+* (*Improved Resilient Back-propagation*) [78] es una mejora del original Rprop [79]. El algoritmo Rprop (*Resilient Back-propagation*) [79]. Como método de BL, es una de las mejores técnicas conocidas en términos de velocidad de convergencia, precisión y robustez con respecto a sus parámetros. *iRProp+* es un procedimiento basado en gradiente, difiriendo de las técnicas clásicas de propagación hacia atrás del error en que, las derivadas parciales de la función error, solo se usan para determinar el sentido en que se deben corregir los parámetros a ajustar, pero no las magnitudes de los ajustes. El modelo de actualización de cada parámetro x_i de una función $f(\boldsymbol{x})$ viene dado por $x_i^{(t+1)} = x_i^{(t)} + \triangle x_i^{(t+1)}$, donde $\triangle x_i^{(t+1)}$ se estima con una función de cambio de signo de la derivada del error entre las iteraciones (t) y $(t-1)$, y en función del tamaño de paso para los parámetros $(\triangle_i)$, tal como se realiza tradicionalmente con las técnicas basadas en gradiente. Así, si el signo de la derivada no cambia en las dos últimas iteraciones, el tamaño de paso se

aumenta. Si la derivada es cero no se modifica y si cambia de signo se disminuye. *iRProp+* aplica una estrategia de *backtracking* para decidir la dirección de corrección del valor de los parámetros, concretamente la idea es que la actualización de los parámetros dependa de la evolución del error. De esta manera, el esquema de entrenamiento combina información local con información global (por ejemplo el valor del error en cada iteración) a la hora de decidir si debe modificar los parámetros. Un cambio de signo en una derivada parcial significa que se ha saltado un mínimo local. Se ha demostrado mediante problemas de prueba o *benchmarks* [78], que *iRProp+* consigue mayor rendimiento que su versión original.

El esquema general del algoritmo *iRProp+* se resume a continuación:

Algoritmo iRprop+

para todo $x_i \in \boldsymbol{x}$ **hacer**
  si $\frac{\partial E^{(t-1)}}{\partial x_i} * \frac{\partial E^{(t)}}{\partial x_i} < 0$ **entonces**
    $\Delta_i^{(t)} = max(\Delta_i^{(t-1)} * \eta^-, \Delta_{min})$

    si $E^t > E^{t+1}$ **entonces**
      $x_i^{(t+1)} = x_i^{(t)} - \Delta x_i^{(t+1)}$
    fin si
    $\frac{\partial E^{(t)}}{\partial x_i} = 0$
  si no si $\frac{\partial E^{(t-1)}}{\partial x_i} * \frac{\partial E^{(t)}}{\partial x_i} > 0$ **entonces**
    $\Delta_i^{(t)} = min(\Delta_i^{(t-1)} * \eta^+, \Delta_{max})$
    $\Delta w_i^{(t)} = -sign(\frac{\partial E^{(t)}}{\partial x_i}) * \Delta_i^{(t)}$
    $x_i^{(t+1)} = x_i^{(t)} + \Delta x_i^{(t+1)}$
  si no
    $\Delta w_i^{(t)} = -sign(\frac{\partial E^{(t)}}{\partial x_i}) * \Delta_i^{(t)}$
    $x_i^{(t+1)} = x_i^{(t)} + \Delta x_i^{(t+1)}$
  fin si
fin para

Donde $\boldsymbol{x}$ es el vector de parámetros a ajustar, E es la función a optimizar, $\Delta_i^{(t)}$ es la magnitud de la variación a realizar sobre x_i en la iteración t, y Δ_{min}

y Δ_{max} son las cotas mínima y máxima de $\Delta_i^{(t)}$. Los valores de los parámetros según [78] son: $\eta^+ = 1.2$, $\eta^- = 0.5$, $\eta^+ = 1.2$, $\Delta_i^{(0)} = 0.0125$, $\Delta_{min} = 0$, $\Delta_{max} = 50$.

Generalmente el algoritmo *iRProp+* se aplica en el entrenamiento de RNAs mediante la optimización del MSE. Teniendo en cuenta que el objeto de estudio de esta tesis son los AH para la clasificación de patrones la función de error utilizada es la función de entropía cruzada, de forma que el vector gradiente viene dado por:

$$\bigtriangledown E(\boldsymbol{\beta}^q, \boldsymbol{w}) = \left(\frac{\partial E}{\partial \boldsymbol{\beta}^q}, \frac{\partial E}{\boldsymbol{w}} \right)$$

donde $\boldsymbol{\beta}^q, \boldsymbol{w}$ son los vectores de parámetros asociados a una red neuronal.

Sea Θ cualquiera de los parámetros de $\boldsymbol{\beta}^q$ y $\boldsymbol{w}$, entonces

$$\frac{\partial E}{\partial \boldsymbol{\Theta}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{q=1}^{Q} y_n^q \frac{1}{g_q\left(\boldsymbol{x}_n, \Theta\right)} \frac{\partial g_q\left(\boldsymbol{x}_n, \boldsymbol{\Theta}\right)}{\partial \boldsymbol{\Theta}}$$

$$\frac{\partial g_q\left(\boldsymbol{x}_n, \boldsymbol{\Theta}\right)}{\partial \boldsymbol{\Theta}} = \frac{1}{\left(1 + \sum_{q=1}^{Q-1} e^{f_q}\right)^2} \left\{ e^{f_q} \frac{\partial f_q}{\partial \boldsymbol{\Theta}} \left(1 + \sum_{q=1}^{Q-1} e^{f_q} \right) - e^{f_q} \sum_{q=1}^{Q-1} e^{f_q} \frac{\partial f_q}{\partial \boldsymbol{\Theta}} \right\}$$

$$\frac{\partial g_q\left(\boldsymbol{x}_n, \boldsymbol{\Theta}\right)}{\partial \boldsymbol{\Theta}} = g_q \frac{\partial f_q}{\partial \boldsymbol{\Theta}} - g_q^2 e^{-f_q} \sum_{q=1}^{Q-1} e^{f_q} \frac{\partial f_q}{\partial \boldsymbol{\Theta}}$$

Donde N representa al número de patrones y Q es el número de nodos de la capa de salida. Finalmente, para los pesos de la capa de salida de la RNA se tiene que:

$$\frac{\partial f_q}{\partial \beta_0^k} = \delta_{kq} = \begin{cases} 0 & k \neq q \\ 1 & k = q \end{cases}, \quad \frac{\partial f_q}{\partial \beta_s^k} = \begin{cases} 0 & q \neq k \\ \boldsymbol{\sigma}_s\left(\boldsymbol{w}, \boldsymbol{x}\right) & q = k \end{cases}$$

y para la capa de salida que

$$\frac{\partial f_q}{\partial w_s^t} = \boldsymbol{\beta}_s^q \frac{\partial \boldsymbol{\sigma}\left(\boldsymbol{w}, \boldsymbol{x}\right)}{\partial w_s^t}, \quad \text{donde} \quad s = 1, 2, \cdots, m; t = 1, 2, \cdots, k$$

Siendo β_0^k el bias del nodo k, β_s^k el peso del arco que une al nodo s de la capa oculta con el nodo k de la capa de salida y w_s^t el peso del arco que une al nodo de entrada t con el nodo s perteneciente a la capa oculta.

Un aspecto importante a tener en cuenta en la utilización del *iRProp+* en la optimización de la función de error en modelos de RNA es el valor inicial del parámetro $\Delta_i^{(0)}$ (igual a 0.0125), pues en superficies de error con alta multimodalidad (como es el caso de la entropía cruzada) el valor de $\Delta_i^{(0)}$ puede llevar a la solución a un espacio cóncavo menos prometedor Figura 2.7. Un análisis del algoritmo *iRProp+* de forma aislada a partir de un punto aleatorio de la superficie de error nos indica que es posible que la solución transite a otra región como ocurre en la Figura 2.7 donde el incremento inicial es más grande de lo necesario. En este caso sería equiprobable que esta nueva región sea más o menos prometedora. Sin embargo cuando el punto inicial es calculado por un algoritmo estocástico, es muy probable que si dicho punto transita a otra superficie, esta sea menos prometedora.

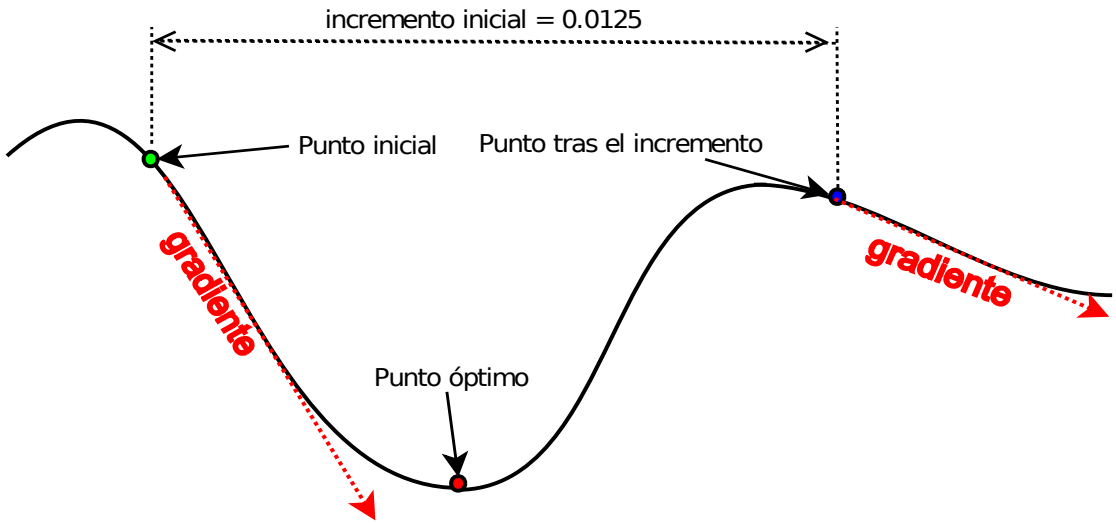


Figura 2.7: Representación del problema presente el el algoritmo iRprop+ en su iteración inicial.

Este problema en superficies multimodales como las funciones de error sobre modelos de RNA puede ser abordado mediante la modificación del algoritmo como se detalla a continuación:

Algoritmo iRprop+

Retroceder = FALSO
para todo $x_i \in \boldsymbol{x}$ **hacer**
si $\frac{\partial E^{(t-1)}}{\partial x_i} * \frac{\partial E^{(t)}}{\partial x_i} < 0$ o Retroceder = VERDADERO **entonces**
Retroceder = FALSO
$\Delta_i^{(t)} = max(\Delta_i^{(t-1)} * \eta^-, \Delta_{min})$

si $E^t > E^{t+1}$ **entonces**
$x_i^{(t+1)} = x_i^{(t)} - \Delta x_i^{(t+1)}$
fin si
$\frac{\partial E^{(t)}}{\partial x_i} = 0$
si no si $\frac{\partial E^{(t-1)}}{\partial x_i} * \frac{\partial E^{(t)}}{\partial x_i} > 0$ **entonces**

si $E^t > E^{t+1}$ **entonces**
Retroceder = VERDADERO
si no
$\Delta_i^{(t)} = min(\Delta_i^{(t-1)} * \eta^+, \Delta_{max})$
$\Delta w_i^{(t)} = -sign(\frac{\partial E^{(t)}}{\partial x_i}) * \Delta_i^{(t)}$
$x_i^{(t+1)} = x_i^{(t)} + \Delta x_i^{(t+1)}$
fin si
si no
$\Delta w_i^{(t)} = -sign(\frac{\partial E^{(t)}}{\partial x_i}) * \Delta_i^{(t)}$
$x_i^{(t+1)} = x_i^{(t)} + \Delta x_i^{(t+1)}$
fin si
fin para

Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt (LM) [80], proporciona una solución numérica al problema de minimizar una función, por lo general no lineal, en un espacio de parámetros de la función. Estos problemas surgen especialmente en la optimización por el método de los mínimos cuadrados de ajuste

de funciones no lineales en problemas de regresión. El LM propone una actualización iterativa que puede ser resumida como:

$$x_{k+1} = x_k - \alpha(\lambda I + H_f)\nabla f \tag{2.22}$$

donde α y λ son números positivos e $\boldsymbol{I}$ es la matriz identidad, H_f es el Hessiano de f y ∇f es el gradiente de la función f. El objetivo es seleccionar un valor de λ de forma que $\lambda I + H_f$ sea definida positiva. La ecuación anterior se aproxima al método del descenso más rápido si $\lambda \to \infty$, y al método de Newton cuando $\lambda \to 0$.

El método LM es más robusto que el de Gauss-Newton (GNA), lo que significa que en muchos casos se encuentra una solución, incluso si comienza muy lejos del mínimo final. Por otra parte, para funciones continuas con pocas curvaturas y razonables parámetros de partida, el LM suele ser un poco más lento que el GNA. LM también puede ser visto como una versión del Gauss-Newton utilizando regiones de confianza.

2.4.1.3. Métodos Heurísticos

La utilización de métodos de naturaleza heurística [81, 82, 83] para el entrenamiento de RNAs surge en la década de los 90 como solución a los problemas de entrenamiento presentados por los algoritmos clásicos. A continuación se citan algunos de los métodos heurísticos utilizados con más frecuencia para el aprendizaje de RNAs:

- Variantes heurísticas de la retropropagación del error [84], BP con velocidad adaptativa y/o optimización de los parámetros de aprendizaje [85], la regla *delta-delta* o la regla *delta-bar-delta* [86], el algoritmo *Quickprop* [87], el algoritmo *iRProp+* [88] y muchos otros más.
- Algoritmos de gradiente conjugado como el *FRCG* de Fletcher-Reeves [89], el *PRCG* de Polak-Ribiére [90] o el de Powell-Beale [91].
- Algoritmos de tipo *Quasi-Newtonianos* como el *DFP* de Davidon-Fletcher-Powell [92] y el *BFGS* de Broyden-Fletcher-Golfarb-Shannon [93].

- Algoritmos que modifican el método *Quasi-Newton* como el *LM* de Levenberg-Marquard [94].

- Métodos heurísticos no basados en gradiente como el enfriamiento simulado (*Simulated Annealing*, SA), la búsqueda tabú (*Tabu Search*, TS), la búsqueda dispersa (*Scatter Search*, SS), los enjambres de partículas (*Particle Swarm*, PS) y los algoritmos evolutivos (*Evolutionary Algorithms*, AEs), siendo éstos últimos los que más impacto han tenido en el diseño de RNAs. Este tipo de heurísticas se han utilizado muy frecuentemente para el aprendizaje en RNAs [62, 95].

Optimización de colonias o enjambres de partículas (Particle Swarm Optimization) (PSO).

Se trata de un algoritmo de optimización global basado en poblaciones. Los individuos de la población, denominados partículas, se agrupan en colonias o enjambres. Cada partícula representa una posible solución del problema de optimización. Cada partícula se mueve a través de un espacio N-dimensional donde N es el número de parámetros a estimar en el modelo a optimizar. Durante su "movimiento" los parámetros del modelo se van ajustando en el espacio de búsqueda, en función de su posición y de la de las partículas vecinas. El efecto es que las partículas "vuelan" hacia el mínimo, realizando la búsqueda alrededor de una amplia área en un entorno óptimo. La proximidad de cada partícula al óptimo se mide según los valores de una función de aptitud previamente fijada y que dependerá de la naturaleza del problema. En el problema que nos ocupa, el entrenamiento de una red neuronal, cada partícula representa el vector de pesos de la red y la función de aptitud puede ser modelada de diferentes formas (ejemplo: a partir del error cuadrático medio de la red sobre el conjunto de entrenamiento). Más detalles sobre este método de optimización pueden encontrarse en "*New Ideas in Optimization*" [96] y en "*Computational Intelligence PC Tools*" [97].

Algunas variantes del método anterior han sido diseñadas para el entrenamiento de redes con nodos de tipo radial [98, 99].

Algoritmo genético (AG)

Un algoritmo genético es una simulación computacional de los procesos de selección, reproducción, mutación y cruce de los individuos de una población en un ambiente natural a lo largo de sucesivas generaciones. En un algoritmo genético, los individuos de la población compiten para sobrevivir. Los individuos están formados en este caso por el vector de pesos de la red, la función de aptitud de cada individuo se establece de varias formas fundamentalmente a partir de una función de error de la red neuronal ante el conjunto de entrenamiento. La aplicación de los algoritmos genéticos al diseño y entrenamiento de redes neuronales con nodos de tipo radial [100, 101, 102] ha validado la utilización de los mismos.

2.4.1.4. Algoritmos Evolutivos (AEs)

Sin embargo, los algoritmos heurísticos que más impacto han tenido, relacionados con diversas arquitecturas y problemas de Redes Neuronales, han sido los basados en modelos evolutivos. El concepto de Algoritmo Evolutivo (AE) describe un conjunto de sistemas para resolver problemas mediante el ordenador, que usan un modelo computacional similar a los procesos evolutivos de 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. Algunas aproximaciones para el entrenamiento de Nodos de Tipo Radial [103, 104, 105, 106, 107, 108, 109] demuestran la competitividad de este entrenamiento frente a otros algoritmos clásicos [110, 111, 112, 113, 114]. De igual manera que los algoritmos evolutivos simulan la competencia de los individuos en la naturaleza, otros comportamientos como la cooperación [115] entre los mismos son simulados para el entrenamiento de Redes Neuronales de Base Radial y RNAs en general.

2.5. Entrenamiento de Redes Neuronales de Base Radial

El entrenamiento de las Redes Neuronales de Base Radial es abordado en dos líneas fundamentales:

- Entrenamiento por capas: donde se entrena en primer lugar la capa de nodos (SRBF) haciendo uso generalmente de técnicas de clustering; y en segundo lugar se entrenan los nodos de la capa de salida bajo el supuesto de que los parámetros de los nodos en la capa oculta ya están ajustados.

- Entrenamiento simultáneo: donde todos los parámetros de los nodos de la red son estimados de forma simultánea.

El teorema de "*No Free Lunch*" propuesto por Wolpert y Macready permite asegurar que *no puede existir un algoritmo capaz de resolver todos los problemas, en media, mejor que cualquier otro algoritmo* [116]. Este teorema motiva a que se sigan buscando nuevos algoritmos de optimización. La utilización de métodos de naturaleza heurística para entrenamiento de Redes Neuronales surge en la década de los 90 [117]. Algunos de los métodos heurísticos que se han utilizado para entrenar Redes Neuronales han sido, entre otros, el método de enfriamiento simulado [118], la búsqueda tabú [119] o, más recientemente, la optimización de enjambre de partículas [120] . También se han propuesto diversas variantes heurísticas del algoritmo BP. Por ejemplo, la regla *delta-delta* o la regla *delta-bar-delta* [86]. Otras alternativas se basan en mejorar la velocidad de convergencia [121], en el uso del gradiente conjugado [122], el algoritmo RProp [79, 123] antes mencionado, algoritmos de tipo Newtoniano [124] (entre los que encontramos el QuickProp [125]) o Quasi-Newtoniano (algoritmo Davidon-Fletcher-Powell [92]) y métodos basados en el algoritmo de Levenberg-Marquardt [80].

2.5.1. Entrenamiento de los Nodos de Tipo Radial SRBF

El entrenamiento de las redes SRBF se encarga de estimar los centros y demás parámetros de las mismas. Los centros de las funciones de base de la red definen la ubicación de las mismas en el espacio de entrada. Cada una de ellas debería estar situada en una zona del espacio de entrada en las que haya ejemplos de entrada/salida que la activen, de forma que contribuya a la salida de la red. Una función que no se active por ningún ejemplo de entrenamiento

solamente contribuirá a aumentar la complejidad de la red inútilmente, además de producir un sistema de ecuaciones mal condicionadas para el posterior cálculo de los pesos de la red.

La estimación de los parámetros de las SRBF ha llevado a la utilización de distintos algoritmos, de entre ellos sobresalen el algoritmo Batch K-Medias [126], OnLine Kmedias [127], Scalable K-Medias [128], C-Mean [129] , Fuzzy C-Mean [130, 114] y el Mean-Traking Clustering Algorithm [131]. Estos algoritmos permiten determinar conglomerados representados por centroides cuyas coordenadas son utilizadas para la inicialización de los parámetros de las SRBF.

2.5.2. Entrenamiento de la Capa de Salida

El entrenamiento de la capa de salida se realiza bajo el supuesto de que los parámetros pertenecientes a los nodos de la capa oculta se encuentren fijos. Para la estimación de los coeficientes de la capa de salida se utilizan diferentes métodos, generalmente algoritmos basados en gradiente sobre una función de error. Entre los métodos más comunes se encuentran la descomposición de Cholesky [132], la descomposición de valores singulares (Singular Values Descomposition, SVD) [133], el método de mínimos cuadrados ortogonales (Ortogonal Least Suare, OLS) [134] y otros ya clásicos como el Gradiente Descendente sobre la función de error.

2.5.2.1. Gradiente Descendente

El algoritmo de Gradiente Descendente propone la optimización de una función mediante la actualización sucesiva de un punto inicial el cual se encuentra situado en el espacio de variables de entrada. Dicha actualización se realiza mediante la substracción de un pequeño diferencial que es proporcional al gradiente de la función en dicho punto. La actualización del punto se puede resumir de la siguiente forma:

$$x_{k+1} = x_k - \alpha \times f'(x_k) \tag{2.23}$$

donde α es un factor de escalamiento. Una aplicación directa de este u otro método de optimización es la minimización de una función de error definida sobre el espacio de variables compuesto por los pesos a optimizar en una Red Neuronal con activación hacia delante.

Capítulo 3

ALGORITMO EVOLUTIVO PARA EL ENTRENAMIENTO Y DISEÑO DE REDES NEURONALES RBF

Desde el surgimiento de los problemas de optimización, la comunidad científica se ha dado a la tarea de diseñar algoritmos que permitan optimizar los modelos matemáticos. Primeramente surgieron un conjunto de algoritmos muy robustos conocidos como algoritmos clásicos. Si bien estos algoritmos clásicos de optimización son efectivos en entornos continuos, lineales o cuadráticos, unimodales y convexos, los algoritmos modernos como los Algoritmos Evolutivos (AEs) se muestran más eficaces en entornos discontinuos, no diferenciables, no convexos, multimodales y con problemas de ruido o faltos de información a priori [135]. Los algoritmos de optimización clásica utilizan reglas deterministas para pasar de un punto del espacio de búsqueda al siguiente y comienzan desde un único punto. Sin embargo, los AEs realizan una búsqueda paralela desde distintos puntos y las transiciones tienen ciertas componentes aleatorias.

En el presente capítulo se describen las principales características de la computación evolutiva. Se describe una metodología propuesta por Deb [136] donde se identifican las fases que caracterizan a cada algoritmo evolutivo. Posteriormente se describe la arquitectura propuesta por Martínez-Estudillo [137] para la evolución de modelos de RNA definiendo el algoritmo NNEP que

servirá de base a los algoritmos propuestos en los capítulos 7 y 8.

3.1. Computación Evolutiva

La Computación Evolutiva es una de las múltiples ramas de la Inteligencia Artificial cuya filosofía se basa en desarrollar algoritmos de búsqueda estocástica mediante técnicas inspiradas en la teoría de la evolución. La evolución es un proceso de optimización, donde el objetivo es mejorar la habilidad de los individuos para sobrevivir adaptándose al medio. Los organismos tienen determinadas características que influyen en su capacidad de adaptación y reproducción. Estas características (genes), se encuentran representadas en los cromosomas. Tras la reproducción sexual se produce una mezcla de material genético generando un hijo con la información genética de los padres. Es de esperar que los hijos vayan heredando las mejores características de sus padres. No obstante, el proceso de la selección natural asegura que los individuos más aptos vayan perpetuándose a lo largo de la evolución, siendo los mejores individuos de la población cada vez más aptos.

Ocasionalmente, se producen mutaciones en los cromosomas que causan cambios en las características de los individuos y que, algunas veces, suelen mejorar las capacidades de los individuos.

La evolución natural puede verse como un proceso de búsqueda aleatorio dentro del espacio de búsqueda de los posibles cromosomas con el objetivo de encontrar un cromosoma que mejore cierta característica. Desde este punto de vista, un algoritmo evolutivo se entiende como un proceso de búsqueda aleatoria de la mejor solución a un problema dado.

La idea que subyace bajo estos algoritmos es siempre la misma: dada una población de individuos en un entorno bajo condiciones de presión selectiva (sobreviven los más aptos) ir mejorando la aptitud de los individuos de la población. Dada una función capaz de medir la capacidad o calidad de los individuos es fácil generar aleatoriamente un conjunto de posibles soluciones, y aplicar sobre éstas la función denominada de aptitud, como una medida abstracta para ver la capacidad de adaptación al medio de cada individuo. Según

esta medida podemos ordenar los individuos de mejor a peor. Los mejores individuos, como regla general, se elegirán como candidatos, también denominados padres, para formar la siguiente generación aplicando técnicas de recombinación o cruce y/o mutación. El cruce o recombinación es un operador que se aplica a más de un padre y cuyo cometido es obtener uno o más individuos -hijos- mediante el intercambio de información. Por otra parte, la mutación siempre se aplica sobre un padre y se obtiene un hijo, modificando de alguna manera al padre. Aplicando dichos operadores a partir del conjunto de individuos padres obtenemos otro conjunto de individuos hijos que compiten entre sí y/o con los padres para formar una nueva población de individuos en la siguiente generación. Este proceso se repite hasta llegar a obtener un individuo con una determinada aptitud suficientemente aceptable, o bien, cuando se ha llegado a un límite computacional fijado de antemano.

La computación evolutiva, ofrece ventajas prácticas para el investigador que se enfrenta a difíciles problemas de optimización. Estas ventajas son múltiples, incluyendo la simplicidad del planteamiento, su respuesta contundente a las cambiantes circunstancias, su flexibilidad, y muchas otras facetas. Un algoritmo evolutivo se puede aplicar a problemas donde las soluciones heurísticas no están disponibles o en general los resultados son poco satisfactorios.

Como resultado, los algoritmos evolutivos han recibido recientemente un mayor interés, en particular con respecto a la forma en que se pueden aplicar para resolver problemas prácticos. Por lo general, agrupados bajo el término computación evolutiva o algoritmos evolutivos, se encuentran los algoritmos genéticos [138], las estrategias de evolución [139, 140], la programación evolutiva [141], y la programación genética [142]. Todos ellos comparten una base conceptual común de simular la evolución de las estructuras de los individuos a través de procesos de selección, mutación, y la reproducción. En comparación con otras técnicas de optimización global, los algoritmos evolutivos (AE) son fáciles de implementar y muy a menudo ofrecen soluciones adecuadas.

3.2. Características de los Algoritmos Evolutivos

El funcionamiento básico de un algoritmo evolutivo se describe como un algoritmo estocástico que mantiene una población de individuos $P(t) = x_1(t), \ldots, x_n(t)$ para cada iteración o generación t. Cada individuo representa una solución potencial al problema a resolver y se implementa mediante alguna estructura de datos. Cada solución *x(t)* es evaluada para dar alguna medida de su ajuste o adaptación. A continuación, se forma una nueva población *(generación t + 1)* mediante la selección de los individuos más adaptados, algunos de los cuales son modificados mediante la aplicación de operadores genéticos de recombinación y mutación para formar nuevas soluciones. La estructura de un programa evolutivo según [143] es la siguiente:

Algoritmo Evolutivo
1. t=0.
2. Inicializa P(t).
3. Evalúa P(t)
4. Mientras no se cumpla alguna condición de parada
t=t+1
Selecciona P(t) a partir de P(t-1)
Modifica P(t)
Evalúa P(t)
Fin del Mientras.

Para llevar a cabo la "Modificación de $P(t)$" se aplican fundamentalmente dos tipos de operadores: en primer lugar los *operadores unarios*, en el sentido de que actúan sobre un único individuo, produciéndole un pequeño cambio para introducir innovación dentro de la población y que suele denominarse mutación; y otros *operadores binarios*, habitualmente denominados de cruce, que crean nuevos individuos a partir de la combinación de partes de otros individuos. La aptitud o fitness de un individuo refleja su valor con respecto a una función objetivo particular a optimizar. El operador de selección impone una búsqueda dirigida dentro del proceso de evolución, eligiendo a los mejores individuos para que sobrevivan y se reproduzcan.

Un elemento de vital importancia es la codificación de los individuos. La codificación es el establecimiento de una aplicación entre el "mundo real" y el "mundo del algoritmo evolutivo". Se trata de buscar una representación de cada solución real para que el AE trabaje en el espacio de dicha representación y se puedan aplicar los operadores de búsqueda o modificación. Desde el punto de vista biológico, podemos hablar de fenotipo (individuo real) y genotipo (codificación cromosómica de dicho individuo). Este paso es fundamental dentro del diseño del AE pues afecta de manera directa a la eficiencia y complejidad del algoritmo.

En los problemas de optimización, la representación de los coeficientes de la función (pesos sinápticos en el caso de las RNA), fue inicialmente afrontada mediante codificación binaria, restringiendo la precisión del algoritmo a la longitud de los cromosomas considerados.

La utilización de la codificación real comienza a aparecer en aplicaciones particulares y, posteriormente, en algunos problemas de optimización. Hasta 1991 no se hacen los primeros estudios teóricos, ya que la codificación real causaba cierta controversia en los investigadores familiarizados con la teoría clásica de los AG. Se demostró entonces que la codificación real tiene mejor comportamiento que la binaria, al menos para este tipo de problemas [144, 145] ya que no se necesita convertir la secuencia de bits en números reales, los operadores se pueden aplicar más rápidamente al decrecer el número de genes y la precisión es mucho mayor. En [146] se realiza una completa revisión de los distintos operadores empleados en AG con codificación real, además de una descripción de los modelos para su estudio teórico.

3.2.1. Metodología de Deb

Deb propone una nueva forma de describir y estructurar los algoritmos de optimización basados en poblaciones de individuos [136]. Según Deb, la mayoría de los algoritmos de optimización basados en poblaciones de individuos generalmente responden a una misma estructura. La mayoría de los algoritmos evolutivos son algoritmos de optimización basados en poblaciones de indivi-

duos, por tanto, se les puede aplicar esta metodología. Igualmente en [136] se le aplica esta metodología a otros métodos de optimización clásicos como el método del simplex propuesto por Nelder and Mead [147], y el método de búsqueda aleatoria adaptativa [148].

La principal ventaja de esta metodología, es que podemos comparar distintos algoritmos de este tipo de una manera sencilla ya que ésta nos propone una descomposición funcional común, en forma de planes o etapas independientes entre sí. De esta manera, nos puede servir para optimizar los algoritmos, a partir de otros, analizando y comparando cada uno de los planes y extrayendo de cada algoritmo lo mejor de cada plan.

Según Deb, el algoritmo comienza con la generación de una población inicial de individuos, posibles soluciones al problema, y en cada generación la población es modificada usando un algoritmo de actualización. Por tanto, si suponemos que nos encontramos en la generación t el conjunto de individuos $B^{(t)}$ (con N $= |B^{(t)}|$) al final de la generación obtendremos una nueva población $B^{(t+1)}$ siguiendo el algoritmo de actualización que consta de los siguientes pasos:

1: Plan de Selección (PS)

Dicho paso consiste en seleccionar los $\underline{p}$ individuos del conjunto $B^{(t)}$ para formar el conjunto $P^{(t)}$. Normalmente se suelen elegir los individuos más aptos pero, como se ha visto anteriormente, en ocasiones se incluyen individuos menos aptos para evitar quedar atrapado en mínimos locales y poder explorar otras zonas dentro del espacio de búsqueda. En la Figura 3.1, podemos observar cómo se seleccionan los individuos marcados de color azul.

Paso 2: Plan de Generación (PG)

En este paso se generan $\underline{c}$ individuos a partir del conjunto $P^{(t)}$ formando el conjunto $C^{(t)}$. Este paso es el más importante y donde suele haber mayor diferencia de uno a otro algoritmo. Dentro de este paso se utilizan los operadores de modificación generando nuevos individuos. En la Figura 3.1, podemos observar cómo se generan los individuos marcados de color rojo a partir de los seleccionados en el paso anterior con el color azul.

Paso 3: Plan de Reemplazamiento (PR)

En este paso se seleccionan $\underline{r}$ individuos del conjunto $B^{(t)}$ formando el conjunto $R^{(t)}$ con el objeto de ser sustituidos. Generalmente, se trata de los individuos menos aptos de la población. En la Figura 3.1, podemos observar cómo se seleccionan los individuos marcados de color verde.

Paso 4: Plan de Sustitución (PSU)

Por último, en este paso se genera la población $B^{(t+1)}$ formada por los individuos de $B^{(t)}$ una vez sustituidos los individuos de $R^{(t)}$ por los individuos seleccionados de los conjuntos $P^{(t)}$ y $C^{(t)}$ utilizando el plan de sustitución. En la Figura 3.1, se sustituyen los individuos marcados en el plan de reemplazamiento de color verde, por los seleccionados en este paso de color rosado. Hay que observar que el número de elementos seleccionados en el plan de reemplazamiento coincide con los elementos seleccionados en el plan de sustitución. De esta forma mantenemos constante el número de individuos de la población, es decir, $|B^{(t)}| = |B^{(t+1)}| = N$, donde N representa el cardinal del conjunto.

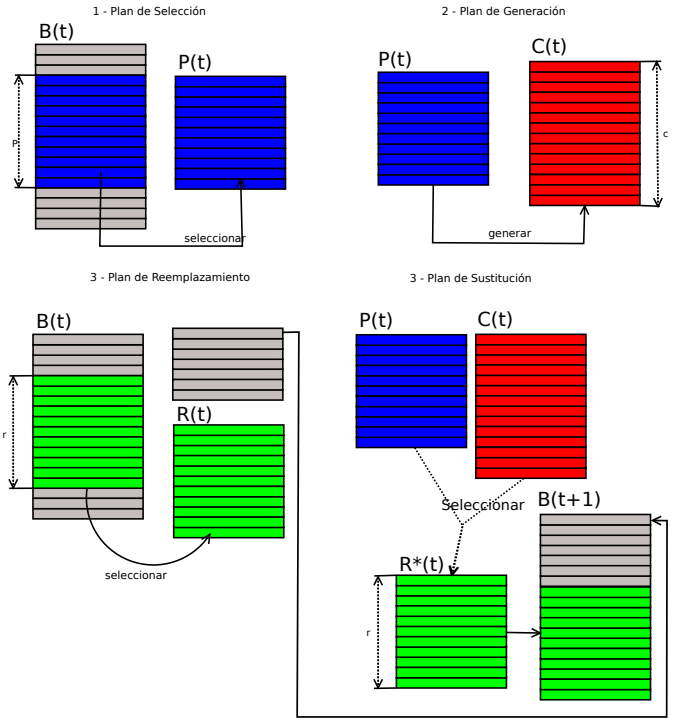


Figura 3.1: Esquema general de la metodología propuesta por Deb.

3.3. Algoritmo Evolutivo para el Entrenamiento y Diseño de Redes Neuronales de Base Radial

La presente sección aborda un algoritmo de PE para el entrenamiento de RNA (*Neural Net Evolutionary Programming* NNEP) propuesto por Martínez-Estudillo [137], basado en diversos trabajos anteriores. El objetivo principal de este algoritmo es el de diseñar la estructura y simultáneamente optimizar los pesos de distintos modelos de RNAs. La búsqueda comienza con una población inicial de redes generadas aleatoriamente, a la que se le aplica un algoritmo poblacional en cada iteración, el cual comparte muchas características con otros algoritmos previos como el propuesto por Angeline [3, 149]. A continuación resumimos sus características fundamentales.

3.3.1. Estructura del Algoritmo Evolutivo

El algoritmo evolutivo se define a continuación siguiendo la metodología de Deb [136] (ver Figura 3.2):

Paso 1: Plan de Selección

Seleccionamos el r por ciento ($r = 90\,\%$) de individuos con mejor aptitud de la población $B_* = B - \{mejor\ \ individuo\ \ de\ \ B\}$ de cardinal $N_R^* = N_R - 1$ y formamos una población P de tamaño $r \times N_R^*/100$. Posteriormente copiamos los individuos de $\boldsymbol{B}$ que no fueron incluidos en $\boldsymbol{P}$ sobre los peores individuos de $\boldsymbol{P}$.

Paso 2: Plan de Generación)

Aplicamos mutación estructural a cada elemento de la población P y mutación paramétrica solo a los $(100-r) * N_R^*$ mejores individuos de P (los individuos a los que se les aplica la mutación paramétrica se corresponden en realidad con el $(100-r)$ por ciento de mejores individuos de la población B^*). La población obtenida después del plan de generación la denominaremos C . Es fácil ver que el cardinal de C es igual a $N_R^* = N_R - 1$.

Paso 3: Plan de Reemplazamiento

El conjunto $R^{(t)}$ está compuesto por la totalidad de los individuos de B_* que serán reemplazados.

Paso 4: Plan de Sustitución

Los elementos de B_* son reemplazados por los elementos de C. Por tanto, la nueva población es $B = \{\ mejor\ individuo\ de\ B\ \} \cup C$. De esta forma, el tamaño de la población se mantiene constante durante la evolución. Se puede observar, por último, que se trata de un algoritmo elitista ya que el mejor individuo de B se mantiene para la siguiente generación.

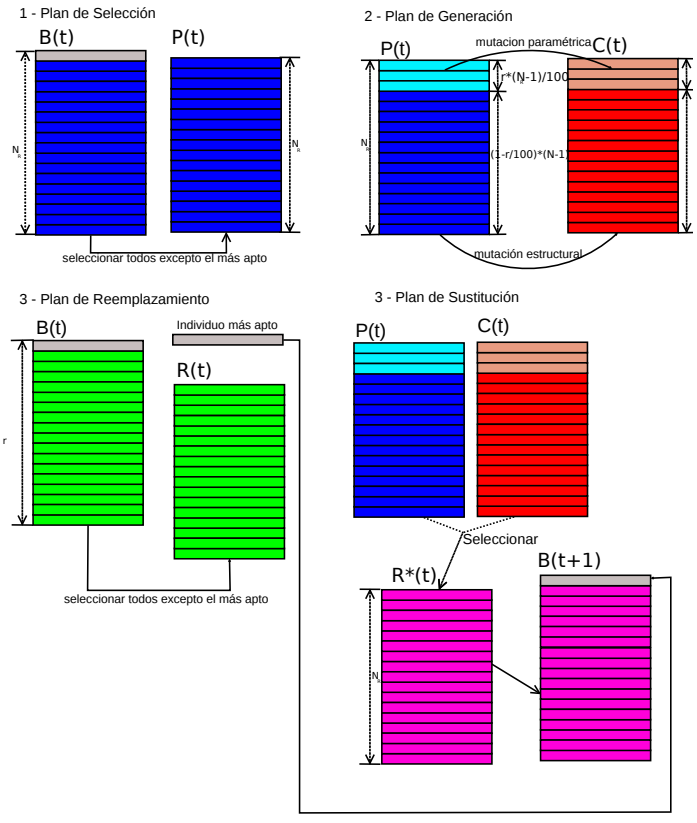


Figura 3.2: Esquema general del algoritmo evolutivo NNEP.

3.3.2. Representación de los Individuos

Variadas son las formas de representar los individuos (RNAs), una de las más básicas es la representación binaria donde cada parámetro de cada individuo a ajustar es modelado como una secuencia de valores binarios cuya unión constituye un único individuo. Existen numerosos trabajos que estiman los pesos de las conexiones de las redes utilizando este tipo de representación [150, 151] que codifica cada peso mediante una cadena de bits. Por otra parte, otra posibilidad consiste en la representación de los pesos de las conexiones de la red directamente mediante números reales [3, 152, 153]. Esta codificación se denomina codificación real. De esta forma, la red se suele representar mediante un vector de números reales.

Cada RNA a representar posee 3 capas: una primera capa de entrada, una siguiente capa oculta y por último una capa de salida. El algoritmo tratará directamente las soluciones al problema y no una representación de las mismas, es decir, utilizará las Redes Neuronales en sí y no una codificación de sus pesos y/o estructura. De esta forma, la información de cada individuo determina tanto la estructura de la red que representa, como sus pesos o coeficientes. Utilizando esta representación, los individuos son sometidos a las operaciones de replicación y mutación, no utilizándose el operador de cruce, debido a las desventajas potenciales que ha demostrado tener en la evolución de RNAs. Con estas características, el algoritmo se puede situar en la categoría de Programación Evolutiva(PE).

Para definir la topología de las RNAs se consideran tres parámetros: m, M_E y M_I. Estos parámetros se corresponden con el número mínimo (m) y máximo (M_E) de nodos ocultos que pueden tener los modelos durante todo el proceso evolutivo y al número máximo (M_I) de nodos ocultos que podrán tener en el proceso de inicialización de la población.

3.3.3. Función de Evaluación

El algoritmo, al aplicarse a problemas de clasificación (binaria o multiclase) utiliza como función de aptitud una transformación estrictamente decre-

ciente del *error de entropía cruzada* $l(\hat{\theta})$, dada por la siguiente expresión:

$$A(\hat{\theta}) = \frac{1}{1 + l(\hat{\theta})} \tag{3.1}$$

de tal forma que $0 \leq A(\hat{\theta}) \leq 1$.

3.3.4. Inicialización de la Población

Se generarán inicialmente 10NP modelos de redes, siendo NP el número de individuos de la población durante el proceso evolutivo. La generación aleatoria de cada red se realizará de la siguiente forma:

1. Se comienza eligiendo el número de nodos de la capa oculta a partir de una distribución uniforme en el intervalo $(0, M_I]$, donde M_I se corresponde con el número máximo de nodos para la capa oculta en la etapa de inicialización. El número de enlaces entre cada nodo de la capa oculta y los nodos de la capa de entrada queda determinado a partir de una distribución uniforme en el intervalo (0, n], siendo n el número de variables de entrada. El número de enlaces entre cada nodo de la capa de salida y los nodos de la capa oculta queda determinado a partir de una distribución uniforme en el intervalo $(0, M]$, siendo M el número de nodos en capa oculta previamente obtenido. Si algún nodo de la capa oculta queda sin conexión, este será conectado con un nodo de la capa de salida elegido aleatoriamente. La capa de salida tiene siempre un número de nodos igual al número de clases del problema menos uno, es decir, $L-1$.

2. Definida la topología de la red, se asigna a cada conexión un valor sináptico o peso en dependencia del tipo de nodo y de la capa a la cual pertenece. Para la inicialización de cada individuo de la población primeramente se procede a inicializar los pesos de los nodos de la capa de salida con valores aleatorios generalmente en el intervalo $[-5, 5]$. Posteriormente las SRBF pertenecientes a la capa oculta se inicializan haciendo uso del algoritmo K-Medias [126, 154] para asignarle un peso inicial a los arcos (links) de cada SRBF. El radio de cada nodo se inicializa con la media

cuadrática de las dos distancias euclídeas relativas a los dos nodos más cercanos.

Por último, la población inicial se forma con las NP redes mejores, asegurándonos así una población inicial con una buena aptitud.

3.3.5. Métodos de Selección

El algoritmo realiza una selección por elitismo en cada generación, que consiste en tomar el 10 % de las mejores redes de la población y sustituirlas por el 10 % de las peores redes de la misma. De este modo, el número de redes (NP) de la población permanece constante a lo largo de la evolución y la presión selectiva es considerablemente alta.

El algoritmo NNEP es un algoritmo elitista, de manera que el individuo con mejor aptitud de la población (y por tanto con menor error de entropía cruzada) pasa sin cambios a la siguiente generación. Sin embargo, en general, la relación entre el CCR y la entropía cruzada depende en gran parte de la estructura de la Base de Datos (BD). Por lo tanto, y, teniendo en cuenta que el objetivo final es el de obtener un alto valor de precisión en el conjunto de generalización, el hecho de utilizar elitismo mediante entropía cruzada es más apropiado en algunas BDs, mientras que en otras, puede ser más adecuado el uso de elitismo en CCR. Por ello Hervás [155] propone mantener el elitismo en ambos criterios y devolver finalmente ambos "mejores individuos", ya que es difícil predecir a priori cuál puede ser el mejor criterio en una BD determinada.

3.3.6. Operadores de Mutación

Las mutaciones realizadas en el algoritmo NNEP serán de dos tipos: paramétricas y estructurales. Las mutaciones paramétricas afectan a los pesos de la red, mientras que las mutaciones estructurales afectan a su topología (nodos ocultos y enlaces). Para tener un mejor control sobre la severidad de las mutaciones se introduce un parámetro de temperatura en cada red, que se define de la siguiente forma:

$$T(\hat{\theta}) = 1 - A(\hat{\theta}), 0 \leq T(\hat{\theta}) \leq 1 \tag{3.2}$$

De esta forma, cuando el proceso de búsqueda se está iniciando, la aptitud de una red es más cercana a cero y por tanto la temperatura es alta, por lo que las mutaciones son más fuertes. A medida que nos acercamos a la solución, disminuye la temperatura, por lo que los cambios que se realizan tanto en los pesos como en la estructura de las redes van siendo más pequeños, afinando así más la búsqueda.

3.3.6.1. Mutaciones Paramétricas

Las mutaciones paramétricas consisten en añadir a cada peso w_{ji} y β_{ij} un ruido gaussiano, donde la varianza de la distribución de Gauss depende de la temperatura de la red. Concretamente, los pesos w_{ji} de la capa intermedia quedan actualizados de la forma:

$$w_{ji}(t+1) = w_{ji}(t) + \xi_1(t) \tag{3.3}$$

donde $i \in [0, \ldots, n], j \in [1, \ldots, M]$, y $\xi_1(t)$ es una variable aleatoria con distribución normal $N(0, \alpha_1(t)T(\hat{\theta}))$. En cuanto a los pesos de la capa de salida se actualizarán de forma análoga:

$$\beta_{ij}(t+1) = \beta_{ij}(t) + \xi_2(t) \tag{3.4}$$

donde $i \in [1, ..., L-1], j \in [0, ..., M]$ y siendo $\xi_2(t)$ es una variable aleatoria con distribución normal $N(0, \alpha_2(t)T(\hat{\theta}))$.

Los cambios en los pesos se realizan de forma secuencial en los nodos de la capa oculta, de manera que, para cada nodo j de la capa oculta, se modifican los pesos w_{ji} y β_{ij} , $1 \leq i \leq L-1$ y, una vez realizado el cambio en los pesos, la aptitud del individuo es recalculada para aplicar un algoritmo estándar de enfriamiento simulado. Si llamamos ΔA a la variación de la aptitud antes y después del cambio en los pesos, resulta que:

$$R(\Delta A) = \begin{cases} Si\ \Delta A \geq 0, & \text{se acepta el cambio} \\ Si\ \Delta A \leq 0, & \text{se acepta el cambio si } e^{\frac{\Delta A}{T}} < \gamma, \text{ donde } \gamma \in U(0,1) \end{cases} \tag{3.5}$$

Este proceso de enfriamiento simulado se repite para cada nodo de la red mutada. Los parámetros α_1 y α_2 que, junto con la temperatura, determinan las varianzas de las distribuciones normales, van cambiando durante el proceso de evolución (al contrario que la versión estática de [3]), realizando en este sentido un aprendizaje adaptativo. De esta forma evitamos quedar atrapados en mínimos locales y aceleramos el proceso de evolución cuando las condiciones del mismo lo permitan. El mecanismo de evolución de los parámetros α_1 y α_2 es la regla de razón de éxito 1/5 de Rechenberg [156], uno de los métodos más simples. Según esta regla, la razón de mutaciones satisfactorias debería ser siempre 1/5. De esta forma, si la razón es mayor que 1/5, la varianza debería incrementarse; en caso contrario, debería disminuirse. Esto es:

$$\alpha_i(t+s) = \begin{cases} (1+\lambda)\alpha_1(t), & \text{si } s_g > 1/5 \\ (1-\lambda)\alpha_1(t), & \text{si } s_g < 1/5 \\ \alpha_1(t), & \text{si } s_g = 1/5 \end{cases} \tag{3.6}$$

siendo $i = 1, 2$, s_g la razón de mutaciones satisfactorias durante s generaciones. En todos nuestros experimentos hemos tomado $\lambda = 0.1$ y $s = 5$.

Por último se muta el radio de las SRBF de igual manera que los pesos w_{ij} adicionando la variable aleatoria $\xi_2(t)$.

3.3.6.2. Mutaciones Estructurales

La mutación estructural modifica el número de nodos ocultos y el número de conexiones entre los nodos de la capa oculta y los nodos de las capas de entrada y salida, afectando de esta forma a la topología de la red y permitiendo explorar nuevas regiones del espacio de búsqueda. Se definen cinco mutaciones estructurales distintas: Añadir Nodos (AN), Eliminar Nodos (EN), Añadir Conexiones (AC), Eliminar Conexiones (EC) y Unir Nodos (UN).

A cada individuo que se le realiza la mutación estructural se le aplicará secuencialmente cada uno de los diferentes tipos de modificación al igual que en [3], teniendo cada mutación una probabilidad de ser aplicada igual a $T(\hat{\theta})$. Si no se aplica ninguna mutación, se elige una al azar. Con el fin de limitar la carga computacional del algoritmo se fija un intervalo $[\Delta min, \Delta max]$ para el número de modificaciones de las mutaciones AN y EN, siendo el número de modificaciones:

$$e_N = \Delta_{min} + [(U(0,1)T(\hat{\theta})(\Delta_{max} - \Delta_{min})] \tag{3.7}$$

siendo e_N el número de nodos finalmente añadidos o eliminados.

Por otro lado, para la adición y eliminación de conexiones se especificarán los valores p_1 y p_2 que determinan el porcentaje actual del número de enlaces de la capa oculta y de salida, respectivamente, que se tomará como máximo. De esta forma, el número de modificaciones para la adición y eliminación de conexiones sería:

$$e_{Ei} = 1 + [(U(0,1)T(\hat{\theta})(p_i/100 * n_i)], con\ i = 1,2 \tag{3.8}$$

siendo p_1 y p_2 el porcentaje de enlaces a modificar en la capa oculta y de salida respectivamente, n_1 y n_2 el número de enlaces actual de dichas capas; y e_{E1} y e_{E2} la cantidad de conexiones añadidas o eliminadas entre la capa de entrada y la oculta y la capa de salida y la oculta, respectivamente.

La mutación UN consiste en elegir aleatoriamente dos nodos a y b, y sustituirlos por otro nuevo nodo c. Si la capa oculta ya posee el número mínimo de nodos preestablecido, la mutación no tiene éxito (se pasaría al siguiente tipo de mutación). Si la capa oculta es híbrida, los nodos elegidas deben ser del mismo tipo. Si no hay dos nodos del mismo tipo, la mutación no tiene éxito (se pasaría al siguiente tipo de mutación). Se conservarán los enlaces en el nodo c con los nodos comunes a los nodos a y b, y también se conservan con una probabilidad de 0.5 los que no sean comunes. El resultado de la fusión se aprecia en la Figura 3.3:

- Enlaces de entrada del nuevo nodo resultado de la unión: Por cada entrada, si ambos nodos originales poseían ese enlace de entrada, el enlace se conserva con un peso igual a la media de ambos pesos. Si solo un nodo poseía ese enlace de entrada, el enlace se conserva con una probabilidad de 0.5 y con el mismo peso. Si ningún nodo poseía ese enlace de entrada, el nodo resultante tampoco lo poseerá.

- Enlaces de salida del nuevo nodo resultado de la unión: Por cada nodo de salida, si ambos nodos originales poseían ese enlace de salida, el enlace se conserva con un peso igual a la suma de ambos pesos. Si solo un nodo poseía ese enlace de salida, el enlace se conserva con una probabilidad de 0.5 y con el mismo peso. Si ningún nodo poseía ese enlace de salida, el nodo resultado tampoco lo poseerá.

$$u_i^c = \frac{u_i^a + u_i^b}{2}; \quad w_i^c = \frac{w_i^a + w_i^b}{2}; \quad \alpha_j^c = \alpha_j^a + \alpha_j^b; \quad \beta_j^c = \beta_j^a + \beta_j^b$$

para i igual al número de nodos en capa oculta y j igual al número de nodos en capa de salida.

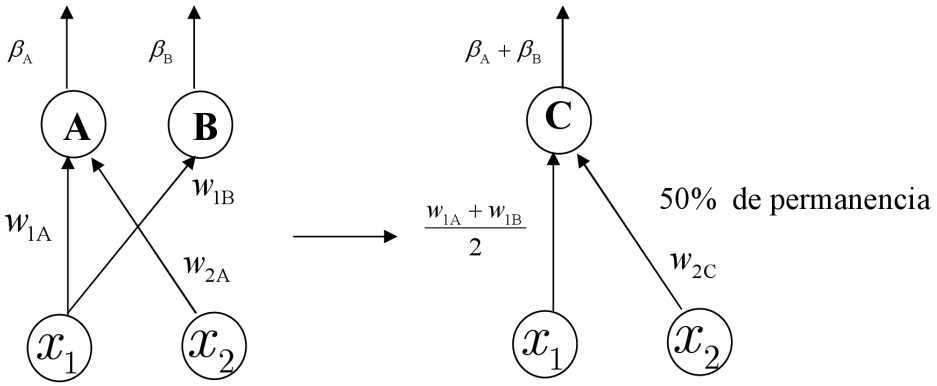


Figura 3.3: Ejemplo de fusión de dos nodos con la mutación UN.

En el caso de los nodos SRBF, la unión debe hacerse de otro modo, ya que el significado de dichos nodos es diferente, encargándose cada una de modelar una zona del espacio cercana a su centro. El hecho de que las funciones combinadas puedan interpretarse como circunferencias de centro y radio perfectamente identificados obliga a que, al combinar ambos nodos, se tenga

en cuenta este conocimiento topológico. Llamemos $\boldsymbol{c}_j$ al centro definido por el nodo j, es decir, $(w_{1j}, w_{2j}, ..., w_{kj})$ y r_j a su radio. En este caso, la mutación UN de dos SRBF, a y b, consistirá en calcular el centro y radio del nodo c como:

$$\boldsymbol{c}_c = \frac{r_a}{r_a+r_b}\cdot\boldsymbol{c}_a + \frac{r_b}{r_a+r_b}\cdot\boldsymbol{c}_b \qquad r_c = \frac{r_a+r_b}{2}$$

En el caso de la mutación AN, uno o más nodos son añadidos a la capa oculta. Las conexiones con los nodos de salida son elegidas aleatoriamente y tienen un peso aleatorio en el intervalo $[-O, O]$. Las conexiones desde la capa de entrada también son elegidas aleatoriamente y tienen un peso aleatorio en el intervalo [-I, I]. La adición de conexión se realizará primero entre capa oculta y capa de entrada (añadiendo e_{E1} elementos, ver fórmula 3.7) y después entre capa de salida y capa oculta (añadiendo e_{E2} elementos, ver fórmula 3.8). Los nodos origen y destino de cada conexión son elegidos aleatoriamente en las capas correspondientes. En la mutación EN uno o más nodos se seleccionan aleatoriamente y son eliminados junto con sus conexiones. La mutación EC se realiza de la misma forma que AC, eligiendo aleatoriamente los nodos origen y destino.

3.3.7. Condición de Parada

El algoritmo se detendrá cuando se cumpla una de las siguientes condiciones:

- Transcurre un número de generaciones sin mejorar la media de aptitud del 10 % de los mejores individuos y la aptitud del mejor individuo.
- Se alcanza un número máximo de generaciones establecido, G.

3.3.8. Esquema del Algoritmo

Con todas estas características, el esquema general del algoritmo NNEP para clasificación es el siguiente:

Algoritmo NNEP

1. Generar una población aleatoria de tamaño $10N_P$ y seleccionar las N_P mejores redes para la población inicial.
2. Repetir hasta que se alcance un número máximo de generaciones G o se cumpla la condición de parada.
Aplicar mutación paramétrica al 10 % de mejores individuos de la población.
Aplicar mutación estructural al restante 90 % de individuos.
Calcular la aptitud de cada individuo de la población.
Ordenar los individuos según su aptitud.
El 10 % de los mejores individuos son replicados y sus réplicas sustituyen al 10 % de los peores individuos (fuerte presión selectiva).
5. Seleccionar el mejor individuo en aptitud, devolviéndolo como solución factible al problema.

3.3.9. Mejoras y variaciones de la metodología NNEP

El algoritmo fue inicialmente desarrollado para la optimización de modelos de Redes Neuronales de UP en regresión [157]. En dicho trabajo, las Redes Neuronales evolutivas de UP aparecían como una alternativa al clásico MLP, demostrando gran capacidad de aproximación en problemas de regresión sintéticos y reales.

Debido a la dificultad que tiene el algoritmo NNEP para converger a la mejor solución posible, el procedimiento de búsqueda global fue hibridado con un algoritmo de optimización local (el algoritmo de Levenberg-Marquardt, [80]) y un procedimiento de agrupamiento o clustering, de nuevo en problemas de regresión y con redes de UP [158]. Frente a los AMs clásicos, el algoritmo híbrido propuesto aplica el procedimiento de búsqueda local solo en ciertas generaciones de la evolución y los individuos optimizados no vuelven a la población, siendo almacenados en una piscina de soluciones. Además, se utiliza el algoritmo de las $K - Medias$ [126, 154] para lograr obtener grupos de individuos con aptitudes cercanas, que se espera se correspondan con regiones relevantes del espacio de búsqueda. De esta forma, la optimización local es solo aplicada a los mejores individuos de cada grupo, reduciendo así el coste computacional.

El siguiente paso supuso la adaptación de los modelos de redes de UP y del algoritmo a problemas de clasificación [155, 159, 160] alcanzándose buenos resultados. El algoritmo NNEP de clasificación también ha sido hibridado con procedimientos de búsqueda local. En una primera aproximación, se reprodujo el esquema de regresión, combinando el AE con el algoritmo de LM y un procedimiento de agrupamiento [161], adaptados ambos al problema de clasificación.

Sin embargo, otra estrategia con la cual se han obtenido mejores resultados para los problemas de clasificación, ha sido la aplicación de un procedimiento híbrido entre la Regresión Logística y el AE, utilizando los nodos de los mejores individuos del AE como nuevas covariables para el modelo de Regresión Logística [162, 163]. El AE es utilizado para determinar la estructura básica del modelo de UPs, y, posteriormente, se aplica la Regresión Logística en el nuevo espacio aumentado con las funciones de base del mejor modelo. Este modelo híbrido supera el rendimiento tanto de la correspondiente parte lineal como de la no lineal y obtiene unos buenos resultados de clasificación en comparación con otras técnicas de aprendizaje.

Por otro lado, NNEP también ha sido adaptado y aplicado en el entrenamiento de modelos híbridos de Redes Neuronales, es decir, modelos formados por distintos tipos de nodos en capa oculta. En el ámbito de regresión, se emplearon modelos con UP y con US que se mostraron eficaces para la resolución de problemas de regresión en los que la tipología especial de las funciones objetivo pueda presentar diferentes comportamientos en diversas regiones del dominio de definición [164]. En el ámbito de la clasificación, también se ha adaptado el algoritmo para la hibridación de distintos tipos de nodos, UP con US [165] y funciones de tipo kernel (RBF) con funciones de tipo proyección (UP y US) [166], superando de nuevo los modelos híbridos a los puros en cuanto a precisión en la clasificación. Esta línea de investigación nos ha llevado a la conclusión de que para algunas BDs, ciertos modelos se comportan mejor, mientras que no aportan buenos resultados para otras. Por lo tanto, es una alternativa muy recomendable el uso de modelos híbridos que puedan unificar las mejores características de cada tipo de nodo.

Capítulo 4

ALGORITMOS HÍBRIDOS PARA EL ENTRENAMIENTO Y DISEÑO DE REDES NEURONALES

4.1. Introducción

El uso de AE para varios de problemas optimización podría ser suficiente para encontrar una solución deseada. Sin embargo, como se informa en la literatura, existen varios tipos de problemas en que un AE no permite obtener una solución conveniente (óptima) [167] por lo que se hace necesario hibridarlo con otras técnicas. El presente capítulo describe las principales técnicas de hibridación para aumentar su eficacia. Además se describen los trabajos previos desarrollados por el grupo AYRNA para aumentar la velocidad de convergencia de los AE.

4.2. Algoritmos Evolutivos Híbridos

Un Algoritmo Híbrido es la combinación de dos o más algoritmos que funcionan de forma conjunta para aumentar su eficacia. Algunas de las posibles razones para la hibridación son [168]:

1. Para mejorar el rendimiento de los algoritmos que componen el AH (por ejemplo: la velocidad de convergencia).
2. Para mejorar la calidad de las soluciones obtenidas.
3. Para incorporar varios algoritmos como parte de un sistema mayor.

Debido a las razones antes mencionadas surge la necesidad de hibridar de los algoritmos evolutivos con otros algoritmos de optimización, técnicas de aprendizaje automático, heurística, etc.

En 1995, Wolpert y Macready [169] muestran que todos los algoritmos que buscan un extremo de una función de costo obtienen resultados "iguales" en promedio sobre todas las funciones de costo posibles. Según los autores, si un algoritmo A supera un algoritmo B sobre algunas de las funciones de costos, en términos generales deben existir otras tantas funciones de costo donde B supera a A. Por lo tanto, desde una perspectiva de solución de problemas es difícil formular un algoritmo de optimización universal que puede resolver todos los problemas, por tanto, la hibridación puede ser la clave para resolver cierto tipo de problemas prácticos.

4.3. Arquitecturas de Algoritmos Evolutivos Híbridos

Variadas son las técnicas y heurísticas/metaheurísticas utilizadas para mejorar la eficiencia general del algoritmo evolutivo. Algunas de las arquitecturas híbridas más utilizadas se resumen a continuación:

1. Algoritmos Evolutivos hibridados con otros Algoritmos Evolutivos.
2. Algoritmos Evolutivos hibridados con Redes Neuronales.
3. Algoritmos Evolutivos hibridados con Lógica Difusa.
4. Algoritmos Evolutivos hibridados con Optimización con Sistemas de Partículas.

5. Algoritmos Evolutivos hibridados con Optimización con Colonia de Hormigas.

6. Hibridación entre Algoritmos Evolutivos y otros algoritmos y heurísticas (ej. Búsqueda Local, Búsqueda Tabú, Recocido Simulado, Escalador de Colinas, Programación Dinámica, Búsqueda Voraz Aleatoria, etc).

4.3.1. Algoritmos Evolutivos hibridados con otros Algoritmos Evolutivos

Variados han sido los acercamientos a esta arquitectura de hibridación. Tan et al. [170] propusieron una técnica híbrida de clasificación evolutiva compuesta por dos fases para extraer reglas de clasificación que se pueden utilizar en la práctica clínica para mejorar la comprensión y la prevención de embarazos no deseados y otros eventos médicos. En la primera fase, un algoritmo evolutivo se utiliza para limitar el espacio de búsqueda para la evolución de un conjunto de buenas reglas candidatas. Luego se aplica PG [171] para evolucionar los atributos nominales de dichas reglas; y un AG para optimizar los atributos numéricos de las reglas concisas de clasificación, sin necesidad de discretización. Estas reglas candidatas se utilizan en la segunda fase para optimizar el orden y número de reglas en la evolución para formar conjuntos de reglas precisas y comprensibles.

Zmuda et al. [172] propusieron un esquema híbrido evolutivo de aprendizaje para crear sistemas para el reconocimiento de patrones multiclases. El algoritmo dedica un gran esfuerzo computacional para la generación de nuevas variables que servirían como entradas de un algoritmo de tipo *Back-End*. Las nuevas variables son generadas mediante una combinación de Programación Genética [171, 173] para sintetizar expresiones aritméticas, algoritmos genéticos [138] para seleccionar un conjunto viable de las expresiones, y la programación evolutiva [174, 175] para optimizar los parámetros dentro de las expresiones. El objetivo fue crear un conjunto compacto de características no lineales que cooperan para resolver un problema de reconocimiento de patrones multiclase.

4.3.2. Algoritmos Evolutivo hibridados con Redes Neuronales

La combinación de los AEs y las RNAs es propuesta por Wang [176] mediante un enfoque híbrido para mejorar el rendimiento de los AEs para un problema de optimización. En la primera fase, las RNA se construyen sobre la base de una colección de muestras de entrenamiento utilizando el algoritmo BP con un parámetro de momento. Seguidamente, el AE realiza la búsqueda de las soluciones en el espacio de las variables. Una vez que el algoritmo evolutivo genera una nueva solución, la RNA se utiliza para determinar su valor de aptitud mediante la creación de una "función objetivo ampliada" donde la estimación de la RNA sobre la nueva solución generada por el AE, es parte de su formulación.

Para mejorar la coherencia y la solidez de los resultados, se sugiere el uso de múltiples de RNAs para proporcionar un rendimiento estadísticamente validado para el algoritmo evolutivo. Para aquellos problemas donde la función objetivo es bien conocida, pero “difícil” de evaluar, las RNA aún se pueden utilizar para proporcionar rápidamente una evaluación del desempeño con el objetivo de mejorar la eficiencia de la búsqueda del AE.

4.3.3. Algoritmos Evolutivos hibridados con Lógica Difusa

Los controladores de lógica difusa (FLC) están compuestos por una base de conocimientos que incluye la información facilitada por un experto en la forma de las normas de control lingüístico, una *interfaz de fuzzificación* que tiene la capacidad de crear conjuntos difusos, un sistema de inferencia que utiliza la base de conocimientos y reglas para realizar la inferencia a través de un método de razonamiento y por último una interfaz de defuzzificación que traduce la acción de control difuso obtenido de este modo a una acción de control real utilizando un método de defuzzificación. Los FLC se han utilizado para diseñar la adaptación de algoritmos evolutivos. La idea principal es utilizar un FLC cuyas entradas son una combinación de medidas de desempeño de un AE y que sus salidas son los parámetros de control de dicho algoritmo. Lee y Takagi [177] propusieron el algoritmo “*Dinamic Parametric Genetic Algorithm*”

que utiliza un FLC para el control de los parámetros de un AG. Las entradas del FLC son una combinación de medidas de desempeño del AG y los resultados son un conjunto de parámetros de control del AG.

4.3.4. Algoritmos Evolutivos hibridados con Optimización con Sistemas de Partículas

PSO incorpora comportamientos observados en las bandadas de aves, de peces, o los enjambres de abejas, e incluso el comportamiento social humano, del cual surgió la idea [178, 179, 180]. Un método híbrido propuesto por Shi et al. [181] realiza una combinación un AE y un PSO. Este enfoque híbrido ejecuta los dos sistemas (un AE y un PSO) al mismo tiempo y selecciona P individuos de cada sistema para ser intercambiados tras N iteraciones. El individuo con mayor aptitud tiene más oportunidades de ser seleccionado. Los principales pasos del método híbrido se muestran a continuación :.

1. Inicializar el AE y el PSO.
2. Ejecutar el AE y el PSO de forma simultánea.
3. Retornar el mejor individuo si alguna de las dos estrategias alcanza su condición de parada.
4. Cada N iteraciones seleccionar P individuos de forma aleatoria de ambos subsistemas e intercambiarlos. Ir al paso 1.

4.3.5. Algoritmos Evolutivos hibridados con Optimización con Colonia de Hormigas

La Optimización con Colonia de Hormigas ACO es un sistema inspirado en el comportamiento de las hormigas reales. Dichos algoritmos son ampliamente utilizados para resolver problemas de optimización discreta [182].

Tseng y Liang [183] proponen un enfoque híbrido que combina ACO, un Algoritmo Genético y un método de Búsqueda Local. El algoritmo se aplica para resolver el Problema de Asignación Cuadrática. El AG en lugar de partir

de una población que se compone de los cromosomas generados aleatoriamente, parte de una población inicial construída por un ACO con el fin de ofrecer un buen comienzo. La feromona actúa como un mecanismo de retroalimentación desde la fase AG a la fase ACO. Cuando el AG alcanza el criterio de terminación, el control se transfiere de nuevo a la fase de ACO. Luego la ACO utiliza las feromonas actualizadas por el AG para explorar el espacio de soluciones y producir una población prometedora para la próxima ejecución de la fase AG. Finalmente un método de búsqueda local se aplica para mejorar las soluciones obtenidas por el ACO y el AG. Otro enfoque híbrido para el mismo problema ha sido propuesto por Vázquez y Whitley [184] donde el AG se combina con Búsqueda Tabú [185].

4.3.6. Algoritmos Evolutivos con conocimiento "a priori"

Existen variados enfoques que no utilizan una población inicial generada aleatoriamente para los algoritmos evolutivos. Si existe un conocimiento previo o puede ser obtenido a un bajo coste computacional, con buenas estimaciones iniciales se pueden generar mejores soluciones con una convergencia más rápida [186, 187] .

Keedwell y Khu [188] mencionan que un enfoque heurístico que provea de una configuración inicial a un AG produce mejoras en el rendimiento sobre problemas de optimización complejos. Algunos autores [188] han propuesto la confección de un autómata celular basado en heurísticas [189], este es un enfoque que proporciona una buena población inicial para un algoritmo evolutivo.

4.3.7. Enfoques Híbridos Incorporando Búsqueda Local

Las hibridaciones entre los algoritmos evolutivos y la búsqueda local se conocen como algoritmos meméticos AM y Algoritmos Híbridos (la diferencia entre ambos radica en que en los AMs los individuos optimizados se incluyen en la población y en los AHs no se incluyen). Ambos tipos de algoritmos han demostrado ser más rápidos y más precisos que los algoritmos evolutivos para diferentes clases de problemas. Como se informa en la literatura, los métodos

híbridos que combinan los métodos probabilísticos y los métodos deterministas han tenido éxito en la solución de problemas complejos de optimización [190]. La Programación Evolutiva (PE) hibridada con procedimientos de optimización determinista fue utilizada por Myung y Kim [191] para problemas de optimización no lineal y cuadrática.

Los algoritmos de programación evolutiva demuestran su capacidad en el modelado de la topología de las redes neuronales así como en la estimación de los parámetros de dichos modelos. Sin embargo, debido a razones fundamentalmente asociadas a la rapidez de convergencia, se ven obligados a ser hibridados con otras técnicas. Con la finalidad de crear algoritmos más eficientes, la comunidad científica en general y el grupo AYRNA de la UCO en particular, se han dado a la tarea de potenciar la hibridación de modelos y algoritmos. Hervás et al. propusieron dos AH: *Hybrid Evolutionary Programming (HEP)* [192] y *Hybrid Evolutionary Programming with Clustering (HEPC)* [192] combinando un AH con el algoritmo LM [94]. A continuación se describe el esquema general de ambos algoritmos .

Algoritmo de Programación Evolutiva Híbrida (HEP) [192]
1. Se inicializa la población con los N mejores individuos de un conjunto de 10N RNAs.
2. Aplicar NNEP hasta que se alcance el criterio de parada.
3. Aplicar LM [94] al mejor individuo obtenido por el NNEP.

Algoritmo de Programación Evolutiva Híbrida Clusterizada (HEPC) [192]
1. Se inicializa la población con los N mejores individuos de un conjunto de 10N RNAs.
2. Aplicar NNEP hasta que se alcance el criterio de parada.
3. K-Medias [126] sobre la sentitividad de los individuos ante los datos de entrenamiento.
4. Aplicar LM [94] al mejor individuo de cada clúster .
5. Seleccionar el mejor individuo dentro de los optimizados en el paso anterior.

Capítulo 5

MODELOS NEURO-LOGÍSTICOS

5.1. Introducción

La combinación de las RNAs y la Regresión Logística da origen a un nuevo tipo de modelo, el modelo *neuro-logístico*, el cual aprovecha la precisión y robustez de los modelos logísticos entrenados generalmente con métodos numéricos, y la capacidad que presentan las RNAs de modelar las interacciones entre las variables independientes.

En el presente capítulo se describe la técnica de RL (sección 5.2), las características de los modelos logísticos (subsección 5.2.1) y la estimación de los coeficientes mediante la estimación máximo verosímil (subsección 5.2.1.1). Posteriormente se describen las características de los modelos *neuro-logístico*s utilizando covariables obtenidas mediante RNAs y cómo pueden ser entrenados. Por último se describe brevemente el problema de multicolinealidad (sección 5.4) de las variables independientes y cómo esta afecta a la estimación en los modelos de RL y en especial a los modelos *neuro-logísticos*.

5.2. Regresión Logística

La regresión logística es una técnica matemática ampliamente utilizada en estadística para predecir el valor de una variable binaria objetivo que re-

presenta la probabilidad de que un determinado evento ocurra, dependiendo de los valores de un conjunto de variables explicativas relacionadas con él.

La utilidad de la RL es ampliamente conocida por su aplicación a diversos campos de estudio, como la medicina o la microbiología, donde la naturaleza probabilística los muchos de sus fenómenos permite la aplicación de modelos logísticos para modelar la probabilidad con que estos ocurren.

5.2.1. Modelo de Regresión Logística

Sea $(\boldsymbol{x}, y)$ un conjunto de patrones de entrenamiento cuyas variables objetivo y son binarias. Para cada vector de variables de entrada $\boldsymbol{x} \in R^K$, la salida es o $y_i = 1$, lo que significa que el patrón pertenece a la *clase positiva* (el evento ocurre), o $y_i = 0$, lo que significa que el patrón pertenece a la *clase negativa* (el evento no ocurre). Para modelar la relación entre cada patrón $\boldsymbol{x}$ y el valor esperado para su salida, se define la **función logística**. Dicha función se formula como:

$$\mu(x, \beta) = \frac{exp(\beta^T \cdot x)}{1 + exp(\beta^T \cdot x)} \tag{5.1}$$

donde $\boldsymbol{\beta}$ es el vector de coeficientes del modelo. Estos coeficientes son inicialmente desconocidos, siendo necesario estimarlos a partir del conjunto de datos de entrenamiento. Además, asumimos que $x_0 = 1$ [193], de forma que β_0 sea un término constante. Así, el modelo de regresión es $y = \mu(x, \beta) + \epsilon$, donde ϵ es el término de error del modelo.

El modelo para clasificación binaria es generalizado para J clases mediante la expresión:

$$\mu_l(\boldsymbol{x}, \boldsymbol{\beta}) = \frac{exp(\beta_l^T \cdot x)}{1 + \sum_{i=1}^{J-1} exp(\beta_i^T \cdot x)} \tag{5.2}$$

donde $\mu_l(x, \beta)$ es la probabilidad de que un patrón pertenezca a la clase j.

Expresado de esta forma, el modelo de regresión logística es un clasificador lineal, resultado de la relación lineal existente entre los coeficientes y

las variables independientes componentes de cada patrón de la base de datos. Esto indica que la regresión logística puede asemejarse a la búsqueda de un hiperplano que sea capaz de separar los datos de la clase positiva de los de la clase negativa [194] .

5.2.1.1. Estimación Máximo Verosímil

En el modelo de RL, la variable dependiente y sigue una distribución binaria con media $\mu(x, \beta)$. Por lo tanto, podemos interpretar la función logística como la probabilidad de que $y_i = 1$, o equivalentemente, la probabilidad de que $\boldsymbol{x}$ pertenezca a la clase positiva. Entonces, podríamos computar la probabilidad para el i-ésimo patrón de que su salida en la base de datos X sea y_i como:

$$\begin{aligned} P(x_i, y_i|\boldsymbol{\beta}) &= \begin{cases} \mu(x_i, \boldsymbol{\beta}) & si \quad y_i = 1 \\ 1 - \mu(x_i, \boldsymbol{\beta}) & si \quad y_i = 0 \end{cases} \\ &= (\mu(x_i, \boldsymbol{\beta}))^{y_i} \cdot (1 - \mu(x_i, \boldsymbol{\beta}))^{1-y_i} \end{aligned} \tag{5.3}$$

A partir de esta expresión se obtiene la función de probabilidad conjunta de las k variables independientes del patrón $\boldsymbol{x}$, como:

$$L(\boldsymbol{x}, y, \boldsymbol{\beta}) = \prod_{i=1}^{K} (\mu(x_i, \boldsymbol{\beta}))^{y_i} \cdot (1 - \mu(x_i, \boldsymbol{\beta}))^{1-y_i} \tag{5.4}$$

donde K es el número de patrones que contiene la base de datos. Maximizando esta ecuación se obtiene la estimación del vector de coeficientes $\boldsymbol{\beta}$ que mejor se adapte a las condiciones del conjunto de entrenamiento. Para ello, debido a que esta ecuación no es lineal, se utiliza algún método numérico.

5.2.2. IRLS

Una alternativa a la maximización numérica de la función de probabilidad del modelo de RL es la técnica IRLS (*Iterative Re-weigthed Least Squared*). Esta técnica usa el algoritmo de *Newton - Raphson* para estimar los coeficientes de las ecuaciones del modelo de regresión logística. El proceso es el siguiente:

En primer lugar se calcula la derivada del logaritmo de la ecuación de probabilidad con respecto a cada uno de los coeficientes que componen el vector $\boldsymbol{\beta}$:

$$\begin{aligned}
\frac{\partial}{\partial \beta_j} \ln L(\boldsymbol{x}, y, \boldsymbol{\beta}) &= \sum_{i=1}^{R} (y_i \cdot \frac{\frac{\partial}{\partial \beta_j}\mu(x_i, \boldsymbol{\beta})}{\mu(x_i, \boldsymbol{\beta})} - (1 - y_i) \cdot \frac{\frac{\partial}{\partial \beta_j}\mu(x_i, \boldsymbol{\beta})}{1 - \mu(x_i, \boldsymbol{\beta})}) \\
&= \sum_{i=1}^{R} y_i \cdot \frac{x_{ij} \cdot \mu(x_i, \boldsymbol{\beta}) \cdot (1 - \mu(x_i, \boldsymbol{\beta}))}{\mu(x_i, \boldsymbol{\beta})} \\
&\quad - \sum_{i=1}^{R} (1 - y_i) \cdot \frac{x_{ij} \cdot \mu(x_i, \boldsymbol{\beta}) \cdot (1 - \mu(x_i, \boldsymbol{\beta}))}{1 - \mu(x_i, \boldsymbol{\beta})} \\
&= \sum_{i=1}^{R} x_{ij} \cdot (y_i - \mu(x_i, \beta)) \qquad (5.5)
\end{aligned}$$

donde $j = 1 \ldots M$ y M es el número de coeficientes que componen el vector $\boldsymbol{\beta}$.

Después, se iguala a cero cada una de las derivadas parciales. Si extendemos la definición de μ de tal forma que $\mu(\boldsymbol{x}, \boldsymbol{\beta}) = (\mu(x_1, \boldsymbol{\beta}), \ldots, \mu(x_R, \boldsymbol{\beta}))^T$, se define el conjunto de ecuaciones como:

$$\boldsymbol{x}^T \cdot (y - \mu(\boldsymbol{x}, \beta)) = 0 \qquad (5.6)$$

Definiendo $h(\beta)$ como el vector de derivadas parciales obtenido anteriormente al derivar el logaritmo de la ecuación de probabilidad con respecto a cada β_j, encontrar una solución para la estimación de los coeficientes equivale a encontrar los ceros de $h(\beta)$. El algoritmo de *Newton - Raphson* encuentra este óptimo a través de aproximaciones lineales sucesivas. Después de elegir un valor inicial para $\hat{\beta}_0$, cada iteración del algoritmo actualiza el valor de $\hat{\boldsymbol{\beta}}$ a través de aproximaciones de primer orden:

$$\hat{\beta}_{i+1} = \hat{\beta}_i + J_h(\hat{\beta}_i)^{-1} \cdot h(\hat{\beta}_i) \qquad (5.7)$$

donde $J_h(\hat{\beta}_i)$ es el Jacobiano de h evaluado para $\hat{\beta}_i$.

El Jacobiano es una matriz donde cada fila es simplemente el gradiente traspuesto de uno de los componentes de h evaluado para $\hat{\beta}_i$, por lo que el ij-ésimo elemento de $J_h(\hat{\beta}_i)$ es $-\sum_{i=1}^{R} x_{ij} \cdot x_{ji} \cdot \mu(x_i, \beta) \cdot (1 - \mu(x_i, \beta))$.

Si definimos $w_i = \mu(x_i, \beta) \cdot (1 - \mu(x_i, \beta))$ y $W = diag(w_1, \ldots, w_R)$, entonces se reescribe el Jacobiano en notación matricial como $-\boldsymbol{x}^T W \boldsymbol{x}$. Usando este hecho, y realizando una serie de simplificaciones, la ecuación de actualización del algoritmo de *Newton - Raphson* se puede reescribir como:

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{i+1} &= \hat{\boldsymbol{\beta}}_i + (\boldsymbol{x}^T W \boldsymbol{x})^{-1} \cdot \boldsymbol{x}^T \cdot (y - \mu(\boldsymbol{x}, \hat{\boldsymbol{\beta}}_i)) \\ &= (\boldsymbol{x}^T W \boldsymbol{x})^{-1} \cdot \boldsymbol{x}^T \cdot (W \boldsymbol{x} \hat{\boldsymbol{\beta}}_i + (y - \mu(\boldsymbol{x}, \hat{\boldsymbol{\beta}}_i))) \\ &= (\boldsymbol{x}^T W \boldsymbol{x})^{-1} \cdot \boldsymbol{x}^T W z \end{aligned} \tag{5.8}$$

donde $z = \boldsymbol{x} \hat{\boldsymbol{\beta}}_i + W^{-1} \cdot (y - \mu(\boldsymbol{x}, \hat{\boldsymbol{\beta}}_i))$.

Aplicando sucesivamente esta ecuación de actualización en un algoritmo iterativo, se entrena el modelo de regresión logística, obteniendo los valores óptimos para cada uno de los coeficientes del modelo. Sin embargo, para poder utilizar este algoritmo, es necesario considerar varios aspectos que aseguren su convergencia, su eficiencia y su eficacia.

5.2.2.1. Criterios de Parada

Como en todos los algoritmos numéricos iterativos, para poder desarrollar el algoritmo de entrenamiento de los modelos de regresión logística, es necesario utilizar algún mecanismo que asegure la convergencia del algoritmo y determine cuándo se han alcanzado los valores óptimos de los coeficientes, y puede, por tanto, finalizarse la ejecución del mismo.

En el caso del IRLS, durante las diferentes iteraciones, la convergencia del algoritmo puede ser controlada mediante dos posibles criterios de parada:

1. ***El cambio relativo en los coeficientes***:

$$\sqrt{\frac{(\hat{\beta}_{i+1} - \hat{\beta}_i)^T \cdot (\hat{\beta}_{i+1} - \hat{\beta}_i)}{\hat{\beta}_i^T \cdot \hat{\beta}_i}} < \epsilon \tag{5.9}$$

2. ***El cambio relativo en la varianza de los coeficientes***:

$$\left| \frac{D(y, \mu_{i+1}) - D(y, \mu_i)}{D(y, \mu_i)} \right| < \epsilon \tag{5.10}$$

donde $D(y, \mu) = -2 \cdot \sum_{i=1}^{R} y_i \cdot \ln(\mu(x_i, \beta)) + (1 - y_i) \cdot \ln(1 - \mu(x_i, \beta))$.

En ambos casos, el parámetro ϵ representa la tolerancia o error permitido en el algoritmo a la hora de estimar los coeficientes del modelo. Este valor es puramente experimental y depende de la precisión que necesita el cálculo de los coeficientes y de la velocidad de computación requerida (a mayor precisión, mayor número de iteraciones del algoritmo necesarias, y por tanto, mayor tiempo de cómputo consumido).

5.2.3. Regularización

Según Duffy y Satner [195] la penalización de la norma del vector de coeficientes presentes en la función de máxima verosimilitud FV provoca un aumento en la precisión de los modelos de regresión logística. Dicha penalización se establece redefiniendo la FV como:

$$L^{\lambda}(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) - \lambda \parallel \boldsymbol{\beta} \parallel^2 \tag{5.11}$$

Donde $l(\boldsymbol{\beta})$ es la función de máxima verosimilitud, $\parallel \boldsymbol{\beta} \parallel^2 = (\sum_{i=1}^{P} \beta_i)^{1/2}$ es la norma del vector de coeficientes y λ es el parámetro que indica el peso de la penalización de la norma del vector de coeficientes.

5.3. Modelos de Regresión Logística utilizando covariables obtenidas mediante Redes Neuronales

Los modelos de regresión logística utilizando covariables obtenidas mediante RNAs incorporan en su formulación el valor de las salidas de las funciones de base presentes en una o varias RNAs previamente entrenadas, con el fin de ampliar el espacio de variables explicativas (Figura 5.1).

Para el entrenamiento de estos modelos recientemente Hervás-Estudillo [196] propusieron un algoritmo (NNEP-RegLog) que hibrida un modelo lineal y las RNAs evolutivas para problemas de *clasificación*. La estimación de los coeficientes del modelo se realiza en tres fases:

1. En esta fase se determina el número de nodos y sus pesos mediante CE. El algoritmo se ejecuta en su forma habitual y una vez se alcanza la última generación, se selecciona al mejor modelo de la población, se extraen sus funciones de base (US, UP o SRBF) asociados a los nodos de capa oculta y se procede a realizar la segunda fase.

2. La segunda fase es un procedimiento estándar de optimización basada en el método de máxima verosimilitud (en concreto, el algoritmo IRLS), donde se determina el resto de coeficientes del modelo lineal en el nuevo espacio que se obtiene al unir las covariables iniciales con las funciones de base estimadas previamente.

3. La última fase es un sencillo mecanismo que 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*.

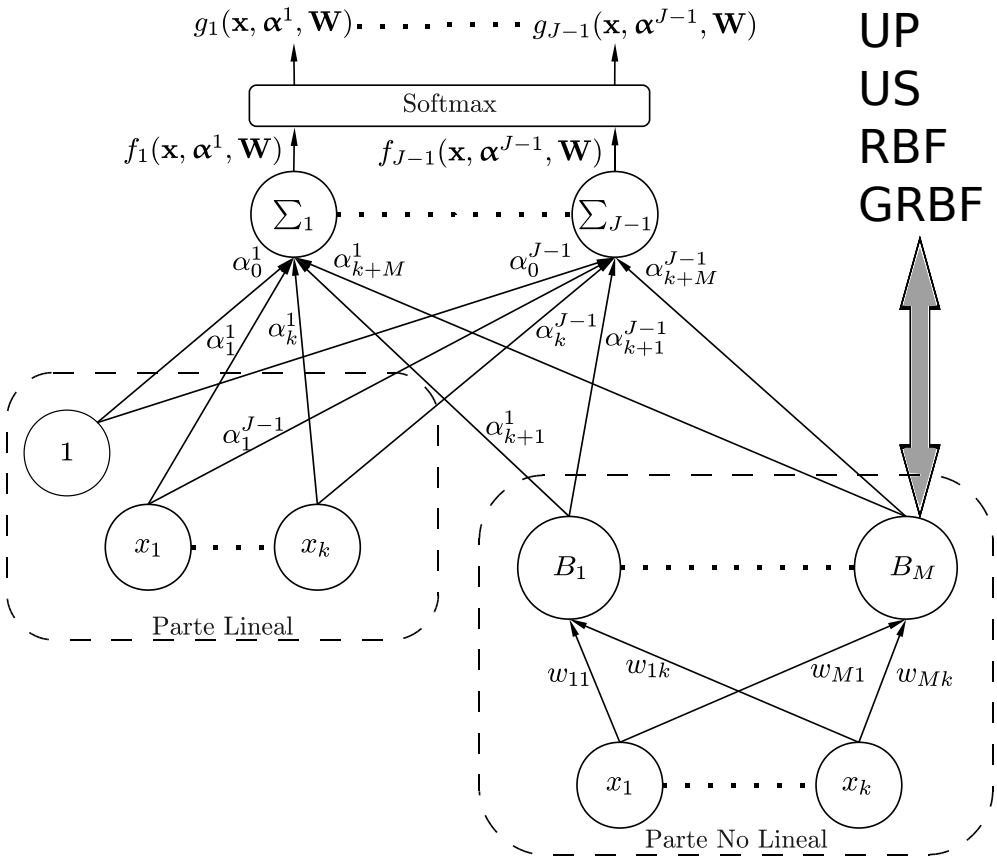


Figura 5.1: Modelo de Regresión Logística utilizando covariables iniciales y unidades de diferente tipo.

Este modelo permite la generación de superficies no lineales de *clasificación* y la identificación de las posibles interacciones de orden superior que puedan 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 es un modelo híbrido que utiliza CE para obtener un conjunto de funciones de base que lleven a resultados lo suficientemente precisos. Una vez obtenidos, se aplica el algoritmo IRLS sobre los nodos (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 $\boldsymbol{\theta}_j = \{\boldsymbol{\alpha}^j, \boldsymbol{W}\}$ es el conjunto de coeficientes asociados al nodo o variable de salida j, el modelo funcional es el siguiente:

$$f_i(\boldsymbol{x}, \boldsymbol{\alpha}^i, \boldsymbol{W}) = \alpha_0^i + \sum_{j=1}^{k} \alpha_j^i x_j + \sum_{j=1}^{M} \alpha_{(k+j)}^i B_j(\boldsymbol{x}, \boldsymbol{w}_j) \tag{5.12}$$

donde B_j representa cada una de las funciones de base del mejor modelo de red obtenido por el algoritmo evolutivo, $\boldsymbol{\alpha}^i = (\alpha_0^i, \alpha_1^i, \ldots, \alpha_k^i, \alpha_{k+1}^i, ..., \alpha_{k+M}^i)$ son los coeficientes asociados a la parte lineal del modelo de Regresión Logística y $\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_M)$ son los coeficientes no lineales asociados a la RNA. Los valores ajustados por el algoritmo IRLS son los que corresponden a los vectores $\boldsymbol{\alpha}^i$, de manera que los pesos $\boldsymbol{W}$ son obtenidos por el algoritmo evolutivo. La arquitectura de estos modelos puede representarse mediante un grafo de red como el de la Figura 5.1.

A continuación se describe de forma general los principales pasos del algoritmo NNEP-RegLog:

Algoritmo NNEP-RegLog
1. Generar una población aleatoria de tamaño $10N_P$ y seleccionar las N_P mejores redes para la población inicial. 2. Repetir hasta que se alcance un número máximo de generaciones G o se cumpla la condición de parada. 2.1 El 10 % de los mejores individuos son replicados y sus réplicas sustituyen al 10 % de los peores individuos (fuerte presión selectiva). 2.2 Aplicar mutación paramétrica al 10 % de mejores individuos de la población. Aplicar mutación estructural al restante 90 % de individuos. 2.3 Calcular la aptitud de cada individuo de la población. 2.4 Ordenar los individuos según su aptitud. 3. Seleccionar el mejor individuo ($\boldsymbol{I}$) en aptitud . 4. Obtener un nuevo espacio de variables ($\boldsymbol{D}$) a partir de las salidas de los nodos de la capa oculta de $\boldsymbol{I}$. 5. Añadir de forma opcional las variables iniciales ($\boldsymbol{X}$) a las variables obtenidas en el paso anterior ($\boldsymbol{D} = \boldsymbol{D} + \boldsymbol{X}$). 6. Aplicar Regresión Logística sobre el espacio de variables $\boldsymbol{D}$.

Trabajos realizados [196] han validado la utilización de la regresión logística sobre transformaciones previas del espacio de variables independientes, constituyendo un campo prometedor para trabajos futuros que hibriden modelos logísticos con transformaciones del espacio de variables independientes obtenidas por otro algoritmo.

En esta tesis se extenderá (capítulo 8) la metodología propuesta en [196] al considerar como nuevas covariables a las variables generadas por los nodos GRBF. Los nodos de tipo radial llevarían a una Regresión Logística con una función de predicción con dos estructuras: una lineal y la otra no lineal formada por funciones locales de tipo Gaussiano Generalizado. De esta forma, la idea sería extraer localizaciones del espacio de entrada, que ayudasen a mejorar la tarea de discriminación.

5.4. Multicolinealidad

5.4.1. Multicolinealidad en los Modelos Logísticos

El término multicolinealidad (o colinealidad) en regresión se refiere a una situación en la que dos o más variables explicativas están fuertemente relacionadas linealmente y, por tanto, resulta difícil cuantificar sus efectos individuales sobre la variable explicada. Este problema reside, por tanto, en la muestra utilizada y/o de la especificación del modelo, y no tiene causas interpretables. Sí existen, en cambio, una serie de situaciones en que la multicolinealidad resulta habitual. Entre los efectos que produce la multicolinealidad se encuentran:

- Las estimaciones individuales de los coeficientes están mal identificadas, pues el valor estimado de un coeficiente puede depender crucialmente del (los) valor(es) estimado(s) de otro(s).
- Se genera un incremento importante de la varianza de los coeficientes estimados.
- Las estimaciones resultan sensibles con respecto a la muestra utilizada, lo que supone que si, por ejemplo, se amplía la muestra con una nueva observación, las estimaciones obtenidas pueden variar sustancialmente.
- Pequeños cambios en los datos producen grandes cambios erráticos en las estimaciones.
- Los coeficientes pueden tener errores estándar grandes y bajos niveles de significación aún cuando sean significativos como grupo y el coeficiente R^2 sea grande.
- Los coeficientes pueden tener el signo equivocado o de magnitud excesivamente grande. Particularmente cuando el rango en que oscilan los valores de una variable es muy pequeño o está compuesto por solo valor (perfecta multicolinealidad), la estimación de los coeficientes del modelo produce valores muy grandes para dar poder discriminante a la variable dentro de la combinación lineal propuesta por los modelos logísticos.

Esta desproporción de los coeficientes provoca el aumento del error en generalización.

Un análisis del proceso de estimación de los coeficientes de los modelos lógisticos nos lleva a tener en cuenta la forma en que el algoritmo IRLS estima los valores de los coeficientes del modelo logśtico mediante el cálculo de la inversa $(\boldsymbol{x}^T W \boldsymbol{x})^{-1}$ (ver fórmula 5.8). En este paso, se presentan los principales problemas introducidos por la multicolinealidad subyacente en las variables independientes. Cuando una de las variables independientes es constante e igual a cero la matriz $\boldsymbol{x}^T W \boldsymbol{x}$ es singular y por tanto no inversible, por ello los coeficientes obtenidos son desproporcionadamente grandes.

Esta característica será tenida en cuenta en la presente tesis en el capítulo 8 donde las variables independientes poseen multicolinealidad afectando a la estimación de la RL.

5.4.2. Multicolinealidad en los Modelos Neuro-Logísticos

Los modelos neuro-logísticos presentan la dificultad de que las covariables generadas por los nodos introducidos en el modelo, pueden presentar multicolinealidad fundamentalmente durante su entrenamiento con algoritmos estocásticos. Durante el ajuste de los pesos de los nodos de forma pseudo-aleatoria se produce en general que algunas de las covariables introducidas toman valor cero.

La causa de que un nodo produzca una covariable "nula" (todos valores iguales a cero) depende de la naturaleza del mismo. Un nodo de tipo UP debido a su estructura multiplicativa (ver 2.4) puede dar lugar a que la potenciación de las variables independientes y su posterior multiplicación produzca valores positivos muy cercanos a cero y menores que el límite de precisión para datos numéricos en un algoritmo numérico.

En el caso de los nodos de tipo US cuando la combinación lineal (función de activación) produce valores (f) negativos suficientemente pequeños ocurre

que:

$$\lim_{f \to -\infty} \frac{1}{1 + e^{-f}} = 0 \tag{5.13}$$

por lo que el nodo de tipo US produce una covariable compuesta por ceros.

Por la importancia que presentan los nodos de tipo RBF para la presente tesis es necesario tener en cuenta que cuando la dimensionalidad aumenta, los patrones tienden a dispersarse en el espacio de las variables independientes, aumentando la magnitud de la distancia euclídea (d). Cuando una unidad se encuentra en un punto alejado de los patrones de entrada ocurre que:

$$\lim_{d \to \infty} e^{-(\frac{d}{r})^2} = 0 \tag{5.14}$$

por lo que los nodos de tipo RBF puede producir una covariable compuesta por ceros.

Atendiendo a esta característica que puede presentarse en los nodos RBF, el entrenamiento de modelos neuro-logísticos mediante algoritmos estocásticos debe tener en cuenta los problemas de multicolinealidad introducidos. El tratamiento de este efecto no deseado se tiene en cuenta en el capítulo 8, donde se entrenan parcialmente modelos neuro-logísticos con Algoritmos Evolutivos.

Capítulo 6

CLASIFICACIÓN DE DATOS DE ALTA DIMENSIONALIDAD

Trabajar con datos de alta dimensionalidad significa trabajar con una muestra de patrones $\{(\mathbf{x}_i, y_i)\}, i = 1, \ldots, N$ donde $x_i \in R^k$, siendo k un número suficientemente alto. Esto significa que cada patrón contiene muchos atributos o características. Los datos espectrales son ejemplos típicos de datos con alta dimensionalidad: dependiendo de la resolución del espectrómetro, los datos espectrales contienen varios cientos de mediciones.

Afortunadamente en el análisis de datos espectrales no todas las coordenadas son independientes: en ese caso, analizando sus dependencias, podemos extraer la información verdaderamente relevante reduciendo de esa manera la dimensionalidad. En términos más generales, la redundancia en los atributos es una condición necesaria para poder reducir la dimensionalidad del espacio de entrada. De hecho, en el caso de que todas los atributos sean independientes, el modelo de regresión lineal simple contendría en este caso tantos parámetros como atributos tenga el espacio de entrada (seleccionaría todas las variables como significativas).

Si el número de muestras disponibles, n, para el aprendizaje es menor que la dimensión del espacio de entrada, k, el problema no está definido (es decir, el modelo no es identificable). Una solución es la de determinar posibles

dependencias entre los atributos con el fin de reducir el número de parámetros del modelo (y así, el número de atributos de los patrones).

Las posibles soluciones para atacar los problemas que surgen en entornos de clasificación de patrones de alta dimensionalidad pasan por:

- Tener en cuenta en el modelo las dependencias entre los atributos, para evitar que el modelo tenga demasiados parámetros que realmente no son necesarios.
- Reducir, cuando sea posible, la dimensionalidad de los datos a través de técnicas de selección y proyección.
- Adaptar el modelo a las características específicas de los espacios de alta dimensionalidad.

6.1. Características de los espacios de Alta Dimensionalidad

Uno de los problemas de los datos en alta dimensionalidad es que escapan a las representaciones mentales humanas, que llegan hasta la tercera dimensión. Aspectos que damos por sentado en espacios de dimensión uno, dos o tres, pueden no ocurrir en espacios de mayor dimensión. A continuación destacaremos algunos hechos extraños que ocurren en espacios de alta dimensionalidad.

6.1.1. El fenómeno del espacio vacío

Scott y Thompson [197] fueron los primeros en señalar algunos hechos que son contrarios a la intuición de la mente humana relacionados con espacios de alta dimensionalidad euclídeos, y describieron lo que ellos denominaron como "el fenómeno del espacio vacío".

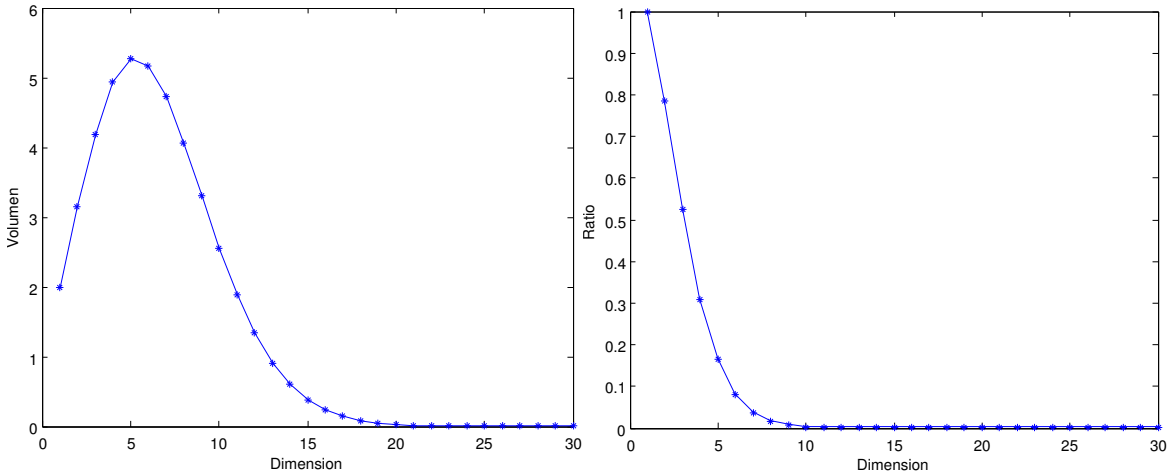
- El volumen de la hiper-esfera de radio uno tiende a cero conforme la dimensión aumenta. El volumen de una esfera de radio r en d dimensiones

se define como:

$$V(d) = \frac{\pi^{d/2}}{\gamma(d/2+1)} r^d \tag{6.1}$$

La Figura 6.1a muestra el volumen para $r = 1$. Como podemos observar el volumen decrece rápidamente con la dimensión d. Por lo tanto, en alta dimensionalidad, la Esfera Unidad esta casi vacía.

- El ratio entre el volumen de una esfera de radio uno y un cubo del mismo radio tiene a cero conforme la dimensión crece, tal y como puede observarse en la Figura 6.1b. En una dimensión, los volúmenes son iguales, mientras que en dos dimensiones el ratio es aproximadamente 0.8; sin embargo en alta dimensionalidad podemos afirmar que el volumen del hiper-cubo se concentra en la esquina (el volumen que no ocupa la esfera). Estas observaciones implican que en espacios de alta dimensionalidad los patrones se encuentran en los bordes del dominio.



(a) Volumen de la Esfera Unidad

(b) Ratio entre el volumen de la Esfera Unidad y el Cubo Unidad

Figura 6.1: Ilustración del efecto de espacio vacío en Alta Dimensionalidad.

6.1.2. El fenómeno de la concentración de la medida

En esta sección analizaremos de una manera más detallada el comportamiento de la distancia Euclídea (es decir, la norma L_2) cuando se aplica a

vectores de alta dimensionalidad.

- La desviación típica de la norma de vectores aleatorios converge a un valor constante conforme aumenta la dimensionalidad, mientras que la media de su norma crece conforme a la raíz cuadrada de la dimensión. De forma más precisa, se demostró que bajo el supuesto de tener variables aleatorias x_i que son independientes e idénticamente distribuidas (i.i.d):

$$\mu_{\|x\|} = E(\|x\|) = \sqrt{an - b} + O(1/n) \quad (6.2)$$
$$\sigma_{\|x\|} = Var(\|x\|) = b + O(1/\sqrt{1}) \quad (6.3)$$

  donde a y b son constantes que dependen solo de los cuatro primeros momentos de las variables x_i.

  Hay que tener en cuenta que estas observaciones también podrían aplicarse a la distancia Euclídea de dos patrones ya que ésta podría considerarse también como un vector aleatorio.

- La diferencia entre las distancias de puntos elegidos aleatoriamente a sus vecinos más lejanos y cercanos decrece conforme aumenta la dimensionalidad. Esto se puede ilustrar por el comportamiento asintótico del contraste relativo [198]:

$$\text{si } lim_{d \leftarrow \infty} Var\left(\frac{\|x\|_k}{E(\|x\|_k)}\right) = 0 \text{ entonces } \frac{D^k_{max} - D^k_{min}}{D^k_{min}} \rightarrow 0 \quad (6.4)$$

  donde D_{min} y D_{max} son las distancias respectivas a los vecinos más lejanos y cercanos a un determinado patrón. Hay que tener en cuenta que la hipótesis del teorema se ha deducido de la ecuación anterior. En el artículo de Beyer [199] podemos encontrar una demostración general del teorema.

La conclusión que podemos obtener de esas observaciones es que en espacios de alta dimensionalidad todos los puntos tienden a estar a igual distancia entre ellos (teniendo en cuenta la distancia Euclídea). Conforme la dimensión aumenta, la distancia observada entre dos patrones tiende a ser constante.

6.1.3. La maldición de la dimensionalidad

La "maldición de la dimensionalidad" es el problema causado por el crecimiento exponencial del volumen de información asociado al incremento del número de dimensiones en el espacio de características. El término fue acuñado por Bellman [200]. Este problema es más intuitivo que los dos anteriores pero a menudo ha sido obviado en los trabajos de investigación. Hace referencia a que en espacios de alta dimensionalidad es necesario una gran cantidad de patrones representados por puntos para cubrir un determinado espacio de entrada, utilizando por ejemplo una malla de patrones equidistantes para abarcar ese determinado espacio. A modo de ejemplo, para cubrir un hiper-cubo en cinco dimensiones ($[0-1]^5$) con una malla de patrones distanciados a 0.1, se necesitan más de 100.000 puntos.

6.2. Consecuencias en el aprendizaje de RNA

Las consideraciones que se han analizado en la sección anterior tienen importantes consecuencias en el aprendizaje de las RNAs. Las siguientes subsecciones detallarán ejemplos de tales consecuencias en contextos específicos.

6.2.1. Aprendizaje supervisado

Cuando modelamos un determinado proceso produciendo una salida en función de unas entradas, tenemos que entrenar un determinado modelo (en nuestro caso RNAs) para unos datos. Cuanto mayor sea el conjunto de datos, mayor será la precisión del modelo. Lo ideal sería que el conjunto de datos pudiera cubrir todo el espacio de entrada para poder asegurarnos de que cualquier valor predicho (es decir, un valor de la salida del modelo) sea el resultado del proceso de interpolación y que se produzcan extrapolaciones extremas.

B. W. Silverman [201] abordó el problema de encontrar el número de patrones necesarios para aproximar una distribución Gaussiana mediante *kernels* Gaussianos. Sus resultados mostraron que el número de patrones necesarios crece de forma exponencial con la dimensión. Fukunaga [202] obtuvo valores

similares con el clasificador k-NN. En su estudio demostró que para una dimensión 4 eran necesarios solo 44 patrones, sin embargo con una dimensión de 128, el número de patrones necesarios era de $3.8e^{57}$.

6.2.2. Modelos Locales

En general, los modelos de RNAs locales (redes RBF) sufren más los efectos de la alta dimensionalidad que los modelos de tipo proyección (redes MLP o redes UP). Cuando se asume una distribución normal de desviación típica $\sigma = 1$, la función de densidad de probabilidad de encontrar un punto a una distancia q del centro de la distribución es la siguiente:

$$f(q, d) = \frac{q^{d-1}}{2^{d/2-1}} \frac{e^{-q^2/2}}{\Gamma(d/2)} \tag{6.5}$$

donde d es la dimensión. En una dimensión, esa función es máxima en el centro de la distribución, su media, (tal y como cabía esperar). Sin embargo, cuando la dimensión crece, el máximo de la función diverge del centro (Figura 6.2). Esto muestra que los kernels Gaussianos no son locales en alta dimensión y que los modelos que son suma de kernels locales (como las redes RBF) no tienen un buen comportamiento en problemas de alta dimensiónalidad.

Esta limitación se verifica en particular en el caso de kernels Gaussianos. Sin embargo, deberíamos enfatizar el hecho de que los modelos globales, como por ejemplo redes MLP, sufren probablemente una limitación similar. De hecho, en muchos casos, la suma de sigmoides, como ocurre en las redes MLP, construyen funciones que toman valores significativos en determinadas regiones limitadas del espacio. Mientras que matemáticamente son diferentes, en muchas ocasiones modelos como redes MLP y redes RBF ofrecen un comportamiento similar. Esto refuerza la convicción de que los modelos locales y globales sufren de igual forma el problema de la maldición de la dimensionalidad, aunque esta convicción es más difícil de demostrar en modelos globales.

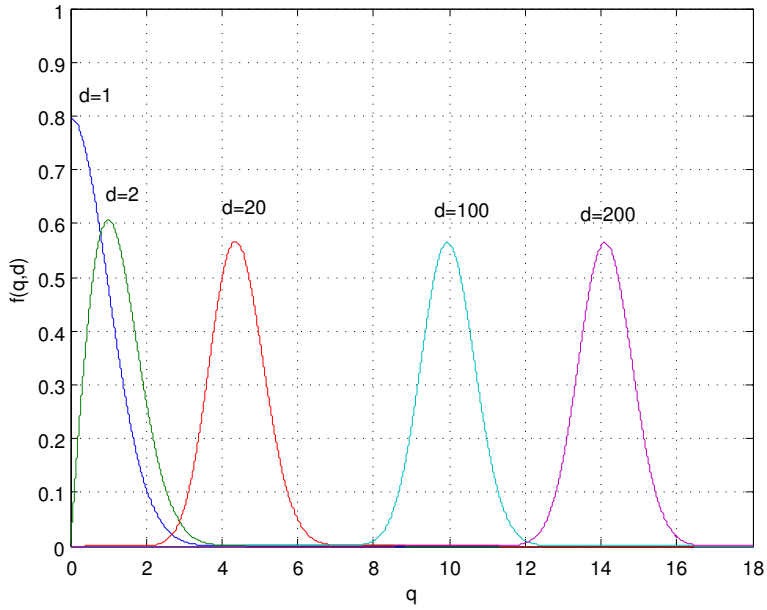


Figura 6.2: Función de densidad de probabilidad de que un punto de una distribución normal esté a una distancia q de la media para diferentes valores de d.

6.2.3. Búsqueda de similaridad y distancia Euclídea

Tanto la mayoría de las RNAs como las técnicas de agrupamiento o *clustering* requieren del cálculo de la distancia entre vectores. En el caso de redes RBF, se requiere del cómputo de la distancia de cada patrón al centro. En el caso de redes MLP, se necesita el cálculo del producto escalar entre el patrón y los pesos de capa de entrada a la capa oculta. Ambas distancias están íntimamente relacionadas con la búsqueda de similaridad en las técnicas de *clustering*, también utilizada en los mapas de Kohonen, el algoritmo k-medias, etc. La búsqueda de similaridad consiste en encontrar en un conjunto de datos el patrón más cercano a un determinado patrón.

En el contexto de las técnicas de *clustering* por ejemplo, se obtiene un análisis *cluster* eficiente cuando los patrones en un *cluster* son muy similares (es decir, cercanos con respecto a la distancia considerada) y cuando los patrones en diferentes cluster están alejados unos de otros. De esta forma, cuando los datos están organizados en *clusters*, el histograma de distancia deberían idealmente mostrar dos picos: uno para la distancia "intra-clusters" y otro para

la distancia "inter-clusters".

Sin embargo, si el histograma de distancias solo contiene un pico o si los dos picos están muy próximos, las técnicas de *clustering* basadas en distancias se complican. Desafortunadamente, tal y como ha apuntado Guo [203], el hecho de que en alta dimensionalidad cualquier histograma de distancias tiende a tener un único pico cada vez más concentrado, provoca que la tarea de determinar *clusters* en los patrones sea bastante complejo. Este problema es una consecuencia directa del fenómeno de concentración de las distancias analizado en secciones anteriores.

6.3. Posibles Soluciones al Problema de la Alta Dimensionalidad

6.3.1. Reducción de la dimensionalidad mediante proyecciones no lineales

Una forma de limitar los efectos de la alta dimensionalidad es reducir la dimensión del espacio de características. Los patrones en problemas reales a menudo se encuentran dentro o cerca de subespacios (*submanifolds*) del espacio de entrada debido a la redundancia de las variables. Una posible solución es proyectar los datos en *submanifolds*. Una forma de proyectar los datos es utilizar la técnica de PCA. El problema de la técnica PCA es que es lineal y en muchos casos, los *submanifolds* no son lineales haciendo que la aplicación de dicha técnica no sea eficiente.

En la actualidad existen métodos alternativos para proyectar los patrones de una forma no lineal. Algunos ejemplos de ello podrían ser los mapas autoorganizados de Kohonen (utilizados generalmente para proyectar los datos en una o dos dimensiones), y métodos basados en preservar las distancias. Éstos últimos incluyen las técnicas de escalado multi-dimensional [204, 205], mapas de Sammon [206], análisis de componentes curvilíneos [207], y extensiones de dichas técnicas. Todas esas técnicas están basadas en el mismo principio: si tenemos n patrones en un espacio d-dimensional, estas técnicas intentan situar

los n patrones en un espacio proyectado m-dimensional mantenimiento iguales las distancias entre cualquier par de patrones en el espacio de entrada y el par correspondiente en el espacio de proyección.

6.3.2. Selección de Características

La selección de características es un proceso que consiste en seleccionar un subconjunto óptimo de características de una base de datos para reducir su dimensionalidad, eliminar ruido y mejorar el desempeño de un algoritmo de aprendizaje: velocidad de aprendizaje, precisión de la predicción (medido con la tasa de error) y comprensibilidad de los resultados producidos.

Las características de una base de datos pueden ser: fuertemente relevantes, débilmente relevantes, irrelevantes y redundantes. El subconjunto óptimo de características está compuesto entonces por las características fuertemente relevantes y las débilmente relevantes, pero no redundantes. Encontrar el subconjunto óptimo de características requiere una búsqueda en el espacio de subconjuntos posibles de características de la base de datos, que es un problema no determinístico polinomial complejo [208]. Dash y Liu [209] apuntaron que un proceso típico de selección de características se ejecuta en 4 pasos:

- Generación del subconjunto, consiste en el mecanismo de búsqueda que produce subconjuntos de características candidatos a ser evaluados. La búsqueda completa garantiza el hallazgo del subconjunto óptimo, sin tener la necesidad de realizar una búsqueda de todos los posibles subconjuntos (2^n) del total de n características, que es una búsqueda exhaustiva [210]. La búsqueda secuencial genera subconjuntos de manera directa, comienza con un subconjunto vacío, para luego agregarle características relevantes de manera progresiva (selección secuencial hacia adelante), o viceversa: comenzar con todo el conjunto y eliminar características irrelevantes de manera progresiva (selección secuencial hacia atrás). La búsqueda aleatoria genera subconjuntos de manera aleatoria, luego aumenta o disminuye características también aleatoriamente para generar el siguiente subconjunto que será evaluado.

- Evaluación del subconjunto, consiste en medir la optimalidad del subconjunto generado en el paso anterior para los fines de un problema de aprendizaje, que en este trabajo es de clasificación. El criterio de evaluación *Filtro* es independiente del algoritmo de aprendizaje (por ejemplo RNAs, SVMs, etc.). La evaluación tipo *envolvente*, depende del algoritmo de aprendizaje que se use. Este último genera un costo computacional mayor al de los algoritmos pertenecientes al Modelo de *Filtro*.

- Criterio de parada, determina cuándo un proceso de selección de características deberá parar. Generalmente el proceso de selección de características se detiene cuando se alcanza el valor de algún parámetro o umbral que se estableció "a priori"; se terminó de realizar la búsqueda completa; o se encontró un subconjunto de características óptimo.

- Validación de resultados, es la evaluación del modelo de selección de características con datos reales y ver si su desempeño lo refleja el resultado de los experimentos.

Los algoritmos de selección de características de tipo *envolvente* se diferencian en el método de búsqueda de la generación del subconjunto. Los principales métodos de búsqueda son:

- Búsqueda Genética, realiza la búsqueda usando el algoritmo genético simple descrito por Goldberg [211].

- Mejor Primero, realiza la búsqueda en el espacio de subconjuntos de características pasando por diferentes capas del espacio de búsqueda [212].

- Búsqueda Aleatoria, realiza la búsqueda comenzando con un subconjunto aleatorio de características, buscando subconjuntos que tengan menor número de características y que generen los menores errores de aprendizaje.

- Búsqueda Aleatoria Optimizada, es una modificación del algoritmo presentado por Stracuzzi [213]. Empieza con el subconjunto completo de características y elimina características irrelevantes aleatoriamente.

6.3.3. Medidas de Distancia Alternativas

La distancia Euclídea es la distancia convencional para determinar la distancia entre dos patrones y, en general, no ha sido cuestionada por la comunidad científica. Sin embargo, en algunas circunstancias, como en el caso de problemas de alta dimensionalidad, sería necesario definir una nueva función de distancia para cuantificar similaridades en alta dimensionalidad. En la práctica, podríamos considerar la r-distancia como una medida de distancia entre dos patrones $\boldsymbol{x}$ e $\boldsymbol{y}$ (con componentes x_i e y_i) con la siguiente forma:

$$\|\boldsymbol{x} - \boldsymbol{y}\|_r = \sqrt[r]{\sum_{i=1}^{d} \mid x_i - y_i \mid^r} \tag{6.6}$$

Como ya se analizó en secciones anteriores, la diferencia entre las distancias de patrones elegidos aleatoriamente a sus vecinos más lejanos y cercanos decrece conforme aumenta la dimensionalidad. Esta afirmación también es aplicable para la distancia definida anteriormente para cualquier valor positivo de r. Sin embargo, el ratio de convergencia de la ecuación 6.4 difiere para diferentes valores de r. La experimentación nos dice que utilizando valores altos de r podemos mitigar el efecto de la perdida de localidad para los kernels Gaussianos.

Debemos recordar que la función GRBF parametriza a la función SRBF de una manera similar a como la r-distancia parametriza a la distancia Euclídea. Por lo tanto, tiene sentido que utilizando valores altos de τ podamos mitigar en cierto sentido la perdida de localidad de los kernels locales.

La Figura 6.3 muestra una ilustración gráfica de porqué la función de base GRBF cuantifica mejor similaridades que la función de base SRBF en espacios de alta dimensionalidad: las líneas rojo y cian representan a funciones SRBFs centradas en el origen y con radios $r = 1.6$ y $r = 10.6$ respectivamente, la línea magenta representa una función de base GRBF centrada en el origen con $\tau = 13$ y $r = 10.6$ y las líneas azul y verde representan a las funciones de densidad de probabilidad de encontrar un patrón de una distribución normal a una distancia q para $d = 2$ y $d = 100$, respectivamente.

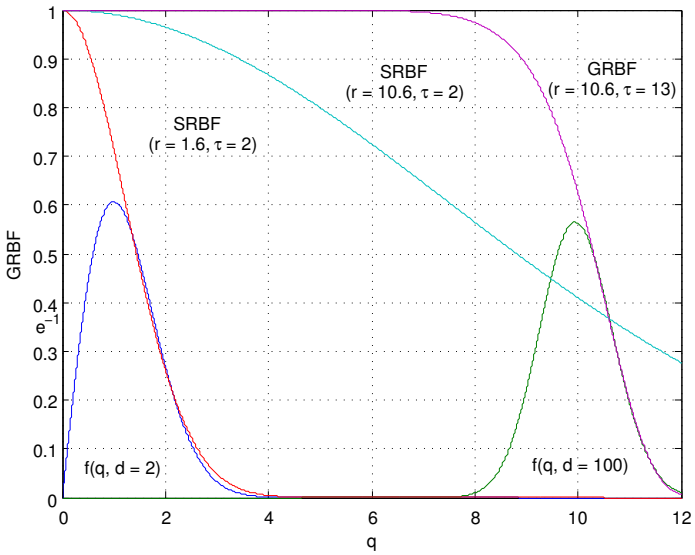


Figura 6.3: Salidas de funciones RBF en función de la distancia a sus centros para dos posibles espacios de dimensión ($d = 2$ y $d = 100$), junto a la distribución de las distancias para datos distribuidos normalmente ($f(q, d)$).

Como podemos observar en la Figura 6.3 cuando $d = 2$, la función de base SRBF demuestra su capacidad para cuantificar similaridades en esa dimensión, asignando valores de pertenencia a los patrones en el intervalo $[0, 1]$ (ver línea roja); sin embargo cuando $d = 100$, la función de base SRBF asigna valores de pertenencia en el intervalo $[0.27, 0.57]$ (ver línea cian) ya que para modelar los patrones en esa dimensión la función SRBF necesita un alto valor del radio. El problema es que la SRBF tiene una curvatura poco pronunciada cuando tiene un alto valor del radio. Por otra parte, la función de base GRBF asigna valores de pertenencia en el intervalo $[0, 1]$ incluso en espacios de alta dimensionalidad (ver línea magenta).

En nuestra opinión, esto justifica que consideremos a la función de base GRBF un kernel válido para cuantificar similaridades en espacios de alta dimensionalidad.

La utilización de nodos GRBF permite asignar altos grados de activación a patrones alejados del nodo, permitiendo tanto "localidad" como cobertura. En la Figura 6.4 se representa con una curva de color negro la probabilidad de encontrar en un conjunto de datos que siguen una distribución multivariante un

patrón a una distancia x; además se observa cómo la SRBF asigna valores de activación en un intervalo reducido y acotado por las ordenadas de los puntos C y D mientras que la GRBF es capaz de asignar valores de activación en un intervalo mucho más amplio acotado por las ordenadas de los puntos A y B:

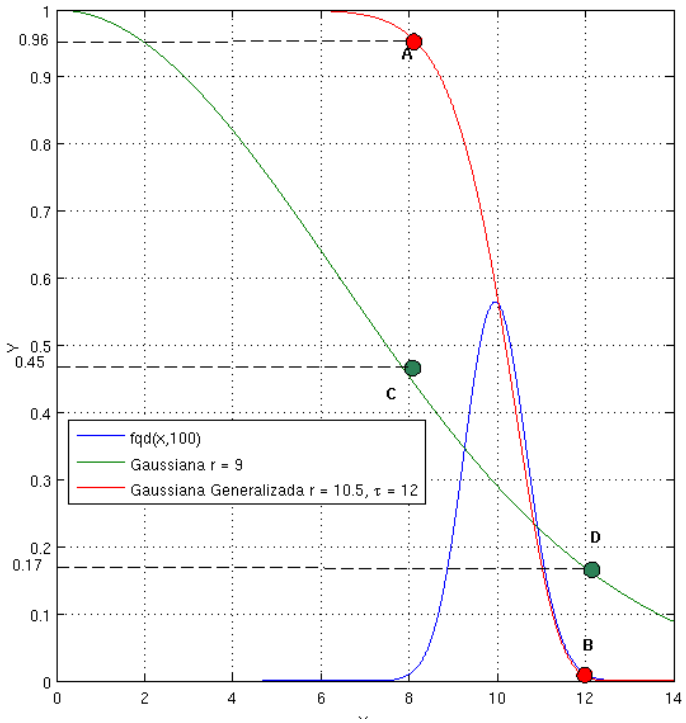


Figura 6.4: Comparación entre la capacidad de ajuste de la función Gaussiana y la función Gaussiana Generalizada.

Capítulo 7

REDES NEURONALES DE NODOS DE TIPO RADIAL GENERALIZADOS

7.1. Introducción

La utilización de modelos locales y en especial de los nodos GRBF ha ido en aumento en la última década siendo utilizados con múltiples fines [214, 106, 107, 215, 109, 216, 217] provocando el interés creciente de los investigadores [218, 219, 220, 221, 222, 108, 223, 224, 225, 226]. Por lo que el empleo de nodos GRBF y su modelado con algoritmos evolutivos constituye un camino "nuevo" y sugerente.

En el presente capítulo se describe la propuesta de un AH para entrenar modelos de RNAs con nodos GRBF [227, 228]. Primeramente se describen brevemente las características del modelo a entrenar y se propone una reformulación de los nodos GRBF (sección 7.2). Posteriormente se presenta el algoritmo propuesto (sección 7.3) y se muestran los resultados obtenidos de la comparación del algoritmo respecto a la aplicación del mismo pero sobre modelos de RNAs con nodos SRBFs, UPs, USs obre datos pertenecientes a la UCI Machine Learning Repository (sección 7.4) .

7.2. Redes Neuronales con Nodos GRBF

Las RNAs con nodos GRBF son modelos locales que permiten una eficaz aproximación de funciones sobre un espacio de variables independientes de alta dimensionalidad gracias a la posibilidad de ajustar su curvatura debido a la introducción del parámetro τ al modelo. No obstante, su ajuste se dificulta pues produce un comportamiento diferente sobre la curvatura de la GG para diferentes valores del radio y fundamentalmente por la característica de que iguales modificaciones en el valor de τ no provoca iguales efectos en la curvatura de la GG. Con el objetivo de eliminar esta dificultad se reformuló la función de la GG en términos del ángulo que forma la recta tangente a la Gaussiana en el punto de inflexión (*cuando* $x = r$) y el eje de las abscisas, como se muestra en la Figura 7.1 luego la función de activación se reformula como:

$$f(x, r, \tau) = e^{-(\frac{x}{r})^{\tau}} \tag{7.1}$$

$$f(x, r, \delta) = e^{-(\frac{x}{r})^{e \times r \times tg\delta}} \tag{7.2}$$

$$f(x, r, \gamma) = e^{-(\frac{x}{r})^{-e \times r \times tg\beta}} \tag{7.3}$$

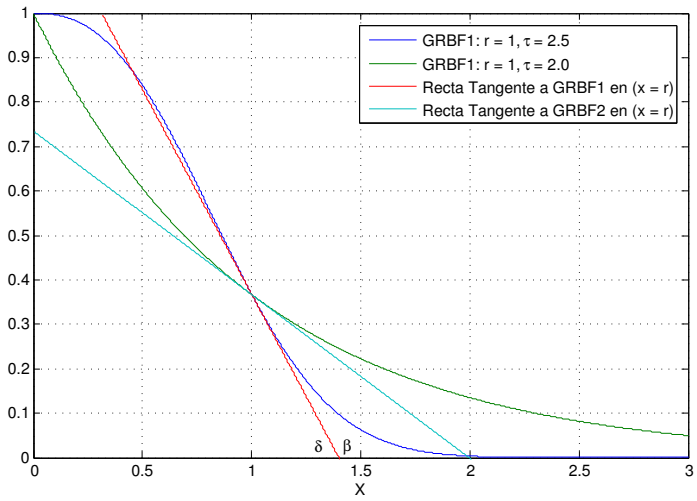


Figura 7.1: Dependencia de la curvatura de la Gaussiana respecto al ángulo que forma la tangente con el eje de abscisas.

A continuación se muestra cómo se obtuvo la expresión $\tau = e \times r \times tg(\delta)$ auxiliándonos de la Figura 7.1 :

1. Se define la función de la GRBF
   $f(x) = e^{-(\frac{x}{r})^{\tau}}$ donde $-\infty < x < \infty$
2. Se deriva la función GRBF respecto x
   $f'(x) = -\frac{e^{-(\frac{x}{r})^{\tau}} \tau (\frac{x}{r})^{\tau-1}}{r}$
3. Se evalúa $f'(x)$ en el punto donde $x = r$
   $f'(r) = -\frac{\tau}{e*r}$
4. Se define la propiedad de la tangente a una curva en un punto
   $f'(r) = tg(\beta)$
5. Se igualan las dos expresiones de $f'(r)$
   $-\frac{\tau}{e \times r} = tg(\beta)$
6. Se despeja τ en la fórmula anterior
   $\tau = -e \times r \times tg(\beta)$
7. Se establece la propiedad de complementariedad de tangentes (ver Figura 7.1)
   $-tg(\beta) = tg(\delta)$
8. Se sustituye fórmula 7 en la fórmula 6
   $\tau = -e * r * tg(\beta) = e * r * tg(\delta)$

La nueva formulación de los nodos GRBF permite separar el ajuste de la curvatura de la Gaussiana, del ajuste del "radio" de influencia del nodo. Un ejemplo demostrativo se observa en la Figura 7.2 donde las GGs representadas de color rojo y verde poseen la misma curvatura y comparten el mismo valor del parámetro δ (0.97) mientras que las GG de color rojo y azul no presentan la misma curvatura a pesar de que poseen el mismo valor del τ (3).

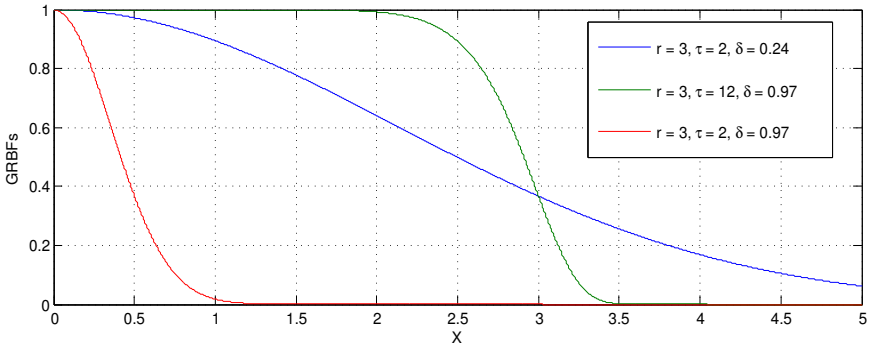


Figura 7.2: Gaussianas Generalizadas (color rojo y verde) con igual curvatura comparadas respecto a una Gaussiana estándar.

Por otra parte para un mismo valor del radio, los nodos GRBFs definidos en términos de δ permiten un ajuste más "rápido" de la curvatura. Un aspecto a tener en cuenta es que el parámetro τ toma valores en el intervalo abierto $[0, \infty)$ y el parámetro δ oscila en el intervalo cerrado $[0, \Pi/2]$. Además, a diferencia de la formulación en términos de τ (ver Figura 2.4) la GG en términos de δ permite realizar cambios en igual magnitud sobre la curvatura de la Gaussiana. En la Figura 7.3 se observan dos pares de GGs con $\delta_1 = 0.634$ y $\delta_2 = 0.834$ para dos valores distintos del radio (0.5 y 6.0); ambos pares de GGs presentan "gráficamente" iguales variaciones en la curvatura independientemente del valor del radio .

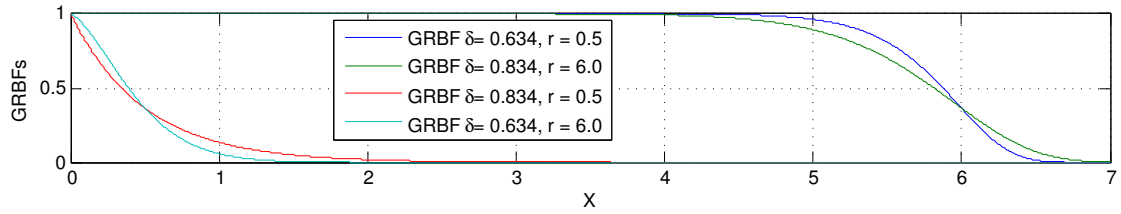


Figura 7.3: Representación de dos pares de GGs con igual radio (cada par: 0.5 y 6.0) e igual curvatura.

Una vez reformulada la GG se elaboró un algoritmo para el entrenamiento de GRBFNN el cual se describe en la siguiente sección 7.3.

7.3. Entrenamiento de las Redes Neuronales de Base Radial Generalizada

Como se planteó con anterioridad la presente tesis propone una extensión del algoritmo NNEP para el entrenamiento de RNA con nodos GRBF. A continuación se representa el esquema general del algoritmo destacando con color verde las secciones que han sido sujetas a cambios para permitir la modelación de los nodos GRBF:

Algoritmo NNEP-GRBF
1. Generar una población aleatoria de tamaño $10N_P$ y seleccionar las N_P mejores redes para la población inicial. 2. Repetir hasta que se alcance un número máximo de generaciones G o se cumpla la condición de parada. 2.1 Aplicar mutación paramétrica al 10 % de mejores individuos de . la población 2.2 Aplicar mutación estructural al restante 90 % de individuos. 2.3 Calcular la aptitud de cada individuo de la población. 2.4 Ordenar los individuos según su aptitud. 2.5 El 10 % de los mejores individuos son replicados y sus réplicas sustituyen al 10 % de los peores individuos (fuerte presión selectiva). 5. Seleccionar el mejor individuo en aptitud, devolviéndolo como solución factible al problema.

La introducción de los nodos GRBF en la capa oculta supone la modificación de su proceso de inicialización (sección 7.3.0.1), la forma en que se realiza la mutación paramétrica de los nodos en cada iteración debido a la introducción del nuevo parámetro δ y por último la forma en que se adiciona un nuevo nodo GRBF o cuando se unen dos nodos durante la mutación estructural. A continuación se detallan las modificaciones incorporadas al algoritmo original.

7.3.0.1. Inicialización de los Individuos

La inicialización de cada individuo de la población y en particular de las GRBFs pertenecientes a la capa oculta se realiza haciendo uso del algoritmo K-Medias [126, 154] para asignarle un peso inicial a los arcos (links) de cada GRBF, quedando solo por estimar los parámetros r y δ de cada nodo. Estos nuevos parámetros dependen de la distribución de la distancia de los puntos de cada cluster a su centroide relativo. Bajo el supuesto de normalidad en la distribución de los datos de entrada, los valores de distancia antes mencionados presentan la característica de que su media se "separa" de cero cuando la dimensionalidad aumenta o cuando los datos se encuentran en el borde del espacio de dimensión D en el que se encuentran los patrones como se describe en la Figura 6.2.

Dada la capacidad de ajuste de los nodos GRBF ante datos de alta dimensionalidad (ver sección 6.2.2) se han producido varios acercamientos a la estimación de los parámetros θ y τ. Francois [229] asumiendo normalidad en la distribución de las variables independientes fija las cotas d_n y d_f, las cuales representan las distancias hasta las que se pueden encontrar el 5 y 95 % de los patrones alrededor del centroide. Planteando el siguiente sistema:

$$e^{-(\frac{d_n}{r})^\tau} = 0.95 \tag{7.4}$$

$$e^{-(\frac{d_f}{r})^\tau} = 0.05 \tag{7.5}$$

Este planteamiento tiene la dificultad de que d_n puede tomar valores menores que cero por lo que redefinimos el sistema como:

$$e^{-(\frac{\mu}{r})^\tau} = 0.5 \quad (7.6)$$

$$e^{-(\frac{d_f}{r})^\tau} = 0.05 \quad (7.7)$$

Donde μ es la media de las distancias al centro del cluster (recordemos que $\tau = e \times r \times tg(\delta)$), permitiendo que nunca se indefina. Resolviendo el sistema se obtiene como resultado:

$$\tau = \frac{\ln \frac{\ln 0.05}{\ln 0{,}5}}{\ln \frac{\mu}{d_f}} ==> e \times r \times tg\delta \quad (7.8)$$

$$r = \frac{d_f}{(-\ln 0.05)^{1/\tau}} \quad (7.9)$$

El valor 0.5 para el valor de activación de la (GG) es fijado en la fórmula 7.6 para simular la probabilidad de que un punto (una distancia) se encuentre antes de la media. Otros enfoques pueden fijar valores cercanos a 1 (ej. 0.95) con el objetivo de darle más grado de pertenencia al patrón respecto al cluster que pertenece pues la mayor probabilidad de encontrar un patrón se encuentra alrededor de la media. Ambos enfoques son válidos y aplicables, no obstante, es necesario puntualizar que el valor de la imagen de la GG para cualquier x $\neq$ 0 no puede alcanzar el valor 1 pues se indefine la función 7.8.

Otra forma de inicializar el valor de $\delta = arctg(\frac{2}{e \times r})$ equivalente a $\tau = 2$ para simular el comportamiento de las SRBF, seguidamente el radio se inicializa con la media cuadrática de las distancias a las dos nodos más cercanos. Esta segunda variante se ajustaría menos a los datos iniciales y dejaría el ajuste de los coeficientes del modelo a las siguientes fases del algoritmo evolutivo. De forma resumida la segunda variante de inicialización se puede expresar como:

$$\tau = 2 ==> e \times r \times tg\delta \tag{7.10}$$

$$r = \sqrt{d_1 \times d_2} \tag{7.11}$$

donde d_1 y d_2 son las distancias a las dos nodos más cercanos.

7.3.0.2. Mutación paramétrica de los nodos GRBF

Las mutaciones paramétricas realizadas por el algoritmo descrito se semejan a las descritas en la sección 3.3.6.1 a la cual se le incorpora la mutación del parámetro δ introducido en los modelos GRBFNN. La mutación de los parámetros del modelo consiste en añadir a cada peso w_{ji} (capa oculta) y β_{ij} (capa de salida) un ruido gaussiano, donde la varianza de la distribución de Gauss depende de la temperatura de la red. Concretamente, los pesos w_{ji} de la capa intermedia quedan actualizados de la forma:

$$w_{ij}(t+1) = w_{ij}(t) + \xi_1(t) \tag{7.12}$$

$$i \in [0, ..., n], j \in [1, ..., M]$$

siendo $\xi_1(t)$ una variable aleatoria con distribución normal $N(0, \alpha_1(t)T(\hat{\theta}))$. En cuanto a los pesos de la capa de salida se actualizarán de forma análoga:

$$\beta_{ij}(t+1) = \beta_{ij}(t) + \xi_2(t) \tag{7.13}$$

$$i \in [1, ..., L-1], j \in [0, ..., M]$$

siendo $\xi_2(t)$ una variable aleatoria con distribución normal $N(0, \alpha_2(t)T(\hat{\theta}))$.

Los cambios en los pesos se realizan de forma secuencial en los nodos de la capa oculta, de manera que, para cada nodo j de la capa oculta, se modifican los pesos w_j, y seguidamente se actualizan los pesos de la capa de

salida β_{ij} donde $1 \leq i \leq L-1$. Una vez realizados los cambios en los pesos, la aptitud del individuo es recalculada para aplicar un algoritmo estándar de enfriamiento simulado. Si llamamos ΔA a la variación de la aptitud antes y después del cambio en los pesos, resulta que:

$$R(\Delta A) = \begin{cases} Si\ \Delta A \geq 0, & \text{se acepta el cambio} \\ Si\ \Delta A \leq 0, & \text{se acepta el cambio si } e^{\frac{\Delta A}{T}} < \gamma, \text{ donde } \gamma \in U(0,1) \end{cases} \tag{7.14}$$

Este proceso de enfriamiento simulado se repite para cada red mutada. El ajuste de los parámetros que intervienen en el algoritmo se describen en la sección 3.3.6.1.

Por último se ajusta el valor del parámetro δ resumido en la expresión:

$$\delta = \delta + signo(z - 0.5) * \frac{\pi}{90} * z \tag{7.15}$$

donde z es un número generado aleatoriamente siguiendo una distribución uniforme en el intervalo $[0, 1]$.

7.3.0.3. Mutación estructural de los nodos GRBF

La mutación estructural modifica el número de nodos ocultos y el número de conexiones entre los nodos de la capa oculta afectando de esta forma a la topología de la red y permitiendo explorar nuevas regiones del espacio de búsqueda. Como se describe en la sección 3.3.6.2, cinco mutaciones estructurales distintas se aplican sobre los individuos de la población : Añadir Nodos (AN), Eliminar Nodos (EN), Añadir Conexiones (AC), Eliminar Conexiones (EC) y Unir Nodos (UN). Entre las posibles mutaciones solo AN y UN necesitan una estrategia para establecer el valor del nuevo parámetro δ.

Cuando el algoritmo se dispone realizar la adición de un nuevo nodo ejecuta los pasos descritos en la sección 3.3.6.2 incorporando la inicialización del parámetro $\delta = arctg(\frac{2}{e \times r})$ para simular un nodo SRBF. La causa fundamental

de esta inicialización es que en el momento de adicionar un nuevo nodo, no se conoce la distribución de los patrones a su alrededor.

Si bien la creación de nuevos nodos GRBF permite colocar nodos en zonas "no exploradas" del espacio de búsqueda, la unión de dos nodos permite obtener modelos de RNAs más simples. Para definir el procedimiento para Unir dos Nodos llamemos $\boldsymbol{c}_j$ al centro definido por el nodo j, es decir, $(w_{1j}, w_{2j}, ..., w_{kj})$ y r_j a su radio. En este caso, la mutación FN de dos GRBF, a y b, consistirá en calcular el centro y radio del nodo c como:

$$\boldsymbol{c}_c = \frac{r_a}{r_a+r_b}\cdot\boldsymbol{c}_a + \frac{r_b}{r_a+r_b}\cdot\boldsymbol{c}_b \qquad r_c = \frac{r_a+r_b}{2}$$

De igual forma se calcula el nuevo valor del parámetro δ como la media de los parámetros pertenecientes a los nodos a unir:

$$\delta_c = \frac{\delta_a+\delta_b}{2}$$

7.4. Experimentación

Tras la elaboración del nuevo algoritmo evolutivo se procedió a pasar a la fase de experimentación descrita a continuación mediante un análisis de las bases de datos utilizadas y los resultados obtenidos por las GRBFNNs sobre datos de la UCI.

7.4.1. Descripción de las bases de datos y del diseño experimental

La metodología propuesta es aplicada sobre 13 bases de datos pertenecientes a la UCI [4]. Para realizar la comparación se establece un marco de experimentación donde se utiliza un único AE (el NNEP) para entrenar modelos de RNAs con la estructura descrita en la Figura 2.2. Al mismo tiempo cada algoritmo NNEP da origen a 4 diferentes versiones de acuerdo al tipo de nodo que compone la capa oculta de cada RNA, siendo los nodos a ajustar de tipo SRBF, UP, US y GRBF, donde este último constituye la propuesta realizada en este capítulo.

Las bases de datos seleccionadas incluyen siete problemas de clasificación binaria y 5 problemas de clasificación multiclase. Cada base de datos presenta un número diferente de clases, características e instancias (ver Tabla 7.2). El mínimo y máximo número de nodos ocultos fueron obtenidos a partir del mejor resultado de un diseño experimental previo Anova I, considerando valores pequeños, medios y grandes [230] para los parámetros m, M_I, M_E. En cada base de datos, se realizaron 10 diferentes simulaciones (no abarcado todo el espacio de combinaciones (27)), realizando el posterior análisis de los resultados mediante una prueba de "Tukey-posthoc" [231]. Si la prueba indica que no existen diferencias significativas entre los resultados (CCR en el conjunto de entrenamiento) de las 30 ejecuciones del algoritmo para cada una de las 10 combinaciones, se seleccionan los menores valores para los parámetros m, M_I and M_E. En caso contrario, si se obtienen diferencias significativas, se seleccionan los valores correspondientes a la configuración para la cual se obtuvieron los mejores resultados. La configuración definida para cada base de datos se describe en la Tabla 7.2.

Dataset	Size	R	B	N	♯ In.	♯ Out.	Distribution	$[m, M_I, M$	♯ Gen.
Labor	57	8	3	5	29	2	(30, 27)	[1, 2, 3]	25
Hepatitis	155	6	13	-	19	2	(32, 123)	[1, 2, 3]	25
Sonar	208	60	-	-	60	2	(98, 110)	[1, 2, 3]	300
BreastC	286	4	3	2	15	2	(201, 85)	[1, 2, 3]	50
Ionos	351	33	1	-	34	2	(126, 225)	[3, 4, 5]	300
Card	690	6	4	5	51	2	(307, 308)	[1, 2, 3]	50
German	1000	6	3	11	61	2	(700, 300)	[2, 3, 4]	300
Newthyr	215	5	-	-	5	3	(150, 35, 30)	[1, 1, 4]	100
PostOp	90	1	-	7	20	3	(2, 24, 64)	[1, 2, 3]	100
Glass	214	9	-	-	9	6	(70, 76, 17, 13, 9, 29)	[7, 8, 9]	500
Zoo	101	1	15	-	16	7	(41, 20, 5, 13, 4, 8, 10,4, 8, 10)	[4, 7, 9]	300
Ecoli	336	7	-	-	7	8	(143, 77, 52, 35, 20, 5, 2, 2)	[4, 7, 9]	300

Tabla 7.2: Configuración definida para la experimentación sobre cada "dataset".

El diseño experimental se realizó mediante un procedimiento de crossvalidación utilizando $3/4\dot{n}$ de los patrones de la base de datos para entrenamiento y $n/4$ patrones para el conjunto de generalización. Para evaluar la estabilidad de los métodos, el AE fue ejecutado 30 veces utilizando al CCR como medida de desempeño.

Todos los parámetros del algoritmo propuesto fueron los mismos a excepción de m, M_I , M_E y el número de generaciones (ver Tabla 7.3). Con el objetivo de no tener conjuntos de datos con atributos desproporcionados que afectasen la calidad de la estimación de los modelos de RNA se realizó un escalado de los valores de cada atributo en el intervalo $[-2, 2]$ (X_i^* representa las variables transformadas). Las conexiones entre la capa oculta y la capa de salida se inicializan con valores pertenecientes al intervalo $[-5, 5]$.

El tamaño de la población a evolucionar por el AE fue establecida en N = 200. Para la mutación estructural, el número de nodos posibles a adicionar o eliminar se encuentra en el intervalo $[1, 2]$, y el número de conexiones a adicionar o eliminar en la capa oculta y la capa de salida durante la mutación estructural se encuentra en el intervalo [1, 7].

7.4.2. Resultados de las GRBFNN

Tras estar establecido el diseño de experimentación se obtuvieron los resultados expuestos en la Tabla 7.3 donde se observa la media y la desviación estándar del CCR en los conjuntos de generalización (C_G) para cada base de datos y cada modelo que utiliza nodos de tipo GRBF, SRBF, US y UP. Cada algoritmo es ejecutado 30 veces, sobre cuyas medias C_G se obtuvo el "ranking" de cada algoritmo en cada base de datos (R = 1 para el mejor modelo y R = 4 para el peor). la Tabla 7.3 incluye además la media de las precisiones ($\overline{C}_G$) y la media de los "rankings" o "ranking" medio ($\overline{R}_{C_G}$) sobre los conjuntos de gene-

ralizacón para cada base de datos. De estos resultados podemos inferir desde un punto de vista puramente descriptivo, que los modelos GRBF obtienen los mejores resultados en C_G para 9 bases de datos y los modelos SRBF alcanzaron el mejor desempeño en 2 bases de datos. Por ello los modelos GRBF alcanzan la mejor media ($C_G = 83.26\,\%$) y "ranking" ($\overline{R}_{C_G} = 1.33$) en C_G. Como se puede observar, los modelos GRBF son especialmente precisos cuando se tratan problemas con "alta dimensionalidad" (e.g., ver las bases de datos German, Card, Ionos and Labor), pues la presencia del parámetro δ o su "equivalente" τ permiten a la GG adaptarse a la distribución de los patrones en el espacio.

Para determinar la significación estadística de las diferencias entre los "rankings" observados para cada uno de los métodos sobre las bases de datos propuestas se efectuó una prueba no paramétrica de Friedman [59] sobre los rankings de las precisiones (C_{G}) de cada algoritmo sobre las bases de datos establecidas. Teniendo en cuenta los resultados de una evaluación previa que verificó la ausencia de normalidad y la no igualdad de la hipótesis de varianza. La ejecución de la prueba (con un nivel de significación de un $5\,\%$) arrojó como resultado que las medias de los rankings de los C_{G} poseen diferencias significativas. Siendo el intervalo de confianza $C_0 = (0, F_{0.05} = 2.89)$ y los valores estadísticos de la F-distribución F^0 para C_{G} y $F^* = 6.50 \in C_0$. De este modo se rechaza la hipótesis nula de lo cual se infiere que los algoritmos tienen un comportamiento igual en cuanto a la media de los rankinigs de las precisiones.

Basándonos en en el rechazo obtenido se realiza una prueba de Nemenyi "post-hoc" [60] para comparar cada clasificador contra los demás. Esta prueba considera que la precisión de dos algoritmos son significativamente diferentes si la media de sus rankings difiere al menos en un valor crítico (CD) que se calcula como:

$$CD = q\sqrt{\frac{K(K+1)}{6D}} \tag{7.16}$$

donde K y D son el número de clasificadores y de bases de datos. Siendo q obtenidos a partir del estadístico de rango tipificado dividido por $\sqrt{2}$ [232, 233]. Sin embargo, es de notar que la idea de comparar cada par de clasificadores

uno contra uno con una prueba "post-hoc" no es tan sensible como comparar todos los clasificadores contra un solo clasificador (método de control). Una forma de realizar este análisis es realizando una prueba de comparación de Bonferroni-Dunn. Esta prueba puede computarse utilizando la Fórmula 7.16 con valores de q apropiadamente ajustados [233].

Los resultados de las pruebas de Bonferroni-Dunn y Nemenyi [60] para $\alpha = 0.10$ y $\alpha = 0.05$ se observan la Tabla 7.4, utilizando los valores críticos correspondientes. Del resultado de estas pruebas se puede concluir diciendo que los modelos de RNA con nodos GRBF obtienen un "ranking" de C_G significativamente mejor cuando se compara con los demás modelos analizados en la comparación.

Tabla 7.3: Resultados estadísticos del AE utilizando diferentes tipos de nodos: Media y Desviación Estándar de la precisión en el conjunto de generalización ($C_G(\%)$), precisión media ($C_G(\%)$) y "ranking" medio (R).

	Method ($C_G(\%)$)			
B. datos	**GRBF**	**RBF**	**PU**	**SU**
Labor	**91.19±8.32**	*90.71±6.26*	83.33±10.15	85.71±9.56
Hepatitis	*85.96 ± 3.21*	**86.84 ± 3.16**	85.08 ± 5.04	85.52 ± 5.01
Sonar	*74.10 ± 3.60*	73.68 ± 3.53	**75.25 ± 4.87**	68.71 ± 4.08
BreastC	**68.87 ± 2.10**	68.45 ± 1.87	67.65 ± 2.49	*68.68 ± 2.14*
Ionos	**93.75 ± 1.76**	90.42 ± 2.60	91.15 ± 2.20	*92.61 ± 2.75*
Card	**87.94 ± 1.38**	76.69 ± 3.33	87.50 ± 2.75	*87.71 ± 1.42*
German	**74.33 ± 2.77**	71.69 ± 1.32	71.24 ± 1.24	*73.07 ± 1.64*
Newthyroid	**97.10 ± 1.93**	95.00 ± 2.01	**96.85 ± 2.71**	94.88 ± 2.26
Post-Op	**79.09 ± 8.06**	*78.93 ± 7.21*	78.93 ± 8.39	76.51 ± 7.17
Glass	**68.93 ± 5.19**	64.91 ± 4.74	65.16 ± 4.17	**67.67 ± 3.49**
Zoo	**94.93 ± 2.77**	75.07 ± 5.00	*94.80 ± 4.48*	92.67 ± 4.34
Ecoli	82.98 ± 3.82	**84.44 ± 2.92**	80.30 ± 5.20	*83.73 ±2.45*
$C_G(\%)$	**83.26**	79.73	81.43	*81.45*
R	**1.33**	2.87	3.04	*2.75*

El mejor resultado está resaltado en negritas
y el segundo mejor resultado esta resaltado en cursiva.

Tabla 7.4: Test de Bonferroni-Dunn.

Test de Nemenyi ($C_G(\%)$)				
			Método (j)	
Método (i)	RBF	PU	SU	GRBF
RBF	-	0.16	0.12	$1.54^{+}_{\bullet}$
PU	-	-	0.25	$1.70^{+}_{\bullet}$
SU	-	-	-	$1.41^{+}_{\bullet}$
Test de BonferroniDunn ($C_G(\%)$)				
			Método Comparado	
Método de Control	RBF	PU	SU	GRBF
GRBF	$1.54^{+}_{\bullet}$	$1.70^{+}_{\bullet}$	$1.41^{+}_{\bullet}$	-
Test de Nemenyi: $CD(\alpha = 0.1) = 1.20$, $CD(\alpha = 0.05) = 1.35$;				
T Test de Bonferroni-Dunn: $CD(\alpha = 0.1) = 1.12$, $CD(\alpha = 0.05) = 1.26$;				
$\bullet$, $\circ$: diferencias significativas con $\alpha = 0.05$ ($\bullet$)				
y $\alpha = 0.1$ ($\circ$); $+$: La diferencia es a favor del método de control o método (j).				

Capítulo 8

MODELOS NEURO-LOGÍSTICOS CON NODOS DE TIPO RADIAL GENERALIZADOS

8.1. Introducción

La composición de los modelos logísticos con transformaciones no lineales de las variables independientes permite crear modelos más precisos pero a su vez más difíciles de entrenar con métodos numéricos. No obstante, Martínez Estudillo [137, 234] propone la utilización de un AH que por su carácter estocástico y guiado por poblaciones, permite el ajuste de los modelos, logrando así unificar el poder de la RL y la CE para obtener modelos más robustos.

En el presente capítulo se describe la propuesta de un AH para construir modelos neuro-logísticos [235, 236, 237] mediante un algoritmo compuesto por dos tres etapas: una primera etapa donde se ejecuta un AE seguido de una segunda etapa donde se amplía el espacio de variables mediante un AE y una tercera etapa donde se define un modelo logístico sobre el nuevo conjunto ampliado de variables.

8.2. Modelos Neuro-Logísticos con Nodos de Tipo Radial Generalizados

Los modelos neuro-logísticos con nodos GRBF se definen como la suma de 2 combinaciones lineales (ver fórmula 5.12): una primera combinación definida sobre las variables independientes y otra definida sobre transformaciones del espacio de variables independientes a través de nodos GRBF como se representa en la Figura 5.1 donde cada nodo B_j es un nodo GRBF.

A continuación se muestra el pseudocódigo del algoritmo GRBFNNEP-RegLog el cual constituye la segunda propuesta de la presente tesis. En el algoritmo se han resaltado de color verde los pasos modificados al algoritmo propuesto por Martínez Estudillo [137, 234].

Algoritmo GRBFNNEP-RegLog

1. Generar una población aleatoria de tamaño $10N_P$ y seleccionar las N_P mejores redes para la población inicial.
2. Repetir hasta que se alcance un número máximo de generaciones G o se cumpla la condición de parada.
   - 2.1 El 10 % de los mejores individuos son replicados y sus réplicas
   - 2.2 Aplicar mutación paramétrica al 10 % de mejores individuos de la población.
   - 2.3 Aplicar mutación estructural al restante 90 % de individuos.
   - 2.4 Calcular la aptitud de cada individuo de la población.
   - 2.5 Ordenar los individuos según su aptitud.
   - 2.6 Evitar colinealidad.

     sustituyen al 10 % de los peores individuos (fuerte presión selectiva).
3. Seleccionar el mejor individuo ($\boldsymbol{I}$) en aptitud .
4. Obtener un nuevo espacio de variables ($\boldsymbol{D}$) a partir de las salidas de los nodos de capa oculta de $\boldsymbol{I}$.
5. Añadir de forma opcional las variables iniciales ($\boldsymbol{X}$) a las variables obtenidas en el paso anterior ($\boldsymbol{D} = \boldsymbol{D} + \boldsymbol{X}$).
6. Aplicar Regresión Logística sobre el espacio de variables $\boldsymbol{D}$.

Para una mayor comprensión de los aspectos novedosos del algoritmo y de su funcionamiento en general, se definen a continuación las 3 etapas funda-

mentales que forman la propuesta:

Etapa 1. Se aplica el algoritmo NNEP a una población de GRBFNNs que servirán de base para determinar la estructura en número de funciones de base y los coeficientes de los nodos GRBF, vectores w_i que serán incluidos en el modelo neuro-logístico:

$$\boldsymbol{B}(\boldsymbol{x}, \boldsymbol{W}) = \{B_1(\boldsymbol{x}, \boldsymbol{w}_1), B_2(\boldsymbol{x}, \boldsymbol{w}_2), ..., B_m(\boldsymbol{x}, \boldsymbol{w}_m)\},$$

correspondientes a la parte no lineal del modelo neuro-logístico. Durante esta etapa y siguiendo un esquema de evolución similar al descrito en el capítulo7, se determina la cantidad de nodos (m) que serán incluidos en el modelo neuro-logístico y la matriz de pesos $\boldsymbol{W} = (\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_m)$ asociados a los nodos de capa oculta del más apto de la población resultante del AE. Para guiar la evolución se define la función de máxima verosimilitud (ver sección 5.2.1.1) sobre un conjunto de N observaciones.

La evolución de los coeficientes de los nodos GRBF implica ajustar el nuevo parámetro δ. Durante el proceso de inicialización, el valor de δ puede ser inicializado como $arctg\left(\frac{\tau=2}{e\times r}\right)$ para emular un nodo SRBF o mediante la metodología descrita en la sección 3.3.6. Por otra parte, al realizarse la mutación paramétrica de los nodos GRBF, se añade a δ un número aleatorio el cual sigue una distribución uniforme en el intervalo [-1,1] radianes. Por último cuando se realiza la mutación estructural de los nodos en cuestión, se realizan adiciones de nodos GRBF donde $\delta = arctg\left(\frac{\tau=2}{e\times r}\right)$ y/o uniones de nodos donde el valor del δ resultante se calcula como el promedio de los deltas de los nodos a unir.

Durante el proceso de evolución se ajustan los pesos de los nodos pertenecientes a la capa de salida $\boldsymbol{\beta}$ de los modelos de RNAs con el objetivo de guiar la evolución de los nodos GRBFs y obtener su configuración final: $\hat{\boldsymbol{W}} = (\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, ..., \hat{\boldsymbol{w}}_m)$, no obstante, los valores obtenidos tras la evolución, para el vector $\boldsymbol{\beta}$, no son incluidos en el modelo neuro-logístico final donde solo las transformaciones inducidas por los nodos pertenecientes a la capa oculta (nodos GRBF) son incorporados al modelo final como se describe en la siguiente etapa donde se estiman los valores finales tanto de $\boldsymbol{\beta}$ como de $\boldsymbol{\alpha}$.

Por último se incorpora un procedimiento computacional para evitar la pérdida de precisión que produce la existencia de multicolinealidad (ver sección 5.4) sobre la calidad de la estimación de la RL. Esta característica se manifiesta cuando una variable presenta un valor constante para todas las observaciones; siendo muy común en las covariables introducidas por nodos GRBFs, las cuales debido al proceso de evolución pueden perder cobertura sobre los patrones que agrupa. Este efecto se puede producir por un "alejamiento" del nodo respecto a los patrones, e incrementado por la presencia de una alta dimensionalidad. El procedimiento consiste en ejecutar una simplificación de los modelos de RNAs que pertenecen a la población sujeta a evolución de forma tal que se eliminen todos los nodos de capa oculta (recordemos que todos son GRBF) que presenten la característica de que: la diferencia entre el mayor y el menor valor de activación no supere un umbral establecido U, donde $U = 0.05$ (el valor fue obtenido por cross-validación). Para compensar el efecto de la eliminación del nodo, se le añade al sesgo de cada nodo de la capa de salida, al cual el nodo eliminado estaba conectado, un valor calculado mediante la siguiente expresión:

$$v_0^j = \frac{\text{máx}\left(B_h(\boldsymbol{x}, \boldsymbol{w_h})\right) + \text{mín}\left(B_h(\boldsymbol{x}, \boldsymbol{w_h})\right)}{2} \times \beta_h^j \tag{8.1}$$

donde j es el índice de la neurona de salida a la cual estaba conectado el h-ésimo nodo GRBF de la capa oculta, v_0^j es el incremento a añadir al sesgo del j-ésimo nodo de salida, $\text{máx}\left(B_h(\boldsymbol{x}, \boldsymbol{w_h})\right)$ y $\text{mín}\left(B_h(\boldsymbol{x}, \boldsymbol{w_h})\right)$ son el máximo y el mínimo valor de activación del h-ésimo nodo GRBF y β_h^j es el peso que está unido al nodo seleccionado para ser eliminado, con el j-ésimo nodo de salida. La eliminación de los nodos reduce el cómputo de las salidas de los nodos y por ende reduce el costo computacional del algoritmo.

Etapa 2. Se procede a unificar en un nuevo espacio más amplio, las variables independientes iniciales del problema con las covariables inducidas por los nodos GRBF obtenidas en el paso 1:

$$H : \mathbb{R}^k \to \mathbb{R}^{k+m}, (x_1, x_2, ..., x_k) \to (x_1, x_2, ..., x_k, z_1, ..., z_m),$$

donde $z_1 = B_1(\boldsymbol{x}, \hat{\boldsymbol{w}}_1), ..., z_m = B_m(\boldsymbol{x}, \hat{\boldsymbol{w}}_m)$.

Etapa 3. En la tercera etapa se maximiza la FV sobre las N observaciones minimizando el valor negativo del logaritmo de la FV:

$$L(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{N} \sum_{n=1}^{N} \left[- \sum_{l=1}^{J} y_n^{(l)} (\boldsymbol{\alpha}^l \boldsymbol{x}_n + \boldsymbol{\beta}^l \boldsymbol{z}_n) + \log \sum_{l=1}^{J} \exp(\boldsymbol{\alpha}^l \boldsymbol{x}_n + \boldsymbol{\beta}^l \boldsymbol{z}_n) \right],$$

donde $\boldsymbol{x}_n = (1, x_{1n}, ..., x_{kn})$ y $\boldsymbol{z}_n = (z_{1n}, ..., z_{mn})$. Siendo el Hessiano de la función a minimizar, semidefinido positivamente sobre las nuevas covariables $x_1, x_2, ..., x_k, z_1, ..., z_m$. Los coeficientes del vector $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{W}})$ determinan el modelo definido en (5.12) con $B_j(\boldsymbol{x}, \boldsymbol{w}_j)$ definido como se describe en la fórmula (7.2).

En esta etapa final, se utiliza la RL para obtener los parámetros pertenecientes al vector $\boldsymbol{\theta}$. No obstante debido a que se poseen las opciones de: utilizar dos algoritmos de RL (SLogistic y MLogistic) así como las alternativas de incluir o no las variables iniciales, el AH propuesto da origen a 4 versiones distintas en dependencia de las opciones seleccionadas.

La validación del algoritmo propuesto se detalla en el próximo capítulo 9.1 el cual se dedica a exponer los principales resultados de la tesis.

Capítulo 9

APLICACIONES (RESULTADOS)

Tras detallar las propuestas de algoritmos en los dos capítulos anteriores, se dedica el presente capítulo a describir los resultados obtenidos de la aplicación de cada algoritmo. Primeramente se aborda el problema de clasificación de microarrays (sección 9.1) mediante la descripción de las características fundamentales de los mismos (subsección 9.1.1), de los datos utilizados en la experimentación y los resultados obtenidos (subsección 9.1.2). De forma análoga se procede a describir el problema de clasificación de incapacidad laboral (sección 9.2).

9.1. Clasificación de Microarrays

Un chip de ADN (del inglés DNA microarray) es una superficie sólida a la cual se une una colección de fragmentos de ADN. Las superficies empleadas para fijar el ADN son muy variables y pueden ser de vidrio, plástico e incluso de silicio. Los chips de ADN se usan para analizar la expresión diferencial de genes, monitorizándose los niveles de miles de ellos de forma simultánea. Su funcionamiento consiste, básicamente, en medir el nivel de hibridación entre la sonda específica (probe, en inglés), y la molécula diana (target), indicándose generalmente mediante fluorescencia y estudiándose mediante análisis de imágenes, lo cual indica el nivel de expresión del gen.

Suelen utilizarse para identificar genes con una expresión diferente bajo condiciones distintas. Por ejemplo, para detectar genes que producen ciertas enfermedades mediante la comparación de los niveles de expresión entre células sanas y células que están desarrollando ciertos tipos de enfermedades.

La tecnología del chip de ADN tiene su origen en una técnica muy usada en biología molecular, el "Southern blot". En la era pre-genómica la biología estudiaba los genes individualmente, uno a uno, por lo que los podía estudiar a fondo. Lo que caracteriza la era post-genómica no es lo que se puede medir sino la cantidad de mediciones simultáneas que se pueden realizar. Para cumplir este objetivo y poder estudiar muchos genes a la vez fue necesario un cambio de paradigma: con los mismos recursos, obtener una imagen de menor resolución pero con una perspectiva más general.

Los chips de ADN se fabrican usando una gran variedad de tecnologías. El gran desarrollo de esta técnica ha llegado al normalizarse el uso de robots que se encargan de la mayor parte del proceso de manipulación de los chips (sintetizar el ADN molde que se une al chip, unirlo, añadir los reactivos necesarios, etc.).

Los chips de ADN se pueden utilizar para detectar ARN, que puede o no ser traducido a proteínas. Los científicos se refieren a esta clase de análisis como "análisis de expresión" o "análisis del transcriptoma". Se suele retrotranscribir el ARN a ADN para estos casos, de manera que lo que usamos como muestra es ADNc.

El uso de chips de ADN para estudiar la expresión de diversos genes fue publicado en 1995, en la prestigiosa revista científica Science y el primer organismo eucariota con todo el genoma (Saccharomyces cerevisiae) dispuesto en un chip de ADN fue publicado en 1997 en la misma revista.

Los chips de ADN se han aplicado al estudio de casi cualquier tipo de problema biológico. El número de publicaciones anuales es muy alto y continúa creciendo. Algunas de sus aplicaciones más frecuentes son:

- Estudio de genes que se expresan diferencialmente ante varias condiciones (sanos/enfermos, mutantes/salvajes, tratados/no tratados).

- Clasificación molecular en enfermedades complejas. Identificación de genes característicos de una patología (firma o "signature").
- Predicción de respuesta a un tratamiento.
- Detección de mutaciones y polimorfismos de un único gen (SNP).

Dada la importancia de los microarrays, en la presente investigación se abordó su estudio desde el problema de la Clasificación de Patrones como se describe en las siguientes subsecciones.

9.1.1. Descripción de las Bases de Datos

Con el propósito de validar el AH propuesto en el capítulo 8 se realizaron una serie de experimentos sobre 6 bases de datos públicas que contienen datos provenientes de microarrays. La Tabla 9.1 muestra las características de las 6 bases de datos utilizadas en la experimentación donde: (FS) describe el tipo de selector de características utilizado, previa aplicación de los algoritmos de clasificación para reducir la dimensionalidad de los datos, (Tam.) es el número de instancias de la base de datos, (R), (B) y (N) son el número de características de tipo real, binaria y nominales respectivamente, (Ent.) es el número total de variables de entrada, (Sal.) es el número de clases, (NPC) número de patrones por clase, ($[M_{\min}, M_{\max}]$) mínimo y máximo número de nodos utilizados en la capa oculta y (#Gen.) el número de generaciones. Estas bases de datos fueron tomadas del campo de estudio de la Bio-Informática siendo comúnmente utilizados para validar la precisión de los clasificadores y de los selectores de características. Debido a la alta dimensionalidad y pequeño número de características presentes en los datos pertenecientes a los microarrays se hace necesaria la selección de las características sobre las cuales se procederá a aplicar el algoritmo de clasificación. Una breve descripción de los datos utilizados se presenta a continuación:

Breast: consta de 97 muestras tomadas de pacientes con cáncer de mama. 46 de ellos son de pacientes etiquetados como enfermos, el resto de la 51

las muestras son de pacientes que mantienen la salud de la enfermedad y que son considerados como no enfermos. Cada muestra se describe por 24481 genes.

CNS (Central Nervous System): compuesto por 60 patrones y 7129 características. La base de datos se obtuvo de pacientes que padecen de tumores embrionarios en el sistema nervioso central. Los patrones son etiquetados en dos clases: sobrevivientes y no sobrevivientes. Las cantidades seleccionadas por cada clase fueron 21 patrones pertenecientes a pacientes que sobrevivieron y 39 patrones pertenecientes a pacientes que fallecieron por la enfermedad.

Colon: utiliza microarrays de oligonucleótidos (creados por Affymetrix) para predecir los niveles de expresión de más de 6500 genes humanos. Las observaciones están compuestas por 40 tejidos con tumor de colon y 22 muestras de tejido normal. Los 2000 genes con la mayor intensidad mínima de los 62 tejidos, fueron utilizados en este análisis.

Leukemia: se refiere a desórdenes primarios en la médula espinal. Esta base de datos contiene 72 patrones de personas que poseen neoplasmas malignos en las células del tejido hematopoiético, de ellos 47 con *leucemia limfobástica aguda* (ALL) y 25 *leucemia mieloide aguda* . El número total de genes analizados fue 7129.

Lung: tiene 12600 genes en 203 muestras. Las 203 muestras consisten en 139 adenocarcinomas de pulmón, 21 casos de carcinoma de células escamosas, 20 tumores carcinoides pulmonares y 6 casos de cáncer de pulmón en células pequeñas, así como 17 muestras pulmonares normales.

GCM: contiene 190 muestras correspondientes a 14 variedades de tumor de las cuales se muestrearon los niveles de expresión de 16063 genes.

En las bases de datos descritas, como en cualquier base de datos típica de microarray, se presenta la particularidad de que poseen miles de características. Para evitar los problemas que implica la alta dimensionalidad de los datos se utilizaron dos algoritmos para seleccionar un número más reducido de características sobre las cuales aplicar nuestra propuesta de algoritmo. Los algoritmos utilizados fueron: Fast Correlation-Based Filter [238] (FCBF) y el Best Agglomerative Ranked Subset [239] (BARS).

En las 6 bases de datos de microarrays, todos los valores de las características referentes a la expresión de los genes son numéricas. Además presentan la particularidad de que toman valores en intervalos de diversa amplitud por lo que es conveniente realizar un reescalado lineal de los valores en el intervalo $[-2, 2]$, siendo X_i^* las variables transformadas tras la selección de características.

Tabla 9.1: Características de las 6 bases de datos utilizadas en la experimentación.

Datos	Fuente	FS	Tam.	R	B	N	Ent.	Sal.	NPC	$[M_{\min}, M_{\max}]$	*Gen*
Breast	**[240]**	BARS	**97**	183	-	-	183	**2**	$(46, 51)$	$[1, 3]$	**100**
		FCBF		493	-	-	493				
CNS	**[241]**	BARS	**60**	187	-	-	187	**2**	$(21, 39)$	$[1, 3]$	**10**
		FCBF		170	-	-	170				
Colon	**[242]**	BARS	**62**	58	-	-	58	**2**	$(40, 22)$	$[1, 3]$	**10**
		FCBF		59	-	-	59				
Leukemia	**[243]**	BARS	**72**	225	-	-	225	**2**	$(42, 25)$	$[1, 3]$	**50**
		FCBF		203	-	-	203				
Lung	**[244]**	BARS	**203**	237	-	-	237	**5**	$(139, 17, 6, 21, 20)$	$[5, 8]$	**100**
		FCBF		250	-	-	250				
Gcm	**[245]**	BARS	**253**	311	-	-	311	**14**	$(11, 10, 11, 11,)$	$[25, 28]$	**400**
		FCBF		264	-	-	264		$(22, 11, 11, 11, 20)$		

9.1.2. Marco de Experimentación

Con el objetivo de validar el algoritmo propuesto se establece un marco de experimentación compuesto por 5 algoritmos pertenecientes al estado del arte dentro del campo de "Machine Learning". Los algoritmos utilizados para comparar la precisión de nuestra propuesta son:

1. *Red RBF* [246] compuesta por una capa de nodos SRBF definida en función de las variables independientes, y entrenada en dos etapas utilizando en la primera etapa el algoritmo $K-Medias$ [247]. Posteriormente se aplica un algoritmo de Regresión Logística sobre las nuevas variables generadas por las SRBF.

2. Regresión Logística Multinomial (MLogistic), algoritmo que establece un modelo de RL incorporando una penalización sobre los coeficientes a estimar, con el objetivo de prevenir el sobreajuste a los datos de entrenamiento [248]. Los coeficientes del modelo son estimados mediante el algoritmo IRLS (ver sección 5.2.2).

3. Regresión Logística Simple (SLogistic): aplica un algoritmo LogitBoost para estimar los coeficientes del modelo. Este algoritmo iterativo determina el número de iteraciones mediante un "5 fold cross-validation" sobre el conjunto de entrenamiento [249].

4. Modelo Logístico de Árboles de Decisión (Logistic Model Tree, LMT) [249].

5. Máquina de Soporte Vectorial (SVM) [250] con kernels SRBF.

Estos algoritmos fueron seleccionados pues se encuentran estrechamente relacionados con nuestra propuesta. Los tres primeros métodos construyen modelos logísticos, el cuarto utiliza el algoritmo de RL (LMT) y el último define una combinación lineal sobre las salidas de nodos SRBF.

Variada ha sido la utilización de estas técnicas para abordar el problema de expresión de genes en datos provenientes de muestras de Microarrays. Una

descripción más detallada de los resultados obtenidos con las técnicas antes mencionadas puede ser consultada en [251, 246, 249].

Para la evaluación de los distintos modelos se ha tomado la precisión como medida de desempeño, siendo conocida también como Porcentaje de Clasificación Correcta o (CCR). Para la evaluación de nuestra propuesta se fijan los intervalos para la inicialización de los pesos de los nodos de la capa de salida de las redes en $[-I, I] = [-5, 5]$ y se crean poblaciones con $N = 500$ individuos. Para realizar la mutación estructural se establece el intervalo $[1, 2]$ describiendo el número mínimo y máximo de nodos que se pueden eliminar o añadir; y se enmarca el número de conexiones a eliminar o establecer dentro del intervalo $[1, 7]$.

Para la selección de los hiperparámetros de las SVM (parámetro de regularización, C, y el radio de las SRBF γ), se ha aplicado una búsqueda de tipo malla mediante un "10-fold cross-validation" sobre el conjunto de entrenamiento utilizando los siguientes rangos:

$$C \in \{2^{-5}, 2^{-3}, \ldots, 2^{15}\} \text{ y } \gamma \in \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}.$$

El diseño experimental fue realizado mediante un procedimiento "holdout cross validation" con $3n/4$ de la totalidad de los patrones para realizar el entrenamiento y el restante $n/4$ como conjunto de datos de generalización. Con el fin de evaluar la estabilidad del experimento, el AE se ejecuta 30 veces permitiendo su validación mediante métodos estadísticos.

Por último, el AE y los algoritmos pertenecientes al marco de experimentación fueron implementados en JAVA. Además se utilizó la librería 'libsvm" [252] para obtener los resultados pertenecientes a las SVM y WEKA[1] para obtener los resultados de los algoritmos contra los que se comparó nuestra propuesta.

En la presente sección se analizan los resultados obtenidos comparando el CCR(la media de la precisión de las 30 ejecuciones sobre cada base de datos) de nuestro algoritmo neuro-logístico y de los restantes algoritmos pertenecientes al marco de experimentación.

[1] http://www.cs.waikato.ac.nz/ml/weka/

En la Tabla 9.2 se representa la media y la desviación estándar de la precisión (C_G) de cada algoritmo sobre el conjunto de generalización en las 30 ejecuciones efectuadas. Del análisis de los resultados se puede concluir desde un punto de vista puramente descriptivo que nuestra propuesta presenta una precisión mejor en media ($\overline{C}_{\mathrm{G}} = 91.08\,\%$) y en rango ($\overline{R}_{C_{\mathrm{G}}} = 1.62$) sobre C_{G} que los demás algoritmos del marco de experimentación.

Tabla 9.2: Comparación del método propuesto respecto a los demás algoritmos que componen el marco de experimentación: $C_{\mathrm{G}}(\%)$ media y desviación estándar (SD) de la precisión perteneciente a las 30 ejecuciones efectuadas sobre cada base de datos en particular, $\overline{C}_{\mathrm{G}}(\%)$ media de las precisiones obtenidas por un clasificador ante las bases de datos propuestas, $\overline{R}$ rango medio, p-Value y $\alpha^{'}$ pertenecientes al test de Holm [1](no paramétrico) realizado sobre C_G con $\alpha = 0.05$ y utilizando el algoritmo propuesto como algoritmo de control.

		Method($C_{\mathrm{G}}(\%)$)							
		RBFN	MLogistic	SLogistic	LMT	SVM	EGRBF	MLEGRBF	MLIEGRBF
Datos	FS	Result	Result	Result	Result	Result	$\mathrm{Mean}_{\mathrm{SD}}$	$\mathrm{Mean}_{\mathrm{SD}}$	$\mathrm{Mean}_{\mathrm{SD}}$
Breast	(1)	*80.00*	*80.00*	72.00	72.00	*80.00*	$73.60_{4.88}$	$77.87_{5.53}$	$\mathbf{80.93_{3.59}}$
	(2)	80.00	*84.00*	*84.00*	*84.00*	76.00	$74.80_{7.50}$	$74.93_{7.71}$	$\mathbf{87.73_{1.01}}$
CNS	(1)	**80.00**	73.33	66.66	66.66	66.67	$70.22_{9.22}$	$70.00_{10.47}$	$\mathit{77.56_{3.27}}$
	(2)	86.66	**100.00**	80.00	80.00	66.67	$84.89_{8.01}$	$84.00_{7.95}$	$\mathit{99.78_{1.22}}$
Colon	(1)	**93.75**	93.75	**100.00**	**100.00**	62.50	$99.17_{2.16}$	$99.17_{2.16}$	$\mathbf{100.00_{0.00}}$
	(2)	**87.50**	75.00	81.25	75.00	62.50	$77.92_{8.16}$	$78.33_{7.83}$	$\mathbf{87.08_{1.59}}$
Leukemia	(1)	*94.44*	**100.00**	88.89	88.89	66.67	$97.22_{3.79}$	$97.59_{3.77}$	$\mathbf{100.00_{0.00}}$
	(2)	*94.44*	*94.44*	83.33	83.33	66.67	$95.93_{4.36}$	$98.15_{3.04}$	$\mathbf{100.00_{0.00}}$
Lung	(1)	96.07	96.07	*98.03*	**98.03**	**98.03**	$93.53_{1.94}$	$97.59_{3.77}$	$\mathit{97.91_{0.50}}$
	(2)	94.11	94.11	**98.03**	**98.03**	94.11	$87.58_{5.00}$	$98.14_{3.04}$	$\mathbf{100.00_{0.00}}$
Gcm	(1)	75.00	73.07	63.49	71.15	75.00	$24.04_{6.50}$	$54.81_{4.84}$	$\mathbf{77.56_{0.93}}$
	(2)	82.00	80.76	71.15	67.30	80.76	$26.24_{6.68}$	$54.09_{7.39}$	$\mathbf{84.35_{2.71}}$
$\overline{C}_{\mathrm{G}}(\%)$		86.99	*87.04*	82.23	82.03	74.63	75.42	82.05	**91.08**
$\overline{R}_{C_G}$		*3.83*	4.08	4.91	5.20	5.75	5.87	4.71	**1.62**
p-Value		0.027	0.013	0.001	3.4E-4	4.0E-5	2.5E-5	0.002	-
$\alpha^{'}_{Holm}$		0.050	0.025	0.012	0.010	0.008	0.007	0.016	-

El mejor resultado se encuentra resaltado en negrita y el segundo mejor en itálica

En los gráficos de la Figura 9.1, cada punto compara nuestra propuesta respecto a los demás algoritmos presentes en el marco de experimentación, sobre cada base de datos específica representada cada una por una columna distinta dentro del gráfico. Cada punto representado muestra en un par de valores ordenados (x,y) donde el valor de las x representa la precisión de nuestra propuesta y el valor de y representa la precisión del algoritmo representado según la simbología en la leyenda. Por tanto si un punto se encuentra por debajo de la línea divisoria roja significa que el algoritmo representado presenta una precisión menor que nuestra propuesta para la base de datos representada por la columna a la cual pertenece dicho punto.

Para determinar la significación estadística de las diferencias entre los "rankings" observados para cada uno de los métodos sobre las bases de datos propuestas se efectuó una prueba no paramétrica de Friedman [59] sobre los rangos de las precisiones (C_{G}) de cada algoritmo en cada una de las bases de datos establecidas, teniendo en cuenta los resultados de una evaluación previa que verificó la ausencia de normalidad y la no igualdad de la hipótesis de varianza de las poblaciones asociadas a los resultados. La ejecución de la prueba (con un nivel de significación de un 10 %) arrojó como resultado que las medias de los rankings de los C_{G} poseen diferencias significativas. Siendo el intervalo de confianza $C_0 = (0, F_{0.05} = 2.13)$ y el valor estadístico de la F-distribución es $F^* = 4.96 \notin C_0$ para C_{G}. De este modo se rechaza la hipótesis nula y se infiere que los algoritmos tienen un comportamiento desigual en cuanto a la media de los rangos de las precisiones.

Tras los resultados obtenidos con la prueba de Friedman [59] se realizó una prueba de Holm "post-hoc" [1] para comparar el rendimiento de cada clasificador repecto a los demás. Esta prueba es un procedimiento de comparación múltiple que funciona estableciendo un algoritmo de control y comparándolo con los restantes métodos [253]. Un análisis pormenorizado de los p-valores (calculados por el test) ordenados ascendente, indica que los "p-valores" siempre son menores que su α'_{Holm} correspondiente: $p_{EGRBF} \leq p_{SVM} \leq p_{LMT} \leq p_{SLogistic} \leq p_{MLEGRBF} \leq p_{MLogistic} \leq p_{RBFN} \leq p_{MLIEGRBF}$ siendo $p_{EGRBF} = 2.5E-5 \leq 0.007, p_{SVM} = 4.0E-5 \leq 0.008, p_{LMT} = 3.4E-4 \leq$

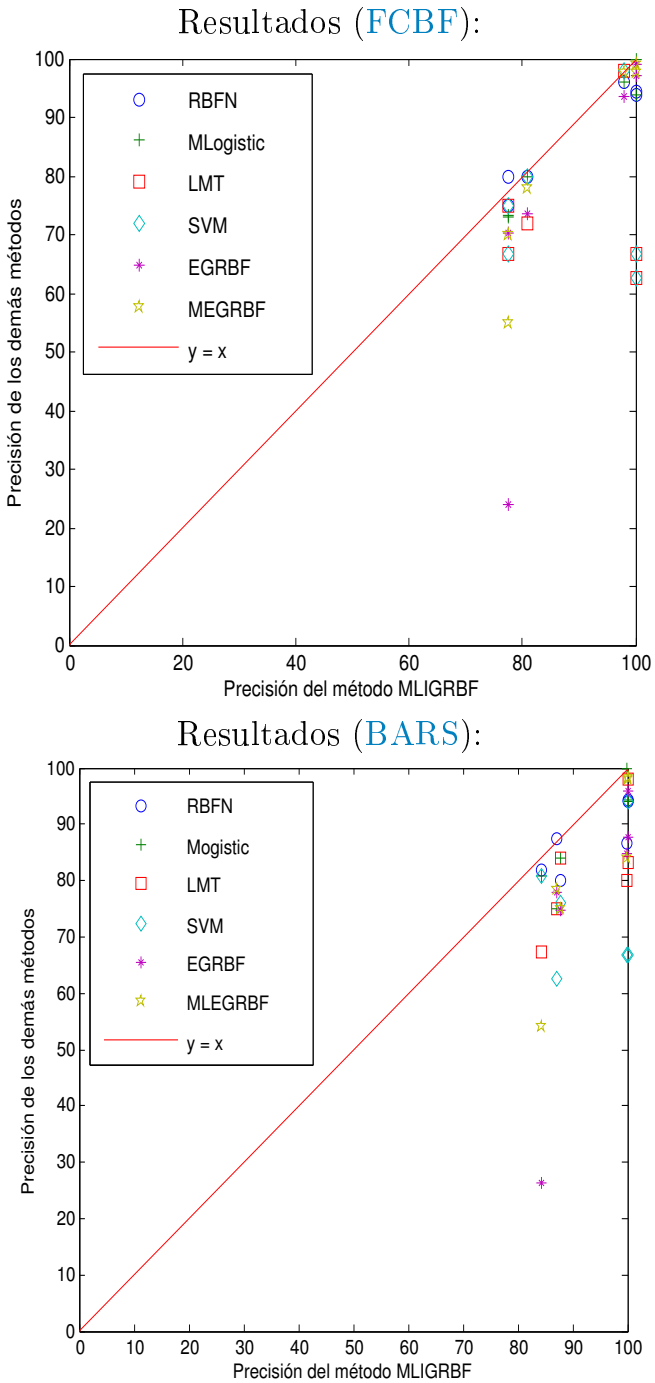


Figura 9.1: Comparación de la precisión de la metodología propuesta respecto a los demás algoritmos pertenecientes al marco de experimentación sobre las 12 bases de datos tras aplicar los algoritmos FCBF y BARS respectivamente.

$0.010, p_{SLogistic} = 0.001 \leq 0.012, p_{MLEGRBF} = 0.002 \leq 0.016, p_{MLogistic} = 0.013 \leq 0.025, p_{RBFN} = 0.027 \leq 0.050$ (ver el final de la Tabla 9.2).

Los resultados de la prueba de Holm para un $\alpha = 0.05$ pueden ser observados en la Tabla 9.2, utilizando los $p - valores$ y α'_{Holm}s correspondientes. Del análisis de los resultados de las pruebas se puede concluir que la metodología propuesta obtiene un "ranking" de C_{G} significativamente más alto que los restantes algoritmos presentes en el marco de experimentación.

9.2. Clasificación de Incapacidad Laboral

La incapacidad laboral es una condición que presentan algunas personas en edad laboral debido a afecciones en su estado de salud. Esta situación médica incide directamente en su situación ocupacional. El análisis de esta condición debe poseer un alto rigor profesional; evitando tener en cuenta otras circunstancias como familia, edad, etc.

9.2.1. Descripción de los Datos

Las situaciones ocupacionales a tener en cuenta para que una persona sea protegida por el estado de incapacidad laboral se pueden resumir como:

- Incapacidad laboral la cual se entiende como la pérdida en el monto de los ingresos de las personas como resultado de una incapacidad laboral permanente o temporal.
- Necesidad de recobrar el bienestar psíquico-físico.
- Necesidad de recibir ayuda financiera durante el periodo de restablecimiento.
- Proceso de reintegrar una persona incapacitada al ambiente laboral, considerando que dicha persona debe ser protegida mediante la selección del empleo a desempeñar.

Dependiendo de la causa que determina las incapacidades laborales se pueden clasificar en los siguientes grados o niveles de afección:

- *Parcial*: cuando una persona se ve disminuida en un 33 % de su capacidad laboral. Sin embargo la afección no lo incapacita para desarrollar la labor fundamental de su ocupación.
- *Total*: cuando un trabajador no puede desarrollar la labor fundamental de su ocupación pero puede realizar otras labores.
- *Absoluta*: cuando el trabajador no es capaz de realizar ninguna profesión.
- *Aguda*: cuando un trabajador posee dificultades que afectan a su capacidad para realizar las funciones básicas de la vida diaria (comer, trasladarse, vestirse, etc).
- *Ninguna*: cuando la persona no posee ninguna afección que impacte sobre su capacidad laboral pero su integridad física se ve disminuida.

La relación de afecciones no incapacitantes se relacionan en la "Ley General de la Seguridad Social" [254].

En caso de accidente ya sea de trabajo o no, la unidad médica que asiste al accidentado en cuanto a su incapacidad emite una síntesis médica en forma de informe para evaluar su incapacidad laboral. Estos informes son utilizados como fuente de información para nuestra experimentación. Las síntesis médicas comprenden diferentes aspectos como:

1. Examen clínico emitido por el médico evaluador.
2. Informes médicos proporcionados por el enfermo.
3. Exámenes complementarios orientados por el médico evaluador.

Los datos utilizados en esta investigación se han obtenido de informes médicos y sesiones de trabajo del equipo encargado de emitir los informes de incapacidad laboral, los cuales fueron recopilados y archivados en ficheros

digitales para su análisis. Algunos datos como edad o sexo han sido extraídos directamente de los documentos mientras otros como la repercusión laboral, han sido recopilados por personal calificado.

Por cada fichero se tuvieron en consideración los siguientes atributos:

- A partir de la síntesis médica: edad, sexo, ocupación, período de recuperación.
- A partir de las sesiones de trabajo del equipo de asesoramiento: Clase (grado de incapacidad), contingencia, período entre exámenes.
- *Repercusión ocupacional*: evaluados en 3 niveles de intensidad (bajo, medio, alto), teniendo en cuenta la repercusión funcional de diferentes enfermedades y la ocupación del trabajador.
- La clasificación de la incapacidad laboral se agrupa en: Ninguna (ND), Incapacitado o Pago.
- Las contingencias pueden ser clasificadas en dos tipos: Común (Enfermedad Común (CD), Accidente Extralaboral (NWA))y Profesional (Enfermedad Ocupacional (OD) o Accidente Laboral (WA).

Para la nomenclatura se ha utilizado el Código Español de Clasificación Ocupacional (CNO-94) [255] para recolectar los datos relacionados con las diversas profesiones. Además, para vincular las enfermedades relacionadas hemos utilizado el documento "International Classification of Diseases" (ICD9-CM). Las variables finales utilizadas en nuestra propuesta de clasificación se muestran en la Tabla 9.3. Gracias a todo el proceso de recópilación de datos se ha podido tener en cuenta un total de 978 registros recolectados entre 2002 y 2003.

9.2.2. Marco de Experimentación

Para el contraste de los resultados obtenidos por nuestra propuesta se aplicaron un conjunto de algoritmos de probada eficacia y que pertenecen al estado del arte en los problemas de clasificación:

Tabla 9.3: Variables definidas sobre el problema de incapacidad laboral.

Variables	
x_1	Edad
x_2	Sexo
x_{3-21}	CNO-94
x_{22}	Tiempo de recuperación
x_{23-42}	Principales Categorías de ICD9-CM
x_{43}	Repercusión ocupacional baja
x_{44}	Repercusión ocupacional media
x_{45}	Repercusión ocupacional alta
x_{46}	Número total de enfermedades
x_{47}	CD contingencia
x_{48}	NWA contingencia
x_{49}	OD contingencia
x_{50}	WA contingencia
x_{51}	Período de tiempo entre exámenes
Clases	
ND	Sin Incapacidad
PD	Incapacitado
F	pago

- k Vecinos Más Cercanos (k-NN) ajustando el valor de k mediante un "10 fold cross-validation" sobre el conjunto de entrenamiento.
- Red RBF (RBFNetwork) accesible en WEKA [246].
- Dos algoritmos de Regresión Logística accesibles también en WEKA: SimpleLogistic (SLogistic) y MultiLogistic (MLogistic).
- Naive Bayes en su versión estándar [246].

Para la experimentación se realizó un "10-fold cross-validation" midiendo la precisión sobre el conjunto de generalización. Para proporcionar robustez estadística al proceso de experimentación se realizaron 10 ejecuciones de los algoritmos estocásticos (métodos propuestos con nodos GRBF y RBF [256]) por cada uno de los 10 "folds" creados. De esta forma se obtienen las medias y desviaciones estándar de las precisiones de los algoritmos en generalización

pertenecientes a cada uno de los 100 modelos. Además se ha realizado un reescalado de los datos de entrada al intervalo $[-2; 2]$ siendo X_i^* el conjunto de variables transformadas para las metodologías que utilizan los nodos RBF y GRBF [228, 227].

9.2.3. Análisis de los Resultados

Con el objetivo de comprobar la significación estadística de las diferencias observadas entre las precisiones de los algoritmos analizados se ha realizado un test de Kolmogorov-Smirnov tras el cual se asume la no normalidad de los resultados obtenidos por los algoritmos a comparar. Seguidamente se realiza una prueba de "Mann-Whitney U rank sum" para cada par de algoritmos. Además se aplicó un "test" de Kruskal-Wallis concluyendo que existen diferencias significativas entre los algoritmos presentes en el marco de experimentación.

De la aplicación de la prueba "Mann-Whitney U rank sum" (ver Tabla 9.4) se concluye que la metodología más competitiva es la SLIGRBF (con una sola comparación adversa) seguida por la SLIRBF. Por consiguiente las unidades GRBFs son más aconsejables para ser aplicadas sobre el problema de Discapacidad Laboral. Una de las características que influyen en la selección de los nodos GRBFs es la simplicidad de los modelos creados donde solo intervienen 10 variables iniciales o independientes.

Tabla 9.4: Media, desviación típica, mínimo y máximo valor de precisión C_G de las 100 ejecuciones del "10-fold cross validation". Número de victorias, empates y derrotas cuando son comparados respecto a los diferentes métodos utilizando el "Mann Whitney U rank sum test" con $\alpha = 0.05$.

	$C_G(\%)$	"Mann Whitney U rank sum test"		
	$Mean \pm SD$	victorias	empates	derrotas
EGRBF	85.26 ± 5.08	5	4	5
MLGRBF	85.76 ± 5.42	5	5	4
SLGRBF	85.30 ± 4.90	5	5	4
MLIGRBF	89.03 ± 3.34	11	1	2
SLIGRBF	**90.70 ± 3.02**	13	1	0
ERBF	79.76 ± 11.36	1	2	11
MLRBF	79.88 ± 11.20	1	2	11
SLRBF	79.56 ± 13.54	1	2	11
MLIRBF	86.39 ± 8.96	5	5	4
SLIRBF	89.86 ± 9.40	12	2	0
k-NN	66.04 ± 8.12	0	0	14
RBFNetwork	86.75 ± 9.30	6	4	4
SLogistic	89.77 ± 9.39	11	2	1
MLogistic	86.54 ± 9.31	5	5	4
Naive Bayes	84.17 ± 9.15	4	0	10

9.2.4. Conclusiones

- Con este enfoque práctico hemos comparado las GRBFNNs Evolutivas y las RBFNNs Evolutivas cuando la dimensión del espacio de entrada es alta, condición sobre la cual, las GRBFs eliminan un conjunto de problemas que influyen negativamente sobre los nodos RBF.
- Se demostró la buena sinergia entre las RNAs Evolutivas y la Regresión Logística.
- Los modelos neuro-logísticos híbridos han demostrado ser una herramienta precisa en la clasificación de discapacidades laborales.
- No se pretende por nuestra parte que el modelo obtenido sea una herramienta ampliamente utilizada para la clasificación de la incapacidad

laboral. Primeramente sería necesario examinar más datos referentes al tema en cuestión. Sin embargo nuestra investigación puede ser utilizada para desarrollar mejores sistemas que brinden asesoramiento sobre la incapacidad laboral.

Capítulo 10

CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo exponemos las conclusiones obtenidas en este trabajo de tesis, las publicaciones asociadas a la misma, y las líneas futuras en las que vamos a trabajar a corto o medio plazo.

10.1. Conclusiones

Como resultado de la presente tesis se llegó a los siguientes resultados:

- Hemos realizado un estudio de las técnicas de entrenamiento de modelos de RNAs con algoritmos clásicos sección 2.4.1 y bioinspirados (capítulo 4). Durante el estudio se analizaron los antecedentes referentes a algoritmos basados en gradiente, haciendo hincapié en el algoritmo $iRProp+$; el cual fue utilizado mediante hibridación con algoritmos evolutivos. Además el estudio abarcó las distintas métricas para la comparación de algoritmos (ver sección 2.3.1) y una descripción de los tests estadísticos para la comparación de los resultados de las métricas utilizadas (ver sección 2.3.2).

- Hemos propuesto una reformulación de los nodos GRBF para facilitar su entrenamiento con algoritmos evolutivos y algoritmos basados en gradiente. En el capítulo 7 se exponen las principales dificultades que presenta

la formulación inicial de las GRBF y cómo se erradican con la nueva reformulación.

- Hemos creado un Algoritmo Híbrido para el entrenamiento de Redes Neuronales de Base Radial Generalizada. Este algoritmo entrena modelos de RNAs con una topología feedforward como muestra la Figura 2.1 donde los nodos pertenecientes a la capa oculta son de tipo GRBF reformulados (ver capítulo 7). El algoritmo hibrida un algoritmo evolutivo base (con nodos GRBF) con características exploratorias con el algoritmo $iRProp+$ como algoritmo de Búsqueda Local para proveer convergencia al AE.

- Hemos creado un Algoritmo Evolutivo para el entrenamiento de modelos Neuro-Logísticos de Base Radial Generalizada. Este algoritmo crea un modelo neuro-logístico sobre las variables independientes iniciales y las covariables resultantes del individuo más apto del algoritmo evolutivo base; el cual fue corregido (ver capítulo 8) para evitar los efectos indeseables introducidos por la multicolinealidad (ver sección 5.4) de las nuevas covariables.

- Hemos validado los algoritmos elaborados con bases de datos internacionales de la UCI Machine Learning Repository (ver sección 7.4) y sobre problemas reales de clasificación, como la clasificación de incapacidad laboral (ver sección 9.2) y en la clasificación de microarrays (ver sección 9.1) .

10.2. Publicaciones asociadas a la tesis

A continuación se exponen las publicaciones asociadas a esta tesis doctoral.

10.2.1. Artículos en revistas

- Adiel Castaño, Francisco Fernández-Navarro, Pedro Antonio Gutiérrez, and César Hervás-Martínez. Permanent disability classifcation by combi-

ning evolutionary generalized radial basis function and logistic regression methods. Expert Syst. Appl., 39(9):8350–8355, 2012.

- Adiel Castaño,Francisco Fernández-Navarro, César Hervás-Martínez, Pedro Antonio Gutiérrez Peña. Neuro-logistic models based on Evolutionary Generalized Radial Basis Function for the microarray gene expression classification problem. Neural Processing Letters, En Prensa, 2011. `http://dx.doi.org/10.1007/s11063-011-9187-8`.

- Adiel Castaño, Francisco Fernández-Navarro, César Hervás-Martínez, M.M. García, Pedro Antonio Gutiérrez Peña. Classification by evolutionary generalised radial basis functions, International Journal of Hybrid Intelligent Systems, 7, (2010) 1-10.

10.2.2. Artículos en congresos internacionales

- Adiel Castaño Méndez, Francisco Fernández-Navarro, Pedro Gutiérrez, Manuel Baena-García, and César Hervás-Martínez. Combining evolutionary generalized radial basis function and logistic regression methods for classification.In Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011, Advances in Intelligent and Soft Computing.

- Adiel Castaño, César Hervás-Martínez, Pedro Antonio Gutiérrez, Francisco Fernández Navarro, and M. M. García. Classification by evolutionary generalized radial basis functions. In ISDA, pages 203-208. IEEE Computer Society, 2009.

10.3. Trabajo futuro

- En cuanto al algoritmo evolutivo base hay varias direcciones de trabajo que se podrían llevar a cabo en el futuro:

  1. Combinar el esquema general del algoritmo con el mecanismo bio-inspirado de los algoritmos asistidos por colonia de bacterias, incorporando el mecanismo de exploración-explotación.
  2. Utilizar mecanismos de selección de características para determinar el subconjunto de variables independientes más prometedoras para definir los nodos pertenecientes a los RNAs durante el proceso de inicialización.
  3. Sustituir el mecanismo de adición de nodos aleatorios a las RNAs por la adición de nodos entrenados por otras RNAs. La edición selectiva de la estructura de las RNAs con mecanismos simples permitiría una exploración más eficaz del espacio de variables independientes.

- Utilizar nodos QRBF en modelos neuro-logísticos para mapear la interacción entre las variables independientes.

Gracias por la atención prestada a la tesis.

BIBLIOGRAFÍA

[1] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, páginas 65–70, 1979.

[2] X. Yao. Evolving artificial neural networks. En *Procs. of the IEEE*, volumen 87, páginas 1423–1447, 1999.

[3] P.J. Angeline, G.M. Saunders, y J.P. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, 1994.

[4] D.J. Newman, S. Hettich, C.L. Blake, y C.J. Merz. UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998. University of California, Irvine, Dept. of Information and Computer Sciences.

[5] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[6] Simon Haykin. *Neural networks: A comprehensive foundation*. MacMillan, New York, 1994.

[7] D. S. Broomhead y D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

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

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

[10] K. Hornik, M. Stinchcombe, y H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, 3(5):551–560, 1990.

[11] Smieja y Muhlenbein. The geometry of multi-layer perceptron solutions. *PARCOMP: Parallel Computing*, 14, 1990.

[12] Michael T. Manry, Jack Fitzer, David P. Klemer, y Multi layer Perceptron. Efficient nonlinear mapping using the multi-layer perceptron, Agosto 17 2000.

[13] S. Mitra, S. K. Pal, y M. K. Pal. Finger print classification using multilayer perceptron. *Neural Computing and Applications*, 2, 1994.

[14] R. Galan, M. Marino, y A. Jiménez. Enhancing multi-layer perceptron. En *IEEE International Conference on Neural Networks (ICNN'94)*, volumen IV, páginas 2661–2666, Orlando, FL, Junio 1994. IEEE. Departamento De Automática, Ingeniería Electrónica E Informática Industrial (DISAM).

[15] Nigel Dodd. Multi-layer perceptron inversion applied to image neighbourhood operations. En Maureen Caudill y Charles Butler, editors, *IEEE First International Conference on Neural Networks (ICNN'87)*, volumen IV, páginas IV–293–IV–300, San Diego, CA, Junio 1987. IEEE.

[16] Ramona-Anne Rosario y Nazif Tepedelenlioglu. A rapid multi-layer perceptron training algorithm. En *IEEE International Joint Conference on Neural Networks (6th IJCNN'92)*, volumen I, páginas 824–829, Baltimore, MD, Junio 1992. IEEE/INNS. FIT.

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

[18] A.C. Martínez-Estudillo. *Modelos de regresión basados en redes neuronales dise nadas y entrenadas mediante algoritmos de optimizaci0́n híbrida. Aplicaciones.* PhD thesis, University of Granada, Department of Computer Science, Mayo 2005.

[19] K.Saito y R.Nakano. Law discovery using neural networks. En *Procs. of the XVth International Joint Conference on Artificial Intelligence (IJ-*

CAI97), páginas 1078–1083, Nagoya, Japan, 1997.

[20] Michael Schmitt. Product unit neural networks with constant depth and superlinear VC dimension. Technical report, 2001.

[21] Kourosh Nourouzi. The geometry of the unit ball of some tensor product spaces. *Appl. Math. Lett*, 20(4):401–404, 2007.

[22] Michael Schmitt. VC dimension bounds for product unit networks. En *IJCNN (4)*, páginas 165–170, 2000.

[23] César Hervás-Martínez, Francisco J. Martínez-Estudillo, y Mariano Carbonero-Ruz. Multilogistic regression by means of evolutionary product-unit neural networks. *Neural Networks*, 21(7):951–961, 2008.

[24] Francisco J. Martínez-Estudillo, César Hervás-Martínez, Pedro Antonio Gutiérrez, y Alfonso C. Martínez-Estudillo. Evolutionary product-unit neural networks classifiers. *Neurocomputing*, 72(1-3):548–561, 2008.

[25] César Hervás-Martínez y Francisco J. Martínez-Estudillo. Logistic regression using covariates obtained by product-unit neural network models. *Pattern Recognition*, 40(1):52–64, 2007.

[26] César Hervás-Martínez, Francisco J. Martínez-Estudillo, y Pedro Antonio Gutiérrez. Classification by means of evolutionary product-unit neural networks. En *IJCNN*, páginas 1525–1532. IEEE, 2006.

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

[28] C.M. Bishop. On the complexity of computing and learning with multiplicative neural networks. *Neural Computation*, 14(2):241–301, 2002.

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

[30] D.J Janson y J.F. Frenzel. Training product unit neural networks with genetic algorithms. *IEEE Expert: Intelligent Systems and Their Applications*, 8(5):26–33, 1993.

[31] R.P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, 27(11):47–64, 1989.

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

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

[34] E. Hartman, J. D. Keeler, y J. M. Kowalski. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation*, 2(2):210–215, 1990.

[35] J. Park y I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

[36] C. Stanfill y D. Waltz. Toward memory based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[37] D. Randall Wilson y Tony R. Martínez. Improved heterogeneous distance functions. *J. Artif. Intell. Res. (JAIR)*, 6:1–34, 1997.

[38] B. G. Batchelor. *Pattern Recognition: Ideas in Practice*. Plenum, 1978.

[39] Manabu Ichino y Hiroyuki Yaguchi. Generalized minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):698–708, 1994.

[40] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, August 2006.

[41] R.O. Duda, P.E. Hart, y D.G. Stork. *Pattern Classification, 2nd Edition*. Wiley Interscience, 2000.

[42] R. Caruana y A. Niculescu-Mizil. Data mining in metric space: An empirical analysis of supervised learning performance criteria. En *Proceedings of the 10th International Conference in Knowledge Discovery and Data Mining*, páginas 69–78, Seattle, USA, August 2004.

[43] M. Sokolova y G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45:427–437, 2009.

[44] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[45] F. Provost y T. Fawcett. Analysis and visualization of the classifier performance: comparison under imprecise class and cost distribution. En *Proceedings of the Third International Conference on Knowledge Discovery (KDD97) and Data Mining*, páginas 43–48, California, USA, August 1997.

[46] F. Provost y T. Fawcett. Robust classification system for imprecise environments. En *Proccedings of the Fithteenth National Conference on Artificial Intelligence*, páginas 706–713, Chicago, USA, July 1998.

[47] N. Lachiche y P.A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves. En *Proceedings of the 20th International Conference on Machine Learning , ICML'03*, páginas 416–423, Washington, USA, August 2003.

[48] R.M. Everson y J.E. Fieldsend. Multi-class roc analysis from a multi-objetive optimisation perspective. *Pattern Recognition Letters*, 27:918–927, 2006.

[49] T.C.W. Langrebe y R.P.W. Duin. Efficient multiclass roc approximation by decomposition via confusion matrix perturbation analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):810–822, 2008.

[50] D. Edwards, C. Metz, y M. Kupinski. Ideal observers and optimal roc hypersurfaces in n-class classification. *IEEE Transactions on Medical Imaging*, 23(7):891–895, 2004.

[51] S. Dreiseitl. Training multiclass classifiers by maximizing the volume under the roc surface. En *Proceedings of the 11th International Conference on Computer Aided Systems Theory, EUROCAST 2007*, volumen 4739, páginas 878–885, Las Palmas de Gran Canaria, España, February 2007.

[52] B. Sahiner, H.P. Chan, y L.M. Hadjiiski. Performance analysis of three-class classifiers: Properties of a 3-d roc surface and the normalized volume under the surface for the ideal observer. *IEEE Transactions on Medical Imaging*, 27(2):215–227, 2008.

[53] X. He y E.C. Frey. The meaning and use of the volume under a three-class roc surface (vus). *IEEE Transactions on Medical Imaging*, 27:577–588,

2008.

[54] D. Mossman. Three-way rocs. *Medical Decision Making*, 19(1):78–89, 1999.

[55] W.J. Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

[56] H.A. Abbass. A memetic pareto evolutionary approach to artificial neural networks. En *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence in Lecture Notes In Computer Science*, páginas 1–12, Adelaide, Australia, December 2001.

[57] H.A. Abbass. An evolutionary artificial neural networks approach for breast cancer diagnosis. *Artificial Intelligence in Medicine*, 25:265–281, 2002.

[58] B.W. Matthews. Comparison to predicted and observed secondary structure of t4 phage lysozyme. *Biochim Biophys Acta*, 405(2):442–451, 1975.

[59] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[60] J.H. Zar. *Biostatistical Analysis*. Prentice Hall, 1996.

[61] Frank Wilcoxon. *Individual Comparisons by Ranking Methods*. International Biometric Society, 1945.

[62] P. Kordík, J. Koutník, J. Drchal, O. Kovárik, M. Cepek, y M. Snorek. Meta-learning approach to neural network optimization. *Neural Networks*, 23:568–582, 2010.

[63] N. Burgess. A constructive algorithm that converges for real-valued input patterns. *International Journal on Neural Systems*, 5(1):59–66, 1994.

[64] R. Setiono. Use of a quasinewton method in feedforward neural-network construction algorithm. *IEEE Transactions on Neural Networks*, 5:54–65, 1995.

[65] R. Reed. Pruning algorithms - a survey. *IEEE Transactions on Neural Networks*, 4:740–747, 1993.

[66] R. Reed y R. Marks. *Neural Smithing*. The MIT Press, 1999.

[67] Y. Chauvin y D.E. Rumelhart. *Backpropagation: Theory, Architectures, and Applications*. Lawrence Erlbaum Associates Inc, 1995.

[68] D.R. Hush y B.G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, 10:8–39, 1993.

[69] M.F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[70] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Department of Applied Mathematics, 1974.

[71] A.H. Kramer y A. Sangiovanni-Vincentelli. chapter Efficient parallel learning algorithms for neural networks, páginas 40–48. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[72] F.M. Silva y L.B. Almeida. Acceleration techniques for the backpropagation algorithm. En L.B. Almeida y C.J. Wellekens, editors, *Procs. of the 1990 EURASIP Workshop on Neural Networks*, páginas 110–119. Springer (Lecture Notes in Computer Science, 412), 1990.

[73] A. Namatame y Y. Kimata. Improving the generalising capabilities of a backpropagation network. *Neural Networks*, 1(2):86–94, 1989.

[74] Serban Radu. Multilayer perceptron trained by the backpropagation algorithm. página 3, Vouliagmeni, Athens, Greece, Mayo 29-31 2003. WSEAS.

[75] M. O. Duff. Backpropagation and Bach's 5th cello suite (sarabande). Poster Presentation at IJCNN-89, 1989.

[76] E. Deprit. Implementing recurrent backpropagation on the connection machine. *Neural Networks*, 2(4):295–314, 1989.

[77] F. J. Pineda. Recurrent backpropagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1(2):161–172, 1989.

[78] C. Igel y M. Husken. Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50(6):105–123, 2003.

[79] M. Riedmiller y H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. En *Procs. of the 1993*

IEEE International Conference on Neural Networks, páginas 586–591, San Francisco, CA, 1993.

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

[81] S. Walczak y N. Cerpa. Heuristic principles for the design of artificial neural networks. *Information and Software Technology*, 41:107–117, 1999.

[82] A. Abraham y B. Nath. Hybrid heuristics for optimal design of artificial neural networks. En *Proceedings of the Third International Conference on Recent Advances in Soft Computing, RASC 2000*, páginas 15–22, Leicester, England, June 2000.

[83] F. Glover y G.A. Kochenberger. *Handbook of metaheuristics*. Kluwer Academic Publishers, 2003.

[84] M. Riedmiller. Advanced supervised learning in multi-layer perceptrons–from backpropagation to adaptive learning algorithms. *Computer Standards and Interfaces*, 16:265–278, 1994.

[85] Z. Zainuddin, N. Mahat, y Y.A. Hassan. Improving the convergence of the backpropagation algorithm using local adaptive techniques. *World Academy of Science, Engineering and Technology*, 1:79–82, 2005.

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

[87] A.C. Veitch y G. Holmes. A modified quickprop algorithm. *Neural Computation*, 3(3):310–311, 1991.

[88] Christian Igel y Michael Hüsken. Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, 50:105–123, 2003.

[89] R. Fletcher y C.M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[90] E. Polak y G. Ribiére. Note sur la convergence de methods de directions conjugres. *Revue Francaise Informat, Reserche Opérationnele*, 16:35–43, 1969.

[91] M.J.D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254, 1977.

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

[93] C.G. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.

[94] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[95] E. Alba y R. Martí. *Metaheuristic Procedures for Training Neural Networks*. Springer, 2006.

[96] D. Corne, M. Dorigo, y F. Glover. *New Ideas in Optimization*. McGraw-Hill, 1999.

[97] R. Eberhart, R. Dobbins, y P. Simpson. *Computational Intelligence PC Tools*. Academic Press; Pap/Dis edition, September 1996.

[98] S. N. Qasem y S. M. Shamsuddin. Generalization improvement of radial basis function network based on multi-objective particle swarm optimization. 2010.

[99] Eysa Salajegheh, Saeed Gholizadeh, y Mohsen Khatibinia. Optimal design of structures for earthquake loads by a hybrid RBF-BPSO method. 2008.

[100] Ludmila Kuncheva. Initializing of an RBF network by a genetic algorithm. *Neurocomputing*, 14(3):273–288, 1997.

[101] Ben Burdsall y Christophe Giraud-Carrier. GA-RBF: A self-optimising RBF network. En *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97)*, páginas 348–351. Springer-Verlag, Abril 1997.

[102] QingNian Zhang, XiangYang He, y JianQi Liu. RBF network based on genetic algorithm optimization for nonlinear time series prediction. En *ISCAS (5)*, páginas 693–696, 2003.

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

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

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

[106] Tong-Seng Quah y Kian-Chong Wong. Utilizing generalized growing and pruning algorithm for radial basis function (GGAP-RBF) network in predicting IPOs performance. En *IJCNN*, páginas 3007–3012. IEEE, 2006.

[107] Ma Li, Abdul Wahab, y Chai Quek. A modified generalized RBF model with EM-based learning algorithm for medical applications. En *CBMS*, páginas 291–296. IEEE Computer Society, 2006.

[108] Guang bin Huang, P. Saratch, Senior Member, y Narashiman Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation, 2005.

[109] Tong seng Quah y Kian chong Wong. Tong-seng quah and kian-chong wong predicting IPOs performance using GGAP-RBF network predicting IPOs performance using generalized growing and pruning algorithm for radial basis function (GGAP-RBF) network, Deciembre 30 2008.

[110] Renato Tinós y Luiz Otávio Murta Jr. Selection of radial basis functions via genetic algorithms in pattern recognition problems. En Marley Maria Bernardes Rebuzzi Vellasco, Marcílio Carlos Pereira de Souto, y Jés Jesus Fiais Cerqueira, editors, *SBRN*, páginas 171–176. IEEE Computer Society, 2008.

[111] Stergios Papadimitriou, Seferina Mavroudi, Liviu Vladutu, y Anastasios Bezerianos. Generalized radial basis function networks trained with instance based learning for data mining of symbolic data. *Appl. Intell*,

16(3):223–234, 2002.

[112] Shayan Mukherjee y Shree K. Nayar. Automatic generation of GRBF networks for visual learning. En *ICCV*, páginas 794–800, 1995.

[113] Automatic generation of GRBF networks for visual learning. Junio 20 1995.

[114] N. B. Karayiannis y G. W. Mi. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural Networks*, 8(6):1492–1506, Noviembre 1997.

[115] Er P. Topchy, Oleg A. Lebedko, Victor V. Miagkikh, y Nikola K. Kasabov. Adaptive training of radial basis function networks based on cooperative evolution and evolutionary programming, 1998.

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

[117] D. Montana y L. Davis. Training feedforward neural networks using genetic algorithms. En *Procs. of the 11th International Joint Conference on Artificial Intelligence*, páginas 762–767, 1989.

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

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

[120] J. Kennedy y R. Eberhart. Particle swarm optimization. En D.S. Touretzky, editor, *Procs. of the IEEE International Conference on Neural Networks*, volumen 4, páginas 1942–1948, Perth, WA, Australia, 1995.

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

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

1992.

[123] H. Braun. *Neuronale Netze: Optimierung durch Lernen und Evolution.* Springer-Verlag, 1997.

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

[125] S.E. Fahlman y C. Lebiere. The cascade-correlation learning architecture. En D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volumen 2, páginas 524–532, San Francisco, CA, 1990. Morgan Kaufmann, San Mateo Inc.

[126] S. P. Lloyd. Least squares quantization in PCM'S. *Bell Telephone Labs Memo*, 1957.

[127] Masa aki Sato y Shin Ishii. On-line EM algorithm for the normalized gaussian network. *Neural Computation*, 12(2):407–432, 2000.

[128] Paul S. Bradley, Usama M. Fayyad, y Cory Reina. Scaling clustering algorithms to large databases. En *KDD*, páginas 9–15, 1998.

[129] G.H. Ball y D.I. Hall. Some fundamental concepts and synthesis procedures for pattern recognition preprocessors. En *Int'l Conf. Microwaves, Circuit Theory, and Information Theory*, páginas 281–297, 1964.

[130] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[131] Paul D Fox y Elmawati L Sutanto. On merging gradient estimation with mean-tracking techniques for cluster identification, Octubre 29 1997.

[132] Héctor Pomares, Ignacio Rojas, Julio Ortega, Jesús González, y Alberto Prieto. A systematic approach to a self-generating fuzzy rule-table for function approximation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(3):431–447, 2000.

[133] Partha Pratim Kanjilal y Debarag Narayan Banerjee. On the application of orthogonal transformation for the design and analysis of feedforward networks. *IEEE Transactions on Neural Networks*, 6(5):1061–1070, Septiembre 1995.

[134] S. Chen, C. F. N. Cowan, y P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991.

[135] X. Yao. Global optimization by evolutionary algorithms. En *Procs. of the 2nd Aizu International Symposium on Parallel Algorithms/Architecutre Synthesis, (pAs'97)*, páginas 282–291, Aizu-Wakamatsu, Japan, 1997.

[136] K. Deb. A population-based algorithm-generator for real-parameter optimization. *Soft Comput*, 9(4):236–253, 2005.

[137] F. J. Martínez-Estudillo, C. Hervás-Martínez, Gutiérrez. P. A., y A. C. Martínez-Estudillo. Evolutionary product-unit neural networks classifiers. *Neurocomputing*, 72(1-2):548–561, 2008.

[138] J. M. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[139] I. Rechenberg. *Evolutions strategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[140] Hans-Paul Schwefel. *Numerische optimierung von computer-modellen mittels der evolutionsstrategie.* Birkhäuser, Basel, 1977.

[141] L. J. Fogel, A. J. Owens, y M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley & Sons, New York, 1966.

[142] J. R. Koza. *Genetic Programming.* MIT Press, 1992.

[143] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs.* Springer-Verlag New York, Inc., New York, NY, USA, 2nd edition, 1994.

[144] C.Z. Janikow y Z. Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. En *Procs. of the 4th International Conference on Genetic Algorithms (ICGA)*, páginas 31–36, 1991.

[145] A.H. Wright. En G.J. Rawlins, editor, *Foundations of genetic algorithms*, chapter Genetic Algorithms for Real Parameter Optimization, páginas 205–218. Morgan Kaufmann, San Mateo, CA, 1991.

[146] F. Herrera, M. Lozano, y J.L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 12(4):265–319, 1998.

[147] J. A. Nelder y R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[148] Brooks S. H. A discussion of random methods for seeking maxima. *Operations Research*, 6:244–253, 1958.

[149] X. Yao y Y Liu. Evolving artificial neural networks through evolutionary programming. En L. Fogel, P. Angeline, y T. Back, editors, *Evolutionary Programming V: Procs. of the Fifth Annual Conference on Evolutionary Programming*, páginas 257–266, 1996.

[150] David J. Janson y James F. Frenzel. Training product unit neural networks with genetic algorithms. *IEEE Expert*, 8(5):26–33, Octubre 1993.

[151] D. Whitley, T. Starkweather, y C. Bogart. Genetic algorithms and neural networks - optimizing connections and connectivity. *Parallel Computing*, 14(3):347–361, Agosto 1990.

[152] N. García-Pedrajas, C. Hervás-Martínez, y J. Muñoz-Pérez. Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). *Neural Networks*, 15(10):1259–1278, 2002.

[153] Xin Yao y Yong Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, Mayo 1997.

[154] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. En *Proceedings of the 5th Symposium on Mathematics, Statistics and Probability*, páginas 281–297, Berkley, 1967. University of California Press.

[155] C. Hervás, F. J. Martínez-Estudillo, y P.A. Gutiérrez. Classification by means evolutionary product-unit neural networks. En *Procs. of the 2006 International Joint Conference on Neural Networks*, páginas 2834–2842, Vancouver, Canada, 2006.

[156] I Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Verlag, Stuttgart, 1973.

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

[158] A.C. Martínez-Estudillo, C. Hervás, F.J. Martínez-Estudillo, y N. García. 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, 2006.

[159] F.J. Martínez-Estudillo, C. Hervás, A.C. Martínez-Estudillo, S. Ventura, y P.A. Gutiérrez. Evolutionary product-unit neural networks for classification. En *Procs. of the 7th International Conference on Intelligent Data and Automated Learning (IDEAL 2006)*, páginas 1320–1328, Burgos, España, 2006. Springer (Lecture Notes in Computer Science, 4664).

[160] F.J. Martínez-Estudillo, C. Hervás, P.A. Gutiérrez, y A.C. Martínez-Estudillo. Evolutionary product-unit neural networks classifiers. *Neurocomputing*, 2007 (Submitted).

[161] F.J. Martínez-Estudillo, C. Hervás, A.C. Martínez-Estudillo, y P.A. Gutiérrez. Hybrid evolutionary product-unit neural networks for classification. En *Procs. of the 10th International Work-Conference on Artificial Neural Networks(IWANN 2007)*, páginas 351–358, San Sebastián, España, 2007. Springer (Lecture Notes in Computer Science, 4507).

[162] C. Hervás y F.J. Martínez-Estudillo. Logistic regression using covariates obtained by product unit neural networks models. *Pattern Recognition*, 40(1):52–64, 2006.

[163] C. Hervás, F.J. Martínez-Estudillo, A.C Martínez-Estudillo, P.A. Gutiérrez, y J.C. Fernández. Aprendizaje mediante la hibridación de técnicas heurísticas y estadísticas de optimización en regresión logística binaria. En *Procs. of the V Congreso Español sobre Metaheurísticas and Algoritmos Evolutivos y Bioinspirados(MAEB07)*, páginas 61–68, 2007.

[164] C. Hervás, F.J. Martínez-Estudillo, P.A. Gutiérrez, y A. Ruíz. Regresión no lineal mediante la evolución de modelos híbridos de redes neuronales. En *Procs. of the IV Congreso Español sobre Metaheurísticas and Algoritmos Evolutivos y Bioinspirados (MAEB05)*, páginas 333–340, Granada,España, 2005.

[165] C. Hervás, F.J. Martínez-Estudillo, P.A. Gutiérrez, J.C. Fernández, y A.J. Tallón-Ballesteros. Clasificación mediante la evolución de modelos híbridos de redes neuronales. En *Procs. of the V Congreso Español sobre Metaheurísticas y Algoritmos Evolutivos y Bioinspirados (MAEB07)*, páginas 77–84, 2007.

[166] Pedro Antonio Gutiérrez, César Hervás, Mariano Carbonero-Ruz, y Juan Carlos Fernández. Combined projection and kernel basis functions for classification in evolutionary neural networks. volumen 72, páginas 2731–2742, 2009.

[167] Chi-Chun Lo y Wei-Hsin Chang. A Multiobjective Hybrid Genetic Algorithm for the Capacitated Multipoint Network Design Problem. En *1999 IEEE International Conference on Communications*, volumen 3, páginas 1573–1576, 1999.

[168] Abhishek Sinha y David E. Goldberg. A survey of hybrid genetic and evolutionary algorithms, Agosto 20 2003.

[169] David H. Wolpert y William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Abril 1997.

[170] Kay Chen Tan, Q. Yu, C. M. Heng, y Tong Heng Lee. Evolutionary computing for knowledge discovery in medical diagnosis. *Artificial Intelligence in Medicine*, 27(2):129–154, 2003.

[171] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[172] D. Wicker, M. M. Rizki, y L. A. Tamburino. A hybrid evolutionary learning system for synthesizing neural network pattern recognition systems. *Lecture Notes in Computer Science*, 1447:619–628, 1998.

[173] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts, Mayo 1994.

[174] D. B. Fogel. *System Identification Through Simulated Evolution: A machine Learning Approach to modeling*. Ginn Press, 160 Gould Street, Needham Heights, MA 01294, 1991.

[175] D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Trans. Neural Networks*, 5(1):3–14, 1994.

[176] Ling Wang. A hybrid genetic algorithm-neural network strategy for simulation optimization. *Applied Mathematics and Computation*, 170(2):1329–1343, 2005.

[177] Michael Lee. Dynamic control of genetic algorithms using fuzzy logic techniques. En Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufman.

[178] M. Clerc y J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE-EC*, 6:58–73, Febrero 2002.

[179] James Kennedy y Russell Eberhart. Particle swarm optimization. En *IEEE International Conference on Neural Networks (ICNN'95)*, volumen 4, páginas 1942–1947, Perth, Western Australia, Noviembre-Deciembre 1995. IEEE.

[180] Sylverin Kemmoé Tchomté y Michel Gourgand. Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems. 2009.

[181] Shi, Liang, Lee, Lu, y Wang. An improved GA and a novel PSO-GA-based hybrid algorithm. *IPL: Information Processing Letters*, 93, 2005.

[182] Gerry Dozier, James Bowen, y Abdollah Homaifar. Solving constraint satisfaction problems using hybrid evolutionary search. *IEEE Transactions on Evolutionary Computation*, 2(1):23–33, Abril 1998.

[183] Lin Yu Tseng y Shih-Chieh Chen. A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of*

Operational Research, 175(2):707–721, 2006.

[184] Manuel Vázquez y L. Darrell Whitley. A hybrid genetic algorithm for the quadratic assignment problem. En L. Darrell Whitley, David E. Goldberg, Erick Cantú-Paz, Lee Spector, Ian C. Parmee, y Hans-Georg Beyer, editors, *GECCO*, páginas 135–142. Morgan Kaufmann, 2000.

[185] Fred Glover. Tabu search–part i. *INFORMS JOURNAL ON COMPUTING*, 1(3):190–206, 1989.

[186] B. C. H. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, Enero 28 2000.

[187] C. F. Liaw. A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Oprational Research*, 124(1):28–42, 2000.

[188] Edward Keedwell y Soon-Thiam Khu. A hybrid genetic algorithm for the design of water distribution networks. *Eng. Appl. of AI*, 18(4):461–472, 2005.

[189] John von Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press, Urbana, Illinois, 1966. Edited and completed by Arthur W. Burks.

[190] E. K. Burke y A. J. Smith. Hybrid evolutionary techniques for the maintenance scheduling problem. *IEEE Power Engineering Society Transactions*, 1999.

[191] Myung H y Kim JH. Hybrid evolutionary programming for heavily constrained problems. *Biosystems*, 1(38):29–43, 1996.

[192] César Hervás-Martínez, P.A. Gutierrez, y J. C. Fernández. Hyperbolic tengent basis function neural networks training by hybrid evolutionary programing for accurrate short-term. ISDA, 2009.

[193] S. le Cessie y J. C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[194] Paul R. Komarek y Andrew W. Moore. Fast robust logistic regression for large sparse datasets with binary outputs, Noviembre 15 2003.

[195] T. J. Santner y D. E. Duffy. *The Statistical Analysis of Discrete Data.* Springer-Verlag New York, Inc., 1989.

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

[197] M. W. Scott y J. R. Thompson. Probability density estimation in higher dimension. En *Proc. 15th Symposium on the Interface, North Holland-Elsevier*, páginas 173–179. Douglas, S.R. (ed): Computer Science and Statistics, 1983.

[198] Charu Aggarwal, Alexander Hinneburg, y Daniel Keim. On the surprising behavior of distance metrics in high dimensional space. En *Database Theory ICDT 2001*, volumen 1973 of *Lecture Notes in Computer Science*, páginas 420–434. Springer Berlin / Heidelberg, 2001.

[199] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, y Uri Shaft. When is "nearest neighbor" meaningful? En *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, páginas 217–235, London, UK, 1999.

[200] L. Bellman. *Adaptative Control Processes: A guided tour*. Priceton University Press, 1st edition, 1961.

[201] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman & Hall, 1st edition, 1986.

[202] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, Boston, MA, 1st edition, 1990.

[203] J. Guo, E. Levina, G. Michailidis, y J. Zhu. Pairwise variable selection for high-dimensional model-based clustering. *Biometrics*, 66(3):793–804, 2010.

[204] R. N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. i. *Psychometrika*, 27(2):125–140, 1962.

[205] R. N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. ii. *Psychometrika*, 27(3):219–246, 1962.

[206] W. Dzwinel. How to make sammon's mapping useful for multidimensional data structures analysis. *Pattern Recognition*, 27(7):949–959, 1994.

[207] P. Demartines y J. Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, 1997.

[208] Avrim L. Blum y Ronald L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117 – 127, 1992.

[209] M. Dash y H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(1-4):131 – 156, 1997.

[210] Huan Liu y Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transaction on Knowledge and Data Engeneering*, 17:491–502, 2005.

[211] D.E. Goldberg. Sizing populations for serial and parallel genetic algorithms. En *Procs. of the 3rd International Conference on Genetic Algorithms*, páginas 70–79, 1989.

[212] Ron Kohavi y George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[213] David J. Stracuzzi y Paul E. Utgoff. Randomized variable elimination. *Journal of Machine Learning Research.*, 5:1331–1362, 2004.

[214] Hadi Sadoghi Yazdi, Javad Haddadnia, y Mojtaba Lotfizad. Duct modeling using the generalized RBF neural network for active cancellation of variable frequency narrow band noise. 2007.

[215] Qing Chen, Shaoyuan Li, Yugeng Xi, y Guang-Bin Huang. Furnace temperature modeling for continuous annealing process based on generalized growing and pruning RBF neural network. En Fuliang Yin, Jun Wang, y Chengan Guo, editors, *ISNN (2)*, volumen 3174 of *Lecture Notes in Computer Science*, páginas 755–760. Springer, 2004.

[216] Ileana Popescu, Athanasios Kanatas, Evangelos Angelou, Ioan Nafornita, y Philip Constantinou. Applications of generalized rbf-nn for path loss prediction, Agosto 26 2002.

[217] Carlos J. Pantaleón, Jose A. García, y Ángel Mediavilla. A nonlinear MESFET model for intermodulation analysis using a generalized radial basis function network, Mayo 21 1999.

[218] Nung Kion Lee y Dianhui Wang. Realization of generalized RBF network. En *CITA*, páginas 82–88, 2003.

[219] Frauke Friedrichs y Michael Schmitt. On the power of boolean computations in generalized RBF neural networks. *Neurocomputing*, 63:483–498, 2005.

[220] Evangelos Dermatas. Direct estimation of polynomial densities in generalized rbf networks using moments. En *ICANN*, páginas 119–126, 2001.

[221] Akio Miyazaki y Tsuyoshi Yamada. A systematic synthesis procedure for feed-forward neural networks by using the GRBF (generalized radial basis function) network technique. En *Proceedings of 1993 IEEE International Conference on Neural Networks (ICNN'93)*, volumen 1, páginas 363–366, Nagoya, Japan, Octubre 1993. IEEE/INNS. Kyushu University.

[222] Bruce E. Segee y Michael J. Carter. Comparative fault tolerance of generalized radial basis function and multilayer perceptron networks. En *Proceedings of 1993 IEEE International Conference on Neural Networks (Joint FUZZ-IEEE'93 and ICNN'93 [IJCNN93])*, volumen III, páginas 1847–1852, San Francisco, California, Marzo-Abril 1993. IEEE/INNS.

[223] Ma Li, Abdul Wahab, y Chai Quek. A modified generalized RBF model with EM-based learning algorithm. En Hamid R. Arabnia, editor, *IC-AI*, páginas 262–270. CSREA Press, 2006.

[224] Jun Li y You-Peng Zhang. Modelling of dynamic systems using generalized RBF neural networks based on kalman filter mehtod. En Derong Liu, Shumin Fei, Zeng-Guang Hou, Huaguang Zhang, y Changyin Sun, editors, *ISNN (1)*, volumen 4491 of *Lecture Notes in Computer Science*, páginas 676–684. Springer, 2007.

[225] Ewaryst Rafajlowicz y Miroslaw Pawlak. Optimization of centers' positions for RBF nets with generalized kernels. En Leszek Rutkowski, Jörg H. Siekmann, Ryszard Tadeusiewicz, y Lotfi A. Zadeh, editors,

ICAISC, volumen 3070 of *Lecture Notes in Computer Science*, páginas 253–259. Springer, 2004.

[226] Frauke Friedrichs y Michael Schmitt. On the power of boolean computations in generalized RBF neural networks, Febrero 06 2008.

[227] Adiel Castaño, Francisco Fernández-Navarro, César Hervás-Martínez, M. M. García, y Pedro Antonio Gutiérrez. Classification by evolutionary generalised radial basis functions. *Int. J. Hybrid Intell. Syst*, 7(4):239–248, 2010.

[228] Adiel Castaño, César Hervás-Martínez, Pedro Antonio Gutiérrez, F. Fernández-Navarro, y M. M. García. Classification by evolutionary generalized radial basis functions. En *ISDA*, páginas 203–208. IEEE Computer Society, 2009.

[229] D. Francois. *High dimentional Data Analisis, From Optimal Metric to Feature Selection*, chapter Seeking on right metric, páginas 54 – 55. VDM Verlag, Saarbrucken, Alemania, 2008.

[230] P. A. Castillo-Valdivieso, J. J. Merelo, A. Prieto, I. Rojas, y G. Romero. Statistical analysis of the parameters of a neuro-genetic algorithm. *IEEE-NN*, 13:1374–1394, Noviembre 2002.

[231] Jr. Miller, R. G. *Simultaneous Statistical Inference*. Springer-Verlag New York, Inc., New York, NY, USA, 2nd edition, 1981.

[232] O.J. Dunn. Multiple comparisons among means. *journal of the American Statistical Association*, (56):52–64, 1961.

[233] Y. Hochberg y A. Tamhane. *Multiple Comparison Procedures*. John Wiley and Sons.

[234] P. A. Gutiérrez, C. Hervás-Martínez, y M. Lozano. Designing multilayer perceptrons using a guided saw-tooth evolutionary programming algorithm. *Soft Computing*, 14(4):599–613, 2010.

[235] A. Castaño, F. Fernández-Navarro, C. Hervás-Martínez, y P. Gutierrez. Neuro-logistic models based on evolutionary generalized radial basis function for the microarray gene expression classification problem. *Neural Processing Letters*, páginas 1–15. 10.1007/s11063-011-9187-8.

[236] Adiel Castaño Méndez, Francisco Fernández-Navarro, Pedro Gutiérrez, Manuel Baena-García, y César Hervás-Martínez. Combining evolutionary generalized radial basis function and logistic regression methods for classification. En Emilio Corchado, Václav Snáidel, Javier Sedano, Aboul Hassanien, y Calvo, editors, *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*, Advances in Intelligent and Soft Computing.

[237] Adiel Castaño, Francisco Fernández-Navarro, Pedro Antonio Gutiérrez, y César Hervás-Martínez. Permanent disability classification by combining evolutionary generalized radial basis function and logistic regression methods. *Expert Syst. Appl.*, 39(9):8350–8355, 2012.

[238] Lei Yu y Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. En Tom Fawcett y Nina Mishra, editors, *ICML*, páginas 856–863. AAAI Press, 2003.

[239] R. Ruiz, J.S. Aguilar-Ruiz, y J.C. Riquelme. Best agglomerative ranked subset for feature selection. *JMLR Workshop and Conference Proceedings*, 4:146–160, 2008.

[240] L. J. Van't Veer, H. Dai, M. J. Van de Vijver, Y. D. He, A. A. M. Hart, M. Mao, H. L. Peterse, K. Van Der Kooy, M. J. Marton, A. T. Witteveen, G. J. Schreiber, R. M. Kerkhoven, C. Roberts, P. S. Linsley, R. Bernards, y S. H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, 2002.

[241] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, y T. R. Golub. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

[242] U. Alon, N. Barka, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, y A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide

arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 96(12):6745–6750, 1999.

[243] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, y E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–527, 1999.

[244] A. Bhattacharjee, WG. Richards, J. Staunton, C. Li, S. Monti, P. Vasa, C. Ladd, J. Beheshti, R. Bueno, M. Gillette, M. Loda, G. Weber, EJ. Mark, ES. Lander, W. Wong, BE. Johnson, TR. Golub, DJ. Sugarbaker, y M. Meyerson. Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences of the United States of America*, 98(24):13790–13795, 2001.

[245] S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. . Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, y T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences of the United States of America*, 98(26):15149–15154, 2001.

[246] I.H. Witten y E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[247] S. P. Lloyd. Least-sqaures quantization in PCM. Technical report, Bell Labs, 1957.

[248] S. le Cessie y J.C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[249] N. Landwehr, M. Hall, y E. Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

[250] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1999.

[251] T. Hastie, R. Tibshirani, y J.H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[252] C.C. Chang y C.J. Lin. Libsvm : a library for support vector machines, 2001.

[253] J. Demsar. Statistical comparisons of clasiffiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[254] Consejo General del Poder Judicial. Ley general de la seguridad social. Madrid, 1994.

[255] Instituto Nacional de Estadística. Clasificación nacional de ocupaciones (cno-94). Madrid: INE, 1994.

[256] Pedro Antonio Gutiérrez, César Hervás-Martínez, y Francisco J. Martínez-Estudillo. Logistic regression by means of evolutionary radial basis function neural networks. *IEEE Transactions on Neural Networks*, 22(2):246–263, 2011.