Universidad de Granada

Aprendizaje de Sistemas Basados en Reglas Difusas Compactos y Precisos con Programación Genética

Tesis Doctoral

Francisco José Berlanga Rivera

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Editor: Editorial de la Universidad de Granada
Autor: Francisco José Berlanga Rivera
D.L.: GR 3106-2010
ISBN: 978-84-693-3294-8

UNIVERSIDAD DE GRANADA

Aprendizaje de Sistemas Basados en Reglas Difusas Compactos y Precisos con Programación Genética

MEMORIA QUE PRESENTA

Francisco José Berlanga Rivera

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

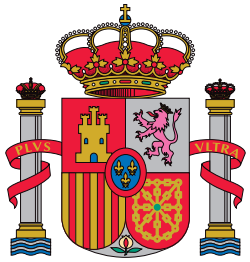
Mayo de 2010

DIRECTORES

Francisco Herrera Triguero
María José del Jesus Díaz

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Tesis Doctoral parcialmente subvencionada por la Comisión
Interministerial de Ciencia y Tecnología con los proyectos
TIN2008-06681-C06-02 y TIN2008-06681-C06-01
y la Junta de Andalucía con el proyecto de excelencia TIC-3928



CICYT

TIN2008-06681-C06-02

y

TIN2008-06681-C06-01

TIC-3928

La memoria titulada "*Aprendizaje de Sistemas Basados en Reglas Difusas Compactos y Precisos con Programación Genética*", que presenta D. Francisco José Berlanga Rivera para optar al grado de doctor, ha sido realizada dentro del programa de doctorado "*Diseño, Análisis y Aplicaciones de Sistemas Inteligentes*" del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de los doctores D. Francisco Herrera Triguero, de la Universidad de Granada, y D^a. María José del Jesus Díaz, de la Universidad de Jaén.

Granada, Mayo de 2010

El Doctorando

Fdo: D. Francisco José Berlanga Rivera

El Director

Fdo: D. Francisco Herrera Triguero

El Director

Fdo: D^a. María José del Jesus Díaz

Agradecimientos

Gracias a todos.

Índice

Índice de figuras

Índice de tablas

Introducción

A Planteamiento

Actualmente se ha incrementado de forma paralela tanto la cantidad de información almacenada como la necesidad de desarrollar algoritmos que permitan extraer conocimiento útil de la misma de forma automática. Estos algoritmos se incluyen dentro del área de extracción de conocimiento en bases de datos (*KDD, Knowledge Discovery in Databases*) [FPS96a, KZ02, TSK06].

La extracción de conocimiento se puede abordar, en función del problema a resolver, desde dos perspectivas distintas: desde el punto de vista predictivo, como un proceso de inducción predictiva que intenta obtener conocimiento que permita pronosticar el comportamiento futuro según los datos disponibles, o desde el punto de vista descriptivo, cuyo objetivo fundamental es descubrir conocimiento de interés dentro de los datos, intentando obtener información que describa el modelo que existe detrás de los datos. La inducción predictiva se realiza bajo enfoques como la clasificación [Han81, WK91], la regresión [BFOS84, McL92] o el análisis de series temporales [BD96].

La clasificación es un tipo de inducción predictiva en la que los datos son objetos caracterizados por atributos que pertenecen a diferentes clases definidas, y el objetivo es inducir un modelo (un *Clasificador* o *Sistema de Clasificación*) capaz de predecir la clase a la cual pertenece un nuevo objeto dado los valores de sus atributos.

En determinados problemas de clasificación tales como el diagnóstico de enfermedades o el reconocimiento de rasgos faciales, entre otros, interviene información compleja, incompleta, imprecisa o con incertidumbre, que los expertos humanos

procesan de forma robusta, pero que es difícil representar y procesar en un Sistema de Clasificación. Para diseñar un esquema de clasificación robusto y con un rendimiento e interpretabilidad altos, es conveniente el uso de una herramienta para tratar con este tipo de información presente en la mayoría de los procesos de clasificación: la *Lógica Difusa*.

La teoría de conjuntos difusos [Zad65] permite manejar imprecisión y tratar el conocimiento humano de una forma sistemática. En el campo de la clasificación, el papel fundamental de los conjuntos difusos es hacer transparentes los esquemas de clasificación que utilizan normalmente los seres humanos a través del desarrollo de un marco formal implementable en un ordenador [Zad77].

Si a la utilización de la Lógica Difusa en un Sistema de Clasificación unimos el diseño de un sistema basado en reglas, entonces obtendremos un *Sistema de Clasificación Basado en Reglas Difusas* (SCBRD), el cual está formado por un conjunto de modelos simples locales, verbalmente interpretables y con un rango de aplicación muy amplio. Los SCBRDs permiten la incorporación en el modelo de toda la información disponible, tanto de la que proviene de expertos humanos que expresan su conocimiento sobre el sistema en lenguaje natural, como de la que tiene su origen en medidas empíricas y modelos matemáticos. Este es un aspecto fundamental en la era de la información en la que nos encontramos, donde el conocimiento humano y su representación e interpretación en los sistemas a desarrollar es cada vez más importante.

En definitiva, el uso de la Lógica Difusa hace posible el tratamiento de información con incertidumbre, muy común en problemas reales de clasificación, y permite el procesamiento de forma eficaz de la información experta disponible. Por otra parte, las reglas difusas permiten representar el conocimiento de una forma comprensible para los usuarios del sistema.

En el proceso de extracción de conocimiento existen distintas tareas o problemas que se pueden enfocar y resolver como problemas de optimización y búsqueda. Los *Algoritmos Evolutivos* (AEs) [BFM97, ES03], entre los que se encuentran los *Algoritmos Genéticos* [Gol89, Hol75] y la *Programación Genética* (PG) [Koz92], imitan los principios de la evolución natural para formar procedimientos de búsqueda y optimización global, lo que les convierte en herramientas especialmente adecuadas para resolver problemas presentes en las distintas etapas del proceso de descubrimiento de conocimiento [Fre02].

En procesos de extracción de reglas, los AEs tratan de forma adecuada las interacciones entre atributos porque evalúan una regla como un todo mediante la

función de adaptación, en lugar de evaluar el impacto de añadir o eliminar una condición de una regla, como ocurre en los procesos de búsqueda local incluidos en la mayoría de los algoritmos de inducción de reglas y árboles de decisión. Por ello, en los últimos años los AEs han sido ampliamente utilizados como herramienta para derivar automáticamente la totalidad o una parte de los SCBRDs, dando lugar a los denominados *Sistemas de Clasificación Basados en Reglas Difusas Evolutivos* [CHHM01, Her08, INN04].

En el diseño de un algoritmo de aprendizaje de SCBRDs, existen dos objetivos principales (y opuestos) que se deben maximizar: la precisión y la interpretabilidad del conocimiento extraido en forma de reglas difusas. Aunque inicialmente se prestó una mayor atención a la mejora de la precisión (mejora que se consiguió a consta de sacrificar la interpretabilidad de los SCBRDs), se ha puesto de manifiesto la necesidad de la existencia de un equilibrio entre la interpretabilidad y la precisión del conocimiento extraido a la hora de diseñar métodos de aprendizaje de SCBRDs [CCHM03a, CCHM03b, CHP$^+$07, INN04].

No obstante, este equilibrio es más dificil de lograr cuando el problema que ha de resolverse presenta una alta dimensionalidad, es decir, un elevado número de variables o características de entrada. El principal inconveniente viene dado por el crecimiento exponencial que se produce en el espacio de búsqueda de reglas difusas con un aumento lineal en el número de variables, lo que popularmente se conoce como el problema de la explosión combinacional de reglas [CA98]. Dicho crecimiento hace que el proceso de aprendizaje se vuelva más complicado, y en la mayoría de los casos, lleva al aprendizaje de un SCBRD que presenta un elevado nivel de complejidad (con respecto al número de reglas, y de variables y etiquetas incluidas en cada regla).

En el diseño de un SCBRD preciso, compacto e interpretable es muy importante la definición adecuada de la semántica de los conjuntos difusos asociados a las etiquetas lingüísticas. Un proceso de aprendizaje que extraiga la mejor base de reglas difusas para una partición lingüística prefijada, puede estar limitado por esta. El aprendizaje de forma simultánea tanto de la base de reglas como de la partición difusa puede permitir extraer un SCBRD con mejor comportamiento. Para que el SCBRD resultante sea lingüístico, esta definición debe ser común para todas las reglas y el proceso de aprendizaje debe verificar algunas restricciones que el modelo de representación de conjuntos difusos basado en 2-tuplas respeta. Para este aprendizaje conjunto de dos partes de un problema muy dependientes, los algoritmos coevolutivos son herramientas muy adecuadas puesto que permiten evolucionar de forma independiente ambas partes del problema.

En esta memoria se abordará el diseño de algoritmos de aprendizaje de SCBRDs compactos y precisos, es decir, que muestren un buen equilibrio entre la interpretabilidad y la precisión de conocimiento extraido, en problemas que presentan una alta dimensionalidad. Para ello haremos uso de los AEs, y en particular de la PG y de los Algoritmos Coevolutivos [Par95].

B Objetivos

El objetivo de esta memoria es estudiar el aprendizaje de SCBRDs en problemas que presentan una alta dimensionalidad (con respecto al número de variables de entrada), y desarrollar nuevos algoritmos basados en el uso de la PG y de los algoritmos coevolutivos para la extracción de conocimiento en forma de reglas difusas que presenten un alto nivel de compacticidad y precisión en problemas con una alta dimensionalidad.

Para desarrollar este objetivo general, definimos los siguientes objetivos particulares:

- Realizar una revisión de los distintos algoritmos de aprendizaje de SCBRDs existentes, prestando especial atención a aquellos que hacen uso de la PG como herramienta para extraer el conocimiento. Como el objetivo es diseñar nuevos algoritmos de aprendizaje de SCBRDs compactos y precisos en problemas con una alta dimensionalidad, el estudio de los sistemas actuales nos servirá para determinar las características más importantes en esta tarea, sus componentes fundamentales y sus objetivos.
- Analizar los problemas a resolver en el diseño de algoritmos evolutivos de extracción de reglas difusas con un buen equilibrio entre interpretabilidad y precisión en problemas con una alta dimensionalidad. Algunos de los aspectos más relevantes son la elección de un esquema de representación de reglas adecuado, el uso de algún mecanismo que mejore la diversidad dentro de la población de reglas, o el uso de medidas de calidad apropiadas para determinar la bondad de la solución.
- Desarrollar un modelo evolutivo basado en PG para el aprendizaje de SCBRDs compactos y precisos en problemas que presentan una alta dimensionalidad. Este modelo permitirá la extracción de reglas difusas en

forma normal disyuntiva (DNF, Disjunctive Normal Form) en las que cada atributo que interviene en la regla puede tomar más de un valor.

- Analizar los principales componentes del modelo desarrollado, para obtener un sistema eficaz para la tarea del aprendizaje de SCBRDs compactos y precisos en problemas que presentan una alta dimensionalidad. Para esto se aplicará el modelo a diversos conjuntos de datos de prueba que presenten una alta dimensionalidad, se estudiará la elección de algunos de sus principales componentes y se analizarán los resultados obtenidos comparándolos con los proporcionados por otros modelos de aprendizaje de SCBRDs existentes en la literatura especializada.

- Desarrollar un modelo coevolutivo para el diseño de SCBRDs compactos y precisos en problemas de alta dimensionalidad. Este modelo aprenderá no sólo el mejor conjunto de reglas difusas sino que también obtendrá la mejor definición para los conjuntos difusos asociados a las mismas, con el objetivo de mejorar los resultados obtenidos por el modelo anterior, tanto desde el punto de vista de compacticidad como de la precisión de los SCBRDs aprendidos.

C Resumen

Para abordar estos objetivos, esta memoria está dividida en cuatro capítulos cuyo contenido se describe brevemente a continuación.

En el Capítulo 1 se introducen los conceptos de extracción de conocimiento en bases de datos y minería de datos, y se describe en profundidad la tarea de inducción predictiva de clasificación. Posteriormente, se describe la computación flexible centrándonos, dentro de las distintas técnicas que la componen, en la descripción de los AEs y la Lógica Difusa. A continuación se introducen los SCBRDs, describiéndose con detalle sus distintos componentes y se presenta una revisión de las propuesta evolutivas basadas en PG para el aprendizaje de SCBRDs. Finalmente, se introduce el problema de la alta dimensionalidad en el aprendizaje de SCBRDs y se muestran las principales propuestas existentes en la literatura para abordar su solución.

En el Capítulo 2 presentamos una propuesta basada en PG para la extracción

de reglas difusas de clasificación compactas y precisas en notación DNF. Se ha realizado un análisis de diversos componentes de dicha propuesta para comprobar su adecuación. Además, se han comparado los resultados de nuestra propuesta con los obtenidos por otros algoritmos de aprendizaje de SCBRDs utilizando problemas que presentan alta dimensionalidad. Finalmente, los resultados obtenidos tanto en precisión como en compacticidad han sido validados mediante el uso de tests estadísticos no paramétricos.

En el Capítulo 3 presentamos una propuesta coevolutiva para el aprendizaje simultaneo de un conjunto de reglas difusas de clasificación compactas y precisas, y de la mejor definición de los conjuntos difusos asociados a las mismas. Para ello, se introducen el paradigma de la coevolución, así como el uso del esquema de representación de 2-tuplas lingüísticas que permite aprender los paramétros de los conjuntos difusos sin que esto suponga una perdida en la interpretabilidad del sistema. A continuación se aplica esta propuesta sobre diferentes problemas que presentan una alta dimensionalidad, y comparamos sus resultados con los del modelo propuesto en el capítulo anterior, describiendo las ventajas que aporta. Por último señalar, que al igual que en capítulo anterior, los resultados obtenidos se han validado mediante el uso de tests estadísticos no paramétricos.

En el Capítulo 4 resumimos el trabajo realizado y los resultados obtenidos en esta memoria, presentamos las conclusiones extraídas sobre los mismos, y planteamos trabajos futuros derivados de la misma.

Por último, se incluye un apéndice con la descripción de los tests estadísticos utilizados en esta memoria. La memoria termina con una recopilación bibliográfica que recoge las contribuciones más destacadas en la materia estudiada.

Capítulo 1

Sistemas de Clasificación Basados en Reglas Difusas

Uno de los problemas a los que nos enfrentamos en la actualidad es que, con el desarrollo del software y el hardware y la facilidad y disminución del costo de almacenamiento de los datos, es necesario trabajar con enormes cantidades de datos.

Por otra parte, es sabido que los datos por sí solos no producen un beneficio directo, sino que su verdadero valor radica en la posibilidad de extraer conocimiento útil para la toma de decisiones o para la exploración y comprensión del fenómeno que produjo dichos datos. Tradicionalmente, este análisis de datos se ha hecho mediante un proceso manual o semiautomático: uno o más analistas, con conocimiento de los datos y con la ayuda de técnicas estadísticas, proporcionaban resúmenes y generaban informes, o bien validaban modelos sugeridos manualmente por los expertos. Sin embargo, este proceso (en especial la generación de modelos) se torna irrealizable conforme aumenta el tamaño de los datos y el número de dimensiones o parámetros, es decir, conforme se produce un incremento en la dimensionalidad de los datos.

Por todo esto, es necesario el uso de metodologías y herramientas de extracción de conocimiento que lleven a cabo un análisis automático e inteligente de los datos [HRF04]. Este es el concepto del proceso de extracción de conocimiento

en bases de datos (*Knowledge Discovery in Databases*), el cual es un conjunto de pasos interactivos e iterativos, entre los que se incluye el preprocesamiento de los datos para corregir los posibles datos erróneos, incompletos o inconsistentes, la reducción del número de registros y/o características encontrando los más representativos, la búsqueda de patrones de interés con una representación particular, y la interpretación de estos patrones (incluso de una forma visual). De todos estos pasos, el más importante es relativo a uso de algoritmos específicos para la extracción de patrones a partir de los datos, el cual es conocido como *Minería de Datos* (*Data Mining*).

La minería de datos es un campo interdisciplinar con el objetivo general de predecir resultados y/o descubrir relaciones en los datos. Así, en función del problema que queramos resolver, la extracción de patrones se puede abordar desde dos perspectivas distintas: desde el punto de vista predictivo, en el que se intenta pronosticar el comportamiento del modelo según los datos disponibles, o desde el punto de vista descriptivo, donde se intenta descubrir patrones que describan los datos. En esta memoria, nos centraremos en aquellos procesos de la minería de datos orientados a la predicción, y más concretamente en aquellos orientados al problema de la clasificación. En un problema de clasificación, los datos son objetos caracterizados por atributos que pertenecen a diferentes clases definidas, y la meta es inducir un modelo para poder predecir la clase a la cual pertenece un objeto, dados los valores de sus atributos [WK91].

Un concepto que se puede aplicar al campo de la minería de datos, es el de la *Computación Flexible* (*Soft Computing*) que agrupa a un conjunto de técnicas, como los Algoritmos Evolutivos (AEs) o la Lógica Difusa, cuya característica principal es la tolerancia a imprecisión e incertidumbre, lo que ayuda a expresar el conocimiento extraído de forma fácilmente interpretable por el experto.

En este capítulo presentaremos los conceptos necesarios para el desarrollo del resto de capítulos de esta memoria. Comenzaremos presentando la definición de Sistema de Clasificación y su diseño a través de un proceso de aprendizaje supervisado. A continuación estudiaremos los Sistemas de Clasificación Basados en Reglas Difusas, analizando sus componentes, funcionamiento y las técnicas existentes en la literatura especializada para el aprendizaje de su Base de Conocimiento. Una vez presentadas estas técnicas, nos centraremos en aquellas propuestas que hacen uso de los AEs para el aprendizaje de Sistemas Basados en Reglas Difusas, prestando especial atención a aquellas propuestas basadas en la Programación Genética. Finalmente, introduciremos el problema de la alta dimensionalidad en el aprendizaje de Sistemas Basados en Reglas Difusas y veremos como dicho pro-

blema se ha abordado en la literatura especializada.

1.1. Extracción de Conocimiento en Bases de Datos y Minería de Datos

A nivel abstracto, la extracción de conocimiento en bases de datos es el área de la computación que incide en el descubrimiento de patrones útiles entre los datos para ayudar en la extracción de información oculta. Históricamente, la búsqueda de patrones interesantes en los datos ha recibido distintos nombres, incluyendo minería de datos, extracción de conocimiento, descubrimiento información, arqueología de datos, o procesamiento de patrones de datos. La frase extracción de conocimiento en bases de datos (*KDD, Knowledge Discovery in Databases*) fue acuñada en 1989 en la *Internacional Joint Conference on Artificial Intelligence (IJCAI-89)* [Pia91] para resaltar que el conocimiento es el producto final del descubrimiento dirigido a los datos. Se ha popularizado en los campos de la inteligencia artificial y el aprendizaje automático. El término *minería de datos* se ha usado fundamentalmente en los campos de la estadística, el análisis de datos, los sistemas de gestión de información o las bases de datos.

Aunque algunos autores utilizan KDD y minería de datos como sinónimos, actualmente se prefiere utilizar KDD para referirse al proceso completo de descubrimiento de conocimiento útil a partir de los datos, y minería de datos para referirse la etapa particular de este proceso que consiste en la aplicación de algoritmos específicos para la extracción de patrones a partir de los datos. En las siguientes subsecciones se detallan los conceptos de KDD y minería de datos.

1.1.1. Descubrimiento de Conocimiento en Bases de Datos

Se puede definir KDD como el *proceso no trivial de extracción de patrones válidos, novedosos, potencialmente útiles, y comprensibles en los datos* [FPS96a].

En esta definición, los *datos* son un conjunto de hechos (como los casos de la base datos), y un *patrón* es una expresión que describe un subconjunto de los datos o un modelo aplicable a este subconjunto. El término *proceso* implica que la extracción de conocimiento incluye distintos pasos, como la preparación los datos,

la búsqueda de patrones, la evaluación del conocimiento, y el refinamiento, todo ello repetido en múltiples iteraciones. *No trivial* implica que es necesaria cierta búsqueda o inferencia; es decir, no es un cálculo directo como puede ser calcular la media de un conjunto de números. Los patrones descubiertos deberían ser *válidos* en nuevos datos con cierto grado de certidumbre dado. Además, los patrones deben ser *novedosos*, al menos para el sistema y preferiblemente para el usuario, y potencialmente *útiles*, suponiendo algún beneficio para el usuario. Finalmente, los patrones deberían ser *comprensibles*, si no inmediatamente, si después de llevar a cabo algún tipo de post procesamiento.

Lo anterior implica que podemos definir medidas cuantitativas para evaluar los patrones extraídos. En muchos casos, es posible definir medidas de certidumbre (como la predicción predictiva estimada sobre nuevos datos) o utilidad (como la ganancia, por ejemplo en dinero ahorrado, debido a las mejores predicciones o al menor tiempo de respuesta del sistema). Ciertos conceptos como novedad y comprensibilidad son subjetivos y por tanto difíciles de plasmar en una medida cuantitativa. En ciertos contextos, la comprensibilidad se puede estimar mediante la simplicidad (por ejemplo el número de bits necesarios para describir un patrón). También suele utilizarse el interés, definido como una medida general del valor de patrón que combina validez, novedad, utilidad, y simplicidad [ST95]. Las funciones de interés se pueden definir de forma explícita o manifestarse implícitamente mediante una ordenación llevada a cabo por el sistema de KDD sobre los patrones o modelos descubiertos. Así, se puede considerar que un patrón es conocimiento si supera cierto umbral de interés, definido por el usuario y específico del dominio.

El KDD es un proceso de extracción de conocimiento que consta de un conjunto de etapas: en primer lugar, supone la utilización de bases de datos junto con algún tipo de selección, preprocesado y transformaciones de esta información; a continuación la aplicación de métodos de minería de datos para la extracción de los patrones; y por último un postprocesado que permita la visualización, evaluación y posible interpretación de los resultados de la etapa de la minería de datos para así identificar el conocimiento obtenido a través del subconjunto de patrones identificados. El proceso completo, mostrado en la Figura 1.1, es iterativo e interactivo, de forma que el experto deberá tomar decisiones en distintas fases.

De forma general, el proceso involucra los siguientes pasos [FPS96b]:

- *Comprensión del dominio de la aplicación.* En esta fase se identifica el conocimiento previo relevante que debe utilizarse, junto con las metas y

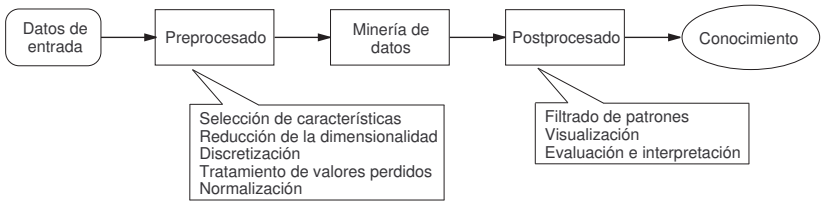


Figura 1.1: Proceso de KDD

los requerimientos del proceso de extracción de conocimiento.

- *Preparación de los datos.* En esta fase se analizan y documentan los datos disponibles y las fuentes de conocimiento del dominio de la aplicación, para estudiar las características de los datos. Además, se aplicará preprocesamiento para mejorar la calidad de los datos disponibles para el proceso de minería, incrementando la eficiencia al reducir el tiempo de cálculo necesario. Involucra cuestiones como:

  - *Limpieza de datos*: consta de ciertas operaciones básicas, como la normalización, manejo de ruido y de valores incompletos o perdidos, reducción de la redundancia, etc. Los datos de fuentes del mundo real suelen contener errores, ser incompletos y/o inconsistentes, y este tipo de datos de baja calidad deben ser limpiados antes de la etapa de minería de datos.
  - *Integración de datos*: juega un importante papel en KDD, e incluye la integración de conjuntos de datos múltiples y heterogéneos generados a partir de diferentes fuentes.
  - *Reducción y proyección de datos*: esto incluye la búsqueda de características útiles de los datos según sea el objetivo final, de forma que se puedan evaluar y desarrollar hipótesis y modelos iniciales. En esta etapa se integran la reducción del número de variables y la proyección de los datos sobre espacios de búsqueda en los que sea más fácil encontrar una solución. Esta cuestión es crítica dentro del proceso global y con frecuencia marca la diferencia entre el éxito y el fracaso del paso posterior de minería de datos.

- *Minería de Datos.* En esta etapa se seleccionan y aplican algoritmos de descubrimiento a los datos para encontrar patrones de interés. En primer

lugar será necesario elegir el tipo de algoritmo de minería de datos a aplicar en función del objetivo del proceso de KDD (clasificación, regresión, segmentación, detección de desviaciones, etc.), y la forma de representación del conocimiento (árboles de decisión, reglas, etc.). Es necesario también especificar un criterio de evaluación que permita definir qué modelo es mejor y la estrategia de búsqueda a utilizar (que normalmente viene impuesta por el algoritmo de minería).

- *Interpretación.* Supone interpretar y evaluar los resultados del proceso de minería de datos en términos del dominio de la aplicación, y realizar nuevos experimentos (regresando posiblemente a alguno de los pasos anteriores) si es necesario. Esto puede involucrar repetir el proceso, quizás con otros datos, con otros algoritmos, con otras metas y/o con otras estrategias. Es una etapa crucial para el que se requiere conocimiento del dominio. La interpretación puede beneficiarse de procesos de visualización, y sirve también para eliminar patrones redundantes o irrelevantes. En este proceso también se comprobará (y se resolverá en su caso) la existencia de conflictos potenciales con creencias previas o con conocimiento previamente extraído.

- *Utilización del conocimiento descubierto.* Este paso incluye la incorporación del conocimiento extraído al sistema, y la realización de acciones basadas en el mismo.

El proceso de KDD puede suponer varias iteraciones y puede contener bucles entre cualesquiera dos pasos. La mayor parte del trabajo previo en extracción de conocimiento se centra en el paso de minería de datos. Sin embargo, el resto de etapas es de igual importancia para la aplicación con éxito del KDD en la práctica.

1.1.2. Minería de Datos

Una vez definidas las nociones básicas e introducido el proceso de KDD, nos centramos en el componente de minería de datos, sobre el que se ha enfocado una gran parte del trabajo en este área. La minería de datos es la etapa de descubrimiento dentro del proceso de KDD, y consiste en el *uso de algoritmos concretos que generan una enumeración de patrones a partir de los datos preprocesados* [FPS96a].

Así, la minería de datos se centra en la aplicación de análisis sobre los datos, y en el desarrollo y aplicación de algoritmos que, bajo limitaciones aceptables de eficiencia computacional, obtengan patrones (o modelos) sobre los datos. Hay que resaltar que el espacio de patrones suele ser infinito, y que la extracción de patrones supone realizar algún tipo de búsqueda en este espacio. Las limitaciones computacionales prácticas establecen límites estrictos sobre el subespacio que puede ser explorado por un algoritmo de minería de datos. El nombre de minería de datos se deriva de las similitudes entre buscar información valiosa en grandes bases de datos y excavar una montaña para encontrar una veta de mineral. Ambos procesos requieren examinar una gran cantidad de material, o investigar de forma inteligente hasta encontrar exactamente dónde residen los valores de interés.

El primer paso dentro del proceso de minería de datos consiste en decidir qué técnica se va aplicar para efectuar la búsqueda de información. El siguiente paso se basa en el estudio de la calidad y validez de esa técnica aplicada al problema. Tras asegurarnos que es la adecuada, procedemos a aplicarla y a evaluar sus parámetros para ajustarlos. A continuación, obtendremos diversas soluciones del problema que serían analizadas antes de avanzar a la siguiente etapa. La evaluación se lleva a cabo en este momento para realimentar el conocimiento sobre la técnica aplicada, con la idea de refinar ajustes o corregir errores, antes de avanzar a la siguiente fase dentro del proceso de KDD.

A continuación se introducen los orígenes de la minería de datos, y se detallan los objetivos que se pueden perseguir con el proceso de minería de datos, así como los tipos de algoritmos de minería de datos que podemos utilizar en función de estos objetivos. Se describen por último los componentes algorítmicos fundamentales de los métodos de minería de datos.

1.1.2.1. Orígenes de la Minería de Datos

Los rápidos avances producidos en los últimos años en las tecnologías de almacenamiento y recolección de datos han permitido la acumulación de enormes cantidades de datos. Sin embargo, la extracción de información útil en dichos conjuntos de datos se torna una actividad compleja y llena de desafios, debido a que en numerosas ocasiones las herramientas y técnicas tradicionales de análisis datos no pueden ser utilizadas con conjuntos de datos de tan elevado tamaño.

La minería de datos surge como una tecnología capaz de mezclar los métodos

tradicionales de análisis de datos con nuevos y sofisticados algoritmos para así poder hacer frente a los retos y desafios formulados por estos grandes volúmenes de datos:

- *Escalabilidad.* Debido a los avances en la generación y recolección de datos, el encontrar conjuntos de datos con tamaños de gigabytes, terabytes o incluso petabytes es algo bastante común. Por lo tanto, los algorimtos de minería de datos deben de ser escalabes si quieren manejar estos conjuntos masivos de datos. En este sentido, muchos algoritmos de minería de datos utilizan estrategias de búsqueda especiales para manejar problemas de búsqueda exponenciales. Por otra parte, la escalabilidad también requiere de la implementación de nuevas estructuras de datos que permitan un acceso eficiente a los registros individuales. Finalmente, el uso de algoritmos parelelos o distribuidos también puede ayudar a la mejorar de la escalabidad.

- *Alta dimensionalidad.* En la actualidad, es bastante común encontrar conjuntos de datos con cientos o incluso miles de atributos. Un ejemplo se encuentra en el campo de la bioinformática, donde los progresos en la tecnología de microarrays han producido expresiones de genes que implican miles de características. Por otra parte, los conjuntos de datos que tienen componentes espaciales o temporales también tienden a presentar una alta dimensionalidad. Por ejemplo, podemos considerar un conjunto de datos que contiene las medidas de las temperaturas en diferentes localizaciones. Si las medidas de la temperatura se toman de forma repetida durante un periodo amplio de tiempo, entonces el número de dimensiones (características) se incrementará en proporción al número de medidas que se hayan tomado. Las técnicas tradicionales de análisis de datos, que fueron desarrolladas en gran medida para trabajar con datos con baja dimensionalidad, a menudo no trabajan bien con datos con alta dimensionalidad. Además, en algunos algoritmos de análisis de datos la complejidad computacional crece rápidamente a medida que la dimensionalidad (el número de características) aumenta.

- *Datos complejos y heterogeneos.* Los métodos tradicionales de análisis de datos suelen trabajar con conjuntos de datos que contienen atributos del mismo tipo, ya sean continuos o categóricos. Sin embargo, a medida que el papel de la minería de datos ha aumentado en campos tales como la medicina, la ciencia o los negocios, también ha surgido la necesidad de técnicas que puedan manejar atributos heterogeneos. Además, en los últimos

años también hemos podido asistir a la aparición de tipos de datos cada vez más complejos. Un ejemplo de estos tipos de datos no tradicionales son las colecciones de páginas Web que contienen texto semiestructurado e hipervínculos, los datos de cadenas de ADN con estructuras secuenciales y en tres dimensiones, y los datos climáticos en forma de series temporales de diferentes medidas (temperatura, presión, etc.) en varias localizaciones de la superficie de la Tierra. Las técnicas de minería de datos desarrolladas para abordar tales tipos complejos de datos deberían tener en cuenta las relaciones existentes en los datos, las autocorrelaciones temporales y espaciales, la conectividad gráfica, y las relaciones padre-hijo presentes entre los elementos en el texto semiestructurado y los documentos XML.

- *Distribución y propiedad de los datos.* En algunas ocasiones, los datos que necesitan ser analizados no se encuentran almacenados en una única localización o no pertenecen a una única organización, sino que están geográficamente distribuidos entre los recursos de diferentes entidades. Esto hace que sea necesario el desarrollo de técnicas de minería de datos distribuidas que incluyan (1) como reducir la cantidad de comunicación necesaria para llevar a cabo una computación distribuida, (2) como consolidar de forma efectiva los resultados de la minería de datos obtenidos de multiples fuentes, y (3) como tratar asuntos relativos a la seguridad de los datos.

- *Análisis no tradicional.* La propuesta estadística tradicional de análisis de datos se basa en el paradigma de hipótesis y prueba. Es decir, se propone una hipótesis, se diseña un experimento para recoger información de los datos, y entonces se analizan los datos con respecto a dicha hipótesis. Desafortunadamente, este tipo de proceso require de una elevada intervención humana. En la actualidad, el análisis de datos requiere a menudo de la generación y evaluación de miles de hipótesis, y por lo tanto es preferible es desarrollo de técnicas de minería de datos que generen y evaluen de forma automática dichas hipótesis.

Motivados por todos estos retos y desafios, los investigadores de diferentes disciplinas han desarrollado nuevas herramientas cada vez más eficientes y escalables que pueden manejar distintos tipos de datos. Todo esto ha culminado en el desarrollo del campo de la minería de datos, el cual se basa en la metodología y algoritmos empleados con anterioridad por dichos investigadores. Por ello, la minería de datos hace uso de ideas tales como (1) el muestreo, la estimación y el testeo de hipótesis estadísticas, y (2) los algoritmos de búsqueda, técnicas de

modelado, y teorías de aprendizaje procedentes de la inteligencia artificial, el reconocimiento de patrones y el aprendizaje automático. Además, la minería de datos también ha adoptado ideas procedentes de otras áreas tales como la optimización, la computación evolutiva, la teoría de la información, el procesamiento de señales, la visualización y la recuperación de información.

Otras áreas que también han jugado un papel importante dentro de la minería de datos son los sistemas de gestión de bases de datos, los cuales proporcionan el soporte para almacenar los datos de forma eficiente, para indexar dichos datos o llevar a cabo el procesado de consultas sobre los datos. En este sentido, la computación de alto rendimiento (computación paralela) también ha sido importante a la hora de poder tratar con los enormes tamaños de algunos de los conjuntos de datos. Finalmente, también es importante señalar que el uso de técnicas distribuidas también ha ayudado a abordar el problema del tamaño de los datos, y además este tipo de técnicas son esenciales para trabajar con los datos cuando estos no se encuentran almacenados en una única localización.

1.1.2.2. Tipos de algoritmos de Minería de Datos en función del objetivo

Los objetivos de la minería de datos se definen por el uso que se pretende del sistema. Podemos distinguir dos tipos de objetivos: la *verificación* y el *descubrimiento*. En la verificación, el sistema se limita a verificar las hipótesis del usuario. En el descubrimiento, el sistema encuentra nuevos patrones de forma autónoma. El descubrimiento se puede descomponer además en:

- *predicción*, donde el sistema encuentra patrones para predecir el comportamiento futuro (utiliza un conjunto de variables de la base de datos para predecir valores futuros desconocidos de otras variables de interés); y
- *descripción*, donde sistema encuentra patrones para presentarlos a un experto en una forma comprensible para él, y que describen y aportan información de interés sobre el problema y el modelo que subyace bajo los datos.

A pesar de los muchos métodos de minería de datos que se encuentran en la bibliografía, hay que resaltar que realmente sólo existen unas pocas técnicas fundamentales. El modelo de representación utilizado en un método concreto suele

ser una composición de un pequeño número de opciones bien conocidas; es decir, que muchos de los métodos se pueden ver como extensiones o híbridos de técnicas y principios básicos. De esta forma, unos algoritmos difieren de otros fundamentalmente en el criterio de calidad o el método de búsqueda utilizados para evaluar el ajuste del modelo.

Como se ha mencionado, los dos objetivos fundamentales de alto nivel en minería de datos de descubrimiento de conocimiento son predicción y descripción. A continuación se detallan los distintos tipos de tareas que se pueden realizar para ambos tipos de modelos:

- Tareas con objetivo *predictivo* [CM07]:
  - *Clasificación.* Los datos son objetos caracterizados por atributos que pertenecen a diferentes clases definidas. La meta es inducir un modelo para poder predecir la clase a la cual pertenece un objeto de datos dados los valores de los atributos [Han81, WK91]. Destaca el aprendizaje supervisado mediante reglas de clasificación [CN89, Coh95, dJHNS04, MMHL86].
  - *Regresión.* En esta tarea, la variable sobre la que se quiere hacer la predicción es continua. La meta es inducir un modelo para poder predecir un valor continuo dados los valores de los atributos [BFOS84, McL92]. Algunos algoritmos de regresión asumen que los valores de la variable objetivo se ajustan con algún tipo de función conocida (lineal, logística, etc.) y entonces determina la mejor función de este tipo que modela a los datos disponibles [Seb84, Wol75]. Sin embargo, otros modelos no asumen ninguna estructura predefinida, como por ejemplo los modelos basados en PG [Koz92] y GA-P [HD95].
  - *Análisis de series temporales.* En el análisis de series temporales [BD96], se examina el valor de un atributo según va cambiando en función del tiempo ([CCW00, LL00], entre otros). Los valores se suelen obtener como instantes de tiempo distribuidos de forma homogénea.
- Tareas con objetivo *descriptivo*:
  - *Agrupamiento.* Consiste en la separación de los datos en subgrupos o clases interesantes; se busca por tanto identificar un conjunto finito de categorías o clusters que describan los datos [JD88, TSM85]. Las

clases pueden ser o bien exhaustivas y mutuamente excluyentes, o bien jerárquicas y con solapamientos.

- *Sumarización.* Son métodos para proporcionar al usuario información comprensible para captar la esencia de grandes cantidades de información almacenadas en una base de datos [AMS+96, ZZ96]. Las técnicas de sumarización suelen aplicarse al análisis de datos interactivos y a la generación automatizada de informes.

- *Asociación.* Es el descubrimiento de relaciones de asociación o correlaciones entre un conjunto de elementos [AIS93]. Suelen expresarse en forma de reglas mostrando parejas atributo-valor que ocurren frecuentemente juntas en un conjunto de datos dado. En estos casos, se utiliza un modelo no supervisado de aprendizaje, como en [AC98, AIS93, AMS+96, WC99], por mencionar algunas, cuyo objetivo es encontrar reglas individuales que definan patrones interesantes en los datos.

- *Descubrimiento de subgrupos.* En esta tarea se lleva a cabo la búsqueda de subgrupos en el conjunto de datos que sean estadísticamente más interesantes, siendo tan grandes como sea posible y que ofrezcan el mayor valor de atipicidad estadística con respecto a la propiedad en que estemos interesados [Klö96, Wro97].

- *Detección de desviaciones, casos extremos y anomalías.* Consiste en detectar los cambios más significativos en los datos con respecto a valores pasados o normales [BC96, BN93, Klö96]. Sirve para filtrar grandes volúmenes de datos que son menos probables de ser interesantes. El problema está en determinar cuándo una desviación es significativa para ser de interés [GMV96, MP96].

- *Descubrimiento de secuencias.* El análisis o descubrimiento de secuencias [AS95, LL04] se utiliza para determinar patrones secuenciales en los datos. Estos patrones se basan en una secuencia temporal de acciones y son similares a asociaciones en las que se encuentran relaciones entre datos, pero en las que la relación está basada en el tiempo.

En esta memoria, nos centraremos en modelos de predictivos para la tarea de clasificación.

1.1.2.3. Componentes de los algoritmos de Minería de Datos

Una vez definido el objetivo de la minería de datos, para construir algoritmos específicos debemos definir tres componentes fundamentales en cualquier algoritmo de minería de datos: el lenguaje de representación del modelo, el criterio de evaluación y el método de búsqueda. Esto constituye una visión simplificada (y como tal incompleta) pero bastante útil para expresar los conceptos clave de un algoritmo de minería datos de forma relativamente unificada y compacta.

- *Lenguaje de representación del modelo.* Es el lenguaje que se utiliza para describir los patrones que se descubren. Es muy importante conocer las restricciones y suposiciones que impone la representación empleada, porque si la representación es demasiado limitada, no podremos producir un modelo adecuado de los datos aunque dediquemos mucho tiempo al proceso de minería de datos. Es importante que el analista comprenda las suposiciones sobre la representación que pueden ser inherentes a un método particular, y que el diseñador del algoritmo establezca claramente qué representación está utilizando en el algoritmo. Hay que resaltar que un mayor poder de representación para los modelos incrementa el peligro de sobreajustar los datos de entrenamiento, pudiendo obtener una precisión predictiva reducida sobre nuevos datos, lo que supone un inconveniente para los algoritmos de inducción predictiva.

- *Criterio de evaluación del modelo.* Se define en forma de función que permite establecer hasta qué punto se ajusta bien un patrón particular (un modelo y sus parámetros) a los objetivos del proceso de KDD. Por ejemplo, los modelos predictivos suelen evaluarse por la precisión predictiva empírica sobre un conjunto de prueba, utilizando técnicas de validación cruzada [Mit97]. Los modelos descriptivos se pueden evaluar a partir de la novedad, la utilidad y la comprensibilidad del modelo. Actualmente se están utilizando las curvas ROC (*Receiver Operating Characteristics*) [PF01] para la evaluación de los algoritmos en determinados tipos de problemas y condiciones.

- *Método de búsqueda.* Consta de dos componentes: la búsqueda de parámetros y la búsqueda del modelo. Una vez fijados el lenguaje de representación y el criterio de evaluación del modelo, el problema de minería datos se reduce a una tarea de optimización: encontrar los parámetros y modelos de la

familia seleccionada que optimicen el criterio de evaluación. En la búsqueda de parámetros, el algoritmo debe buscar los parámetros que optimicen el criterio de evaluación del modelo, dados los datos y un modelo fijo de representación. La búsqueda de modelos es un bucle sobre el método de búsqueda de parámetros en el que se cambia la representación del modelo para poder considerar una familia de modelos. Algunos de los métodos de búsqueda utilizados son la búsqueda exhaustiva, la vuelta atrás o la búsqueda probabilística.

1.2. Sistemas de Clasificación

La clasificación es un tipo de inducción predictiva que ha recibido gran atención por parte de los investigadores. En la clasificación los datos son objetos caracterizados por atributos que pertenecen a diferentes clases definidas, y el objetivo es inducir un modelo (un *Clasificador* o *Sistema de Clasificación*) capaz de predecir la clase a la cual pertenece un nuevo objeto dados los valores de sus atributos [Han81, WK91].

A continuación definiremos formalmente el problema de la clasificación, y haremos un repaso de algunos enfoques clásicos de diseño de Sistemas de Clasificación a través de un proceso de aprendizaje inductivo supervisado.

1.2.1. Definición

El problema de la clasificación, desde el punto de vista del aprendizaje supervisado, consiste en el establecimiento de una regla de decisión que permita determinar el valor de salida (*la clase*) de un nuevo objeto en una de las clases existentes y conocidas, $C = \{C^j \ / \ j = 1, \ldots, n_c\}$ (siendo n_c el número de valores distintos que puede tomar la clase).

Cada uno de estos objetos, a los que tambien se denomina ejemplos $E = \left\{e^h / \ h = 1, \ldots, n_e\right\}$ (siendo n_e el número de ejemplos), viene descrito mediante un conjunto de observaciones $X(e^h) = (e^h_1, \ldots, e^h_{n_v})$ (siendo n_v el número de observaciones) a las que se denomina variables, atributos o características.

El diseño de un Sistema de Clasificación se puede ver como la búsqueda de

una correspondencia

$$D : X(E) \longrightarrow C \tag{1.1}$$

optimal en el sentido de un cierto criterio $\delta(D)$ que nos determine la bondad del Sistema de Clasificación. Normalmente el objetivo final es el diseño de Sistema de Clasificación que asigne clases a cualquier punto del espacio de atributos con el mínimo error posible.

1.2.2. Diseño de un Sistema de Clasificación a través de un proceso de aprendizaje inductivo supervisado

Como se ha mencionado, el diseño de un Sistema de Clasificación a través de un *proceso de aprendizaje supervisado* parte de un conjunto de ejemplos descritos mediante un conjunto de variables o características, de los cuales se conoce su clasificación correcta. De este conjunto de ejemplos de entrenamiento, el proceso de aprendizaje inductivo extrae la información necesaria, descrita en la estructura de representación del conocimiento elegida, para la clasificación de los nuevos ejemplos en alguna de las clases determinadas previamente. Este proceso se describe en la figura 1.2.

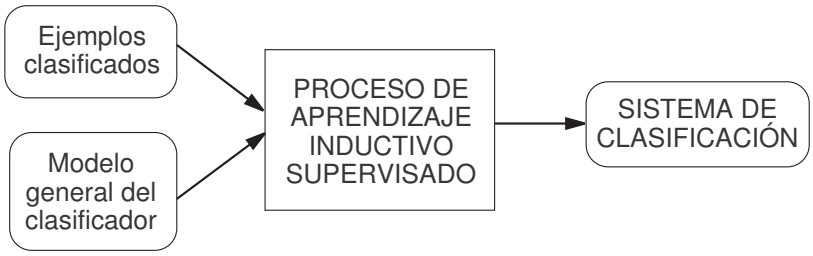


Figura 1.2: Fase de aprendizaje de un Sistema de Clasificación

El *proceso de clasificación* propiamente dicho, se realiza cuando el Sistema de Clasificación recibe un patrón de datos admisible cualquiera con clase desconocida y toma la decisión sobre la clase a la cual pertenece. Este proceso puede verse en la figura 1.3.

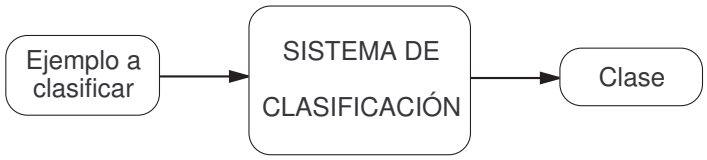


Figura 1.3: Funcionamiento de un Sistema de Clasificación

Habitualmente se realiza una estimación de la capacidad de predicción del Sistema de Clasificación con alguna técnica de estimación de error [WK91] basada en el porcentaje de acierto alcanzado por el Sistema de Clasificación sobre ejemplos de prueba no utilizados en el proceso de aprendizaje.

1.2.3. Técnicas clásicas de aprendizaje supervisado de Sistemas de Clasificación

A continuación se describen algunas de las técnicas clásicas de aprendizaje supervisado de sistemas de clasificación:

- *Redes Neuronales.* En este campo se incluye un grupo amplio de técnicas que, de forma general, obtienen Redes Neuronales consistentes en capas de nodos interconectados, en las que cada nodo produce una función no lineal de su entrada (procedente de otros nodos o directamente de la entrada de datos). Algunos de los nodos constituyen la salida de la red. De esta forma, la red al completo representa un conjunto complejo de interdependencias que puede incorporar cualquier grado de no linealidad y permite el modelado de funciones muy generales.

  La Red Neuronal más sencilla es el perceptrón. En la bibliografía especializada existen distintas propuestas para su diseño, así como el de redes multicapa más complejas [WK91, MST94].

- *Máquinas de Soporte Vectorial.* Las máquinas de soporte vectorial (*SVM*, *Support Vector Machine*) son procedimientos de clasificación y regresión basados en la teoría estadística del aprendizaje [Vap95]. Podemos definir una SVM como una clase específica de algoritmos preparados para el entrenamiento eficaz de una máquina de aprendizaje lineal en un espacio inducido por una función núcleo (o kernel), de acuerdo a unas reglas de generalización

empleando técnicas de optimización (ver [MM06] para una revisión completa).

Las dos ideas fundamentales para la construcción de un clasificador SVM son la transformación del espacio de entrada en un espacio de alta dimensión y la localización en dicho espacio de un hiperplano separador óptimo. La transformación inicial se realiza mediante la elección de una función kernel adecuada. La ventaja de trabajar en un espacio de alta dimensión radica en que las clases consideradas serán linealmente separables con alta probabilidad, y por tanto, encontrar un hiperplano separador óptimo será poco costoso desde el punto de vista computacional. Además, dicho hiperplano vendrá determinado por unas pocas observaciones, denominadas, vectores soporte por ser las únicas de las que depende la forma del hiperplano.

Una de las principales dificultades en la aplicación de este método radica en la elección adecuada de la función kernel. Es decir, construir la función de transformación del espacio original a un espacio de alta dimensión es un punto crucial para el buen funcionamiento del clasificador. La forma final de la regla de clasificación para un clasificador binario (dos clases, +1 y -1) queda como sigue:

$$f(x) = b + \sum_i \alpha_i K(x, x_i)$$

donde b y α_i son parámetros aprendidos por el clasificador durante el proceso de entrenamiento,) $K(x, x_i)$ es el valor de la función kernel para los puntos x y x_i. Si $f(x)$ es mayor que un umbral entonces la clase estimada para un punto x será "+1", y "-1" en caso contrario.

- *Árboles de decisión.* El propósito de estos métodos de clasificación es crear una estructura de árbol donde las ramas son las condiciones establecidas sobre las características de los objetos y las hojas son las clases consideradas. El proceso de clasificación comienza en la raíz y las ramificaciones del árbol se deciden a partir de las características del objeto más significativas. Dado un objeto, se desplaza a lo largo de las ramas del árbol hasta que finalmente se determina como su clase aquella que define la última hoja.

  Existen diferentes algoritmos para construir un árbol de decisión: ID3, C4.5, CART, entre otros [BFOS84, Qui93]. Sin embargo, la idea fundamental en todos ellos es la misma: los ejemplos en cada rama han de ser homogéneos,

mientras que el tamaño del árbol ha de ser pequeño (es decir, la descripción ha de ser lo más simple posible).

El algoritmo general de árbol de clasificación comienza considerando todas las posibles divisiones del conjunto inicial en subconjuntos. Se estima la calidad de cada una de las divisiones y se elige la mejor de todas ellas (la de menor entropía). El proceso se repite para cada una de las particiones realizadas, hasta que la entropía en cada uno de las hojas creadas es mejor que un valor predeterminado.

- *Redes Bayesianas.* Las redes bayesianas [Pea88] son clasificadores empleados para representar distribuciones conjuntas de modo que permitan calcular la probabilidad a posteriori de un conjunto de clases dado un conjunto de características observadas en los objetos, y así clasificar los objetos en la clase más probable. Una red bayesiana se compone de un grafo dirigido en el cual cada nodo está asociado con una característica y con una distribución de probabilidad condicional. El grafo representa la estructura y las distribuciones de probabilidad los parámetros de la red. La idea general consiste en usar una estrategia que pueda buscar de modo eficiente en el espacio de posibles estructuras y extraer aquella que dé mejores resultados de clasificación.

- *k vecinos más cercanos.* El algoritmo de clasificación de los k vecinos más cercanos (*k-NN*, *k Nearest Neighbor*) [FH51] se basa en la suposición de que los ejemplos cercanos pertenecen a la misma clase. Su fase de aprendizaje es muy simple, pues se limita a almacenar los ejemplos del conjunto de entramiento. La fase de clasificación también es simple, aunque más costosa en eficiencia. El clasificador busca los k ejemplos más cercanos al dato que se quiere clasificar y le asigna la clase más frecuente entre ellos.

En la siguiente sección abordaremos el concepto de *Computación Flexible* [Bon97] que agrupa a una serie de técnicas de mineria de datos tales como los algoritmos evolutivos o la lógica difusa, las cuales serán las herramientas que utilizaremos en esta memoria para el desarrollo de nuevos algoritmos de clasificación. Estos nuevos algoritmos aprenderán *Sistemas de Clasificación Basados en Reglas Difusas* para abordar el problema de la *alta dimensionalidad* en los datos, siendo ambos conceptos también descritos en las siguientes secciones de este capítulo.

1.3. Computación Flexible

1.3.1. Introducción

El término computación flexible fue acuñado a mediados de la década de los 90 y agrupa a un conjunto de metodologías tolerantes a imprecisión e incertidumbre, lo que le confiere una capacidad de adaptación que permite solucionar problemas en entornos cambiantes de forma robusta y con bajo costo. La definición propuesta por Zadeh [Zad94] en 1994 establece que:

> "*Básicamente, la computación flexible no es un cuerpo homogéneo de conceptos y técnicas. Es más bien una mezcla de distintos métodos que de una forma u otra cooperan desde sus fundamentos. En este sentido, el principal objetivo de la computación flexible es aprovechar la tolerancia que conllevan la imprecisión y la incertidumbre, para conseguir manejabilidad, robustez y soluciones de bajo costo. Los principales ingredientes de la computación flexible son la lógica difusa, la neuro-computación y el razonamiento probabilístico, incluyendo este último a los algoritmos genéticos, las redes de creencia, los sistemas caóticos y algunas partes de la teoría de aprendizaje. En esa asociación de lógica difusa, neuro-computación y razonamiento probabilístico, la lógica difusa se ocupa principalmente de la imprecisión y el razonamiento aproximado, la neuro-computación del aprendizaje, y el razonamiento probabilístico de la incertidumbre y la propagación de las creencias*".

Según la definición de Zadeh, la computación flexible difiere de la computación tradicional en su tolerancia a la imprecisión, a la incertidumbre y a verdades parciales. De esta forma, la computación flexible constituye "un enfoque emergente de computación, que se equipara a la destacable capacidad de la mente humana de razonar y aprender en un entorno de imprecisión e incertidumbre" [JM97].

Hasta que Zadeh dio la primera definición de computación flexible, se hacía referencia a los conceptos que maneja de manera aislada. Es importante resaltar que la computación flexible no es una simple mezcla de lógica difusa, redes neuronales, computación evolutiva y razonamiento probabilístico. Más bien es una sociedad en la que cada uno de los socios contribuye con una metodología diferente para tratar problemas en sus dominios de aplicación que, de otra forma, serían irreso-

lubles. Desde esta perspectiva, las principales contribuciones de estas técnicas son complementarias y sinérgicas más que competitivas entre ellas, conduciendo a lo que se denominan "sistemas inteligentes híbridos". El principio que subyace en la computación flexible es la hibridación de técnicas para el desarrollo de métodos computacionales que obtengan una solución aceptable a bajo costo mediante la búsqueda de una solución aproximada a un problema formulado de forma precisa o imprecisa [TT01].

A continuación se detallan las técnicas más importantes utilizadas en el ámbito de la computación flexible:

- *Lógica difusa*: la lógica difusa trata con conjuntos difusos [Zad65] y conectores lógicos para modelar los problemas de razonamiento del mundo real de la misma forma que lo hacen los seres humanos, con tratamiento de la vaguedad e incertidumbre. Un conjunto difuso, a diferencia de los conjuntos convencionales, utiliza valores de pertenencia que varían en el intervalo [0,1].

- *Redes neuronales artificiales*: las redes neuronales [Gol96] son métodos predecibles no-lineales que aprenden a través del entrenamiento y semejan la estructura de una red neuronal biológica [Zur92]. La aplicación más común de una red neuronal artificial es el aprendizaje automático, en el que se necesita un período de entrenamiento para actualizar los parámetros de la red hasta que se alcanza un estado de equilibrio.

- *Algoritmos evolutivos*: la computación evolutiva se basa en el empleo de modelos de procesos evolutivos para el diseño e implementación de sistemas de resolución de problemas. Los distintos modelos computacionales que se han propuesto dentro de esta filosofía suelen recibir el nombre genérico de algoritmos evolutivos [BFM97]. Existen cuatro tipos básicos de algoritmos evolutivos bien definidos que han servido como base a la mayor parte del trabajo desarrollado en el área: los algoritmos genéticos, las estrategias de evolución, la programación evolutiva y la programación genética. De forma simplificada, un algoritmo evolutivo se basa en mantener una población de posibles soluciones del problema a resolver, realizar una serie de alteraciones genéticas sobre las mismas y efectuar una selección para determinar qué soluciones permanecen en generaciones futuras y cuáles son eliminadas.

- *Razonamiento probabilístico*: permite trabajar con incertidumbre, uno de los principales problemas para muchas de las técnicas de minería de datos, debido al uso explícito de la teoría de la probabilidad para cuantificar

la incertidumbre. El mayor exponente de estos métodos son las redes bayesianas [Pea88], que consisten en una representación gráfica de dependencias para razonamiento probabilístico.

En el proceso de extracción de conocimiento en bases de datos y, en concreto, en el proceso de minería de datos existen distintas tareas o problemas que se pueden enfocar y resolver como problemas de optimización y búsqueda. Los algoritmos evolutivos imitan los principios de la evolución natural para formar procedimientos de búsqueda y optimización global y son aplicables tanto para el desarrollo de algoritmos de minería de datos propiamente dichos, como para el desarrollo de algoritmos de preprocesamiento o postprocesamiento o como herramientas para la optimización de los parámetros de otros algoritmos [Fre02].

Por otro lado, uno de los objetivos a considerar en minería de datos, además de la precisión predictiva y el interés, es la comprensibilidad de los resultados para el usuario. En este aspecto, la lógica difusa constituye una herramienta de representación del conocimiento que permite modelar incertidumbre e imprecisión de una forma sencilla y directamente interpretable por el usuario.

Como hemos comentado antes, además del uso de las diferentes técnicas en los dominios en los que son apropiadas, la potencia de la computación flexible está en la hibridación de unas técnicas con otras para abordar la solución a problemas que de otra forma estarían fuera de su ámbito de acción.

Uno de los enfoques más populares es la hibridación entre la lógica difusa y los algoritmos evolutivos, que da lugar a los algoritmos evolutivos de extracción de reglas difusas [CHHM01, INN04]. Un algoritmo evolutivo de extracción de reglas difusas es un sistema difuso que incluye un proceso de aprendizaje basado en un algoritmo evolutivo [ES03]. Los sistemas difusos son una de las áreas más importantes para la aplicación de la teoría de los conjuntos difusos. Por lo general se consideran estructuras de modelos en forma de Sistemas Basados en Reglas Difusas (SBRDs). Los SBRDs son una extensión de los sistemas clásicos basados en reglas, puesto que tratan con reglas de tipo "SI - ENTONCES", cuyos antecedentes y consecuentes están formados por sentencias de lógica difusa en lugar de clásicas. Han demostrado su capacidad para resolver problemas de control [PDH97], modelización [Ped96b], clasificación o minería de datos [IY04, Kun00], por mencionar algunas entre un gran número de aplicaciones.

En los últimos años ha aumentado el interés sobre los algoritmos evolutivos de extracción de reglas difusas debido a su alta potencialidad. Al contrario que las redes neuronales, el agrupamiento, la inducción de reglas y otros enfoques de

aprendizaje automático, los algoritmos evolutivos aportan una forma adecuada para codificar y evolucionar operadores de agregación de antecedentes de reglas, diferentes semánticas de reglas, operadores de agregación de bases de reglas y métodos de defuzificación. En la sección 1.4 se muestra con mayor detalle los componentes de un SBRD y se explican los aspectos necesarios para su aprendizaje a través de un proceso evolutivo.

En las siguientes secciones se describen brevemente los algoritmos evolutivos y la lógica difusa, ya que estas técnicas de la computación flexible se han utilizado en esta memoria.

1.3.2. Algoritmos Evolutivos

El paradigma de la computación evolutiva consta de algoritmos estocásticos de búsqueda basados en abstracciones de la teoría de la evolución de Darwin. Estos algoritmos se están utilizando cada vez con mayor frecuencia para reemplazar a los métodos clásicos en la resolución de problemas reales. Existen distintos tipos de algoritmos evolutivos, pero casi todos tienen ciertos elementos básicos comunes [BFM97]:

- Utilizan el proceso de aprendizaje colectivo de una población de individuos. Normalmente cada individuo representa un punto dentro del espacio de búsqueda de todas las soluciones potenciales para un problema dado, es decir, codifica una solución candidata. Los individuos pueden incorporar adicionalmente otra información, como pueden ser los parámetros de la estrategia del algoritmo evolutivo. En el área de la computación evolutiva, a la solución codificada se le denomina genotipo y a la solución decodificada (lo que realmente representa cada individuo en el contexto del problema) se le denomina fenotipo.

- Los descendientes de los individuos se generan mediante procesos no determinísticos que tratan de modelizar los procesos de *mutación* y *cruce*. La mutación corresponde a una auto-replicación errónea de los individuos, mientras que el cruce intercambia material genético entre dos o más individuos ya existentes. Ambos operadores son estocásticos, aplicándose con probabilidades definidas por el usuario. Normalmente se establece una probabilidad de mutación muy inferior a la probabilidad de cruce, ya que una

probabilidad de mutación muy elevada convertiría el proceso de búsqueda evolutivo en un proceso de búsqueda aleatoria. No obstante, la mutación es necesaria para incrementar la diversidad genética de individuos dentro de la población y para alcanzar valores de genes que no estén presentes en la población y que de otra forma serían inalcanzables, puesto que el operador de cruce solo intercambia genes (ya existentes) entre individuos.

- Se asigna una medida de calidad (denominada habitualmente medida de adaptación o *fitness*) a cada individuo mediante el proceso de evaluación. El operador de selección actúa en base a esta medida y favorece, en el proceso de reproducción, a individuos mejores respecto a aquellos con peor valor de la función de adaptación.

Se pueden identificar cuatro modelos computacionales básicos dentro de los algoritmos evolutivos:

- *Algoritmos Genéticos* [Gol89, Hol75], que modelan la evolución genética, por lo que las características de los individuos se expresan mediante genotipos. Dan más importancia al operador de cruce como elemento fundamental de búsqueda y consideran la mutación como un operador menor, aplicado normalmente con una probabilidad muy baja. En los primeros AGs los individuos se representaban mediante cadenas binarias, pero actualmente se utilizan representaciones más elaboradas, como cadenas reales, o permutaciones, entre otras.

- *Estrategias de Evolución* [Sch95], que se orientan hacia la modelización de los parámetros estratégicos que controlan la variación en la evolución, es decir, la evolución de la evolución. Suelen utilizar vectores reales para la representación de los individuos. Aunque inicialmente utilizaban la mutación como principal operador de búsqueda, actualmente se utiliza tanto la mutación como el cruce. Los individuos suelen representar tanto las variables del problema como los parámetros que controlan la evolución.

- *Programación Evolutiva* [Fog88], derivada de la simulación de comportamiento adaptativo en evolución. Fue desarrollada inicialmente para evolucionar máquinas de estado finito, pero actualmente se suele utilizar para evolucionar individuos formados por vectores reales. No utilizan en general el operador de cruce.

- *Programación Genética* [Koz92, Koz94b], basada en los algoritmos genéticos, pero en las que los individuos son programas que se representan mediante árboles. Estos individuos constan no solo de estructuras de datos, sino también de funciones (u operadores) aplicados a estas estructuras de datos.

El funcionamiento de cualquier algoritmo evolutivo se puede describir de la siguiente forma: se mantiene una población de posibles soluciones para el problema, se realizan modificaciones sobre las mismas y se seleccionan, en función de una medida de adaptación del individuo al entorno, aquellas que se mantendrán en generaciones futuras y las que serán eliminadas. La población evoluciona a través de las mejores regiones del espacio de búsqueda mediante los procesos de modificación y selección. Las modificaciones sobre la población permiten mezclar información de los padres que debe pasar a los descendientes (operador de cruce) o introducir innovación dentro de la población (operador de mutación).

Una característica importante de los algoritmos evolutivos es que realizan un proceso de búsqueda global. El hecho de trabajar con una población de soluciones candidatas en lugar de con una solución individual y utilizar operadores estocásticos reduce la probabilidad de caer un óptimo local e incrementa la probabilidad de encontrar el máximo global.

Además, este carácter de búsqueda global hace a los algoritmos evolutivos especialmente adecuados para resolver problemas presentes en las distintas etapas del proceso de descubrimiento de conocimiento [Fre02]. Por ejemplo, en procesos de extracción de reglas, los algoritmos evolutivos tratan de forma adecuada las interacciones entre atributos porque evalúan una regla como un todo mediante la función de adaptación en lugar de evaluar el impacto de añadir y/o eliminar una condición de una regla, como ocurre en los procesos de búsqueda local incluidos en la mayoría de los algoritmos de inducción de reglas y árboles de decisión. En [BFM97] se puede encontrar una descripción completa de los distintos tipos de algoritmos evolutivos.

Entre las distintas clases de algoritmos evolutivos, los AGs y la PG son los más utilizados en la actualidad en el campo de la minería de datos, y específicamente en el descubrimiento de reglas. Estas dos clases de algoritmos difieren fundamentalmente en la representación de los individuos. En los AGs, los individuos se representan como una cadena lineal de condiciones, y en el caso de reglas cada condición suele ser una pareja atributo-valor, mientras que en PG un individuo suele representarse mediante un árbol, y en este caso particular los nodos hoja o terminales son condiciones de reglas y/o valores de atributos, y los nodos internos

representan las funciones.

En esta memoria, se abordará la extracción de reglas de clasificación utilizando PG. En las siguientes secciones se describen brevemente los AGs y el paradigma de la PG.

1.3.2.1. Algoritmos Genéticos

Los AGs son algoritmos estocásticos de optimización y búsqueda inspirados en los procesos de evolución natural. Fueron definidos inicialmente por Holland [Hol75] y han sido posteriormente estudiados en profundidad por otros autores [Gol89, Mic96]. La figura 1.4, en la que $Pob(t)$ denota la población en la generación t, muestra la estructura general de un AG básico.

```
Procedimiento Algoritmo Genético
  EMPEZAR
        t = 0;
        inicializar Pob(t);
        evaluar Pob(t);
        MIENTRAS NO (condición de parada) HACER
           EMPEZAR
                 t = t + 1;
                 seleccionar Pob'(t) a partir de Pob(t − 1);
                 cruzar y mutar Pob'(t);
                 Pob(t) = Pob'(t);
                 evaluar Pob(t);
           FIN
  FIN
```

Figura 1.4: Estructura básica de un Algoritmo Genético

El funcionamiento descrito de forma sencilla, es el siguiente [Mic92]: el sistema parte de una población inicial de individuos que codifican soluciones candidatas al problema propuesto mediante alguna representación genética. Esta población de individuos (denominados cromosomas) evoluciona en el tiempo a través de un proceso de competición y variación controlada. Cada cromosoma de la población tiene asociada una medida de adaptación para determinar qué cromosomas serán seleccionados para formar parte de la nueva población en el proceso de competición. La nueva población se creará utilizando operadores genéticos de cruce y mutación. Este ciclo evolutivo continúa hasta que se verifique una determinada condición de parada como puede ser, por ejemplo, que se haya realizado un de-

terminado número máximo de evaluaciones de individuos, que la población haya evolucionado durante un número máximo de generaciones, que se haya alcanzado una solución con un determinado valor de la función de adaptación (o de parte de ella), que la población no evolucione por no generarse individuos nuevos durante un determinado número de generaciones, etc. Bäck, Fogel y Michalewicz en [BFM97] dan una descripción completa de los AGs.

En su propuesta original, Holland distingue a los AGs de otros algoritmos evolutivos por tres características [Hol75]: el esquema de codificación binario, el método de selección (proporcional a la función de adaptación) y el método básico para producir variaciones en la población (el operador de cruce). Es fundamentalmente esta tercera característica la que hace a los AGs diferentes del resto de los AEs. En propuestas posteriores a la de Holland se utilizan métodos alternativos de selección y se adoptan esquemas de codificación adaptados a los problemas a resolver y menos restrictivos que el esquema de codificación binario.

La aplicación de un AG para resolver un problema debe determinar:

- Una representación genética de las soluciones del problema.
- Una forma de crear una población inicial de soluciones.
- Una función de evaluación que proporcione un valor de adaptación de cada cromosoma.
- Operadores que modifiquen la composición genética de la descendencia durante la reproducción.
- Valores para los parámetros utilizados (como tamaño de la población, probabilidades de aplicación de los operadores genéticos, etc.).

A continuación se comentan brevemente los aspectos básicos relacionados con los AGs, como son la representación de las soluciones, el mecanismo de selección y los operadores genéticos de cruce y mutación.

Representación de las soluciones

El esquema de codificación es un factor clave en la aplicación de los AGs, ya que éstos manipulan directamente una representación codificada del problema, por lo que el esquema escogido puede limitar la forma en la que el AG afronta

el problema. Existen distintos esquemas generales de codificación entre los que destacamos los siguientes:

- *Codificación binaria*: Es la primera que se utilizó [Gol89, Hol75] y se basa en la representación de los cromosomas como cadenas de bits de modo que, dependiendo del problema, cada gen del cromosoma puede estar formado por una subcadena de varios bits.

- *Codificación real*: Es más adecuada para problemas que incluyen variables definidas sobre dominios continuos [HLV98], y se ha estudiado ampliamente en los últimos años. En este esquema de representación, cada variable del problema se asocia a un único gen que toma un valor real dentro del intervalo especificado, por lo que no existen diferencias entre el genotipo (la codificación empleada) y el fenotipo (la propia solución codificada).

- *Codificación basada en orden*: Este esquema está diseñado específicamente para problemas de optimización combinatoria en los que las soluciones son permutaciones de un conjunto de elementos determinado [Gol89, Mic96].

El mecanismo de selección

El mecanismo de selección es el encargado de seleccionar la población intermedia de individuos que formará la nueva población del AG en la siguiente generación una vez aplicados los operadores de cruce y mutación. De este modo, si notamos por Pob la población actual formada por n cromosomas, $C_1, \ldots, C_n$, el mecanismo de selección se encarga de obtener una población intermedia Pob', formada por copias de los cromosomas de Pob (véase la figura 1.5). El número de veces que se copia cada cromosoma depende de su adecuación, por lo que generalmente aquellos que presentan un valor mayor en la función de adaptación suelen tener más oportunidades para contribuir con copias a la formación de Pob'.

Existen diferentes formas de poner en práctica la selección [BS91], como la selección proporcional [Hol75], por ranking [Bak87], o por torneo [MG95], entre otras. El mecanismo de selección puede ser complementado por el modelo de selección elitista, basado en mantener un número determinado de los individuos mejor adaptados de la población anterior en la nueva población (la obtenida después de llevar a cabo el proceso de selección y de aplicar los operadores de cruce y mutación) [Gol89, Mic96].

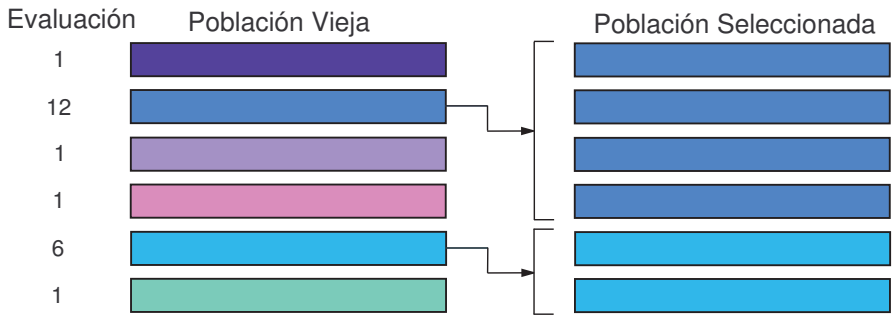


Figura 1.5: Ejemplo de aplicación del mecanismo de selección

El operador de cruce

Este operador constituye un mecanismo para compartir información entre cromosomas. Combina las características de dos cromosomas padre para obtener dos descendientes, con la posibilidad de que los cromosomas hijo, obtenidos mediante la recombinación de sus padres, estén mejor adaptados que éstos. No suele ser aplicado a todas las parejas de cromosomas de la población intermedia sino que se lleva a cabo una selección aleatoria en función de una determinada probabilidad de aplicación, denominada probabilidad de cruce (P_c).

El operador de cruce juega un papel fundamental en los AGs, puesto que debe permitir la explotación del espacio de búsqueda refinando las soluciones obtenidas hasta el momento mediante la combinación de las buenas características que presenten. Tanto la definición del operador de cruce como la del operador de mutación dependen directamente del tipo de representación empleada. Por ejemplo, para codificación binaria se suele emplear el cruce simple en un punto, en el que se selecciona aleatoriamente un punto de cruce y se intercambia el código genético de los dos cromosomas padre a partir de dicho punto (en la figura 1.6 se muestra un ejemplo de aplicación de este operador de cruce) o el cruce multipunto, que es igual que el anterior, pero utilizando dos o más puntos de cruce.

También se pueden emplear ambos operadores cuando se trabaja con el esquema de codificación real, aunque existe una serie de operadores diseñados para su uso específico con esta representación [HLV98]. Entre éstos, destacaremos una familia de operadores que manejan técnicas basadas en lógica difusa para mejorar el comportamiento del operador de cruce [HLV97].

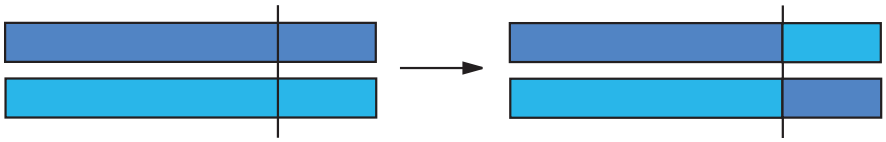


Figura 1.6: Ejemplo de aplicación del operador de cruce simple en un punto

El operador de mutación

Altera arbitrariamente uno o más genes del cromosoma seleccionado para aumentar la diversidad de la población. Se establece una probabilidad de mutación para todos los genes de los cromosomas existentes. En este caso, la propiedad de búsqueda asociada al operador de mutación es la exploración, ya que la alteración aleatoria de una de las componentes del código genético de un individuo suele conllevar el salto a otra zona del espacio de búsqueda que puede resultar más prometedora.

El operador de mutación clásicamente empleado en los AGs con codificación binaria se basa en cambiar el valor del bit seleccionado para mutar por su complementario en el alfabeto binario, tal y como recoge la figura 1.7.

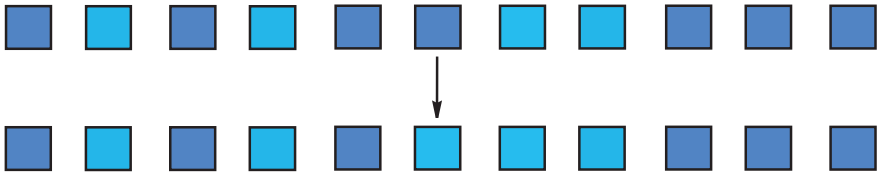


Figura 1.7: Ejemplo de aplicación del operador de mutación

Este operador puede trasladarse al campo de los AGs con codificación real, de forma que el nuevo valor del gen mutado se escoja aleatoriamente dentro del intervalo de definición asociado. Al igual que en el caso del operador de cruce, existen distintos operadores de mutación específicos para trabajar con esta codificación [HLV98, Mic96].

Modelos de población

Un elemento importante en el proceso evolutivo es el modelo de población utilizado. En la literatura se encuentran dos modelos diferentes: el modelo generacional y el modelo estacionario.

En el modelo generacional [Hol75], durante cada iteración se crea una población completa con nuevos individuos obtenidos mediante la aplicación de los operadores de cruce y mutación. Después de cada generación, se reemplaza la población completa por sus descendientes, obteniendo la siguiente generación.

En el modelo estacionario [WK88], no se modifica la población completa a la vez, sino solamente parte de ella. Durante cada iteración se escogen dos padres de la población (utilizando diferentes mecanismos de muestreo) y se les aplican los operadores genéticos. El/los descendiente/s reemplazan a uno/dos individuo/s de la población inicial. Desde su introducción en el algoritmo GENITOR [WK88], el modelo de estado estacionario se ha estudiado y empleado ampliamente. El modelo estacionario produce una alta presión selectiva (y por tanto una rápida convergencia) cuando se reemplazan los peores individuos de la población.

1.3.2.2. Programación Genética

La PG es un paradigma dentro del campo de la computación evolutiva diseñado para encontrar programas de ordenador que realicen una determinada tarea definida por el usuario. Se trata de una especialización de los AGs donde cada individuo representa un programa de ordenador (normalmente codificado como un árbol sintáctico). Por tanto, puede considerarse una técnica de aprendizaje automático usada para optimizar una población de programas de ordenador según una heurística definida en función de la capacidad del programa para realizar una determinada tarea computacional (el cálculo de una función matemática, la ejecución de un algoritmo, etc.) definida por el usuario.

Los primeros resultados de la aplicación de la PG fueron presentados por Stephen F. Smith [Smi80] y Nichael L. Cramer [Cra85]. Sin embargo, John R. Koza es considerado el padre de este paradigma, siendo quien lo aplicó a la resolución de varios problemas complejos de optimización y búsqueda [Koz89, Koz92, Koz94b].

El paradigma de la PG ha sido aplicado a una amplia variedad de dominios tales como la regresión simbólica [Koz94a], el diseño de circuitos electrónicos [KBA+97], la minería de datos [BLF00], el control de robots [KR92] o el reconocimiento de patrones [Koz94c], entre otros.

La figura 1.8 muestra la estructura básica la PG, donde $Pob(t)$ denota la población en la generación t, n es el tamaño de la población, C_1 y C_2 son dos cromosomas, y P_r, P_c y P_m son las probabilidades de los operadores de reproducción, cruce y mutación, respectivamente.

```
Procedimiento Programación Genética
  EMPEZAR
      t = 0;
      inicializar Pob(t);
      evaluar Pob(t);
      MIENTRAS NO (condición de parada) HACER
        EMPEZAR
            PARA i = 0 HASTA n HACER
              EMPEZAR
                  SEGUN operador(P_r, P_c, P_m)
                    EMPEZAR
                        CASO "Reproducción":
                          seleccionar C_1 de Pob(t);
                          copiar C_1 en Pob'(t);
                        CASO "Cruce":
                          seleccionar C_1 y C_2 de Pob(t);
                          cruzar C_1 y C_2;
                          copiar C_1 y C_2 en Pob'(t);
                          i = i + 1;
                        CASO "Mutación":
                          seleccionar C_1 de Pob(t);
                          mutar C_1;
                          copiar C_1 en Pob'(t);
                    FIN
              i = i + 1;
              FIN
            Pob(t) = Pob'(t);
            evaluar Pob(t);
            t = t + 1;
        FIN
  FIN
```

Figura 1.8: Estructura básica de la Programación Genética

A continuación, comentaremos brevemente los aspectos básicos relacionados con la PG: la representación de los cromosomas, la generación de la población inicial, el mecanismo de selección, y los operadores de cruce y mutación.

Representación de las soluciones

La mayoría de las propuestas basadas en PG emplean una estructura en árbol

para representar a los diferentes programas de ordenador que evolucionan dentro de la población de individuos. Dichos árboles están formados por una serie de primitivas elegidas entre:

- Un *conjunto de terminales* T, el cual hace referencia a los nodos situados en las hojas de los árboles, y que puede estar formado por constantes numéricas o booleanas, las variables de entrada al problema, o bien por cualquier función que no requiera argumentos.

- Un *conjunto de funciones* F, el cual se corresponde con los nodos interiores (que no son hojas) de los árboles, y que puede estar formado por cualquier función que reciba algún argumento. Algunos ejemplos de funciones son los operadores matemáticos o lógicos, las sentencias condicionales (if-then-else), o las sentencias iterativas (while), entre otros.

En la PG, los conjuntos de terminales y funciones deben elegirse de forma que estos satisfagan las dos siguientes propiedades:

- **Clausura**: Cada función en F debe estar bien definida para cualquier combinación de argumentos (funciones o terminales) que pueda encontrarse. Si la propiedad de la clausura no prevalece, las dos alternativas son:

  1. Descartar los individuos que no evalúan resultados con el dominio adecuado.
  2. Penalizar la adaptación de tales individuos.

- **Suficiencia**: El conjunto de funciones y terminales usados deben ser capaces de resolver el problema. Como resulta evidente la elección de los conjuntos de funciones y terminales afecta directamente a las soluciones que pueden ser obtenidas. Por lo tanto, el usuario de la PG debe conocer o creer que alguna composición de estas funciones y terminales puede producir una solución al problema.

En la figura 1.9 se muestra un ejemplo de un programa (en concreto una función matemática) codificado como un árbol.

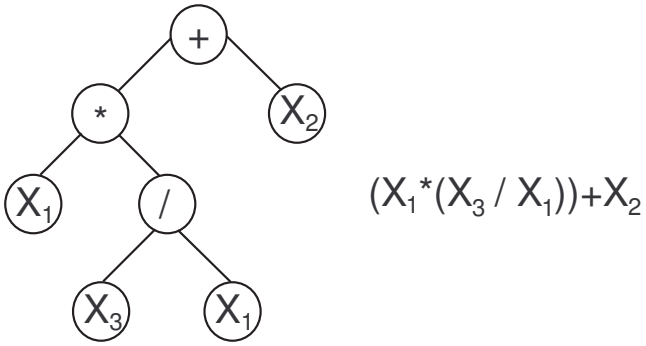


Figura 1.9: Ejemplo de un programa codificado como un árbol

La generación de la población inicial

Los cromosomas de la población inicial se inicializan mediante árboles sintácticos generados de forma recursiva y cuyos nodos son elegidos aleatoriamente de entre los elementos de los conjuntos de funciones y terminales.

Existen varios métodos para inicializar la población de cromosomas, requiriendo todos ellos que el usuario especifique el valor de la profundidad máxima que los árboles pueden alcanzar:

- *Grow*: Para cada nodo en el árbol se elige aleatoriamente un elemento de los conjuntos de funciones o terminales. Si el nodo elegido es una función, entonces para cada uno de sus hijos se vuelve a repitir el mismo proceso. Si se alcanza el límite de profundidad máximo en el árbol, el nodo sólo se puede eligir de entre los elementos del conjunto de terminales. Este método permite obtener árboles irregulares, ya que al insertar un nodo terminal evitamos que esa rama pueda seguir expandiéndose, incluso aunque no se haya alcanzado el límite máximo de profundidad en el árbol.

- *Full*: Este método solo permite elegir elementos del conjunto de funciones mientras que los nodos se encuentren a una profundidad menor que el límite máximo especificado. Los nodos que se encuentran en el límite máximo de profundidad sólo pueden ser escogidos de entre los elementos del conjunto de terminales. Por lo tanto, este método obtiene siempre árboles balanceados, donde cada rama siempre tiene la máxima longitud.

- *Ramped half-and-half*: Este método, recomendado por Koza en [Koz92], básicamente consiste en generar el 50 % de la población inicial mediante el método *grow* y 50 % restante mediante el método *full*. Además, Koza también recomienda eliminar aquellos árboles que estén duplicados en la población inicial.

El mecanismo de selección

El mecanismo de selección sólo depende del valor de adaptación de los diferentes cromosomas en la población, y por lo tanto la estrutura que estos tengan no influye en su aplicación. Por ello, en la PG se puede hacer uso de cualquier mecanismo de selección que pueda ser usado en un AG, como por ejemplo los mencionados en la sección 1.3.2.1.

El operador de cruce

Es el principal operador de búsqueda dentro de la PG. El operador de cruce estandar genera dos hijos a partir de dos padres de la siguiente forma: Se selecciona aletoriamente un nodo (que actuará de punto de cruce) en cada uno de los padres y entonces se intercambian los subárboles que cuelgan de cada uno de esos nodos (en la figura 1.10 se muestra un ejemplo de aplicación de este operador).

Koza indica en [Koz92] que la probabilidad de seleccionar como puntos de cruce aquellos nodos del árbol que representen a una función debe ser del 90 %, mientras que la de seleccionar nodos que representen un terminal debe ser del 10 %. Además, Koza también señala que si al realizar el cruce alguno de los hijos excede el límite máximo de profundidad permitido, entonces ese hijo debe ser descartado y sustituido por su correspondiente padre en la nueva población.

Por otra parte, es importante señalar que, a diferencia de como ocurre en los AGs, en la PG el cruce de un individuo consigo mismo generalmente da lugar a dos hijos totalmente diferentes, salvo que se escoja como punto de cruce el mismo nodo en ambos padres. Esto presenta algunas ventajas e incovenientes:

- El tamaño de la población puede ser pequeño y sin embargo mantener la diversidad en la población.

- La proporción de individuos con buen y mal fitness está gobernada por la operación de reproducción y el mecanismo de selección de los padres.

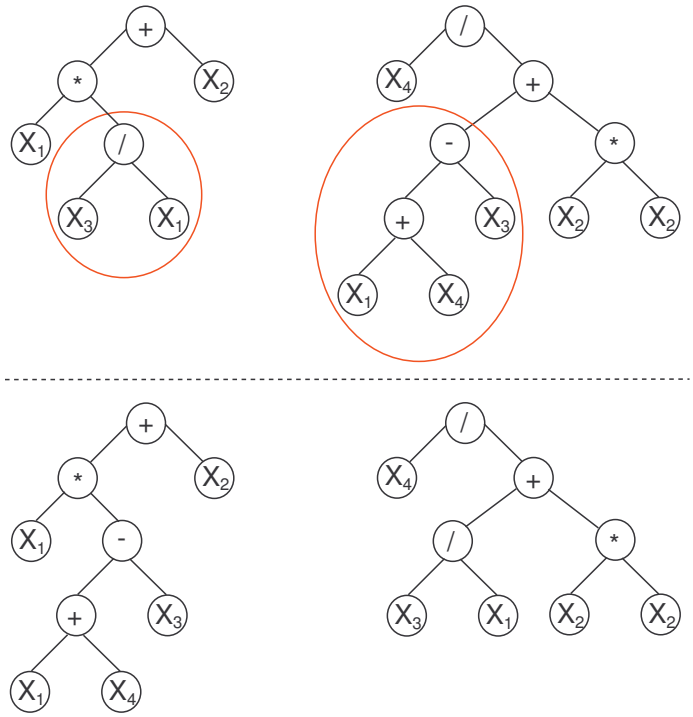


Figura 1.10: Ejemplo de aplicación del operador de cruce simple en un punto

- El cruce de dos individuos muy próximos y con un buen valor de adaptación cuando el algoritmo está proximo a converger no tiene porque necesariamente producir una solución adecuada. Como consecuencia de esto, la PG hace una buena exploración del espacio de soluciones, pero también presenta una convergencia lenta en las fases finales del algoritmo.

El operador de mutación

Este operador no tiene tanta importancia en la PG como en los AGs, ya que como se ha indicado anteriormente la diversidad en la población se mantiene gracias a las características del operador de cruce. Este operador puede actuar de dos formas distintas:

1. Se elige un nodo aleatoriamente en el árbol y este se reemplaza por otro nodo

del mismo tipo: si el nodo codificaba una función entonces se reemplaza por otra función del conjunto F, y si codificaba un terminal entonces por otro terminal del conjunto T.

2. Se elige un nodo aleatoriamente en el árbol y entonces el subárbol que cuelga de él se reemplaza por otro subárbol generado alteatoriamente mediante el método *grow*. Un ejemplo de este tipo de mutación se muestra en la figura 1.11.

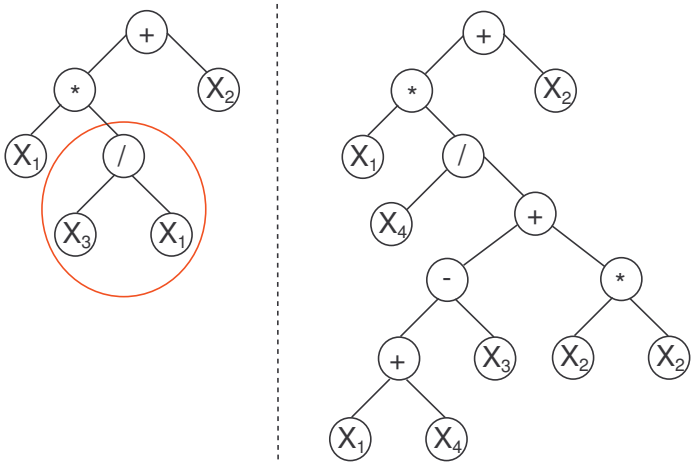


Figura 1.11: Ejemplo de aplicación del operador de mutación

1.3.3. Lógica Difusa

La lógica difusa permite modelar conocimiento impreciso y cuantitativo, así como transmitir y manejar incertidumbre y soportar, en una extensión razonable, el razonamiento humano de una forma natural. Desde que Zadeh propuso la teoría de conjuntos difusos [Zad65] y el concepto de variable lingüística [Zad75], se ha aplicado en múltiples áreas de investigación, fundamentalmente por su cercanía al razonamiento humano y por proporcionar una forma efectiva para capturar la naturaleza aproximada e inexacta del mundo real.

La forma de representación del conocimiento ha sido una de las áreas de mayor interés investigadas en la disciplina de las ciencias de la computación y la inteligencia artificial. Uno de los principales aspectos es la representación de conocimiento lingüísticamente impreciso, para lo que se han demostrado ineficaces las técnicas convencionales. Así, el desarrollo de la lógica difusa se vio motivado por la necesidad de disponer de un marco conceptual que pudiera ser aplicado con éxito al tratamiento de la información en entornos de incertidumbre e imprecisión léxica [Zad92].

Podemos considerar que la lógica difusa es una extensión a la lógica clásica, en la que se incorporan nuevos conceptos para trabajar con el problema de representación en un ambiente de incertidumbre e imprecisión. La diferencia fundamental entre las proposiciones de la lógica clásica y las proposiciones difusas está en el rango de valores de verdad. Mientras que en las proposiciones clásicas sólo existen dos posibles valores de verdad (verdadero o falso), el grado de verdad o falsedad de las proposiciones difusas puede tomar distintos valores numéricos. Asumiendo que la verdad y la falsedad se representan con 1 y 0 respectivamente, el grado de verdad de cada proposición difusa se expresa como un valor en el intervalo [0,1]. La lógica difusa es, en realidad, una forma de lógica multivaluada. Su finalidad última es aportar una base para el *razonamiento aproximado* con proposiciones imprecisas utilizando la teoría de conjuntos difusos como herramienta principal.

La lógica difusa se basa en el concepto de conjunto difuso, que puede definirse como una generalización de los conjuntos clásicos. En éstos, la función de pertenencia sólo puede tomar dos valores (0 ó 1, pertenencia o no pertenencia). En cambio, en los conjuntos difusos, la función de pertenencia asigna a cada elemento un grado de pertenencia dentro del intervalo [0,1]. Esto permite representar conceptos con límites borrosos, pero cuyo significado sí está definido de forma completa y precisa. En [KY95] se puede encontrar una descripción detallada sobre teoría de conjuntos difusos.

Un conjunto difuso es una entidad más compleja que un conjunto clásico pero constituye una representación con mayor precisión para conceptos reales. Esta expresividad permite simplificar las reglas y los sistemas basados en ellos.

Una partición difusa de una variable determina una distinción de niveles en los valores de la misma mediante conjuntos difusos y permite solapamiento en las fronteras, al igual que ocurre en el razonamiento humano cuando trabajamos con valores lingüísticos [Zad75]. Así, una variable se describe mediante distintos términos lingüísticos de los que se utilizan habitualmente en el razonamiento humano

con variables numéricas (por ejemplo *Bajo*, *Medio*, y *Alto*). Cuando utilizamos estos conceptos, mentalmente pensamos en ellos con un cierto solapamiento entre algunos términos. De ahí que, cuando decidimos trabajar con una variable considerándola una variable *lingüística* que toma como valores términos *lingüísticos*, definimos una partición difusa que considere siempre una división del dominio en términos lingüísticos con cierto nivel de solapamiento. El significado de cada término viene especificado por un conjunto difuso y por tanto por una función de pertenencia. Esta función de pertenencia determinará -de forma precisa- el grado de pertenencia al conjunto difuso correspondiente para cualquier valor de la variable.

Los sistemas basados en reglas que utilizan conjuntos difusos para describir los valores de sus variables se denominan sistemas basados en reglas difusas. Ante una situación dada, la regla se podrá aplicar si el antecedente de la misma describe la zona del espacio a la que pertenece el ejemplo, es decir, si el grado de compatibilidad del antecedente de la regla y el ejemplo es mayor que cero. Este grado se calcula de la siguiente forma:

- Para cada variable se calcula el grado de pertenencia al conjunto difuso correspondiente. Si en la regla, para una variable se indica una disyunción de términos (por ejemplo, *"Nivel = Bajo O Muy Bajo"*), esto se interpreta como la unión de los conjuntos difusos correspondientes y el grado de pertenencia se calcula con una t-conorma (operador de unión difusa) que habitualmente es el máximo, aunque existen otras definiciones para el mismo. Según esto, el grado de pertenencia de un valor actual de la variable nivel a *"Bajo O Muy Bajo"* será igual al máximo entre el grado de pertenencia del valor actual del nivel a *"Bajo"* y el grado de pertenencia a *"Muy Bajo"*.

- Habitualmente el antecedente de una regla difusa está formado por una conjunción de condiciones para las distintas variables. En este caso, el grado de compatibilidad del ejemplo con la regla se calcula con una t-norma (operador de intersección difusa). Existen múltiples expresiones posibles para las t-normas, pero es frecuente el uso del operador mínimo o producto. Si utilizamos el mínimo, el grado de compatibilidad se calcula como el mínimo entre los grados de pertenencia de las variables implicadas en el antecedente a los conjuntos difusos correspondientes. Como se puede observar, si se elige la t-norma mínimo o producto, cuando una de las variables no pertenece al conjunto difuso implicado, el grado de compatibilidad del ejemplo con la regla es cero.

- Si en el antecedente las condiciones para las variables se combinan con el operador de disyunción (por ejemplo, *"Nivel = Bajo O Edad = Joven"*), se utilizará el operador de unión difusa (t-conorma) para el cálculo del grado de compatibilidad.

Cuando en el antecedente de las reglas sólo hay conjunciones de condiciones formadas por pares variable/valor, éstas se denominan *reglas canónicas*. Por ejemplo:

```
SI (Zona = Norte Y Mejora imagen = Media Y Distintas
alturas = No) ENTONCES Eficiencia = Baja
```

Sin embargo, si en estas condiciones aparecen disyunciones en los valores de cada variable, se denominan *reglas en forma normal disyuntiva* (*DNF*, *Disjunctive Normal Form*). Por ejemplo:

```
SI (Zona = (Norte O Sur) Y Mejora imagen = (Media O Alta)
Y Diferentes alturas = No) ENTONCES Eficiencia = Baja
```

Las reglas difusas se pueden considerar modelos locales simples, lingüísticamente interpretables y con un rango de aplicación muy amplio. Permiten la incorporación de toda la información disponible en el modelado de sistemas, tanto de la que proviene de expertos humanos que expresan su conocimiento sobre el sistema en lenguaje natural, como de la que tiene su origen en medidas empíricas y modelos matemáticos.

Los sistemas difusos se pueden clasificar en dos familias. La primera incluye modelos lingüísticos basados en colecciones de reglas SI-ENTONCES, cuyos antecedentes y consecuentes utilizan valores difusos. Utilizan razonamiento difuso, y el comportamiento del sistema se puede describir en términos *naturales*. El modelo *Mamdani* [MA75] cae dentro de este grupo. El segundo grupo, basado en sistemas de tipo *Sugeno* [TS85] utilizan una estructura de regla con antecedente difuso y consecuente *funcional*. Este enfoque aproxima un sistema no lineal mediante una combinación de varios sistemas lineales, descomponiendo el espacio de entrada en varios espacios parciales difusos y representando cada espacio de salida con una ecuación lineal. Estos modelos son capaces de representar tanto información cualitativa como cuantitativa y permiten una aplicación relativamente sencilla de potentes técnicas de aprendizaje para su identificación a partir de los

datos. Pueden aproximar cualquier función continua sobre un conjunto compacto con cualquier grado de precisión [BF99].

En estos sistemas, es necesario establecer un compromiso entre legibilidad y precisión. Si nos interesan soluciones más precisas, no nos preocuparemos tanto de la interpretabilidad lingüística. En estos casos, los sistemas de tipo Sugeno son más adecuados. En otro caso, la opción será un sistema de tipo Mamdani.

Es necesario destacar que no todos los sistemas difusos son sistemas basados en reglas difusas. Los conjuntos difusos se utilizan también, por ejemplo, en algoritmos de agrupamiento con el objetivo de obtener conjuntos difusos que permitan diferenciar los grupos con mayor expresividad, o también en algoritmos de modelado para predicción o regresión. Por ejemplo, se puede obtener un modelo que utilice conjuntos difusos para representar el conocimiento extraído, pero no necesariamente mediante reglas difusas.

En minería de datos, uno de los aspectos que determinan la calidad del conocimiento extraído, y por tanto del algoritmo utilizado, es la interpretabilidad del mismo y en este sentido los sistemas basados en reglas difusas lingüísticas son los más utilizados dentro del campo de la lógica difusa. Podemos destacar como motivos para el uso de la lógica difusa en minería de datos los siguientes:

- Cualquier proceso de minería de datos tiene como objetivo principal la identificación de patrones interesantes y la descripción de los mismos de una forma concisa y con significado [FPS96a]. Los modelos difusos representan una descripción de los datos orientada al usuario a través de un conjunto de reglas cualitativas que establecen relaciones significativas y útiles entre variables. Los conjuntos difusos permiten establecer límites flexibles entre los distintos niveles de significado, sin ignorar ni enfatizar en exceso los elementos cercanos a las fronteras, de igual forma que ocurre en la percepción humana. En todo proceso de extracción de conocimiento hay un componente de interacción humana y los conjuntos difusos permiten representar el conocimiento de forma lingüística, incorporar conocimiento previo de forma fácil y proporcionar soluciones interpretables.

- En muchos algoritmos de minería de datos y, especialmente, en la evaluación de los patrones extraídos, aparece el concepto de interés [Fre99], que agrupa distintas características como validez, novedad, utilidad y simplicidad, y que puede cuantificarse de manera muy conveniente a través de conjuntos difusos. De forma adicional, algunos algoritmos incorporan, como parámetro

adicional de interés, una medida de diferenciación difusa entre un patrón descubierto y un vocabulario definido por el usuario.

Los conjuntos difusos manejan de forma natural conocimiento de dominio lingüístico y permiten la obtención de soluciones más interpretables, por lo que la teoría de conjuntos difusos tiene una especial importancia en el ámbito de la minería de datos [Yag96]. Se han implementado diversos visualizadores de datos utilizando la teoría de conjuntos difusos [Bal96] para facilitar el manejo simultáneo de diferentes tipos de variables (datos categóricos, simbólicos y numéricos) en el análisis de datos del mundo real en minería de datos. Pedrycz [Ped98] discute algunos recursos computacionales constructivos y dirigidos por conjuntos difusos y establece la relación entre la minería de datos y la modelización difusa.

La lógica difusa se ha aplicado en tareas de agrupamiento [Ped96a, RL99], clasificación [CdJH98, CdJH99, CYP96, INYT94, Jan98, Kun00], reglas de asociación [AC99, CW02, FWS$^+$98, HKC99], dependencias funcionales [BPU99] o sumarización de datos [CCW00].

En la siguiente sección nos centraremos en la aplicación de los AEs, y más concretamente de la PG, y lógica difusa para extracción de conocimiento en forma de reglas difusas.

1.4. Sistemas de Clasificación Basados en Reglas Difusas

1.4.1. Definición

Un Sistema de Clasificación Basado en Reglas Difusas (al que en lo sucesivo denominaremos SCBRD) está compuesto por una *Base de Conocimiento* (BC) y un *Método de Razonamiento Difuso* (MRD) que, utilizando la información de la BC, determina una clase para cualquier patrón de datos admisible que llegue al sistema. Esta estructura se refleja en la figura 1.12.

1.4.1.1. La Base de Conocimiento

La BC está formada por dos componentes:

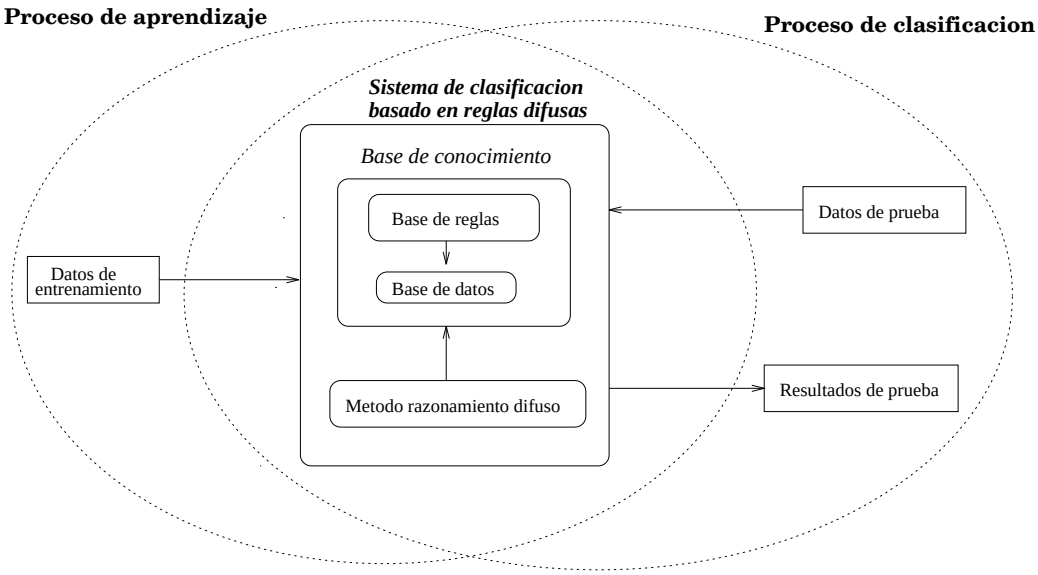


Figura 1.12: Diseño de un SCBRD (aprendizaje y clasificación)

- La *Base de Datos* (BD), que contiene la definición de los conjuntos difusos asociados a los términos (o etiquetas) lingüísticos utilizados en la Base de Reglas.

  Cada variable i en el problema tendrá asociada una partición difusa de su dominio de definición, la cual representa los distintos conjuntos difusos asociados a cada uno de los términos lingüísticos ($L_i^m \ / \ m = 1, \ldots, l_i$). En la figura 1.13 se muestra un ejemplo de una partición difusa de Ruspini con cinco etiquetas lingüísticas y conjuntos difusos triangulares.

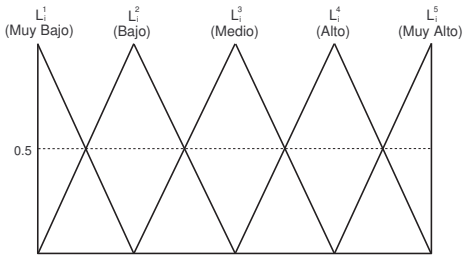


Figura 1.13: Ejemplo de partición difusa

- La *Base de Reglas* (BR), formada por un conjunto de reglas de clasificación $R = \left\{R^k \ / \ k = 1, \ldots, n_r\right\}$ (siendo n_r el número total de reglas que forman la BR) de uno de los tipos siguientes utilizados en la literatura especializada para SCBRDs [CdJH99]:

  (a) **Reglas difusas con una clase en el consecuente**

$$R^k : \text{Si } X_1 \text{ es } A_1^k \text{ y } \ldots \text{ y } X_{n_v} \text{ es } A_{n_v}^k \quad \text{entonces } Clase \text{ es } C^k \tag{1.2}$$

  donde $X_1, \ldots, X_{n_v}$ son las variables asociadas a los diferentes atributos del sistema de clasificación, $A_1^k, \ldots, A_{n_v}^k$ son las etiquetas lingüísticas utilizadas para discretizar los dominios continuos de las variables ($A_i^k = L_i^m / \ m = 1, \ldots, l_i$), y $Clase$ es la variable que indica la clase C^k a la cual pertenece el patrón.

  (b) **Reglas difusas con una clase y un grado de certeza asociado a la clasificación para esa clase en el consecuente**

$$R^k : \text{Si } X_1 \text{ es } A_1^k \text{ y } \ldots \text{ y } X_{n_v} \text{ es } A_{n_v}^k \quad \text{entonces } Clase \text{ es } C^k \text{ con } GC^k \tag{1.3}$$

  donde GC^k es el grado de certeza asociado a la clasificación en la clase C^k para los ejemplos pertenecientes al subespacio difuso delimitado por el antecedente de la regla.

  (c) **Reglas difusas con grados de certeza asociados a cada una de las clases en el consecuente**

$$R^k : \text{Si } X_1 \text{ es } A_1^k \text{ y } \ldots \text{ y } X_{n_v} \text{ es } A_{n_v}^k \quad \text{entonces } (GC^{k1}, \ldots, GC^{kn_c}) \tag{1.4}$$

  donde GC^{kj} es el grado de fuerza de la regla R^k para predecir la clase C^j para un ejemplo perteneciente a la región difusa representada por el antecedente de la regla.

El último modelo de regla extiende a los dos anteriores empleando distintos valores para los parámetros del vector $(GC^{k1}, \ldots, GC^{kn_c})$. Así, con

$$GC^{kv} = 1, \quad GC^{kj} = 0, \quad j \neq v, \quad j = 1, \ldots, n_c$$

tenemos una regla difusa tipo (a), y con

$$GC^{kv} = GC^{k}, \quad GC^{kj} = 0, \quad j \neq v, \quad j = 1, \ldots, n_c$$

una regla tipo (b).

En algunas propuestas de SCBRDs se utilizan reglas con consecuente difuso, es decir, reglas difusas en las que el consecuente es una etiqueta lingüística definida a través de un conjunto difuso. Este tipo de reglas, en un Sistema de Clasificación, requieren un proceso de defuzzificación adicional que permita la obtención de un valor nítido de clasificación, y no serán consideradas en esta memoria.

Por otra parte, en esta memoria estamos especialmente interesados en el aprendizaje de SCBRDs compactos, que presenten una alta interpretabilidad. En este sentido, existe un tipo de representación de reglas que permite obtener SCBRDs con un alto grado de legibilidad al proporcionar una descripción más compacta de las relaciones difusas, especialmente en el antecedente de las reglas. A las reglas obtenidas mediante este tipo de representación se las denomina reglas en *forma normal disyuntiva* (o reglas DNF), las cuales tienen la siguiente estructura:

$$\begin{array}{c} R^k : \text{Si } X_1 \text{ es } \hat{A}_1^k \text{ y } \ldots \text{ y } X_{n_v} \text{ es } \hat{A}_{n_v}^k \\ \text{entonces } (GC^{k1}, \ldots, GC^{kn_c}) \end{array} \tag{1.5}$$

donde cada variable de entrada X_i toma como valor un conjunto de etiquetas lingüisticas $\hat{A}_i^k = \left\{ L_i^1 \ o \ \ldots \ o \ L_i^{l_i} \right\}$ unidas por el operador de disjunción lógica (comúnmente conocido como operador o).

El uso de reglas DNF permite cambios en la granularidad del sistema al combinar terminos lingüísticos mediante el operador o, y por su naturaleza permite la ausencia de alguna de las variables de entrada en cada regla (simplemente haciendo que $\hat{A}_i^k$ sea igual al todo el conjunto de etiquetas lingüísticas de la variable i).

Finalmente, es importante señalar que la representación en forma normal disyuntiva puede combinarse con cualquiera de los tres tipos de reglas vistos anteriormente, habiéndose elegido en la expresión (1.5) una regla del tipo (c) por ser esta la más general. No obstante, es importante señalar que para las propuestas descritas a lo largo de esta memoria utilizaremos reglas DNF del tipo (b).

1.4.1.2. El Método de Razonamiento Difuso

El MRD es un procedimiento de inferencia que utiliza la información de la BC para predecir una clase ante un ejemplo no clasificado. La potencia del razonamiento difuso permite obtener un resultado (una clasificación) incluso cuando no tenemos una compatibilidad exacta (con grado 1) entre el ejemplo y el antecedente de las reglas.

Tradicionalmente, en la literatura especializada [INT92, CYP96] se ha utilizado el MRD del máximo, también denominado MRD clásico o de la regla ganadora, que considera la clase indicada por una sóla regla, aquella con la que el ejemplo tiene mayor grado de compatibilidad (calculando este a través de alguna t-norma aplicada al grado de pertenencia del ejemplo a los distintos conjuntos difusos presentes en el antecedente). Sin embargo, también es posible encontrar otros MRDs que combinan la información aportada por todas las reglas que representan el conocimiento de la zona a la que pertenece el ejemplo [CdJH99]. En esta memoria utilizaremos, además del MRD clásico, el MRD de la suma normalizada (o de combinación aditiva) de los grados de asociación de los consecuentes de las reglas para cada clase, ya que estos son los MRDs que mejores resultados obtienen habitualmente en clasificación [CdJH99, INM99a].

A continuación se presenta el modelo general de razonamiento difuso que combina la información proporcionada por las reglas difusas compatibles con el ejemplo. En el proceso de clasificación del ejemplo $e = (e_1, \ldots, e_{n_v})$, los pasos del modelo general de un MRD son los siguientes:

1. Calcular el grado de emparejamiento del ejemplo con el antecedente de las reglas.
2. Calcular el grado de asociación del ejemplo a la clase consecuente de cada regla mediante una función de agregación entre el grado de emparejamiento y el grado de certeza de la regla con la clase asociada.
3. Determinar el grado de asociación del ejemplo con las distintas clases.

4. Clasificación. Para ello aplicaremos una función de decisión F sobre el grado de asociación del ejemplo con las clases, que determinará, en base al criterio del máximo, la clase C^v a la que corresponda el mayor valor.

En el punto (3) es donde se distinguen los dos métodos usados en esta memoria, esto es, utilizar la función del máximo para seleccionar la regla con mayor grado de asociación para cada clase, y utilizar el funcional suma sobre los grados de asociación de las reglas asociadas a cada clase.

1.4.2. Tareas en el aprendizaje de Sistemas de Clasificación Basados en Reglas Difusas

El aprendizaje de un SCBRD mediante un proceso de aprendizaje inductivo supervisado comienza con un conjunto de ejemplos correctamente clasificados y tiene como objetivo final la obtención de un SCBRD que determine, con el mínimo error posible, la clase para un conjunto de ejemplos no clasificados. Una vez obtenido el SCBRD, se determina su rendimiento sobre datos de prueba para tener una estimación del error real del SCBRD. Este proceso se describe en la figura 1.12.

El análisis de los componentes y del funcionamiento de un SCBRD nos permite determinar que el proceso de aprendizaje implica principalmente la realización de dos tareas complementarias entre sí:

1. el establecimiento de las particiones difusas para los dominios de las variables del problema, y

2. la generación de un conjunto de reglas difusas.

Ambas tareas se engoblan dentro del proceso de generación de la BC, el cual debe obtener un conjunto de reglas difusas que representen, de la forma más precisa posible, las relaciones entre las variables del problema de clasificación y permitan la clasificación correcta de nuevos ejemplos que se presenten al sistema.

La primera de las tareas consiste en la obtención del conjunto de etiquetas lingüísticas (y conjuntos difusos asociados) óptimo para cada variable. Este es un problema difícil que se ha afrontado de diferentes formas en la literatura especializada:

- Estableciendo una partición difusa fija para cada variable en base a conocimiento experto o a una división equidistante del dominio con un tipo de función de pertenencia (triangular, trapezoidal, gaussiana, ...) preestablecida o determinada tras una experimentación. Algunas propuestas clásicas donde es posible encontrar este enfoque son la extensión a problemas de clasificación del proceso de generación de Wang y Mendel propuesto por Chi y otros en [CYP96] o en el método de generación desarrollado por Ishibuchi y otros en [INT92].

- Utilizando técnicas de agrupamiento que determinan grupos de datos con determinadas relaciones entre las características y construyendo los conjuntos difusos mediante proyecciones. Algunos métodos dentro de esta línea de trabajo se pueden encontrar en [CC04, CC05, BM07].

- Utilizando distintas particiones difusas para cada variable, que se diferencian en el número de términos utilizados, como hacen Ishibuchi y otros [INT94, IYN05].

Con respecto a la segunda de las tareas, la literatura especializada también muestra que es posible utilizar diferentes métodos para la generación de un conjunto de reglas difusas:

- Métodos iterativos que

  - exploran el conjunto de ejemplos de entrenamiento y determinan la mejor regla para cada ejemplo [CYP96], o
  - exploran el espacio de patrones y determinan la mejor regla para cada subregión del espacio determinada por la partición difusa [INT92].

- Métodos basados en Redes Neuronales (también conocidos como métodos o sistemas neurodifusos) [NK97, CFM02, CP04].

- Métodos basados en técnicas de agrupamiento [AL95, AT97, CF00, SR00, WL02].

- Métodos basados en la derivación de reglas difusas a partir de árboles de decisión [CY96, ARS03, MJG05].

- Métodos basados en Algoritmos Evolutivos: La generación de un conjunto de reglas difusas se puede tratar como un problema de optimización o búsqueda. Es bien sabido que los AEs y, particularmente los AGs, son técnicas de búsqueda que han mostrado de forma teórica y empírica capacidad robusta de búsqueda en espacios complejos. Por esta razón, y aunque los AEs no se pueden considerar algoritmos de aprendizaje, los AEs constituyen una herramienta de optimización robusta, independiente del dominio y aplicable a esta y otras tareas que componen el aprendizaje de SCBRDs.

El aprendizaje de los distintos componentes de un SCBRD mediante AEs a dado lugar a los denominados Sistemas de Clasificación Basados en Reglas Difusas Evolutivos (SCBRDEs), los cuales se presentan a continuación.

1.4.3. Sistemas de Clasificación Basados en Reglas Difusas Evolutivos

En los últimos años, la investigación en el campo de los sistemas difusos ha evolucionado hasta dar lugar a un marco de trabajo más general que contempla la integración de la Lógica Difusa con otras técnicas tales como los AEs [Bäc96], las Redes Neuronales y el Razonamiento Probabilístico. Esta nueva área ha recibido el nombre de computación flexible.

En concreto, la combinación de la Lógica Difusa y los AEs ha obtenido unos resultados muy prometedores en los últimos años [CAAFR07, CC09, CGH$^+$04, CHHM01, CHP$^+$07, CP07, Her08, INN04]. La definición automática de un sistema difuso se puede afrontar como un proceso de optimización o búsqueda, y los AEs, en particular los AGs, están considerados en la actualidad como la técnica de búsqueda global más conocida y empleada. Además, la codificación genética que emplean les permite incorporar conocimiento a priori de una forma muy sencilla y aprovecharlo para guiar la búsqueda. Todas estas razones han incrementado el empleo de los AEs para el diseño de sistemas difusos a lo largo de los últimos años, lo que ha dado lugar a la creación de los denominados *sistemas difusos evolutivos* [CHHM01, Her08].

Dentro de los sistemas difusos evolutivos, los más conocidos y estudiados son los denominados SBRDEs [CHHM01, Her08], en los que se hace uso de los AEs para obtener de un modo automático la totalidad o una parte del SBRD. En [Her08], podemos encontrar una taxonomía de los diferentes tipos de SBRDEs

existentes en la literatura especializada, según las partes del SBRD que se codifiquen dentro del modelo evolutivo. En esta memoria, estamos interesados especialmente en aquellos SBRDEs diseñados para el aprendizaje (total o parcial) de la BC. En este tipo de SBRDEs, uno de los aspectos de diseño más importantes es la elección del esquema de codificación de las reglas difusas dentro de la población de individuos. En concreto, las diferentes propuestas en la literatura especializada siguen uno de estos enfoques:

- El enfoque "Cromosoma = Conjunto de reglas", también llamado enfoque *Pittsburgh*, en el que cada individuo representa un conjunto completo de reglas, de forma que los individuos compiten entre si a lo largo del proceso evolutivo [Smi80]. Thrift propone en [Thr91] un método pionero que sigue este enfoque de codificación.

- El enfoque "Cromosoma = Regla", en que cada individuo codifica una única regla y el conjunto completo de reglas se obtiene al combinar varios individuos de la población (cooperación entre reglas) o bien al combinar los individuos obtenidos en varias ejecuciones del proceso evolutivo (competición entre reglas). De hecho, podemos encontrar tres propuestas genéricas con este tipo de enfoque:

  - El enfoque *Michigan*, en el que cada individuo codifica una única regla. Los sistemas que utilizan este enfoque se denominan habitualmente sistemas clasificadores. Son sistemas de paso de mensajes, basados en reglas, que utilizan aprendizaje por refuerzo y un AG para aprender las reglas guiando su ejecución en un entorno dado [Kov04]. Casillas y otros proponen en [CCB07] un método que sigue este enfoque de codificación, el cual extiende el algoritmo clásico XCS [Wil95] para aprender reglas difusas.
  - El enfoque *IRL (Iterative Rule Learning)*, en el que cada cromosoma representa una regla, y la solución global está formada por las mejores reglas obtenidas al ejecutar el algoritmo en sucesivas ocasiones. 2SLAVE [GP01] y MOGUL [CdJHL99] son ejemplos de propuestas que siguen este enfoque.
  - El enfoque *cooperativo-competitivo (GCCL)*, en que la toda la población, o un subconjunto de ella, codifica la BR. En este modelo, los cromosomas compiten y cooperan de forma simultanea. Ishibuchi

y otros proponen en [INM99b] un método que usa este enfoque de codificación.

1.4.4. Aprendizaje de Sistemas Basados en Reglas Difusas Evolutivos mediante Programación Genética

Aunque la mayoría de propuestas de aprendizaje de SBRDEs en la literatura especializada hacen uso de los AGs, también es posible encontrar propuestas que utilizan otros tipos de AEs. Un ejemplo son aquellas que hacen uso de la PG. A continuación se presenta una revisión de las principales propuestas existentes en la literatura especializada:

- Un trabajo inicial dentro de este campo es *Fuzzy GP*, desarrollada por Geyer-Schulz [GS95], el cual combina un simple AG que opera en un lenguaje libre de contexto mediante el uso de un lenguaje de reglas difusas libres de contexto. Fuzzy GP es un método basado en gramáticas, que permite codificar un conjunto de reglas por individuo de la población. La gramática usada en Fuzzy GP también incluye algunos modificadores lingüísticos que permiten desplazar las funciones de pertenencia, o bien aplicar operadores de concentración o dilatación a dichas funciones de pertenencia. Por lo tanto, Fuzzy GP es un método que aprende BCs.
- En [TAKJ96], Tunstel y otros proponen varios paradigmas para el aprendizaje automático dentro del campo del control de robots. Uno de estos paradigmas es un algoritmo basado en PG que usa un conjunto de funciones y terminales para codificar un conjunto de reglas en un único individuo de la población. Este algoritmo solo aprende la BR, ya que los parámetros de las funciones de pertenencia son prefijados y no varian durante todo el proceso evolutivo. Este mismo paradigma, se utiliza posteriormente en [AKTJ00] para realizar otra tarea diferente dentro del mundo del control de robots: la navegación desde un punto a otro en la misma planta de un edificio.
- Una ligera modificación del paradigma visto en el punto anterior, se puede encontrar en [HBT99], donde los autores proponen la coevolución de las funciones de pertenencia y de los conjuntos de reglas difusas, es decir, que se aprende toda la BC.
- Alba y otros, proponen en [ACT99] un método basado en PG para el problema del centrado de un carrito, el cual usa una gramática BNF que produce

una serie de árboles los cuales representan (cada uno de ellos) un conjunto de reglas. Se utiliza una PG tipada (typed-GP) con el objetivo de evitar que se generen reglas no válidas y sin sentido. Finalmente, hay que señalar que esta propuesta aprende la BR asociada al controlador mientras que las funciones de pertenencia permanente fijas.

- Sánchez y otros proponen en [GGS99] y [SCC01] un par de métodos de aprendizaje de SCBRDs que combinan los operadores de la PG con un AG y el Enfriamiento Simulado (Simulated Annealing), respectivamente, para establecer las funciones de pertenencia (por lo tanto se aprenden BCs completas). Estos procesos están basados en gramáticas y codifican un conjunto de reglas por individuo de la población.

- Bastian propone en [Bas00] una representación en árbol de modelos difusos que utiliza la PG para identificar de forma simultanea la estructura de la regla (las variables de entrada y salida, y el número de términos de cada variable lingüística) y los parámetros de las funciones de pertenencia.

- En [MdBVFN01], Mendes y otros proponen CEFR-MINER, un algoritmo coevolutivo que incluye un algoritmo basado en PG para obtener un conjunto de reglas difusas y un algoritmo evolutivo para evolucionar las funciones de pertenencia (se aprende por lo tanto toda la BC). La PG aprende reglas difusas en notación DNF y sigue el enfoque Pittsburgh para codificar las reglas dentro de la población. CEFR-MINER no hace uso de ningún tipo de gramática, y en su lugar lo que hace es introducir algunas restricciones sintácticas en los árboles con el fin de obtener individuos que sean válidos. CEFR-MINER también incluye un proceso de poda de los árboles para evitar los efectos del crecimiento (code bloating) en los mismos.

- Hoffmann y Nelles, proponen en [HN01] un algoritmo basado en PG para identificar la partición óptima del espacio de entrada en una serie de conjuntos difusos gausianos ortogonales. Esta partición óptima se usa después para identificar la parte del antecedente de una serie de modelos lineales neuro-difusos, que se describen por medio de una serie de reglas difusas tipo Takagi-Sugeno-Kang (TSK). Los parámetros lineales en el consecuente de las reglas TSK se estiman mediante el uso de un algoritmo de mínimos cuadrados local con pesos.

- En [CLH02], Chien y otros aprenden funciones discriminantes con atributos difusos por medio de la PG. Estas funciones se pueden trasformar a

posteriori en una serie de reglas difusas que pueden ser usadas dentro de un sistema experto. Este algoritmo aprende una única función por cada clase en el problema, de manera que la BR completa se obtiene al combinar las funciones (reglas) de todas las clases del problema. Las funciones de pertenencia se fijan y no cambian durante el proceso evolutivo.

- Tsakonas investiga en [Tsa06] la efectividad de una serie de estructuras inteligentes generadas mediante la PG para tareas de clasificación. En concreto, presenta cuatro gramáticas libres de contexto para el aprendizaje de árboles de decisión, SBRDs, redes neuronales y redes de Petri difusas. Con respecto al aprendizaje de SBRDs, la gramática propuesta permite codificar varias reglas por individuo de la población (enfoque Pittsburgh), mientras que las funciones de pertenencia (de tipo gausiano) se fijan a priori. Por lo tanto, solo se aprende la BR.

- En [ASD08], Akbarzadeh y otros proponen un algoritmo coevolutivo que incluye un algoritmo basado en PG para obtener un conjunto de reglas difusas relacionales para problemas de clasificación y una estrategia evolutiva para obtener las funciones de pertenencia (se aprende por lo tanto toda la BC). La PG aprende reglas difusas relacionales en notación DNF (se imponen algunas restricciones en la estructura de los individuos) y sigue el enfoque Pittsburgh para codificar las reglas dentro de la población. Es importante señalar que en cada ejecución del algoritmo coevolutivo sólo se obtienen las reglas y las funciones de pertenencia asociadas a una sóla de las clases del problema, y que por lo tanto el algoritmo se debe ejecutar tantas veces como clases tenga el problema a resolver.

- Mucientes y otros proponen en [MVBL09] un algoritmo basado en PG para obtener reglas difusas TSK para estimar los tiempos de procesamiento de una serie de máquinas utilizadas en la industria de la fabricación de muebles. Este algoritmo hace uso de una gramática libre de contexto para generar las reglas difusas TSK y sigue el enfoque GCCL de codificación de reglas, de manera que la solución final está formada por un subconjunto de los individuos de la población final.

1.5. El problema de la alta dimensionalidad en el aprendizaje de Sistemas Basados en Reglas Difusas

A la hora de diseñar cualquier método de aprendizaje de SBRDs, existen dos objetivos principales (y a la vez opuestos) que se deben intentar maximizar: la precisión y la interpretabilidad del conocimiento extraido en forma de reglas difusas.

Durante los años 90, las diferentes propuestas desarrolladas prestaron una mayor atención a la mejora de la precisión de los SBRDs, aunque dicha mejora se consiguió a consta de sacrificar la interpretabilidad de los mismos. No obstante, algunos estudios recientes [AAFHO07, CCHM03a, CCHM03b, GRP$^+$07, INN04, IN07, VAD07] han puesto de manifiesto la necesidad de la existencia de un equilibrio entre la interpretabilidad y la precisión del conocimiento extraido a la hora de diseñar métodos de aprendizaje de SBRDs.

Este equilibrio es más dificil de lograr cuando el problema que ha de resolverse presenta una alta dimensionalidad, es decir, un elevado número de variables o características de entrada, y/o un elevado número de instancias o ejemplos. En esta memoria nos ocuparemos de aquellos problemas que presentan alta dimensionalidad con respecto al número de variables de entrada. En este tipo de problemas, los principales inconvenientes vienen dados por el crecimiento exponencial que se produce en el espacio de búsqueda de reglas difusas con un aumento lineal en el número de variables, lo que popularmente se conoce como el problema de la explosión combinacional de reglas [CA98]. Dicho crecimiento hace que el proceso de aprendizaje se vuelva más complicado, y en la mayoría de los casos, lleva al aprendizaje de un SBRD que presenta un elevado nivel de complejidad (con respecto al número de reglas, y de variables y etiquetas incluidas en cada regla).

El análisis de la literatura especializada muestra que existen dos soluciones principales para abordar el aprendizaje de SBRDs que presenten un buen equilibrio entre interpretabilidad y precisión en problemas con alta dimensionalidad:

1. *Llevar a cabo un proceso de Selección de Características*:

   Para un problema dado, el número de características o variables de entrada puede ser elevado por alguno de los siguientes motivos [LM98]:

   - Los datos no se han obtenido para un fin específico, por lo que ca-

racterísticas que son necesarias para una tarea, serán redundantes o irrelevantes para otra.

- Los datos se han recogido sin conocimiento claro de las variables relevantes para el problema a resolver. En estas situaciones se suelen introducir todas las características consideradas remotamente relevantes. Como consecuencia, y de forma inevitable, algunas de ellas serán redundantes o irrelevantes.
- Un determinado problema puede necesitar datos de distintas fuentes de información, y si la cantidad de información de cada una de ellas es moderadamente grande, la unión será enorme.

En estos casos, es necesario aplicar un algoritmo de *Selección de Características* [BG05, GE08, HYLW08, KJ97, LY05], antes o durante el proceso de aprendizaje inductivo del SBRD, que determine las características más relevantes para alcanzar el máximo rendimiento posible. Algunos resultados inmediatos de la selección de características son:

- Menor cantidad de datos, de forma que se mejora la eficiencia del algoritmo de aprendizaje al poder este aprender más rápidamente.
- Mayor precisión, que hará que el SBRD obtenido pueda generalizar mejor.
- Resultados más simples y más fáciles de entender, lo que mejorará la interpretabilidad del SBRD obtenido.

Podemos distinguir dos tipos de algoritmos de selección de características, según la forma en que se evaluen los distintos subconjuntos de características:

a) *Filtro*, donde la evaluación de un subconjunto de características se hace en base a un cierto criterio o medida que permite filtrar características irrelevantes, y que es independiente del algoritmo de aprendizaje utilizado. Algunas de esas medidas son:
   - De información, basadas en medidas de incertidumbre respecto a la clase verdadera.
   - De dependencia, asociación o correlación entre una característica dada y la variable de clase.

- De distancia, separabilidad, divergencia o discriminación entre clases.
- De consistencia, que tratan de encontrar el mínimo número de características que puedan separar las clases de la misma forma en que lo hace el conjunto completo de variables.

b) *Envolventes*, donde la evaluación de un subconjunto dado de características se hace en base a la precisión del clasificador inducido por un algoritmo de aprendizaje usando dicho subconjunto de características.

En la literatura especializada es posible encontrar procesos de selección de características basados en el uso de medidas de información [BG05, CCFM05, SJ04, SB00], sistemas difusos evolutivos [CCdJH01, GP01, HCHC04, YBS02], sistemas neuro-difusos [CFM02, CP04, LMB02, RZ01], así como otras técnicas distintas [RRZ00, RZ00, RSA03].

2. *Compactar y reducir un conjunto de reglas previamente aprendido, en una etapa de post-procesamiento*:

   Algunos procesos de aprendizaje de SBRDs tienden a generar un número muy elevado de reglas difusas cuando el problema a resolver es de alta dimensionalidad, con la consiguiente pérdida de interpretabilidad que esto supone. Para resolver este problema, diversos autores han propuesto incluir un proceso de compactación y/o reducción del conjunto de reglas después del proceso de aprendizaje, el cual minimizará el número de reglas incluidas en la BR y mejorará la interpretabilidad del SBRD.

   Los métodos de reducción de reglas actúan combinando y/o seleccionando un subconjunto de reglas del conjunto de reglas devuelto por el proceso de aprendizaje, con el objetivo de minimizar el número de reglas utilizadas mientras que se mantiene (o incluso se mejora) la precisión del SBRD. Aquellas reglas que son conflictivas, que están mal definidas o que son innecesarias pueden ser eliminadas a través de este proceso de reducción de reglas, mejorando así tanto la interpretabilidad como el rendimiento de SBRD obtenido.

   En la literatura especializada es posible encontrar métodos de reducción basados en sistemas difusos evolutivos [CCdJH05, INYT95, IY04], transformaciones ortogonales [SB01, YBY99, YW99] y en medidas de similaridad [ARS03, BM07, CL04, PT06].

Otra posible solución al problema de la alta dimensionalidad es el uso de la PG para diseñar algoritmos que permitan la ausencia de algunas de las va-

riables de entrada en las reglas, lo que permite obtener SBRDs con un buen equilibrio interpretabilidad-precisión en problemas con alta dimensionalidad. En los siguientes capítulos se describen las propuestas desarrolladas en esta memoria para abordar este problema.

Capítulo 2

Un modelo basado en programación genética para el aprendizaje de sistemas de clasificación basados en reglas difusas compactos y precisos para problemas con alta dimensionalidad

En este capítulo se describe una propuesta evolutiva basada en PG para el aprendizaje de SCBRDs compactos y precisos para problemas con alta dimensionalidad, el algoritmo GP-COACH (Genetic Programing based learning of COmpact and ACcurate fuzzy rule based classification systems for High dimensional problems).

Se trata de un algoritmo evolutivo que aprende reglas difusas DNF generadas por medio de una gramática libre de contexto, y que utiliza un mecanismo de competición entre reglas para mantener la diversidad de la población, el cual elimina reglas irrelevantes durante el proceso evolutivo, lo que permite obtener conjuntos compactos de reglas difusas, pero que presentan una buena capacidad de generalización.

Se ha realizado un análisis de componentes del modelo en el que se examinan los resultados obtenidos con la inclusión de este mecanismo de competición entre reglas, con el uso de algunos operadores genéticos específicos, y con el uso de un algoritmo basado en PG en vez de un AG tradicional.

Se ha llevado a cabo un estudio experimental con problemas de clasificación de alta dimensionalidad, donde se ha comparado los resultados obtenidos por GP-COACH con los obtenidos por otros métodos de aprendizaje de SCBRDs, bien conocidos en la literatura especializada.

Finalmente, y atendiendo a las recomendaciones hechas por Demšar en [Dem06], se ha considerado el uso de tests estadísticos no paramétricos para la validación de los resultados obtenidos, tanto desde el punto de vista de la precisión como de la compacticidad de los SCBRDs obtenidos.

2.1. Introducción

El objetivo que perseguimos es el desarrollo de un algoritmo basado en PG para el aprendizaje de SCBRDs que presenten un buen equilibrio entre interpretabilidad y precisión en problemas con alta dimensionalidad con respecto al número de variables de entrada. Esta tarea no es sencilla, pues presenta algunos inconvenientes:

1. El incremento en el número de variables de entrada del problema, produce un crecimiento exponencial en el especio de búsqueda de reglas difusas, lo que puede hacer que el algoritmo de aprendizaje de SCBRDs se quede estancado en un óptimo local.

2. En este tipo de problemas existe la posibilidad de que se produzca *sobreaprendizaje*, es decir, que el algoritmo sea capaz de aprender un SCBRD que cubra muy bien los ejemplos de entrenamiento, pero que sin embargo presente una elevada tasa de error ante nuevos ejemplos de prueba.

3. Existe una alta probabilidad de obtener SCBRDs con un elevado nivel de complejidad (con respecto al número de reglas, y de variables y etiquetas incluidas en cada regla).

Por ello, a la hora de diseñar cualquier algoritmo basado en PG para abordar esta tarea, debemos planteanos una serie de cuestiones:

1. *¿Que tipo de esquema de codificación es conveniente emplear para representar la solución?*

   Como ya hemos visto en la Sección 1.4.3, existen dos enfoques distintos a la hora de representar un conjunto de reglas difusas:

   a) el enfoque Pittsburgh, donde cada cromosoma representa un conjunto de reglas, y
   b) el enfoque "cromosoma = regla", donde cada cromosoma representa una única regla.

   La elección de uno u otro enfoque dependerá en gran medida del tipo de tarea a resolver dentro del campo de la minería de datos. En ámbito de la clasificación, que es el que nos ocupa en esta memoria, se busca evaluar la calidad del conjunto de reglas como un todo (más que evaluar la calidad de cada regla de forma separada). En este sentido, el enfoque Pittsburgh parece ser el más natural a emplear, y es por ello que dicho enfoque ha sido tradicionalmente el más utilizado en el diseño de SCBRDs mediante PG (ver Sección 1.4.3). Por contra, el enfoque "cromosoma = regla" parece más adecuado para aquellas tareas dentro de la minería de datos en las que el objetivo es obtener un conjunto pequeño de reglas de alta calidad, pero en las que cada regla es evaluada independientemente del resto.

   No obstante, y a pesar del hecho de que el enfoque Pittsburgh permite directamente tener en cuenta la interacción entre las reglas cuando se calcula el fitness de cada individuo, es importante señalar que este enfoque también presenta algunos inconvenientes que desaconsejan su uso cuando el objetivo es obtener SCBRDs compactos y precisos para problemas de alta dimensionalidad.

   En concreto, los algoritmos de aprendizaje de reglas difusas basados en este enfoque tienden a generar individuos muy grandes y complejos, lo que decrementa considerablemente la interpretabilidad y compacticidad de los

SCBRDs obtenidos. Algunas soluciones propuestas para abordar dicho problema han sido el diseño de operadores genéticos específicos que eviten la generación de invididuos muy complejos, la introducción de restricciones en el tamaño máximo alcanzable por los individuos, o el uso de algún tipo de mecanismo de poda que elimine reglas irrevantes en cada individuo, disminuyendo así su tamaño y complejidad. Sin embargo, todas estas soluciones normalmente conllevan un incremento en el coste computacional del algoritmo, o bien en la complejidad del diseño del mismo. Por otra parte, también pueden llevar a que el algoritmo no sea capaz de alcanzar un buen nivel de precisión debido a las restricciones impuestas en los individuos.

Por contra, en las propuestas de aprendizaje basadas en el enfoque "cromosoma = regla" los individuos son mucho más pequeños y simples, lo que lleva a reducir el coste computacional de los algoritmos, así como a simplificar su diseño. Sin embargo, la gran dificultad en este enfoque consiste en conseguir un conjunto completo de reglas que presente una elevada calidad (tanto a nivel de interpretabilidad como de precisión), y no sólo a nivel individual de cada regla.

Una posible solución, denominada enfoque IRL, consiste en obtener el conjunto de reglas al ejecutar varias veces el algoritmo: en cada ejecución se devuelve una única regla, siendo por tanto necesario el uso de algún mecanismo que evite que el algoritmo devuelva siempre la misma regla. Este enfoque, que ha sido utilizado en diferentes propuestas de aprendizaje de SCBRDs existentes en la literatura, permite obtener conjuntos de reglas que presentan una buena cooperación. Sin embargo, el hecho de que dichas reglas se obtengan en ejecuciones independientes hace que la competición entre las mismas sea completamente nula, lo que lleva a que la interpretabilidad de los SCBRDs obtenidos no siempre sea la mejor al poder existir reglas redundantes o irrelevantes entre el conjunto de reglas final.

En este sentido, el uso del enfoque GCCL, en donde todos los individuos compiten y cooperan de forma simultanea permitiendo así la eliminación de reglas irrelevantes durante el propio proceso evolutivo, puede lograr obtener conjuntos de reglas difusas con un alto nivel de interpretabilidad y precisión. No obstante, el uso de este enfoque hace necesario introducir algún mecanismo para mantener la diversidad en la población, evitando así que todos los individuos convergan a la misma zona del espacio de búsqueda de reglas difusas.

2. *¿Que mecanismo es adecuado utilizar para favorecer la diversidad dentro de la poblacion?*

   En la literatura especializada se pueden encontrar varias propuestas para favorecer la diversidad dentro de una población de individuos. Dichas propuestas normalmente se basan en la creación de nichos dentro de la población, lo que evita que todos los individuos convergan a una misma zona dentro del espacio de búsqueda de reglas difusas.

   Algunas técnicas clásicas de creación de nichos dentro del campo de los AGs son los denominados métodos de proporción o fitness sharing, los métodos de multitud o crowding, y los métodos de aclarado o clearing [PHH03]. Todos estos métodos se basan en el calculo de algún tipo de medida de similaridad entre los invididuos. Sin embargo, el cálculo de esta medida no es sencillo cuando utilizamos PG, ya que los individuos suelen codificarse como un árbol, siendo dificil calcular la similitud entre dos árboles. Para evitar este problema, necesitamos utilizar algún mecanismo que mantenga la diversidad en la población y que no tenga en cuenta la estructura de los individuos, sino los ejemplos que cada uno de estos cubre. Este es el caso por ejemplo de la Competición de Tokens [WL00].

3. *¿Que medidas de calidad se utilizarán para calcular la bondad de cada solución?*

   En la literatura es posible encontrar una amplia variedad de medidas de calidad que se pueden utilizar para medir la bondad de una regla o de un conjunto de reglas. Algunas de estas medidas son subjetivas y por tanto requieren de la intervención humana, mientras que otras, por el contrario, son objetivas y no requieren de dicha intervención. Obviamente, en nuestro caso, dado que lo que queremos es obtener un proceso de inducción automática de reglas difusas sólo podremos aplicar medidas o criterios de calidad objetivos.

   Dado que en nuestra propuesta vamos a utilizar un enfoque GCCL en el que cada individuo representa una regla difusa, se debería utilizar una función de fitness que permita evaluar la bondad de una única regla difusa. Por otra parte, dado que estamos interesados en obtener conjuntos de reglas compactos y precisos, esta función debería premiar a aquellas reglas difusas que cubran muchos ejemplos de la clase representada en su consecuente, y que su vez los cubran correctamente. Por tanto, dicha función de fitness deberá incluir medidas de calidad objetivas que midan:

- la precisión de una regla, es decir, la confianza de que el consecuente indicado en la regla es el correcto para aquellos ejemplos que emparejan con el antecedente de dicha regla, y
- el porcentaje de acierto dentro del conjunto de ejemplos pertenecientes a la clase representada en el consecuente de la regla.

Por otra parte, a pesar de que cada individuo codifica una única regla difusa, al final todos los individuos de la población se deben unir para formar el conjunto de reglas difusas que será devuelto como solución al problema. Por ello, también es necesario el uso de alguna función de fitness que calcule la bondad de toda la población de reglas. Como ya hemos indicado, estamos interesados en obtener conjuntos de reglas que presenten un buen equilibrio entre interpretabilidad y precisión, así que dicha función de fitness debería incluir medidas de calidad objetivas que midan:

- la complejidad del conjunto de reglas, la cual vendrá dada tanto por el número de reglas en el conjunto, como por el número medio de variables y etiquetas por regla, y
- la precisión predictiva del conjunto de reglas, que se define como el número de instancias o ejemplos correctamente clasificados.

4. *¿Que tipo de operadores genéticos se deben utilizar?*

   Como es bien sabido, el principal operador genético dentro de la PG, y en general en cualquier AE, es el operador de cruce. Este operador intercambia el material genético de dos padres con el objetivo de explotar el espacio de búsqueda y refinar los individuos obtenidos hasta el momento. Por lo tanto, este operador debe jugar un papel fundamental en cualquier propuesta de aprendizaje de reglas difusas, debiendo aplicarse con una alta probabilidad.

   En cuanto al operador de mutación, aunque en PG no tiene la importancia del operador de cruce, también nos va a permitir explorar el espacio de reglas, encontrando así nuevas reglas que pueden llevar a zonas más prometedoras dentro del espacio de búsqueda. Por lo tanto, también deberemos considerar el uso de este operador en nuestra propuesta, aunque con una probabilidad menor que la del operador de cruce.

   Por otra parte, dado que estamos interesados en obtener conjuntos de reglas compactos, que contengan reglas simples con pocas variables y etiquetas, es interesante hacer uso de algún tipo de operador que permita generalizar las

reglas de la población al eliminar alguna de sus variables. Este operador se deberá aplicar con una probabilidad baja porque sino correríamos el riesgo de obtener reglas muy generales que cubrirían muchos ejemplos y que podrían hacerlo de forma no correcta.

Para evitar que el operador anterior introduzca un sesgo en la búsqueda también puede ser interesante el usar un operador que haga justo lo contrario, es decir, que dada una regla, intente hacerla un poco más específica para que no cubra tantos ejemplos y que sin embargo los cubra bien. Al igual que el operador anterior, este operador también se debería aplicar con una probabilidad baja.

Finalmente, nos podríamos plantear que ocurriría si alguno de estos operadores generase individuos poco adecuados en la población. Esto podría hacer que buenos individuos (los padres) desaparecieran en favor de otros peores. Sin embargo, este inconveniente tiene una fácil respuesta: para evitar este problema debemos elegir un esquema de reemplazamiento de la población correcto.

5. *¿Que tipo de esquema de reemplazamiento de la población se debe utilizar?*

   Existen dos esquemas clásicos de reemplazamiento de la población dentro del campo de los AGs:

   - el esquema generacional, donde la nueva población de hijos reemplaza por completo a la antigua población de padres, y
   - el esquema estacionario, donde solo los peores individuos de la población son reemplazados por los nuevos descendientes, y sólo en el caso de que estos último sean mejores.

   El problema es que ninguno de estos esquemas puede ser usado en nuestro algoritmo porque ambos requieren que cada individuo sea por si solo una solución completa al problema, cosa que no sucede en nuestro caso pues la solución completa viene dada por toda la población de reglas difusas.

   Por otra parte, como hemos visto estamos interesados en utilizar un mecanismo que mantenga la diversidad en la población y que al mismo tiempo permita que los individuos compitan entre si con el objetivo de eliminar reglas irrelevantes y obtener conjuntos compactos de reglas difusas. Por lo tanto, podemos dejar que todos los individuos generados, es decir padres e hijos, compitan entre sí y que sea el propio mecanismo de mantenimiento

de la diversidad el que controle quienes deben formar parte de la siguiente población y quienes deben ser borrados. El uso de este esquema de reemplazamiento nos va a permitir tener un tamaño de población variable, el cual va a cambiar durante todo el proceso evolutivo, de forma que así podamos obtener un conjunto compacto de reglas difusas a pesar de que el tamaño inicial de la población sea elevado.

2.2. GP-COACH: Genetic Programing based learning of COmpact and ACcurate fuzzy rule based classification systems for High dimensional problems

Se presenta aquí el modelo evolutivo para el aprendizaje de SCBRDs compactos y precisos en problemas de alta dimensionalidad (con un elevado número de variables de entrada), denominado GP-COACH (Genetic Programing-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems). Sus principales características son las siguientes:

- Utiliza una gramática libre de contexto que aprende reglas difusas DNF y que permite la ausencia de algunas de las variables de entrada en las reglas. Esta forma de representación permite obtener reglas difusas simples, con pocas variables y etiquetas (reglas compactas).
- Sigue el enfoque *GCCL* de codificación de reglas, que codifica una única regla por individuo, de forma que la BR esta formada por todos los individuos de la población. El uso de este tipo de enfoque en la codificación hace necesario definir dos funciones de evaluación (fitness) diferentes:
  - Una función de fitness local que evalúa la bondad de cada una de las diferentes reglas en la población. A partir de ahora denominaremos a esta función simplemente como *función de fitness*.
  - Una función de fitness global que evalúa la bondad de toda la población de individuos, es decir, de toda la BR. A partir de ahora nos referiremos a esta función como *fitness global*.
- GP-COACH incluye un mecanismo para mantener la diversidad de la población y así evitar que todos los individuos converjan a una misma zona

del espacio de búsqueda de reglas difusas. En concreto, hemos utilizado la *Competición de Tokens* [WL00], que hace que las reglas compitan entre si durante el proceso evolutivo, eliminando reglas irrelevantes, y por lo tanto generando un número más reducido de reglas difusas pero que presentan una elevada capacidad de clasificación. Por otra parte, es importante indicar que GP-COACH utiliza un tamaño variable de población, lo cual es lo que nos permite obtener conjuntos pequeños y compactos de reglas difusas. En [KZB07, KZB09], podemos encontrar estudios sobre el uso de poblaciones de tamaño variable en PG.

- Hace uso de un mecanismo de inferencia jerárquico con dos niveles, aprendiéndose dos tipos distintos de reglas: *reglas primarias*, que son reglas fuertes y generales generadas por los operadores genéticos, y *reglas secundarias*, que son reglas más débiles y específicas generadas después de la competición de tokens con el objetivo de incrementar la diversidad de la población.

- Finalmente, GP-COACH utiliza un esquema de reproducción en el que cada descendiente se genera aplicando uno solo de los operadores genéticos, y en el que tanto padres como hijos compiten entre si con el objetivo de formar parte de la nueva población.

A continuación, se describen con detalle cada uno de estos componentes, mostrándose también una completa descripción del algoritmo, así como un pseudocódigo del mismo.

Tabla 2.1: Ejemplo de gramática

$Comienzo \longrightarrow [Si],\ antec,\ [entonces],\ consec,\ [.].$
$antec \longrightarrow descriptor1,\ [y],\ descriptor2.$
$descriptor1 \longrightarrow [nada].$
$descriptor1 \longrightarrow [X_1\ es]\ etiqueta.$
$descriptor2 \longrightarrow [nada].$
$descriptor2 \longrightarrow [X_2\ es]\ etiqueta.$
$etiqueta \longrightarrow \{member(?a,\ [B,\ M,\ A,\ B\ o\ M,\ B\ o\ A,\ M\ o\ A,\ B\ o\ M\ o\ A])\},\ [?a].$
$consec \longrightarrow [Clase\ es]\ descriptorClase.$
$descriptorClase \longrightarrow \{member(?a,\ [C^1,\ C^2,\ C^3])\},\ [?a].$

2.2.1. Definición de la gramática libre de contexto

Como se ha indicado anteriormente, GP-COACH hace uso de una gramática libre de contexto para aprender reglas difusas en notación DNF:

$$R^k : \text{Si } X_1 \text{ es } \hat{A}_1^k \text{ y } \ldots \text{ y } X_{n_v} \text{ es } \hat{A}_{n_v}^k$$
$$\text{entonces } Clase \text{ es } C^k \text{ con } GC^k$$

donde cada variable de entrada X_i toma como valor un conjunto de etiquetas lingüísticas $\hat{A}_i^k = \left\{ L_i^1 \; o \; \ldots \; o \; L_i^{l_i} \right\}$ unidas por el operador de disjunción lógica (comúnmente conocido como operador o), y donde GC^k es el grado de certeza asociado a la clasificación en la clase C^k para los ejemplos pertenecientes al subespacio difuso delimitado por el antecedente de la regla.

Dicha gramática permite la ausencia de algunas de las variables de entrada, lo que nos lleva a obtener reglas simples, con pocas variables y etiquetas. En la Tabla 2.1, se muestra un ejemplo de gramática para un problema de clasificación con dos variables de entrada (X_1, X_2), tres etiquetas lingüísticas por variable (*Bajo*, *Medio*, *Alto*) y tres clases (C^1, C^2, C^3). En esta gramática, el símbolo $?a$ que aparece en alguna de la reglas de producción representa a uno, y sólo uno, de los valores separados por comas dentro de los corchetes. En la Figura 2.1, se muestra como una regla difusa (generada con esta gramática) se puede codificar como un árbol.

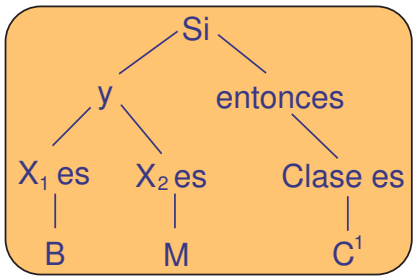


Figura 2.1: Codificación de una regla como un árbol

En GP-COACH cada regla difusa DNF generada tiene asociado un grado de certeza ($GC^k \in [0, 1]$), el cual representa la confianza en la clasificación en la clase

indicada en el consecuente de la regla (C^k). Concretamente, en nuestra propuesta dicho grado de certeza se obtiene como sigue:

$$GC^k = \frac{\sum_{\substack{e^h \in C^k \\ h=1,\ldots,n_e}} m^k(e^h)}{\sum_{h=1,\ldots,n_e} m^k(e^h)} \tag{2.1}$$

donde $m^k(e^h)$ es el grado de compatibilidad entre un ejemplo y el antecedente de una regla difusa, es decir, el grado de pertenencia del ejemplo al subespacio difuso delimitado por el antecedente de la regla, el cual se denomina grado de emparejamiento. En nuestro método, dicho grado de emparejamiento se calcula de la siguiente forma:

$$\begin{aligned} m^k(e^h) = T(TC(\mu_{L_1^1}(e_1^h), \ldots, \mu_{L_1^{l_1}}(e_1^h)), \ldots, \\ TC(\mu_{L_{n_v}^1}(e_{n_v}^h), \ldots, \mu_{L_{n_v}^{l_{n_v}}}(e_{n_v}^h))) \end{aligned} \tag{2.2}$$

donde $L_i^{l_i}$ es la etiqueta lingüística número l_i de la variable i; $\mu_{L_i^{l_i}}(e_i^h)$ es el grado de emparejamiento para el valor de la $i-esima$ variable del ejemplo e^h al conjunto difuso correspondiente a la etiqueta lingüística l_i para dicha variable (i); T es la t-norma seleccionada para representar el significado del operador Y (la intersección difusa) que en nuestro caso es la t-norma del mínimo; y TC es la t-conorma seleccionada para representar el significado del operador O (la unión difusa) que en nuestro caso es el t-conorma del máximo. Otras formas diferentes de calcular el grado de certeza se pueden encontrar en [IY05].

Como se puede ver, nuestra gramática libre de contexto sólo permite aprender una regla difusa DNF por individuo en la población. Esto se debe al esquema de codificación que se ha utilizado en nuestro modelo (el enfoque *GCCL*).

2.2.2. Evaluando un individuo: Función de fitness

Cada uno de los individuos (reglas) de la población se evalúa mediante una función de fitness que se basa en la estimación de dos medidas:

- la *Confianza*, que mide la precisión del individuo, es decir, la confianza de que lo indicado en el consecuente es verdad si se verifica el antecedente (la confianza se calcula de la misma manera que el grado de certeza en la ecuación 2.1), y
- el *Soporte*, el cual mide la generalidad del conocimiento representado en el individuo y se calcula de la siguiente forma:

$$Soporte(R^k) = \frac{\sum\limits_{\substack{e^h \in C^k \\ h=1,\ldots,n_e}} m^k(e^h)}{N_{C^k}} \tag{2.3}$$

donde N_{C^k} es el número de ejemplos que pertenecen a la misma clase que la indicada en el consecuente del individuo (R^k).

Ambas medidas se combinan para formar la siguiente función de fitness:

$$F\_fitness(R^k) = \alpha * Confianza(R^k) + (1 - \alpha) * Soporte(R^k) \tag{2.4}$$

donde α es un parámetro que nos permite dar mayor o menor importancia a cada una de esas medidas.

Finalmente, es importante señalar que cada vez que un individuo se evalúa es también necesario modificar su grado de certeza de acuerdo a su valor de confianza.

2.2.3. Evaluando una población: Fitness global

GP-COACH utiliza otra función de fitness denominada *fitness global* para obtener la mejor población generada durante todo el proceso evolutivo. En concreto, dicho fitness global se define como:

$$\begin{aligned} Fitness\_global = w_1 * Precision + w_2 * (1{,}0 - Var_N) + \\ + w_3 * (1{,}0 - Cond_N) + w_4 * (1{,}0 - Reg_N) \end{aligned} \tag{2.5}$$

donde Var_N y $Cond_N$ son los valores normalizados del número de variables y condiciones (etiquetas) en las reglas, y Reg_N es el número normalizado de reglas de la población.

Para normalizar las medidas anteriores, se ha utilizado la normalización min-max:

$$Z_N = \frac{(Z - Z_{min})}{(Z_{max} - Z_{min})} \tag{2.6}$$

donde Z_N es el valor normalizado, Z es la variable original (sin normalizar), y Z_{max} y Z_{min} son los valores máximo y mínimo que la variable Z puede tomar, respectivamente. En la Tabla 2.2 se muestran los valores mínimos y máximos utilizados para normalizar cada una de las medidas anteriormente descritas.

Tabla 2.2: Valores mínimos y máximos considerados para normalizar

	Var	**Cond**	**Reg**
MIN	1	1	n_c
MAX	n_v	$n_v * (l_i - 1)$	n_e

Finalmente, es importante señalar que en GP-COACH no es posible utilizar un enfoque evolutivo multiobjetivo para abordar el calculo de la función de fitness global. Esto se debe al modelo de aprendizaje de reglas difusas utilizado (el enfoque GCCL), el cual sólo codifica una única regla difusa por cromosoma, formando la solución completa mediante la unión de todos los individuos de la población. Por lo tanto, no es posible evolucionar un pareto de soluciones. El diseño de un enfoque evolutivo multiobjetivo en GP-COACH supondría el uso del esquema de codificación Pittsburgh, el cual presenta algunos incovenientes que desaconsejan su uso para la obtención de SCBRDs compactos y precisos: problemas de eficiencia, problemas por la codificación de tamaño variable, problemas por el rápido crecimiento de los árboles (code bloating), etc.

2.2.4. Competición de tokens: Manteniendo la diversidad en la población

La competición de tokens [WL00] ha sido el mecanismo escogido para mantener la diversidad de la población. Su idea es la siguiente: En la naturaleza, cuando un individuo encuentra un buen lugar para vivir (un nicho) intenta explotar sus recursos al mismo tiempo que previene la llegada al mismo de otros individuos que también quieran compartir tales recursos. Los nuevos individuos estarán forzados

por lo tanto a explorar y encontrar sus propios nichos, a menos que estos sean más fuertes que el individuo que inicialmente ocupaba el nicho. De esta forma se incrementa la diversidad de la población.

Basándonos en esta idea, supondremos que cada ejemplo en el conjunto de entrenamiento puede proporcionar un recurso al que denominaremos *token*. Los individuos de la población competirán para capturar tantos tokens como les sea posible. Si un individuo (una regla) consigue emparejar con un ejemplo, este establecerá una bandera para indicar que ha capturado el token asociado a dicho ejemplo. De esta forma, otros individuos más débiles que él no podrán conseguir el token al no encontrarse este ya disponible (aunque ellos también emparejen con el ejemplo al que dicho token está asociado).

El orden en que los individuos compiten para capturar los tokens vendrá dado por su fortaleza o valor de fitness. Aquellos individuos que presenten valores altos de fitness, podrán capturar tantos tokens como les sea posible. Sin embargo, otros individuos más débiles que entren al mismo nicho verán penalizado su valor de fitness al no poder competir con los otros más fuertes. Esto se logra introduciendo una modificación en el valor de fitness de cada individuo, la cual se basa en el número de tokens que cada individuo ha podido conseguir:

$$Fitnesss\_modificado(R^k) = \begin{cases} F\_fitness(R^k) * \frac{count(R^k)}{ideal(R^k)}, & si\ ideal(R^k) > 0 \\ 0, & en\ otro\ caso \end{cases} \tag{2.7}$$

donde *F_fitness(R^k)* es el valor de fitness obtenido por la regla en la función de evaluación, *contador(R^k)* es el número de tokens que la regla R^k ha podido capturar y *ideal(R^k)* es el número total de tokens que la regla podía haber conseguido, y que es igual al número de ejemplos con los que la regla empareja.

Como resultado de la competición de tokens, es posible que existan individuos que no han podido conseguir ningún token. Estos individuos se consideran irrelevantes y pueden borrarse de la población debido al hecho de que todos los ejemplos que ellos cubren ya se encuentran cubiertos por otros individuos más fuertes.

En la Figura 2.2 se muestra un ejemplo de aplicación de la competición de tokens. En este ejemplo, hay una población con cinco reglas $(R^1, \ldots, R^5)$, ordenadas en orden decreciente de su valor de fitness. Antes de la competición de tokens todos los ejemplos de entrenamiento $(e^1, \ldots, e^{12})$ tienen sus tokens libres.

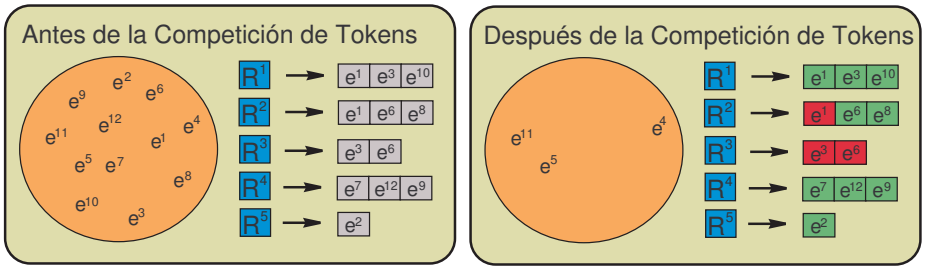


Figura 2.2: Ejemplo de aplicación de la Competición de Tokens

Una vez que la competición de tokens ha empezado, podemos ver como R^1 puede conseguir todos sus tokens ya que es la regla más fuerte. La regla R^2 solo puede capturar dos de sus tres tokens debido a que la regla R^1 ya ha capturado con anterioridad el token asociado al ejemplo e^1, así que R^2 debe modificar su fitness. La regla R^3 debe eliminarse de la población ya que todos sus tokens son capturados previamente por otras reglas más fuertes, es decir, R^3 se considera como una regla irrelevante. Finalmente, R^4 y R^5 no necesitan modificar sus valores de fitness ya que ambas son capaces de conseguir todos sus tokens.

2.2.5. Reglas secundarias: Incrementando la diversidad de la población

Una vez que la competición de tokens ha finalizado, es posible que existan algunos ejemplos de entrenamiento que no hayan sido cubiertos por ninguna de las reglas de la población (este es el caso de los ejemplos e^4, e^5 y e^{11} en el ejemplo de aplicación de la competición de tokens mostrado en la Figura 2.2). En tal caso, la generación de nuevas reglas específicas que cubran dichos ejemplos, incrementa la diversidad de la población y facilita al proceso evolutivo el descubrimiento de reglas más fuertes y más generales que cubran tales ejemplos.

Tal y como se indicó anteriormente, GP-COACH utiliza un proceso jerárquico de inferencia con dos niveles, de forma que se aprenden conjuntos de reglas que contienen dos tipos distintos de reglas difusas: Un núcleo de reglas fuertes y generales (*reglas primarias*) que cubren la mayoría de los ejemplos, y un conjunto pequeño de reglas más débiles y específicas (*reglas secundarias*), las cuales solo se tienen en cuenta en el caso de que no exista ninguna regla primaria que empareje

con los ejemplos.

Concretamente, este proceso jerárquico de inferencia con los niveles funciona de la siguiente forma: Cuando un nuevo ejemplo e llega a uno de los SCBRDs aprendidos, este tratará de encontrar la clase que mejor empareja con dicho ejemplo e, usando para ello la información almacenada en su base de conocimiento. Sin embargo, el MRD solo considerará aquellas reglas que han sido etiquetas como primarias para obtener la clase de dicho ejemplo e. Las reglas secundarias solo se tendrán en cuenta en el caso de que no exista ninguna regla primaria en la BR que empareje con el ejemplo e.

Este proceso de inferencia jerárquico de inferencia con dos niveles permite que GP-COACH pueda incrementar la precisión de los conjuntos de reglas aprendidos, evitando introducir errores de clasificación provocados por el uso de las reglas secundarias cuando existen reglas primarias (más generales) que pueden aplicarse.

Para generar reglas secundarias, GP-COACH sigue el método propuesto por Chi y otros [CYP96] (el cual es una extensión para problemas de clasificación del método propuesto por Wang y Mendel en [WM92]):

1. Si después de aplicar la competición de tokens existen algunos ejemplos de entrenamiento con sus tokens libres, entonces se elige aleatoriamente uno de esos ejemplos y se genera una nueva regla que lo cubra. Esta nueva regla contendrá todas las variables de entrada y cada una de esas variables tendrá asociada una única etiqueta lingüística (aquella para la que el ejemplo presenta el mayor grado de emparejamiento). La clase de la nueva regla será la clase asociada al ejemplo que ha sido elegido.

2. Se eliminan todos los ejemplos que emparejen que con esta nueva regla, de entre aquellos que estaban con sus tokens libres tras la competición de tokens.

3. Si aun existen ejemplos con sus tokens libres, entonces se repiten los dos pasos previos. En otro caso, se evalúan las nuevas reglas que se han generado y estas se unen con el resto de reglas de la población actual.

Es importante señalar que GP-COACH también puede aprender conjuntos de reglas que no contienen ninguna regla secundaria, debido a que sus reglas primarias son lo suficientemente fuertes como para cubrir todos los ejemplos de entrenamiento.

2.2.6. Operadores genéticos

GP-COACH utiliza cuatro operadores genéticos diferentes para generar nuevos individuos (el procedimiento de selección de estos operadores se explica en la siguiente sección):

1. *Cruce*: Una parte en el primer padre se selecciona de forma aleatoria y se intercambia por otra parte seleccionada en el segundo padre, pero siempre bajo la restricción de que el descendiente generado debe de ser válido según las reglas de producción de la gramática. Es importante indicar que no se permite elegir como puntos de corte aquellos que coincidan en mitad de una disyunción de etiquetas.

   Este operador produce dos hijos, pero solo uno de ellos (elegido aleatoriamente) se devuelve como descendiente. Un ejemplo del comportamiento de este operador se muestra en la Figura 2.3.

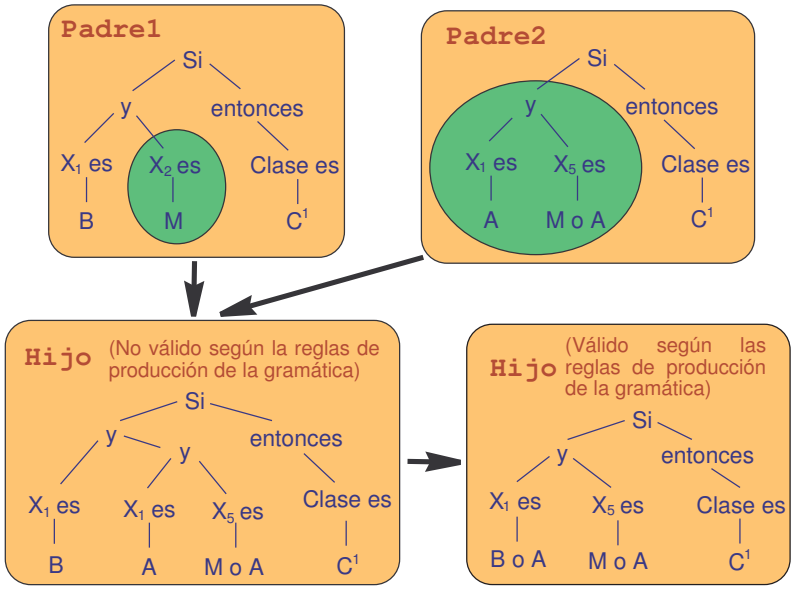


Figura 2.3: Operador de cruce

2. *Mutación*: Se elige aletoriamente una variable en la regla y entonces se lleva a cabo una de las tres siguientes acciones (de nuevo, elegida aleatoriamente):

a) Se añade una nueva etiqueta al conjunto de etiquetas.

b) Se elimina una etiqueta del conjunto de etiquetas.

c) Una etiqueta se reemplaza por otra que no estuviera ya incluida en el conjunto de etiquetas.

3. *Inserción*: Se buscan todas las variables presentes en la regla y entonces se añade otra diferente que no estuviera previamente incluida en ella. El conjunto de etiquetas lingüísticas asociado a esta nueva variable se elige de forma aleatoria, aunque debe contener al menos una etiqueta y debe ser diferente del conjunto "nada"(ver Tabla 2.1).

4. *Dropping Condition*: Debido a la naturaleza probabilística de la PG, se pueden generar descriptores de variables redundantes, es decir, que no aportan verdadero conocimiento a la regla pues este viene dado por el resto de descriptores. Por lo tanto, es necesario generalizar la reglas, para representar el conocimiento de una forma más concisa. El operador dropping condition selecciona aleatoriamente una variable en la regla y la transforma a "nada"(el conjunto de etiquetas lingüísticas asociado a dicha variable también se borra). De esta forma la variable no se considera nunca más en la regla, y por lo tanto la regla puede representar el conocimiento de una forma más general. Un ejemplo de este operador se muestra en la Figura 2.4.

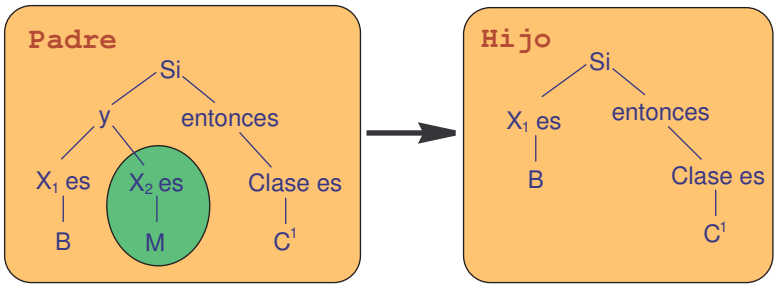


Figura 2.4: Operador *dropping condition*

2.2.7. Etapa de reproducción: Selección, aplicación de los operadores genéticos y reemplazamiento

Un aspecto importante en todo proceso evolutivo consiste en indicar como se genera una nueva población de individuos a partir de la población actual de padres. Esto implica explicar la forma en que se seleccionan los padres, como se aplican los diferentes operadores genéticos, el número de hijos de que se generan a partir de los padres previamente seleccionados y, finalmente, como se forma la nueva población usando esos hijos.

Antes que nada, es importante señalar que en cada iteración de nuestro algoritmo se generan un número de descendientes igual al tamaño de la población actual. Dicho tamaño no es constante y puede variar a lo largo de proceso evolutivo debido a la acción de la competición de tokens.

- *Selección*: Los padres de la población actual se seleccionan mediante el esquema de selección por torneo. Este esquema selecciona un número fijo de individuos de la población (en nuestros experimentos hemos usado un tamaño de torneo igual a 2) y devuelve como ganador del torneo a aquel de ellos que presenta el mejor fitness.

- *Aplicación de los operadores genéticos*: Cada uno de los individuos elegidos durante la etapa de selección se usa para generar un nuevo hijo mediante la aplicación de uno solo de los cuatro operadores genéticos descritos anteriormente. La elección del operador genético a aplicar se lleva a cabo de una manera probabilística. En concreto, se suman las probabilidades de aplicación de los cuatro operadores genéticos en una única medida $P = P_c + P_m + P_i + P_{dp}$ (donde P_c, P_m, P_i y P_{dp} representan las probabilidades de cruce, mutación, inserción y dropping condition, respectivamente) y entonces se obtiene un valor aleatorio $u \in [0, P]$, eligiéndose el operador genético a aplicar tal y como se muestra en la Tabla 2.3.

Tabla 2.3: Elección del operador genético

Cruce	$0 \leq u < P_c$
Mutación	$P_c \leq u < (P_c + P_m)$
Inserción	$(P_c + P_m) \leq u < (P_c + P_m + P_i)$
Dropping Condition	$(P_c + P_m + P_i) \leq u < P$

Por otro lado, es necesario tener en cuenta algunas consideraciones adicionales:

- La primera de ellas se refiere a la aplicación del operador de cruce. Como ya se ha indicado anteriormente, este operador genera un hijo de dos padres. Sin embargo, el esquema de selección por torneo solo devuelve un padre, por lo que es necesario seleccionar un segundo padre de la población para así poder aplicar dicho operador. No obstante, y al contrario de como se seleccionaba el primer padre, este segundo padre se elige de forma aleatoria de entre todos los individuos de la población
- Por otra parte, se debe señalar que el operador de cruce puede intercambiar información entre individuos que representan reglas de diferentes clases. Esto puede incremetar la diversidad en la población y si dicho intercambio generase alguna regla con información inapropiada, nuestro mecanismo de competición de tokens la borraría de la población, dejando sólo las mejores reglas.
- Otro punto importante es que no siempre es posible aplicar todos los operadores genéticos. Por ejemplo, no es posible aplicar el operador de dropping condition a una regla que solo tiene una variable, o aplicar el operador de inserción a una regla que ya está usando todas las variables. Cuando esto ocurre, se selecciona otro operador genético de entre los restantes.

- *Reemplazamiento*: Un punto de gran importancia en GP-COACH es que la nueva población de hijos no reemplaza a la población actual. En vez de eso, se forma una nueva población conjunta, resultado de unir ambas poblaciones. Los individuos de esta nueva población conjunta se ordenan según su valor de fitness, y entonces se lleva a cabo la competición de tokens.

2.2.8. Descripción del algoritmo

El algoritmo GP-COACH comienza creando una población inicial de acuerdo a las reglas de producción de la gramática libre de contexto. Se evalúa cada uno de los individuos de esa población inicial, y entonces esta se guarda como la mejor de todo proceso evolutivo (también se calcula el valor de fitness global de la población inicial y este se guarda como el de la mejor población). Entonces se copia la población inicial como la actual y empieza el proceso evolutivo:

1. Se crea una población de descendientes, con el mismo tamaño que la población actual. Los padres se seleccionan mediante el esquema de selección por torneo binario y los hijos se crean usando uno de los cuatro operadores genéticos. Cada hijo de esta población de descendientes se evalúa.

2. Una vez que se ha creado la población de descendientes, esta se une a la población actual, creándose una nueva población conjunta cuyo tamaño es el doble del de la población actual. Los individuos de esta nueva población se ordenan según su valor de fitness y se aplica la Competición de Tokens. Los individuos irrelevantes (aquellos que no pueden conseguir ningún token) se eliminan de la población y, si es necesario, se crean reglas secundarias (reglas específicas que cubren ejemplos cuyos tokens han quedado libres tras la Competición de Tokens) para incrementar la diversidad de la población.

3. Se calcula el valor de fitness global de la nueva población, y si este mejora al de la mejor población entonces se reemplaza la mejor población por la nueva (y también se actualiza el valor de fitness global de la mejor población). En cualquier caso, la nueva población debe reemplazar a la actual para poder aplicar así otra iteración del proceso evolutivo.

El proceso evolutivo acaba cuando se verifica la condición de parada (que se haya alcazado un número máximo de llamadas a la función de evaluación). Entonces, la población que estaba guardada como la mejor se devuelve como solución al problema.

El pseudocódigo del algoritmo GP-COACH se muestra en la Figura 2.5.

2.3. Estudio experimental

En esta sección se describe la metodología seguida en nuestro estudio experimental. En primer lugar, se muestran los conjuntos de datos utilizados (Sección 2.3.1). A continuación, se describen brevemente los métodos de aprendizaje de SCBRDs utilizados en la comparativa con los resultados obtenidos por GP-COACH, mostrándose los valores utilizados para sus parámetros (Sección 2.3.2). Finalmente, para llevar a cabo un análisis adecuado de los resultados es necesario utilizar diferentes tests estadísticos no paramétricos, los cuales se introducen en la Sección 2.3.3.

```
Inicializar(Pob_inicial)
Para cada Pob_inicial[i] hacer
   Pob_inicial[i].fitness ← Evaluar(Pob_inicial[i])
Fin
Copiar Pob_inicial en Mejor_pob
Mejor_pob.fitness_global ← Fitness_global(Mejor_pob)
Copiar Pob_inicial en Pob_actual
Mientras (condición de parada ≠ verdadero) hacer
    Pob_hijos = ∅
    Mientras (Tamaño(Pob_hijos) ≠ Tamaño(Pob_actual)) hacer
        Padre ← Torneo_binario(Pob_actual)
        Hijo ← Operador_genetico(Padre)
        Evaluar(Hijo)
        Añadir Hijo a Pob_hijos
    Fin
    Pob_conjunta ← Pob_actual ∪ Pob_hijos
    Nueva_pob_actual ← Competicion_tokens(Pob_conjunta)
    Nueva_pob_actual.fitness_global ← Fitness_global(Nueva_pob_actual)
    Si Nueva_pob_actual.fitness_global > Mejor_pob.fitness_global entonces
        Mejor_pob.fitness_global ← Nueva_pob_actual.fitness_global
        Copiar Nueva_pob_actual en Mejor_pob
    Fin
    Copiar Nueva_pob_actual en Pob_actual
Fin
Devolver(Mejor_pob)
```

Figura 2.5: Pseudocódigo del algoritmo GP-COACH

2.3.1. Conjuntos de datos

Nuestra propuesta se ha analizado mediante el uso de 24 conjuntos de datos de clasificación, obtenidos del repositorio de la UCI[1] [AN07] y del proyecto DELVE[2]. Las características más importantes de estos conjuntos de datos se muestran en la Tabla 2.4.

Por otra parte, hemos de señalar que hemos usado la técnica de la *validación cruzada con 10 particiones* para particionar dichos conjuntos: El conjunto original de ejemplos D se divide en $D_j,\ j\ =\ \{1,\ldots,10\}$ conjuntos disjuntos de igual tamaño. Entonces se construyen 10 conjuntos de entrenamiento $TR_i,\ i\ =\ \{1,\ldots,10\}$ y sus complementarios 10 conjuntos de prueba TS_i, de acuerdo a las siguientes ecuaciones:

$$\begin{aligned} &TR_i\ =\ \cup_{j\in J}\ D_j, \\ &J\ =\ j/1 \leq j \leq (i-1)\ y\ (i+1) \leq j \leq 10 \end{aligned} \tag{2.8}$$

[1]Hemos utilizado dos versiones diferentes del conjunto de datos Hill-Valley que se corresponden con los datos sin (1) y con (2) ruido, respectivamente

[2]URL: http://www.cs.toronto.edu/∼delve/

$$TS_i = D/\ TR_i \tag{2.9}$$

Debido a que todos los métodos utilizados en la comparativa son no deterministas, en nuestros experimentos hemos usado 3 semillas distintas para cada partición. Por lo tanto, para cada conjunto de datos se mostrarán los resultados medios obtenidos en 30 ejecuciones.

Tabla 2.4: Características de los conjuntos de datos

Nombre	**Num. Ejem.**	**Num. Var.**	**Num. Clases**	**Nombre**	**Num. Ejem.**	**Num. Var.**	**Num. Clases**
Bupa	345	6	2	Cleveland	297	13	5
Ecoli	336	7	8	Flare	1066	9	2
Glass	214	9	6	Hill-Valley1	1212	100	2
Hill-Valley2	1212	100	2	Iris	150	4	3
Libras Mov.	360	90	15	Magic	19020	10	2
Page-blocks	5472	10	5	Parkinsons	195	22	2
Pen-based	10992	16	10	Pima	768	8	2
Quadruped	5000	46	4	Ringnorm	7400	20	2
Satimage	6435	36	6	Segment	2310	19	7
Sonar	208	60	2	Spambase	4597	57	2
Twonorm	7400	20	2	Wdbc	569	30	2
Wine	178	13	3	Yeast	1484	8	10

2.3.2. Métodos de aprendizaje de SCBRDs

Los resultados obtenidos por GP-COACH se han comparado con los obtenidos por otros métodos de aprendizaje de SCBRDs, los cuales llevan a cabo un proceso de selección de características (antes o durante el proceso de aprendizaje de reglas) con el fin de obtener SCBRDs que presenten un buen equilibrio entre interpretabilidad y precisión. A continuación se describen brevemente cada uno de los métodos considerados en la comparativa:

- **PCA-Ravi**: Ravi y otros [RRZ00] desarrollan un proceso para obtener reglas difusas para problemas de clasificación con alta dimensional. Su propuesta utiliza un conjunto reducido de variables extraídas a partir

de las originales mediante el uso del análisis de componentes principales (PCA) [ZPR09], y un algoritmo de aceptación de umbral modificado [RZ00] para construir un subconjunto compacto de reglas que presente una alta precisión.

- **2SLAVE**: Gonzalez y Pérez proponen en [GP01] el algoritmo 2SLAVE, que es un método basado en el uso de un AG para aprender reglas difusas DNF, que sigue el enfoque IRL para codificar reglas dentro de la población de individuos, y que incluye un proceso de selección de características durante el proceso de aprendizaje de reglas.

- **PG-PITT-Tsakonas**: Tsakonas [Tsa06] diseña un proceso de aprendizaje de SCBRDs basado en el uso de la PG. Este método utiliza una gramática libre de contexto para generar conjuntos completos de reglas por individuo de la población (enfoque Pittsburgh).

- **GCCL-Ishibuchi**: Ishibuchi y otros proponen en [INM99b] un método que sigue el esquema GCCL de codificación de reglas. Este método utiliza un tamaño fijo de la población y etiquetas *"Don't Care"* para generalizar el conocimiento representado en las reglas difusas. Los consecuentes de las reglas están formados por una de las clases de salida del problema y por un factor de certeza. Tanto la clase como el factor de certeza de la regla se obtienen mediante el uso de un procedimiento heurístico, el cual se aplica antes del cálculo del fitness de la regla. Por lo tanto, el AG solo opera sobre los antecedentes de las reglas.

- **FRBCS_GP**: Una primera propuesta basada en el uso de la PG para el aprendizaje de SCBRDs desarrollada por los autores, cuyas principales características son las siguientes:

  - FRBCS_GP aprende conjuntos de reglas que contienen un único tipo de regla difusa, y por lo tanto no utiliza un proceso jerárquico de inferencia con dos niveles tal y como hace GP-COACH.
  - El tamaño de la población actual permanece constante durante todo el proceso evolutivo (a diferencia de GP-COACH que utiliza un tamaño de población variable).
  - FRBCS_GP no hace uso de ningún tipo de función de fitness global para determinar la mejor población generada durante todo el proceso evolutivo, y la solución al problema siempre es la última población generada.

- FRBCS_GP utiliza una función de fitness no difusa, que se basa en el número de ejemplos positivos y negativos cubiertos por cada regla.
- Finalmente, FRBCS_GP hace uso de un esquema de selección por ranking para seleccionar padres, y no hace uso del operador genético de inserción para generar descendientes.

Los parámetros usados en cada uno de estos algoritmos (que son los recomendados por sus respectivos autores) se muestran en la Tabla 2.5.

En GP-COACH, los pesos (w_i) de la función de fitness global se han determinado de forma heurística, tratando de representar un compromiso entre precisión, simplicidad de cada regla individual y simplicidad de la BR completa. Por otra parte, es importante señalar que los valores utilizados para dichos pesos han sido los mismos para todos los conjuntos de datos considerados en la experimentación.

Tabla 2.5: Parámetros usados en los algoritmos

Algoritmo	**Parámetros**
PCA-Ravi	$Umbral\_PCA = 75\,\%, U = 0,95\,\%, thresh = 0{,}035, thrtol = 10^{-8}$ $eps = 0{,}35, acc = 10^{-6}, old = 9999, itrmax = 100, W_{NCP} = 10, W_S = 1$
2SLAVE	$itrmax = 1000, itr\_sin\_mejora = 50, Pob = 20, \lambda = 0{,}8, k_1 = 0, k_2 = 1$ $P_{AND} = 0,1, P_{OR} = 0,1, P_c = 0,6, P_m\ (por\ gen) = 0,05, P_{Rotation} = 0,05$
GP-PITT-Tsakonas	$tamaño\_max\_individuo\ = 650\ nodos, Pob = 2000, itrmax = 10000$ $Torneo = 6, P_c = 0,35, P_m\ (por\ nodo) = 0,4, P_{Shrink} = 0,6$
GCCL-Ishib.	$Eval = 20000, Pob = 100, N_{rep} = 20, P_c = 1,0, P_m = 0,1, P_{don't\ care} = 0,9$
FRBCS_GP	$Eval = 20000, Pob = 200, P_c = 0,5, P_m = 0,4, P_{dp} = 0,1, soporte\_min = 0,01$
GP-COACH	$Eval = 20000, Pob = 200, \alpha = 0,7, P_c = 0,5, P_m = 0,2, P_{dp} = 0,15, P_i = 0,15$ $Torneo = 2, w_1 = 0,8, w_2 = w_3 = 0,05\ y\ w_4 = 0,1$

2.3.3. Tests no paramétricos para el análisis estadístico de los resultados

Como hemos indicado anteriormente, en nuestro estudio experimental hemos utilizado un procedimiento de validación cruzada con 10 particiones para estimar el error del clasificador. Este procedimiento permite entrenar el clasificador con un conjunto de ejemplos independiente del conjunto de prueba para cada una de las particiones, pero al mismo tiempo también permite probar el clasificador con

todos los ejemplos (si consideramos las 10 particiones). No obstante, los resultados obtenidos de este tipo de validación no son completamente independientes, ya que sus resultados no siguen una distribución normal ni tampoco presentan homogeneidad de varianza.

En la situación anterior, y siguiendo las recomendaciones hechas por Demšar en [Dem06], se debe considerar el uso de test no paramétricos. Estos test no paramétricos pueden aplicarse tanto a porcentajes de acierto de clasificación, como a ratios de error, o a cualquier otra medida que permita evaluar los métodos a estudiar, incluyendo el tamaño de los modelos. Resultados empíricos sugieren que los test no paramétricos son incluso más potentes que los paramétricos. Demšar recomienda usar un conjunto de test no paremétricos simples, seguros y robustos para llevar a cabo una comparación estadística entre clasificadores. Concretamente, nosotros hemos considerado dos métodos alternativos basados en el uso de test no paramétricos para analizar los resultados experimentales obtenidos:

- Uso de los tests de Friedman e Iman-Davenport, y del método de Holm como procedimiento a posteriori (post-hoc). Los dos primeros test se pueden usar para detectar si existen diferencias significativas entre los algoritmos de un cierto grupo. Si se detectan esas diferencias, entonces se puede emplear el test de Holm para comparar el mejor algoritmo (algoritmo de control) contra el resto.
- Utilización del test de ranking de signos de Wilcoxon para comparar directamente los resultados de dos algoritmos.

El uso de estos tests estadísticos no paramétricos se explica con mayor detalle en el Apéndice A.

2.4. Análisis experimental de GP-COACH

En esta sección, llevamos a cabo un estudio experimental para mostrar la adecuación de algunos componentes de GP-COACH:

- el mecanismo de la competición de tokens,
- el uso de los operadores genéticos de inserción y dropping condition, y

- el uso de un algoritmo basado en PG en vez de un AG tradicional.

En primer lugar, y debido a que en nuestros experimentos hemos utilizado dos MRDs diferentes, el clasico o del máximo (*Max*) y la suma normalizada (*Sum*), es necesario analizar los resultados de GP-COACH usando ambos MDRs para ver que configuración es la mejor.

Aplicamos un test de Wilcoxon para determinar el mejor MRD (el análisis estadístico se ha realizado considerando el porcentaje de acierto en prueba), mostrándose los resultados en la Tabla 2.6, donde R^+ y R^- indican los rankings obtenidos por los MRDs clásico y de la suma, respectivamente. La mejor configuración (resaltada en negrita) es aquella que presenta el mayor valor de ranking.

Tabla 2.6: Test de Wilcoxon para determinar el mejor MRD en GP-COACH, $p = 0.05$

$\mathbf{R}^+$ (MRD Max)	$\mathbf{R}^-$ (MRD Sum)	**Valor Crítico**	**Dif. sig.?**
7,5	**292,5**	81	Si

El test de Wilcoxon detecta que existen diferencias significativas en el uso del MRD de la suma normalizada. Por lo tanto, sólo consideraremos este MRD para analizar los diferentes componentes de GP-COACH. En concreto, hemos analizados los siguientes:

1. *Influencia de la Competición de Tokens*: Hemos realizado un estudio comparativo entre el algoritmo GP-COACH y una modificación de este que no contiene el mecanismo de mantenimento de la diversidad de la competición de tokens (denominado *GP-COACH Sin Comp. Tokens*).

2. *Efectividad de los operadores genéticos de Inserción y Dropping Condition*: Se ha creado un algoritmo GP-COACH modificado que no tiene estos operadores genéticos (denominado *GP-COACH Sin Inser. Ni Drop.*).

3. *Ventajas de usar un algoritmo basado en PG para el aprendizaje de reglas difusas en vez de usar un AG tradicional*: Se ha implementado un nuevo algoritmo basado en un AG para el aprendizaje de reglas difusas DNF (denominado *GA-COACH*). Este algoritmo codifica una única regla difusa DNF por cromosoma, utilizando codificación binaria. Como operadores genéticos se ha utizado un operador de cruce uniforme y un operador de mutación que cambia aletoriamente el valor de un gen (no se ha considerado el uso de los operadores especificos de inserción y dropping condition).

Los resultados obtenidos se muestran en la Tabla 2.8, donde $\overline{\%Entr}$ y $\overline{\%Pru}$ representan el porcentaje de acierto en entrenamiento y prueba, respectivamente.

En la Tabla 2.7 se muestra el análisis estadistico realizado mediante el test de Wilcoxon (considerando el porcentaje de acierto en prueba), el cual muestra la robustez de nuestro algoritmo GP-COACH tradicional, al ser este estadísticamente mejor que el resto de las propuestas desarrolladas para analizar sus componentes.

Tabla 2.7: Test de Wilcoxon para analizar algunos componentes de GP-COACH, $p = 0{,}05$

GP-COACH$_{Sum}$ vs	**R$^+$**	**R$^-$**	**Valor crítico**	**Dif. sig.?**
GP-COACH$_{Sum}$ Sin Comp. Tokens	**300,0**	0,0	81	Si
GP-COACH$_{Sum}$ Sin Inser. Ni Drop.	**265,5**	34,5	81	Si
GA-COACH$_{Sum}$	**262,0**	38,0	81	Si

2.5. Análisis comparativo con otros métodos de aprendizaje de SCBRDs

En esta sección, se analizan estadisticamente la compacticidad y precisión de GP-COACH, comparando sus resultados con los obtenidos por otras propuestas de aprendizaje de SCBRDs bien conocidas en la literatura especializada.

En primer lugar, tenemos que determinar el mejor MRD para cada uno de los métodos de aprendizaje de SCBRDs considerados en esta comparativa (con excepción de el método de Tsakonas, donde se ha usado su propio MRD [Tsa06]). Para ello, hemos aplicado un test the Wilcoxon (considerando el porcentaje de acierto en prueba), mostrándose los resultados en la Tabla 2.9 donde R^+ y R^- se corresponden con los rankings obtenidos por el MRD clásico y de la suma normalizada, respectivamente. Las mejores configuraciones (resaltadas en negrita) son aquellas que han obtenido los valores más altos.

Tabla 2.8: Resultados del análisis de componentes de GP-COACH

	GP-COACH$_{Max}$		**GP-COACH**$_{Sum}$		**GP-COACH**$_{Sum}$ **Sin Comp. Tokens**		**GP-COACH**$_{Sum}$ **Sin Inser. Ni Drop.**		**GA-COACH**$_{Sum}$	
Conj. Datos	**%Entr**	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**
Bupa	66,94	61,58	69,04	63,63	58,54	57,13	65,40	60,64	66,12	62,10
Cleveland	63,91	54,31	64,93	55,23	53,84	53,52	63,14	56,79	67,60	52,75
Ecoli	83,10	77,52	83,58	77,72	50,64	50,31	77,80	75,01	69,19	65,88
Flare	67,70	67,38	67,70	67,45	59,02	58,81	65,14	64,41	66,73	66,01
Glass	69,68	65,63	71,26	65,33	49,21	47,93	61,44	56,88	60,87	57,05
HillValley1	53,76	52,56	53,96	52,89	51,67	50,80	53,29	51,82	55,30	53,05
HillValley2	55,59	52,99	55,68	53,99	50,63	49,56	55,26	54,15	56,78	54,76
Iris	97,78	97,56	97,78	97,56	93,90	90,67	97,14	97,56	94,89	94,00
Libras Mov.	73,88	45,28	74,23	45,56	17,69	15,93	74,85	50,00	93,02	57,87
Magic	78,82	78,78	79,78	79,82	65,64	65,64	76,45	76,32	75,75	75,74
Page-blocks	90,39	90,30	91,30	91,23	89,78	89,78	91,25	91,19	90,51	90,47
Parkinsons	88,60	84,62	89,74	86,48	75,48	75,57	87,88	84,97	86,38	82,01
Pen-based	78,78	78,77	82,29	82,20	35,40	35,44	78,84	78,54	74,56	74,18
Pima	75,93	73,90	77,02	74,37	65,01	65,06	76,93	73,68	74,60	73,33
Quadruped	100,00	100,00	100,00	100,00	77,26	77,25	100,00	100,00	99,28	98,67
Ringnorm	86,95	86,75	91,24	91,13	64,22	64,10	77,68	77,40	86,67	86,10
Satimage	66,58	66,51	72,83	72,50	41,06	40,96	72,40	72,25	64,27	64,02
Segment	84,87	84,78	86,55	85,96	42,42	42,66	74,32	73,62	66,41	66,32
Sonar	76,16	67,12	80,25	67,46	62,27	58,42	78,49	65,83	79,72	60,40
Spambase	77,34	76,60	83,17	82,80	63,56	63,64	76,06	76,10	88,44	81,07
Twonorm	77,25	76,70	85,42	84,83	78,61	78,17	85,16	84,77	88,03	88,25
Wdbc	94,17	92,02	95,09	93,90	88,68	87,81	93,12	91,97	92,57	90,51
Wine	97,94	94,31	98,96	95,10	89,57	83,89	96,36	89,66	91,84	89,48
Yeast	41,84	40,67	49,74	48,56	34,33	33,65	42,46	41,71	43,77	42,34
MEDIA	77,00	73,61	79,23	75,65	60,77	59,86	75,87	72,72	76,39	71,93

Tabla 2.9: Test de Wilcoxon para determinar el mejor MDR, $p = 0{,}05$

Algoritmo	**R^+ (MRD Max)**	**R^- (MRD Sum)**	**Valor crítico (p = 0,05)**	**Dif. sig.?**
PCA-Ravi	149,0	**151,0**	81	No
2SLAVE	86,5	**213,5**	81	No
GCCL-Ishibuchi	**197,0**	103,0	81	No
FRBCS_GP	21,5	**278,5**	81	Si

El test de Wilcoxon sólo detecta diferencias significativas en el uso de un MRD u otro en el método FRBCS_GP. Sin embargo, en nuestro estudio consideraremos para cada método aquella configuración en la que el MRD presente el valor más alto (independientemente de que existan diferencias significativas o no).

Nuestro estudio se ha dividido en tres partes diferentes: En la primera se lleva a cabo un análisis estadístico con respecto a la precisión de los seis métodos de aprendizaje de SCRBDs considerados en nuestro estudio experimental. El análisis estadístico de la compacticidad e interpretabilidad de los resultados obtenidos se lleva a cabo en la segunda parte. Finalmente, en la tercera parte se muestra un resumen del comportamiento en precisión y compacticidad de GP-COACH con respecto al resto de métodos, y también se muestra un ejemplo de las bases de reglas obtenidas por los distintos métodos en estudio.

2.5.1. Análisis de la precisión

Los resultados de precisión obtenidos para cada uno de los conjuntos de datos y métodos considerados en nuestro estudio experimental se muestran en la Tabla 2.11. El estudio estadístico se ha llevado a cabo considerando el porcentaje de acierto en prueba obtenido por los diferentes métodos.

En la Figura 2.6 se muestran los valores de los rankings obtenidos por cada algoritmo mediante el test de Friedman. Cada columna representa el ranking medio obtenido por un algoritmo. En concreto, si un cierto algoritmo logra los rankings 1, 3, 1, 4 y 2 sobre cinco conjuntos de datos, entonces su ranking medio será $\frac{1+3+1+4+2}{5} = \frac{11}{5}$. La altura de la columna es proporcional al ranking y *cuanto más baja sea una columna, entonces mejor es su algoritmo asociado.*

Aplicamos entonces los test de Friedman e Iman-Davenport (considerando un nivel de significancia $\alpha = 0{,}05$) para ver si existen diferencias estadísticas entre

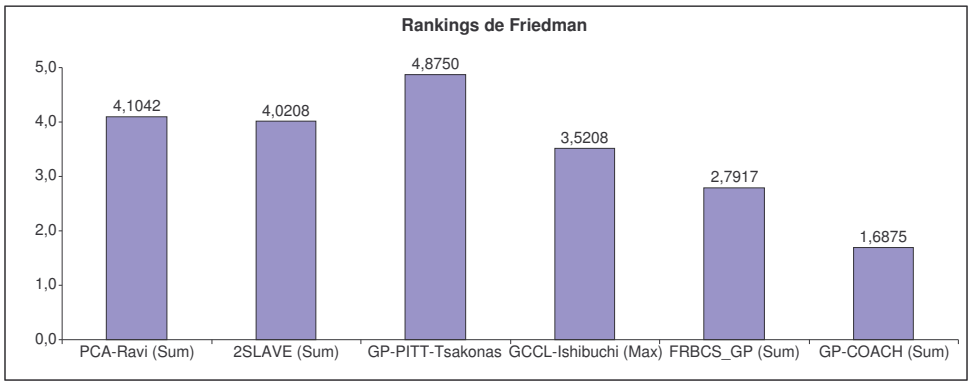


Figura 2.6: Rankings de Friedman (precisión)

los métodos. En la Tabla 2.10 se muestran los resultados de aplicar ambos tests.

Tabla 2.10: Estadísticos y valores críticos para los tests de Friedman e Iman-Davenport (precisión), $\alpha = 0{,}05$

Estadístico de Friedman	**Valor crítico**	**Hipótesis**
43,2976	11,0705	Rechazada
Estadístico de Iman-Davenport	**Valor crítico**	**Hipótesis**
12,9832	2,2932	Rechazada

Los valores de los estadísticos de Friedman e Iman-Davenport son mayores que sus correspondientes valores críticos, lo que indica que la hipótesis de equivalencia de resultados se rechaza, y que por tanto existen diferencias significativas entre los métodos estudiados. Es necesario aplicar un test a posteriori para detectar si el algoritmo de control (GP-COACH en este caso, ya que es el algoritmo que ha obtenido el ranking más bajo en el test de Friedman) es significativamente mejor que el resto de algoritmos. En la Tabla 2.12 se muestran todas las posibles hipótesis de comparación entre el algoritmo de control y el resto, ordenadas por su valor p y con un nivel de significancia α.

El método de Holm rechaza todas las hipótesis. Por lo tanto, y de acuerdo a dicho método, el algoritmo de control (GP-COACH) es estadísticamente mejor (con un valor de $p = 0{,}05$) que el resto de métodos desde el punto de vista de la precisión.

Tabla 2.11: Resultados de precisión de GP-COACH y otros métodos de aprendizaje de SBCRDs

Conj. Datos	PCA-RAVI$_{Sum}$		2SLAVE$_{Sum}$		GP-PITT-TSAK.		GCCL-ISHIB.$_{Max}$		FRBCS_GP$_{Sum}$		GP-COACH$_{Sum}$	
	%Entr	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**	**%Entr**	**%Pru**
Bupa	67,08	54,46	64,75	58,58	59,80	56,45	60,36	58,27	64,49	62,20	69,04	63,63
Cleveland	79,97	49,24	54,18	46,19	61,93	56,46	62,83	54,15	60,91	56,69	64,93	55,23
Ecoli	86,19	55,46	58,18	57,49	45,77	43,94	74,47	71,17	81,22	76,75	83,58	77,72
Flare	67,65	66,48	43,54	42,64	67,90	67,23	67,31	65,92	64,81	64,36	67,70	67,45
Glass	74,18	46,56	49,29	44,39	48,03	45,12	69,63	60,69	61,28	56,61	71,26	65,33
HillValley1	52,35	51,48	52,52	51,76	50,86	49,97	20,26	20,02	50,43	49,78	53,96	52,89
HillValley2	52,00	50,85	52,53	51,21	50,57	49,20	28,62	28,00	51,28	50,69	55,68	53,99
Iris	94,27	88,44	94,67	94,67	54,10	48,44	95,55	94,67	97,65	97,11	97,78	97,56
Libras Mov.	74,83	42,41	33,15	25,83	10,08	5,28	28,70	20,74	56,24	47,69	74,22	45,56
Magic	77,47	77,59	74,23	74,29	64,89	64,79	76,02	76,02	74,59	74,51	79,78	79,82
Page-blocks	91,15	90,70	91,40	91,42	93,03	92,92	90,41	90,34	91,24	91,09	91,30	91,23
Parkinsons	88,20	73,63	84,10	81,75	77,87	74,53	84,22	83,27	86,86	85,75	89,74	86,48
Pen-based	81,36	81,81	81,32	81,16	44,86	44,67	82,53	82,18	75,74	75,53	82,29	82,20
Pima	81,77	68,31	67,00	66,45	65,85	64,28	70,37	69,11	74,79	73,16	77,02	74,37
Quadruped	100,00	100,00	99,99	99,99	30,03	28,51	99,99	99,99	99,94	99,89	100,00	100,00
Ringnorm	38,40	30,12	80,12	79,64	50,87	50,51	91,81	91,70	94,10	93,84	91,24	91,13
Satimage	78,36	76,54	33,39	33,45	23,82	23,82	63,14	63,12	68,08	68,06	72,83	72,50
Segment	80,17	78,50	73,37	72,81	21,85	21,62	84,64	84,07	81,23	80,38	86,55	85,96
Sonar	92,90	27,65	78,45	70,72	65,24	52,42	83,49	72,40	83,30	71,15	80,25	67,48
Spambase	81,70	64,93	69,87	70,14	82,30	81,89	69,77	69,87	75,03	74,55	83,17	82,80
Twonorm	24,81	20,04	84,67	84,35	49,02	48,80	90,70	90,12	92,40	91,97	85,42	84,83
Wdbc	94,73	86,77	92,42	91,80	65,66	63,09	92,69	91,09	95,60	95,02	95,09	93,90
Wine	99,33	93,17	92,22	91,53	46,30	38,19	97,98	91,21	95,84	91,13	98,96	95,10
Yeast	58,94	39,45	15,22	14,51	32,36	31,76	50,69	49,01	52,79	52,16	49,74	48,56
MEDIA	75,74	63,11	67,52	65,70	52,62	50,16	72,34	69,88	76,24	74,17	79,23	75,66

Tabla 2.12: Tabla de Holm (precisión) (GP-COACH es el algoritmo de control)

i	Algoritmo	z	p	α/i	Hipótesis
5	GP-PITT-Tsakonas	5,9021	$3{,}5890 \cdot 10^{-9}$	0,0167	Rechazada
4	PCA-Ravi$_{Sum}$	4,4748	$7{,}6484 \cdot 10^{-6}$	0,01	Rechazada
3	2SLAVE$_{Sum}$	4,3205	$1{,}5568 \cdot 10^{-5}$	0,0125	Rechazada
2	GCCL-Ishibuchi$_{Max}$	3,3947	$6{,}8710 \cdot 10^{-4}$	0,025	Rechazada
1	FRBCS_GP$_{Sum}$	2,0445	0,0409	0,05	Rechazada

2.5.2. Análisis de la compacticidad

Una vez que hemos demostrado el buen rendimiento de nuestra propuesta en términos de precisión, en este segundo apartado nos disponemos a analizar la compacticidad e interpretabilidad de los resultados obtenidos por los diferentes algoritmos en estudio.

En primer lugar, es importante señalar que algunos de los métodos de aprendizaje de SCBRDs considerados en nuestro estudio aprenden reglas difusas DNF, mientras que otros aprenden reglas canónicas (no DNF). Como ya hemos visto en la Sección 1.4.1.1 de esta memoria, las reglas difusas DNF son un tipo especial de regla que puede englobar a varias reglas canónicas. Por lo tanto, no parece lógico comparar métodos que obtienen dos tipos de reglas diferentes. Debido a que GP-COACH aprende reglas difusas DNF, en nuestro estudio de la compacticidad sólo hemos considerado aquellos métodos que también aprenden reglas difusas DNF.

Los resultados de compaticidad se muestran en la Tabla 2.13, donde $\overline{Reg}$ es el número medio de reglas, $\overline{Var}$ el número medio de variables por regla, $\overline{Cond}$ el número medio de condiciones (etiquetas) por regla, y IC es un índice de compacticidad que hemos calculado para medir la compacticidad e interpretabilidad de una BR Para ello hemos utilizado la siguiente expresión:

$$IC = Reg_N + Var_N + Cond_N \tag{2.10}$$

donde Reg_N, Var_N y $Cond_N$ representan los valores normalizados de las medidas previamente indicadas. Dichos valores normalizados se han obtenido utilizando los valores máximos y mínimos indicados en la Tabla 2.2. El análisis estadístico se ha llevado a cabo tomando como medida el índice de compacticidad IC (cuanto más bajo es este índice, entonces mejor es la compacticidad).

Tabla 2.13: Resultados de compacticidad de GP-COACH y otros métodos de aprendizaje de SBCRDs

Conj. Datos	2SLAVE$_{Sum}$				FRBCS_GP$_{Sum}$				GP-COACH$_{Sum}$			
	Reg	**Var**	**Cond**	**IC**	**Reg**	**Var**	**Cond**	**IC**	**Reg**	**Var**	**Cond**	**IC**
Bupa	4,57	4,96	14,55	1,40	18,53	4,83	12,40	1,31	10,07	1,38	2,49	0,16
Cleveland	12,37	9,19	21,00	1,10	34,53	3,82	10,12	0,52	23,83	3,05	7,44	0,35
Ecoli	10,37	4,49	11,35	0,97	30,40	3,31	7,44	0,70	25,57	2,88	6,21	0,57
Flare	2,97	5,98	12,39	0,94	15,23	3,47	9,42	0,55	8,13	1,80	4,11	0,19
Glass	8,80	6,32	14,69	1,07	23,47	3,71	8,16	0,63	17,43	2,64	5,56	0,39
HillValley1	6,63	21,38	48,74	0,32	26,93	16,10	50,54	0,30	7,27	3,18	7,65	0,04
HillValley2	6,40	47,73	157,42	0,88	34,33	8,22	24,39	0,16	6,90	3,20	8,16	0,04
Iris	3,93	2,72	6,71	0,96	3,00	1,51	2,69	0,28	3,23	1,22	1,75	0,13
Libras Mov.	25,53	37,93	73,35	0,65	49,77	3,75	7,09	0,16	113,93	53,36	76,03	1,07
Magic	4,23	6,31	15,03	0,96	33,33	5,53	13,82	0,83	9,33	1,71	4,33	0,16
Page-blocks	7,53	6,45	15,95	0,99	39,87	3,95	8,92	0,53	14,97	1,61	3,51	0,13
Parkinsons	3,43	9,96	18,86	0,64	7,53	4,23	10,87	0,29	6,40	1,67	3,77	0,09
Pen-based	39,97	11,76	26,55	1,12	87,07	3,25	7,67	0,26	89,70	4,27	9,35	0,35
Pima	3,80	4,24	10,60	0,74	27,90	4,46	11,76	0,88	17,23	2,46	5,15	0,36
Quadruped	6,00	18,22	29,01	0,53	4,27	1,24	1,75	0,01	4,57	1,24	1,55	0,01
Ringnorm	4,60	12,54	35,27	1,02	39,60	13,19	18,38	0,87	17,50	5,45	9,90	0,35
Satimage	9,83	18,34	40,45	0,77	93,40	11,00	25,59	0,47	27,53	5,82	13,29	0,22
Segment	10,47	9,21	21,58	0,73	38,8	6,30	14,00	0,48	23,30	3,28	6,85	0,21
Sonar	9,33	15,50	28,40	0,40	20,97	8,58	25,22	0,32	14,03	2,78	6,35	0,12
Spambase	7,90	22,25	49,19	0,59	43,93	13,46	31,57	0,37	10,27	3,77	7,48	0,08
Twonorm	24,40	14,02	42,74	1,22	99,27	16,85	49,08	1,46	51,67	4,11	9,15	0,27
Wdbc	5,47	9,90	21,24	0,50	16,30	3,57	9,02	0,18	4,90	1,17	3,03	0,03
Wine	5,73	6,55	15,87	0,77	9,60	3,60	7,98	0,40	7,57	1,90	4,65	0,18
Yeast	13,27	4,69	11,50	0,86	64,03	3,29	6,98	0,56	32,20	2,99	6,44	0,47

En la Figura 2.7 se muestran los valores de rankings obtenidos por los distintos algoritmos mediante el test de Friedman. Aplicamos entonces los test de Friedman e Iman-Davenport (con un nivel de significancia $\alpha = 0{,}05$) para comprobar si existen diferencias entre los métodos estudiados.

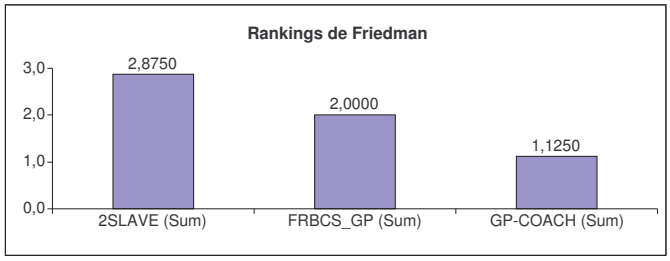


Figura 2.7: Rankings de Friedman (compacticidad)

Los valores mostrados en la Tabla 2.14 indican que tanto el estadístico de Friedman como el de Iman-Davenport son mayores que sus valores críticos asociados, y que por lo tanto la hipótesis de equivalencia de resultados se rechaza. Aplicamos entonces un test a posteriori para ver si el algoritmo de control (el que presenta menor ranking en el test de Friedman) es significativamente mejor que el resto. En la Tabla 2.15 se muestran todas las posibles hipótesis de comparación entre el algoritmo de control y el resto, ordenadas por su valor p y con un nivel de significancia α.

Tabla 2.14: Estadísticos y valores críticos para los tests de Friedman e Iman-Davenport (compacticidad), $\alpha = 0{,}05$

Estadístico de Friedman	**Valor crítico**	**Hipótesis**
36,7500	5,9915	Rechazada
Estadístico de Iman-Davenport	**Valor crítico**	**Hipótesis**
75,1333	3,1996	Rechazada

El método de Holm rechaza todas las hipótesis, lo que demuestra que GP-COACH es estadísticamente mejor que los algoritmos 2SLAVE y FRBCS_GP con respecto a la compacticidad de sus resultados, para un valor de $p = 0{,}05$.

Tabla 2.15: Tabla de Holm (compacticidad) (GP-COACH es el algoritmo de control)

i	**Algoritmo**	**z**	**p**	α/**i**	**Hipótesis**
2	2SLAVE$_{Sum}$	6,0622	$1{,}3429 \cdot 10^{-9}$	0,025	Rechazada
1	FRBCS_GP$_{Sum}$	3,0311	0,0024	0,05	Rechazada

2.5.3. Resumen

En la Tabla 2.16 mostramos un resumen del comportamiento de GP-COACH con respecto al resto de métodos considerados en el estudio. Dicha tabla contiene tres columnas: La primera indica el nombre del método, mientras que en la segunda y tercera pueden aparecer los símbolos + , - o = para indicar si el método de Holm ha detectado la existencia o no de diferencias significativas entre GP-COACH y el algoritmo especificado en la fila (el símbolo * indica que se desconoce la existencia o no de diferencias significativas debido a que el análisis estadístico no se ha podido llevar a cabo).

Tabla 2.16: Cuadro resumen del comportamiento de GP-COACH en precisión y compacticidad

Algoritmo	**Precisión**	**Compacticidad**
PCA-Ravi$_{Sum}$	+	*
2SLAVE$_{Sum}$	+	+
PG-PITT-Tsakonas	+	*
GCCL-Ishibuchi$_{Max}$	+	*
FRBCS_GP$_{Sum}$	+	+

Por lo tanto, podemos ver como GP-COACH es un método capaz de obtener SCBRDs que presentan un buen equilibrio entre interpretabilidad y precisión, ya que es el algoritmo que presenta mejor ranking y además gana estadísticamente al resto de métodos considerados en al menos uno de los dos criterios analizados (precisión y compacticidad).

Finalmente, en la Tabla 2.17 se muestra un ejemplo de las bases de reglas difusas que se han obtenido por los diferentes métodos en estudio para el conjunto de datos Iris como ejemplo ilustrativo del mencionado equilibrio entre interpreta-

Tabla 2.17: Bases de reglas difusas obtenidas para el conjunto de datos Iris

$PCA\text{-}Ravi_{Sum}$ (N. reglas: 4, %Pru: 73,33)
R^1 : *Si* P_1 *es* $L_1^{1(2)}$ *entonces Clase es* C^1
R^2 : *Si* P_1 *es* $L_1^{2(2)}$ *entonces Clase es* C^3
R^3 : *Si* P_1 *es* $L_1^{4(5)}$ *entonces Clase es* C^3
R^4 : *Si* P_1 *es* $L_1^{2(3)}$ *entonces Clase es* C^2

$2SLAVE_{Sum}$ (N. reglas: 4, %Pru: 93,33)
R^1 : *Si* X_3 *es* $(L_3^1$ *o* $L_3^5)$ *y* X_4 *es* $(L_4^1$ *o* L_4^2 *o* $L_4^5)$ *entonces Clase es* C^1
R^2 : *Si* X_3 *es* $(L_3^1$ *o* L_3^2 *o* $L_3^5)$ *y* X_4 *es* $(L_4^1$ *o* $L_4^5)$ *entonces Clase es* C^1
R^3 : *Si* X_2 *es* $(L_2^1$ *o* L_2^2 *o* L_2^3 *o* $L_2^5)$ *y* X_3 *es* $(L_3^1$ *o* L_3^2 *o* L_3^3 *o* $L_3^5)$ *y* X_4 *es* $(L_4^3$ *o* $L_4^5)$ *entonces Clase es* C^2
R^4 : *Si* X_1 *es* L_1^4 *y* X_3 *es* $(L_3^4$ *o* $L_3^5)$ *y* X_4 *es* $(L_4^4$ *o* $L_4^5)$ *entonces Clase es* C^3

GP-PITT-Tsakonas (N. reglas: 49, %Pru: 46,67)
R^1 : *Si* X_1 *es* L_1^5 *entonces Clase es* C^1
R^2 : *Si* X_1 *es* L_1^4 *entonces Clase es* C^3
...
R^{48} : *Si* X_3 *es* L_3^3 *entonces Clase es* C^2
R^{49} : *Si* X_4 *es* L_4^2 *entonces Clase es* C^1

$GCCL\text{-}Ishibuchi_{Max}$ (N. reglas: 12, %Pru: 86,67)
R^1 : *Si* X_4 *es* L_4^1 *entonces Clase es* C^1 *con GC* $1,0$
R^2 : *Si* X_4 *es* L_4^3 *entonces Clase es* C^2 *con GC* $0,82$
...
R^{11} : *Si* X_2 *es* L_2^1 *entonces Clase es* C^2 *con GC* $0,65$
R^{12} : *Si* X_1 *es* L_1^2 *y* X_3 *es* L_3^3 *entonces Clase es* C^2 *con GC* $0,92$

$FRBCS\_GP_{Sum}$ (N. reglas: 3, %Pru: 93,33)
R^1 : *Si* X_4 *es* L_4^1 *entonces Clase es* C^1 *con GC* $1,0$
R^2 : *Si* X_2 *es* $(L_2^1$ *o* L_2^2 *o* L_2^3 *o* $L_2^5)$ *y* X_3 *es* L_3^3 *y* X_4 *es* L_4^3 *entonces Clase es* C^2 *con GC* $0,95$
R^3 : *Si* X_4 *es* $(L_4^4$ *o* $L_4^5)$ *entonces Clase es* C^3 *con GC* $0,85$

$GP\text{-}COACH_{Sum}$ (N. reglas: 3, %Pru: 93,33)
R^1 : *Si* X_4 *es* L_4^1 *entonces Clase es* C^1 *con GC* $1,0$
R^2 : *Si* X_3 *es* $(L_3^2$ *o* $L_3^3)$ *y* X_4 *es* L_4^3 *entonces Clase es* C^2 *con GC* $0,96$
R_3 : *Si* X_4 *es* $(L_4^4$ *o* $L_4^5)$ *entonces Clase es* C^3 *con GC* $0,85$

bilidad y precisión[3].

2.6. Conclusiones

En este capítulo hemos propuesto un método basado en PG para el aprendizaje de SCBRDs compactos y precisos para problemas con alta dimensionalidad (con un elevado número de variables de entrada), denominado GP-COACH. Sus principales características son las siguientes:

[3]En el método PCA-Ravi$_{Sum}$, la variable "P_1"hace referencia a una nueva variable extraida (usando el Análisis de Componentes Principales, PCA) a partir de las variables originales (X_i). Por otra parte, los números entre paréntesis en las etiquetas L indican el número de conjuntos difusos por variable, ya que el método de Ravi utiliza simultaneamente cuatro particiones difusas para cada atributo con 2, 3, 4 y 5 etiquetas lingüísticas, respectivamente.

- Representa el conocimiento extraido mediante reglas difusas DNF.
- Utiliza una gramática libre de contexto para aprender dichas reglas difusas DNF, que permite la ausencia de alguna de las variables de entrada en las reglas, lo que nos lleva a obtener reglas que tienen pocas variables y etiquetas lingüísticas en sus antecedentes.
- Sigue el enfoque *GCCL* de codificación de reglas, el cual codifica una única regla por individuo, estando la BR formada por todos los individuos (reglas) de la población.
- Incluye un mecanismo para incrementar la diversidad de la población, la *Competición de Tokens*, el cual hace que las reglas compitan entre si durante el proceso evolutivo. Esto permite obtener conjuntos de reglas compactos que también presentan una alta capacidad de generalización.
- Usa un proceso de inferencia jerárquico con dos niveles, el cual nos permite mejorar la precisión de las BR obtenidas, al evitar errores de clasificación generados por el uso de reglas muy específicas (*reglas secundarias*) que cubren pocos ejemplos, cuando otras reglas más generales (*reglas primarias*) se pueden usar. Es más, se ha podido comprobar que en la mayoría de los problemas esas reglas secundarias no suelen formar parte de los conjuntos de reglas obtenidos al final del proceso evolutivo, ya que dichas reglas suelen eliminarse en las primeras etapas del proceso evolutivo debido a la acción de la Competición de Tokens.

Para analizar nuestro método, se ha considerando el uso de varios conjuntos de datos de alta dimensionalidad en un estudio experimental dividido en dos partes distintas:

1. El análisis de la adecuación de varios componentes de nuestro método:
   - el uso de la competición de tokens,
   - el uso de los operadores genéticos de inserción y dropping condition, y
   - el uso de la PG en vez de un AG tradicional.

   Se han utilizado tests estadísticos no paramétricos para validar los resultados obtenidos, mostrándose la robustez de GP-COACH.

2. Un estudio comparativo de los resultados obtenidos por GP-COACH y los obtenidos por otros cinco métodos de aprendizaje de SCBRDs bien conocidos en la literatura especializada.

   Al igual que en el estudio anterior, también se han utilizado tests estadísticos no paramétricos para comparar y analizar la compacticidad y precisión de los SCBRDs obtenidos. En este sentido, se han derivado las siguientes conclusiones:

   a) Nuestra propuesta mejora en precisión al resto de los algoritmos estudiados. Se ha demostrado estadísticamente que GP-COACH es capaz de aprender SCBRDs que tienen una alta capacidad de generalización para problemas con alta dimensionalidad.

   b) GP-COACH es capaz de aprender SCBRDs compactos e interpretables para problemas de alta dimensionalidad, mejorando sus resultados a los obtenidos por otros métodos (2SLAVE y FRBCS_GP) que también aprenden reglas difusas DNF.

Capítulo 3

Aprendizaje coevolutivo de sistemas de clasificación basados en reglas difusas compactos y precisos para problemas con alta dimensionalidad

En este capítulo se describe una propuesta coevolutiva para el aprendizaje de SCBRDs compactos y precisos para problemas con alta dimensionalidad, el algoritmo GP-CO^{2}ACH (Genetic Programing based COevolutionary learning of COmpact and ACcurate fuzzy rule based classification systems for High dimensional problems). Se trata de un algoritmo que aprende de forma simultanea toda la BC mediante la coevolución paralela (GP-CO^{2}ACH-P) o secuencial (GP-CO^{2}ACH-S) de dos especies distintas que cooperan entre si:

1. Una población de reglas difusas DNF que formarán la BR del problema a resolver.
2. Una población de cromosomas, cada uno de ellos codificando una posible

definición para la BD mediante el esquema de representación de 2-tuplas lingüísticas[1].

Para mostrar la eficacia de GP-CO^{2}ACH, se ha llevado a cabo un estudio experimental con problemas de clasificación de alta dimensionalidad, donde los resultados obtenidos por dicho algoritmo se han comparado con los obtenidos por:

a. El algoritmo GP-COACH, que sólo aprende BRs lingüísticas con partición difusa predeterminada ya que la BD se define inicialmente y permanece invariable a lo largo de todo el proceso evolutivo.

b. Un nuevo algoritmo, al que denominaremos *GP-COACH + Ajuste Lateral*, resultante de aplicar un proceso a posteriori de ajuste lateral (usando la representación de 2-tuplas lingüísticas) de la BD inicialmente definida, una vez que ha terminado la ejecución del algoritmo GP-COACH.

Finalmente, y al igual que en capítulo anterior, hemos considerado el uso de tests estadísticos no paramétricos para validar los resultados obtenidos, tanto desde el punto de vista de la precisión como de la compacticidad de los SCBRDs obtenidos.

3.1. Introducción

De entre las distintas tareas que deben llevarse a cabo en el diseño de cualquier algoritmo de aprendizaje de SCBRDs, la más importante (y a la vez dificil) consiste en derivar una BC (BR y BD) apropiada al problema que se pretende resolver. Esta tarea se vuelve más crucial aun si cabe cuando el problema a resolver presenta una alta dimensionalidad, ya que la BC obtenida no sólo debe representar el conocimiento de la forma más precisa posible sino que además debe de hacerlo manteniendo un buen nivel de interpretabilidad y compacticidad.

Algunas propuestas clásicas de aprendizaje de la BC utilizan un enfoque que permite alterar los valores de los distintos parámetros que definen las funciones de pertenencia de los conjuntos difusos mediante la realización de desplazamientos y/o ensanchamientos en los mismos, con el objetivo de mejorar la precisión

[1] El esquema de las 2-tuplas lingüísticas permite mantener la forma original de las funciones de pertenencia (simétricas), modificando lateralmente la localización del soporte.

del SCBRD. Sin embargo, el uso de este enfoque puede comprometer gravemente la interpretabilidad del SCBRD obtenido, ya que puede dar lugar a particiones difusas muy complejas y difíciles de interpretar por parte de un experto. Para solucionar dicho problema, se pueden imponer una serie de restricciones de diseño con el objeto de mantener un buen nivel de interpretabilidad en las BCs derivadas. No obstante, el uso de estas restricciones hacen menos flexible el proceso de extracción de la BC, lo que incrementa la complejidad dentro del espacio de búsqueda.

Recientemente, se ha propuesto el uso del esquema de representación de las 2-tuplas lingüísticas [HM00], el cual permite el desplazamiento lateral del soporte de las etiquetas considerando un único parámetro (pequeños desplazamientos a la izquierda/derecha de las funciones de pertenencia originales) con el objetivo de obtener etiquetas que contengan un conjunto de ejemplos con mejor cubrimiento (búsqueda de precisión) y que conservan su forma original (mantenimiento de la interpretabilidad), al tiempo que se reduce el espacio de búsqueda de la BC con respecto al enfoque clásico que considera 3 parámetros (en el caso de funciones de pertenencia triangulares), lo que facilita la obtención de modelos óptimos.

Por otra parte, el análisis de la literatura especializada muestra que existen dos enfoques distintos, ambos con sus ventajas e inconvenientes, a la hora de derivar la BC de un SCBRD:

1. *Derivación simultanea*: Hace referencia a aquellas propuestas que obtienen toda la BC, es decir, la BD y la BR, de forma simultanea a partir de los datos disponibles. A este enfoque se le suele denominar simplemente como *proceso de aprendizaje*.

2. *Derivación secuencial*: La tarea de derivar la BC completa se divide en dos o más etapas secuenciales, cada una de ellas llevando a cabo una derivación parcial o total de la BC. La mayoría de propuestas que siguen este enfoque llevan a cabo una etapa inicial que aprende la BR, y luego a posteriori aplican una etapa que ajusta la BD previamente aprendida/definida mediante ligeras modificaciones que permiten incrementar el rendimiento del sistema. A esta última etapa se la denomina como *proceso de ajuste*.

La derivación secuencial de la BC presenta la ventaja de que se reduce el espacio de búsqueda, puesto que se trabaja con espacios confinados en cada etapa. No obstante, el hecho de que los componentes (BR y BD) se obtengan en etapas independientes entre sí puede hacer que los SCBRDs obtenidos no lleguen a alcanzar

un equilibrio óptimo entre interpretabilidad y precisión en problemas que presentan una alta dimensionalidad. Por otro lado, en la derivación simultánea de la BC se considera de mejor forma la fuerte dependencia existente entre los componentes, lo que permite obtener SCBRDs con un mejor equilibrio entre interpretabilidad y precisión. Sin embargo, este tipo de derivación presenta el inconveniente de que el proceso se vuelve mucho más complejo debido a que el espacio de búsqueda crece significativamente, haciendo fundamental la elección de una técnica de búsqueda apropiada.

En los últimos años, el interés por el paradigma de la coevolución [Par95] ha crecido gracias a la habilidad que esta técnica evolutiva presenta para manejar espacios de búsqueda de gran dimensión y para tratar con problemas que pueden descomponerse. En este sentido, la descomposición directa del proceso de derivación de la BC en dos componentes interdependientes que aprenden la BR y la BD, respectivamente, hace que los algoritmos coevolutivos cooperativos [PD00] se muestren como una técnica muy útil y apropiada para la tarea del aprendizaje de SCBRDs que presenten un alto nivel de equilibrio entre interpretabilidad y precisión para problemas con una alta dimensionalidad.

En la siguiente subsección se presentan algunos preliminares tales como la interpretabilidad en el aprendizaje/ajuste de la BC, el aprendizaje/ajuste de la BC mediante el esquema de representación de las 2-tuplas lingüísticas, y el funcionamiento detallado de los algoritmos coevolutivos.

3.2. Preliminares

3.2.1. Interpretabilidad en el aprendizaje/ajuste de la Base Conocimiento

La forma más común de obtener la BD dentro del proceso de derivación de la BC de un SCBRD consiste en alterar los valores de los distintos parámetros que definen las funciones de pertenencia de los conjuntos difusos mediante la realización de desplazamientos y/o ensanchamientos en los mismos. Esto se puede conseguir aprendiendo/ajustando directamente cada uno de los parámetros de dichos conjuntos. Por ejemplo, si consideramos la siguiente función de pertenencia triangular:

$$\mu(x) = \begin{cases} \frac{x-a}{b-a}, & si\ a \leq x < b \\ \frac{c-x}{c-b}, & si\ b \leq x \leq c \\ 0, & en\ otro\ caso \end{cases}$$

alterar los parámetros a, b y c supone variar la forma del conjunto difuso asociado a la función de pertenencia (véase la Figura 3.1), afectando así al comportamiento del SCBRD. Lo mismo ocurre en el caso de los demás tipos de funciones de pertenencia (trapezoidales, gaussianas, sigmoidales, etc.).

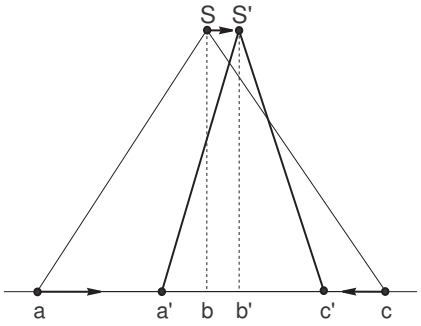


Figura 3.1: Definición de conjuntos difusos alterando sus parámetros básicos

Sin embargo, este tipo de aproximación al aprendizaje/ajuste de los conjuntos difusos puede comprometer gravemente la interpretabilidad del SCBRD, ya que el uso de este enfoque puede dar lugar a particiones difusas muy complejas en la BD, como la mostrada en la Figura 3.2, que hacen dificil la interpretación del sistema por parte de un experto.

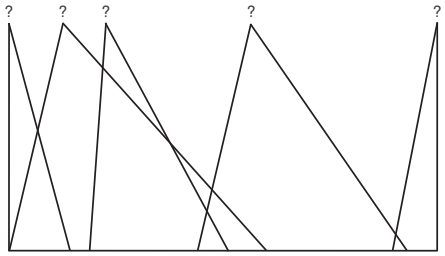


Figura 3.2: Perdida de interpretabilidad en la partición difusa

Una posible solución para asegurar una buena interpretabilidad durante el proceso de aprendizaje/ajuste de los conjuntos difusos consiste en imponer una serie de restricciones de diseño con objeto de obtener una BD que mantenga la comprensibilidad del sistema en la mayor medida posible. Algunas de las propiedades más importantes que han de satisfacer las BDs aprendidas/ajustadas son:

- Propiedad de cobertura: Cada valor del universo de discurso ha de pertenecer a, al menos, un conjunto difuso. Alternativamente, se puede considerar un criterio más estricto exigiendo un grado mínimo de cobertura para todos los elementos del dominio.
- Propiedad de normalidad: Cada función de pertenencia debe tener un grado de pertenencia máximo con, al menos, un elemento del universo de discurso. Es decir, los conjuntos difusos deben ser normales.
- Propiedad de distinguibilidad: Cada término lingüístico debe tener un significado claro y su conjunto difuso asociado debe especificar claramente un intervalo del universo de discurso. En resumen, las funciones de pertenencia de la partición difusa deben ser lo suficientemente distintas entre sí.

La introducción de estas restricciones necesarias para asegurar la integridad semántica de la BD hacen menos flexible el proceso de derivación de la misma, al tiempo que incrementan su complejidad. Por otra parte, dicha complejidad también aumenta a medida que crece el número de variables y el número de etiquetas por variable del problema a resolver.

3.2.2. Aprendizaje/ajuste de la Base de Conocimiento mediante el esquema de representación de las 2-tuplas lingüísticas

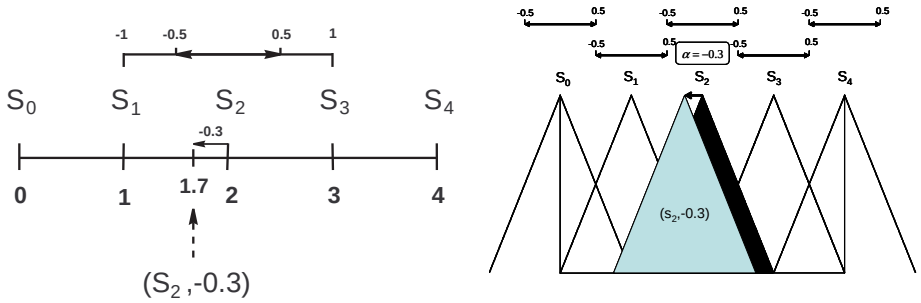
Una alternativa al enfoque clásico de aprendizaje/ajuste de la BD, y que evita los problemas de falta de interpretabilidad y aumento de la complejidad derivados del mismo, consiste en el uso de la representación de las 2-tuplas lingüísticas, la cual se basa en el concepto de translación simbólica [HM00].

La translación simbólica de una etiqueta lingüística es un número (α) dentro de un intervalo $[-\delta, \delta)$, determinando este intervalo el dominio de la etiqueta cuando esta es movida entre sus dos etiquetas laterales adyacentes (ver la Figura 3.3(a)).

Consideremos un conjunto de etiquetas S ($S = S_1, S_2, \ldots, S_n$) representando una partición difusa. Formalmente, para representar la translación simbólica de una etiqueta en S tenemos el par,

$$(S_i,\ \alpha_i),\ S_i\ \in\ S,\ \alpha_i\ \in\ [-\delta,\ \delta).$$

La translación simbólica de una etiqueta implica el desplazamiento lateral de la función de pertenencia asociada. Como ejemplo, la Figura 3.3 muestra la translación simbólica de una etiqueta representada por el par $(S_2, -0{,}3)$ junto con el desplazamiento lateral de la función de pertenencia correspondiente.



(a) Translación simbólica de una etiqueta, $\delta = 0{,}5$

(b) Desplazamiento lateral de una función de pertenencia

Figura 3.3: Translación simbólica de una etiqueta lingüística y desplazamiento lateral de la función de pertenencia asociada

Por lo tanto, el esquema de representación de las 2-tuplas lingüísticas reduce el espacio de búsqueda al utilizar un único parámetro por etiqueta lingüística, consiguiéndose así una mejora potencial de la precisión del sistema al tiempo que se mantiene la interpretabilidad en un alto grado. De esta forma se alcanzan los dos objetivos principales que son el obtener una partición difusa que cubra al conjunto de ejemplos con el mejor grado posible (mejora de la precisión), y que dicha partición difusa conserve su forma original (mantenimiento de la interpretabilidad) al tiempo que se reduce el espacio de búsqueda del proceso de aprendizaje/ajuste de la BD con respecto al enfoque clásico.

3.2.3. Algoritmos Coevolutivos

Los algoritmos coevolutivos [Par95] son técnicas evolutivas avanzadas propuestas para resolver problemas complejos que se pueden descomponer. Estos se componen de dos o más especies (poblaciones) que interactúan entre ellas mediante una función de adaptación conjunta. De ese modo, a pesar de que cada especie tiene su propio esquema de codificación y operadores de reproducción, a la hora de evaluar un individuo, su bondad se calcula considerando algunos individuos de las otras especies. Esta coevolución hace más fácil encontrar buenas soluciones en problemas con espacios de búsqueda complejos.

A la hora de diseñar cualquier algoritmo coevolutivo, debemos plantearnos una serie de cuestiones a resolver:

1. *¿Que tipo de interacción tendrá lugar entre las especies?*

   Se pueden considerar diferentes tipos de interacciones entre las especies según las dependencias existentes entre los subcomponentes de la solución. En general, podemos mencionar dos tipos distintos de interacción, que dan lugar a dos tipos de algoritmos coevolutivos:

   - *Algoritmos coevolutivos competitivos* [RB97]: Aquellos donde cada especie compite contra el resto (por ejemplo, para obtener exclusividad de un recurso limitado). En este caso, el incremento de la adecuación de un individuo de una especie implica una disminución de la adecuación en el resto de las especies, es decir, el éxito ajeno supone el fracaso personal.
   - *Algoritmos coevolutivos cooperativos o simbióticos* [PD00]: Aquellos donde todas las especies cooperan para construir una solución al problema. En ese caso, la adecuación de un individuo dependerá de su capacidad para colaborar con individuos de otras especies.

     En los algoritmos coevolutivos cooperativos, la evolución de las distintas especies se realiza según el esquema mostrado en la Figura 3.4. En cada generación, y para cada población, se selecciona un conjunto de *cooperadores*. Para evaluar un individuo en una determinada especie, se combina su información con los cooperadores del resto de especies, formando así una solución al problema. El valor de adaptación para ese

individuo dependerá del mecanismo de asignación de crédito escogido para agregar la evaluación de cada una de las soluciones generadas.

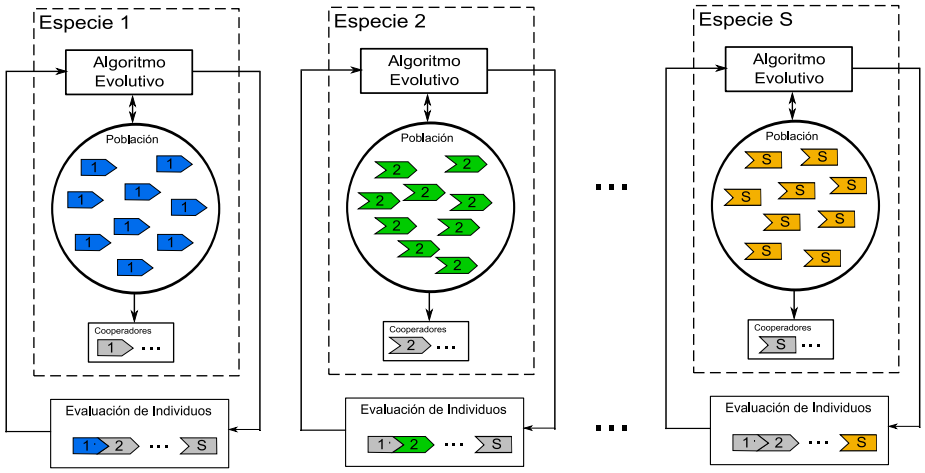


Figura 3.4: Cooperación coevolutiva

La coevolución cooperativa es recomendable cuando el problema a resolver tiene las siguientes características [PRS01]:

a) El espacio de búsqueda es complejo,

b) El problema, por definición, se puede descomponer,

c) Se manejan diferentes tipos de valores y conceptos, y

d) Hay una fuerte interdependencia entre los componentes de la solución.

El problema del aprendizaje de la mejor BC (BR y BD) para un SCBRD presenta todas estas características anteriores, por lo que la coevolución cooperativa se muestran como una herramienta adecuada la resolución óptima de dicha tarea.

2. *¿Cual será el esquema de interacción entre especies?*

Una vez determinado el tipo de interacción entre las especies (competición o cooperación), el siguiente paso en el diseño de un algoritmo coevolutivo consiste en definir la forma en que se determinarán los competidores/cooperadores de entre los individuos de cada especie, y en especificar como dichos competidores/cooperadores van a interactuar entre sí a la hora

de evaluar a los distintos individuos en cada una de las especies. Todo esto se traduce en especificar algunos atributos básicos del algoritmo coevolutivo [WLD01]:

- *El tamaño de la piscina de competición/cooperación*, es decir, el número de competidores/cooperadores de cada especie que son utilizados para obtener una evaluación de fitness de un individuo dado.

  La forma más obvia, y también más costosa desde un punto de vista computacional, de obtener el fitness de un individuo en una especie dada consiste en hacer que dicho individuo interactue con todos los competidores/cooperadores del resto de especies. En el extremo opuesto, tenemos el caso en el que la evaluación de un individuo en una especie depende exclusivamente de una única interación con competidores/cooperadores de las otras especies.

  Entre estas dos opciones extremas, existe un amplio abanico de posibilidades que implican varias interacciones en las que los competidores/cooperadores de cada especie se eligen de formas muy diversas, y que dependen de la *presión de selección*.

- *La presión de selección* es el grado en que los competidores/cooperadores son elegidos en función de su valor de fitness.

  De esta forma tenemos una fuerte presión de selección cuando todos los competidores/cooperadores se eligen de entre los mejores individuos en la especie durante la anterior generación. Por el contrario, la presión de selección será baja cuando todos los competidores/cooperadores son elegidos de forma aleatoria.

  Entre estas dos opciones, existe una amplia variedad de formas de elegir a los competidores/cooperadores que hacen que varie la presión de selección y que dependerán del número de competidores/cooperadores que son elegidos de entre los mejores y de los que son elegidos aleatoriamente.

- *El mecanismo de asignación de crédito*, es decir, la forma en que se obtiene un único valor de fitness para un individuo dado cuando en su evaluación tienen lugar varias interacciones entre competidores/cooperadores.

  Existen diversas formas de obtener un único valor cuando se llevan a cabo varias interacciones para calcular el fitness de un individuo en una especie del coevolutivo. Algunas de las más comunes son:

- **Optimista**: Es el método más tradicional de asignación de crédito, y consiste en asignar el valor de la mejor interacción del individuo como su valor de fitness.
- **Media**: El valor de fitness de un individuo se calcula como la media de los valores obtenidos para todas sus interacciones. Este tipo de mecanismo de asignación de crédito es usado normalmente en la coevolución competitiva.
- **Pesimista**: Se asigna como valor de fitness del individuo a aquel obtenido en su peor interacción.

3. *¿Cual será la frecuencia de interacción entre las especies?*

   Un aspecto importante y que es necesario especificar a la hora de diseñar un algoritmo coevolutivo es la frecuencia de interacción entre las especies, es decir, cuando y como son procesadas y actualizadas las distintas especies durante el proceso coevolutivo. En general, podemos mencionar dos formas distintas de procesamiento y actualización de las especies [Wie04]:

   - *Paralela*: Las distintas especies son todas procesadas en paralelo, por lo que todas están activas en un instante dado del proceso coevolutivo. Para evaluar el fitness de los individuos de las distintas especies se utilizan los competidores/cooperadores guardados con anterioridad para cada especie.

     Este esquema de procesamiento y actualización de las especies permite distribuir la ejecución del coevolutivo entre multiples procesadores con el objetivo de disminuir el tiempo de computación.

   - *Secuencial*: Las distintas especies del coevolutivo evolucionan por turnos, es decir, en un instante dado del proceso coevolutivo sólo una de las especies está activa, mientras que el resto permanecen como si estuvieran congeladas. La especie activa evalua sus individuos usando los competidores/cooperadores guardados en los estados congelados del resto de las especies. Cuando la especie activa termina de ser procesada, actualiza sus competidores/cooperadores y cambia su estado al de congelada, pasando a estar activa alguna de las especies que antes estaba detenida.

     En este esquema de procesamiento y actualización de las especies, los cambios que se producen en las especies procesadas en un primer lugar afectan a las especies que son procesadas con posterioridad.

Por otra parte, en [PD06] se introducen los conceptos de *puntos de interacción* entre especies, y de *época*. En concreto, se denominan puntos de interacción a los instantes dentro del proceso coevolutivo en que las distintas especies interaccionarán entre sí, y época al periodo de tiempo existente entre dos puntos de interacción consecutivos. Es importante señalar que aunque el concepto de época mide un periodo de tiempo entre interacciones, su tamaño viene expresado como un cierto número de generaciones o iteraciones dentro del proceso evolutivo de cada una de las especies del coevolutivo.

Utilizando el concepto de época, es también posible alterar la frecuencia de interacción entre las especies, simplemente haciendo que estas actualicen sus respectivos competidores/cooperadores al final de una época, es decir de varias generaciones/iteraciones en su proceso evolutivo, y no de una única generación/iteración. De esta forma basta con incrementar el tamaño de época para disminuir la frecuencia de interacción entre las especies, y viceversa.

3.3. GP-CO^{2}ACH: Genetic Programing based COevolutionary learning of COmpact and ACcurate fuzzy rule based classification systems for High dimensional problems

A continuación se presenta el modelo coevolutivo cooperativo para el aprendizaje de BCs compactas y precisas en problemas de alta dimensionalidad, denominado GP-CO^{2}ACH (Genetic Programing based COevolutionary learning of COmpact and ACcurate fuzzy rule based classification systems for High dimensional problems).

Se trata de un algoritmo en el que dos especies distintas, una de ellas aprendiendo la BR y otra la mejor definición para la BD, evolucionan en paralelo (GP-CO^{2}ACH-P) o secuencialmente (GP-CO^{2}ACH-S), cooperando entre sí para formar una solución completa a un problema dado.

A continuación se describen los procesos evolutivos utilizados respectivamente en cada una de esas dos especies, y finalmente se muestra una descripción completa de los procesos coevolutivos propuestos (paralelo y secuencial), donde se explicará

como ambas especies cooperan entre sí para obtener BCs compactas y precisas para problemas con una alta dimensionalidad.

3.3.1. Especie 1: Aprendizaje de la Base de Reglas

Dado que nuestro objetivo final es el de aprender BCs compactas y precisas, en esta primera especie del coevolutivo vamos a hacer uso del algoritmo GP-COACH para el aprendizaje de la BR, ya que como hemos visto en el capítulo anterior, este algoritmo ha demostrado ser una herramienta adecuada para la obtención de BR compactas y precisas en problemas con una alta dimensionalidad.

Como el funcionamiento del algoritmo GP-COACH ya ha sido presentado con todo detalle en la sección 2.2 del capítulo anterior de la presente memoria, nos centraremos en mostrar como la segunda especie aprende la mejor definición de la BD mediante la representación de 2-tuplas lingüísticas.

3.3.2. Especie 2: Aprendizaje de la Base de Datos

Como hemos indicado anteriormente, esta segunda especie será la encargada de aprender una BD mediante el uso del esquema de representación de 2-tuplas lingüísticas, con el fin de obtener una BC que presente un buen nivel de interpretabilidad al tiempo que se mejora la precisión.

En concreto, esta especie hace uso de un AG específico, el algoritmo CHC [Esh91], que es un AG que presenta un buen equilibrio entre exploración y explotación, siendo una buena elección en problemas con espacios de búsqueda complejos. Este modelo hace uso de un proceso de *Selección basado en Población* para realizar una búsqueda global adecuada, el cual para formar la siguiente población selecciona los M mejores individuos de entre los M padres y sus correspondientes hijos. Para introducir diversidad en la búsqueda, en lugar del bien conocido operador de mutación, CHC hace uso de dos mecanismos: un mecanismo de prevención de incesto y un proceso de reinicialización.

El mecanismo de prevención de incesto se utiliza para determinar cuándo se aplica el operador de cruce, es decir, dos padres se cruzan si su distancia de hamming dividida entre 2 es superior a un valor umbral establecido por el experto, L.

Dado que vamos a considerar un esquema de codificación real (ver sec-

ción 3.3.2.1), necesitamos transformar cada gen mediante un Código Gray para poder calcular la distancia de hamming. Así, el valor umbral es inicializado a la mayor distancia posible entre dos padres dividida por cuatro, y es decrementado en uno cada vez que no se incluye un nuevo individuo en la población en una generación. El algoritmo aplica el mecanismo de reinicialización cuando L llega a un valor menor que 0.

El pseudocódigo del algoritmo CHC considerado para esta especie se muestra a continuación (un esquema del mismo se muestra en la figura 3.5):

1. Se genera la población inicial con M cromosomas e inicializamos el valor umbral L.
2. Se aplica el operador de cruce para generar los descendientes, teniendo en cuenta el mecanismo de prevencion de incesto.
3. Se ejecuta el proceso de selección basado en población para generar la nueva población.
4. Se actualiza el valor umbral si no hay nuevos individuos en la población.
5. Si L es menor que 0 aplicamos el proceso de reinicialización.
6. Si no se verifica la condición de parada se vuelve al paso 2, en otro caso se acaba.

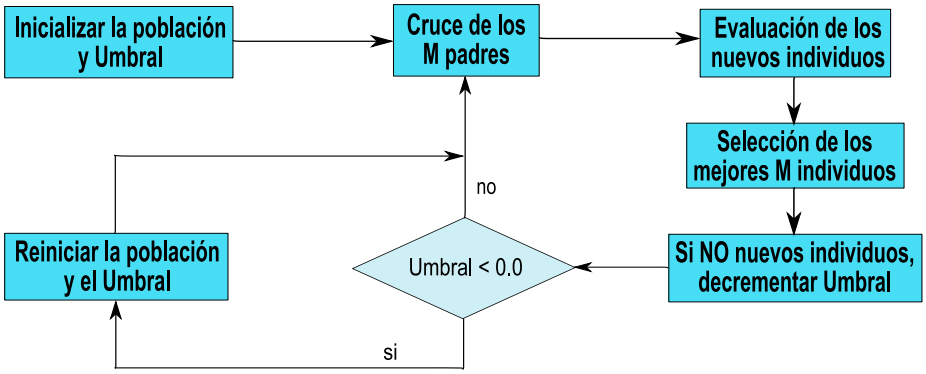


Figura 3.5: Esquema de CHC

En las siguientes subsecciones veremos en detalle cada uno de los componentes necesarios para diseñar este proceso evolutivo:

- codificación de la BD,
- evaluación de los cromosomas,
- generación de la población inicial,
- operador de cruce (junto con la prevención de incesto considerada) y
- mecanismo de reinicialización.

3.3.2.1. Codificación de la Base de Datos

Debido a que estamos usando el esquema de representación de 2-tuplas lingüísticas, el cual hace uso de un único parámetro por cada función de pertenencia en la BD, cada cromosoma consistirá en un vector de números reales con la siguiente forma:

$$C = (c_1^1, \ldots, c_1^{L_1}, c_2^1, \ldots, c_2^{L_2}, ..., c_{n_v}^1, \ldots, c_{n_v}^{L_{n_v}})$$

donde n_v es el número de variables de entrada, L_i es el número de etiquetas lingüísticas de la variable i, y cada gen c_i^j es un número real dentro del intervalo $[-\delta, \delta]$ que representa el desplazamiento lateral para la etiqueta j de la i-ésima variable.

3.3.2.2. Evaluación de los cromosomas

El valor de fitness asignado a cada uno de los cromosomas de la población del algoritmo CHC se define como el porcentaje de acierto en entrenamiento ($\%Entr$) obtenido por el SCBRD resultante de unir la BD codificada en el cromosoma y la BR formada por todos los individuos (reglas) presentes en la población de la otra especie de nuestro modelo coevolutivo (BR que como ha indicado anteriormente se obtiene utilizando el algoritmo GP-COACH).

3.3.2.3. Población inicial

La población inicial se crea de la siguiente manera:

- Un primer cromosoma con todos sus genes con el valor '0,0', es decir, representando a la partición original sin ningún tipo de desplazamiento.
- El resto de cromosomas se crean asignando a cada gen un valor aleatorio en el intervalo $[-\delta, \delta]$.

3.3.2.4. Operador de cruce

En nuestro algoritmo hemos utilizado un operador basado en el concepto de entornos, es decir, en el que los descendientes son generados alrededor de los padres (explotación). Este tipo de operadores presentan una buena cooperación cuando son introducidos en modelos evolutivos en los que se fuerza la convergencia presionando sobre los descendientes (como es el caso de CHC). Particularmente, nosotros hemos utilizado el operador *Parent Centric BLX* (PCBLX) [HLS03], basado en el operador BLX-α [ES93]. La Figura 3.6 muestra el comportamiento de este tipo de operadores.

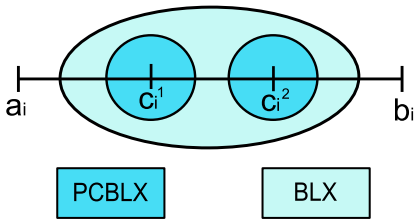


Figura 3.6: Esquema del comportamiento de los operadores BLX y PCBLX

El operador PCBLX se describe como sigue. Supongamos que $X = (x_1 \cdots x_{n_v})$ y $Y = (y_1 \cdots y_{n_v})$, $(x_i, y_i \in [a_i, b_i] \subset \Re, i = 1 \cdots n_v)$, son dos cromosomas con codificación real que quieren ser cruzados. El operador PCBLX genera los siguientes dos descendientes:

- $O_1 = (o_{11} \cdots o_{1n_v})$, donde o_{1i} es un valor elegido aleatoriamente (uniformemente) dentro del intervalo $[l_i^1, u_i^1]$, con $l_i^1 = max\{a_i, x_i - I_i\}$, $u_i^1 = min\{b_i, x_i + I_i\}$, y $I_i = \mid x_i - y_i \mid$.
- $O_2 = (o_{21} \cdots o_{2n_v})$, donde o_{2i} es un valor elegido aleatoriamente (uniformemente) dentro del intervalo $[l_i^2, u_i^2]$, con $l_i^2 = max\{a_i, y_i - I_i\}$ y $u_i^2 = min\{b_i, y_i + I_i\}$.

Por otro lado, el mecanismo de prevención de incesto solo se considera para determinar si se aplica o no el operador PCBLX. En nuestro caso, dos padres serán cruzados si su distancia de hammming dividida por 2 es mayor que un umbral predeterminado, L. Al considerar un esquema de codificación real necesitamos transformar cada gen considerando un Código Gray (código binario) con un número fijo de bits por gen ($BITSGENE$), que es determinado por el experto del sistema. Así, el valor umbral es inicializado como:

$$L = (\#Genes \cdot BITSGENE)/4{,}0.$$

donde $\#Genes$ es igual a la longitud total del cromosoma. Siguiendo el esquema original de CHC, L es decrementado en uno cada vez que no se introduce un nuevo individuo en la población en una generación. Debido a que la convergencia es muy lenta, en nuestro caso, L también será decrementada en uno cuando no se consiga ninguna mejora respecto al mejor cromosoma de la generación anterior.

3.3.2.5. Mecanismo de reinicialización

Al no utilizarse un operador de mutación se considera un mecanismo de reinicialización para evitar los óptimos locales [Esh91], que será aplicado cuando el valor umbral L sea menor que cero. En este caso, se generan aleatoriamente todos los cromosomas de la población con sus genes tomando un valor dentro del intervalo $[-\delta, \delta]$. Además, la mejor solución global encontrada hasta el momento por el algoritmo CHC también se incluye en la población con el fin de incrementar la convergencia del algoritmo.

3.3.3. Descripción de nuestra propuesta coevolutiva en paralelo (GP-CO²ACH-P)

Nuestra propuesta coevolutiva en paralelo, basada en las ideas originales propuestas en [PD00], comienza creando una población inicial para cada una de las dos especies: Por un lado, se genera una población de reglas difusas de acuerdo a las reglas de producción de la gramática libre de contexto (ver sección 2.2), obteniéndose así la BR inicial. Por otra parte, se inicializa también la población de cromosomas del algoritmo CHC, según se ha indicado en la sección 3.3.2.3.

Una vez creadas ambas poblaciones iniciales, es necesario evaluar los individuos

de ambas especies. En concreto, cada individuo i en la población inicial de reglas difusas se evalua mediante la función de fitness definida en 2.2.2 y haciendo uso de la partición difusa original sin desplazamientos, mientras que el fitness de cada individuo j del algoritmo CHC se obtiene como el porcentaje de acierto en entrenamiento ($\%Entr$) obtenido por el SCBRD resultante de unir la BD codificada en el cromosoma j y la BR formada por todos los individuos (reglas) presentes en la población actual de la otra especie.

A continuación, se calcula el valor de la función de fitness global (sección 2.2.3) para la mejor BC inicial, es decir, la formada por la BR inicial y el mejor cromosoma del algoritmo CHC (que contiene la mejor definición de la BD). Dicha BC inicial se guarda como la mejor de todo proceso evolutivo.

Entonces, y mientras no se verifique la condición de parada, se aplica el siguiente proceso evolutivo:

1. Se lleva a cabo una evolución en paralelo de ambas especies, aplicando una única iteración de sus respectivos procesos evolutivos.

2. Se evaluan los individuos de ambas especies: Cada individuo i de la BR actual se evalua usando el mejor individuo del algoritmo CHC de la iteración anterior (ya que se desconoce cual es el mejor individuo de la iteración actual al estar siendo evaluados paralelamente los individuos del algoritmo CHC). El fitness de cada individuo j de la población del algoritmo CHC se obtiene como el porcentaje de acierto del SCBRD formado por la definición de la BD codificada en él y la BR actual. En la Figura 3.7, se muestra gráficamente como se evaluan los nuevos individuos de ambas especies.

3. Se calcula el valor de fitness global de la BC formada por la BR actual y la mejor definición actual para la BD, y si este valor mejora al de la mejor BC almacenada hasta el momento, entonces reemplazamos la mejor BC por la BC actual (y también se actualiza el valor de fitness global de la mejor BC).

Cuando el proceso coevolutivo acaba, se devuelve como solución al problema la BC que estaba guardada como la mejor.

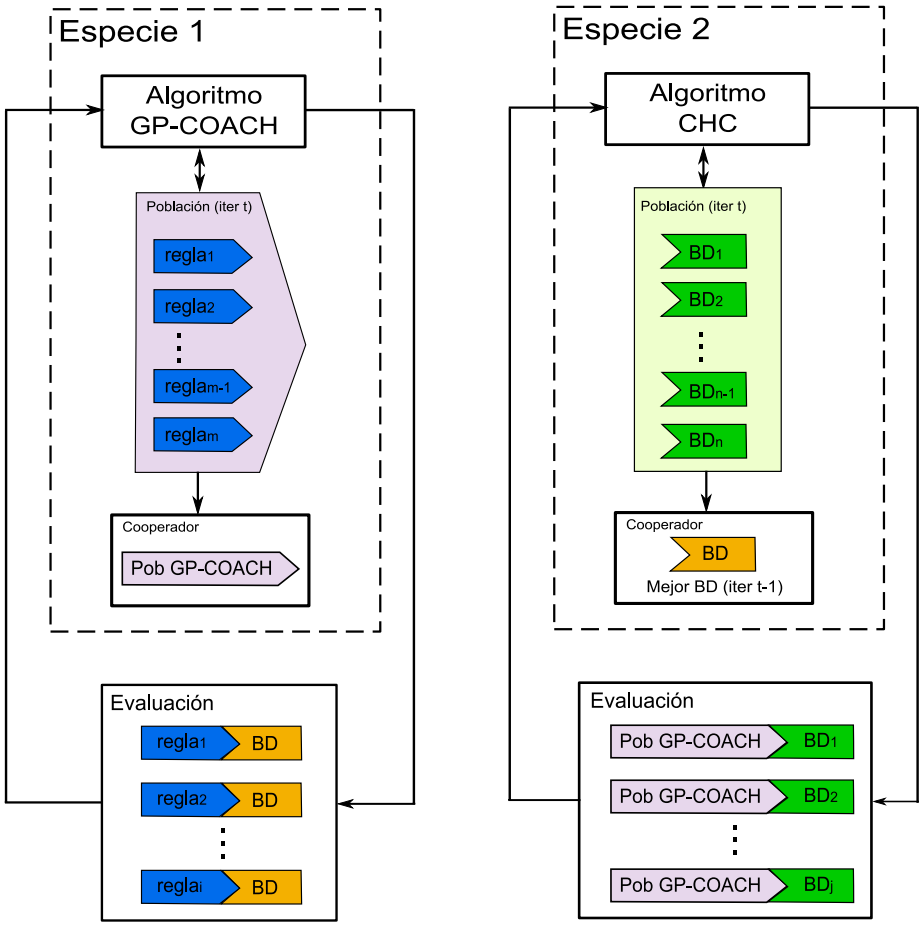


Figura 3.7: Evaluación de los individuos de ambas especies en la propuesta de cooperación coevolutiva en paralelo (GP-CO^{2}ACH-P)

3.3.4. Descripción de nuestra propuesta coevolutiva secuencial (GP-CO^{2}ACH-S)

Nuestra propuesta coevolutiva secuencial se basa en las ideas para mejorar la cooperación secuencial propuestas en [PD06], que hacen uso del concepto de *época* para variar la frecuencia de interacción entre las especies. En concreto, en nuestra propuesta hemos utilizado una época de tamaño igual a 2 iteraciones. El esquema de nuestra propuesta es el siguiente:

Se crea una población inicial para la especie que aprende la BR, en la que cada individuo (regla) se genera de forma aleatoria de acuerdo a las reglas de producción de la gramática libre de contexto. Mientras tanto, la especie que aprende la BD se queda congelada y sin inicializar, por lo que es necesario especificar algún cooperador para ella. En nuestro caso, se ha establecido como tal cooperador a la definición inicial de la BD sin ningún tipo de desplazamiento.

La especie que aprende la BR evoluciona una época completa dentro de su proceso evolutivo. Durante esa época, cada individuo (regla) i se evalua utilizando el cooperador congelado en la especie que aprende la BD.

Entonces, se calcula el valor de la función de fitness global para la BC actual, es decir, la formada por la BR actual y la definición inicial de la BD sin desplazamientos. Dicha BC actual se guarda como la mejor de todo proceso evolutivo hasta el momento.

Congelamos la especie que aprende la BR, estableciéndose como su cooperador la BR formada por todos los individuos en la población actual, y descongelamos la población del algoritmo CHC, la cual se inicializa según lo indicado en la sección 3.3.2.3.

La especie que aprende la BD evoluciona una época completa en su proceso evolutivo. Durante esa época, cada individuo j de la población del CHC se evalua utilizando el cooperador congelado en la especie que aprende la BR.

Se vuelve a calcular el valor de la función de fitness global para la BC actual, es decir, la formada por la BR congelada en la especie 1, y la mejor definición para la BD en la especie 2. En el caso de que dicha medida mejore a la de la mejor BC almacenada hasta el momento, entonces reemplazamos la mejor BC por la BC actual (y también se actualiza el valor de fitness global de la mejor BC).

Entonces, y mientras no se verifique la condición de parada, se aplica el siguiente proceso evolutivo:

1. Se congela la especie de la BD y su mejor individuo se establece como su cooperador.

2. Se activa la especie de la BR y se avanza una época en su proceso evolutivo. Cada individuo i de la BR se evalua usando el cooperador almacenado en la especie congelada de la BD. En la Figura 3.8, se muestra gráficamente como se evaluan los nuevos individuos de la especie que aprende la BR.

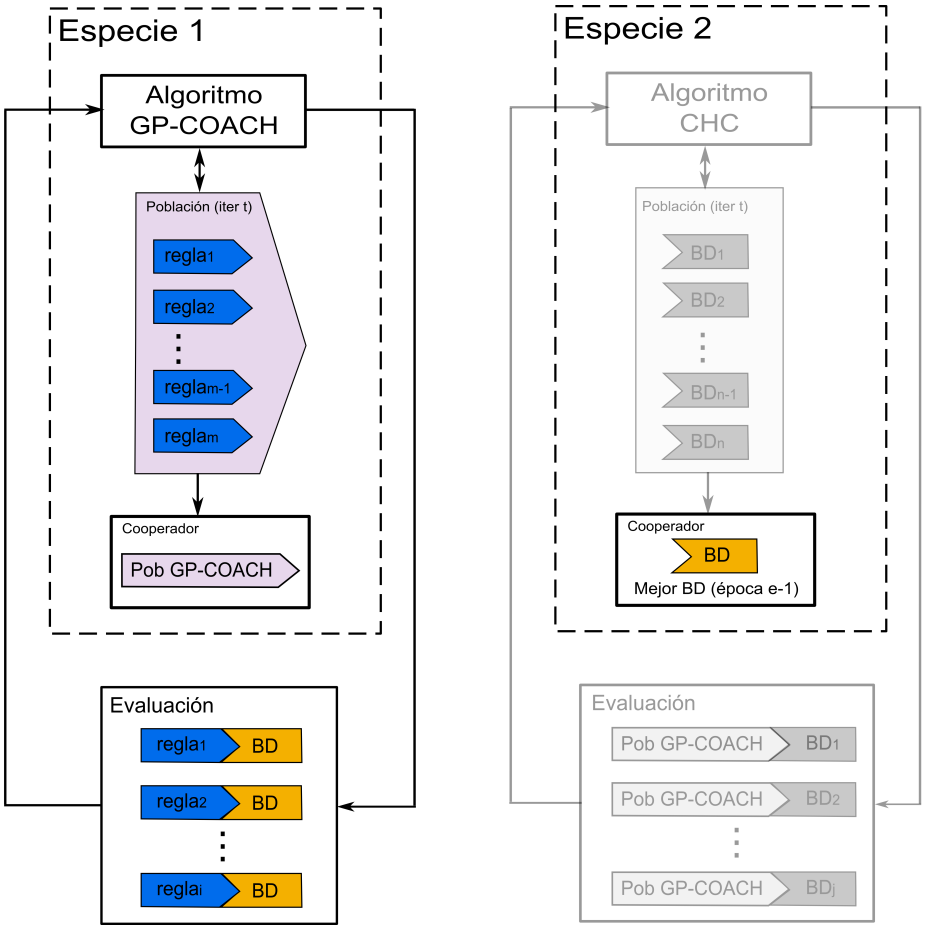


Figura 3.8: Evaluación de los individuos de la especie que aprende la BR en la propuesta de cooperación coevolutiva secuencial (GP-CO²ACH-S)

3. Se calcula el valor de fitness global de la BC formada por la BR actual y la

mejor definición actual para la BD, y si este valor mejora al de la mejor BC almacenada hasta el momento, entonces reemplazamos la mejor BC por la BC actual (y también se actualiza el valor de fitness global de la mejor BC).

4. Se congela la especie de la BR y la base de reglas formada por toda la población actual se establece como su cooperador.

5. La especie que aprende la BD, que se encontraba inactiva, se activa y se avanza una época en su proceso evolutivo. Cada individuo j de la BD se evalua usando el cooperador almacenado en la especie congelada de la BR. En la Figura 3.9, se muestra gráficamente como se evaluan los nuevos individuos de la especie que aprende la BD.

6. Se calcula el valor de fitness global de la BC formada por la BR actual y la mejor definición actual para la BD, y si este valor mejora al de la mejor BC almacenada hasta el momento, entonces reemplazamos la mejor BC por la BC actual (y también se actualiza el valor de fitness global de la mejor BC).

Cuando el proceso coevolutivo acaba, se devuelve como solución al problema la BC que estaba guardada como la mejor.

3.4. Estudio experimental

Para analizar la eficacia de nuestra propuesta coevolutiva, hemos realizado un estudio experimental utilizando los 24 conjuntos de datos de clasificación introducidos en el capítulo anterior (y cuyas características más importantes se muestran en la Tabla 2.4).

Los resultados de nuestra propuesta coevolutiva se han comparado con los obtenidos por el algoritmo GP-COACH (que sólo aprende BRs), y con los resultados obtenidos por un nuevo algoritmo evolutivo resultante de aplicar a GP-COACH un proceso de ajuste lateral a posterior usando la representación de 2-tuplas lingüísticas, algoritmo al que denominaremos GP-COACH + Ajuste Lateral. El esquema de dicho algoritmo es el siguiente:

1. Se aplica el algoritmo GP-COACH descrito en la sección 2.2 del capítulo anterior de esta memoria, obteniéndose una BR compacta y precisa a partir del uso de una BD que contiene unas particiones difusas prefijadas.

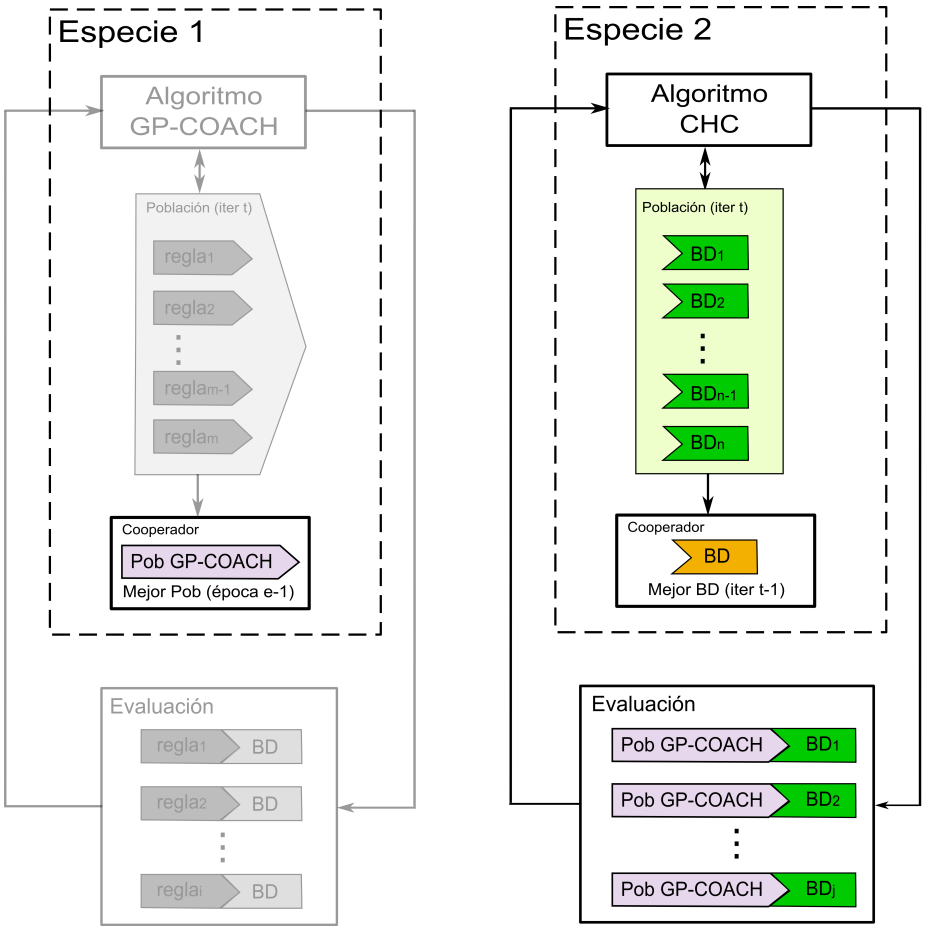


Figura 3.9: Evaluación de los individuos de la especie que aprende la BD en la propuesta de cooperación coevolutiva secuencial (GP-CO^{2}ACH-S)

2. Se aplica un proceso de ajuste a posteriori de las particiones difusas de la BD utilizada anteriormente por el algoritmo GP-COACH, con el objetivo de mejorar la precisión del SCBRD obtenido anteriormente. Este proceso de ajuste se basa en el uso del esquema de representación de las 2-tuplas lingüísticas y utiliza el algoritmo CHC descrito en la sección para obtener la mejor definición de la BD.

   Es importante señalar que dicho proceso de ajuste no altera la BR previamente obtenida por el algoritmo GP-COACH.

Los parámetros usados en cada uno de esos algoritmos se muestran en la Tabla 3.1.

Tabla 3.1: Parámetros usados en los algoritmos

Algoritmo	**BC**	**Parámetros**
GP-COACH	BR	$Eval = 20000, Pob = 200, \alpha = 0{,}7, P_c = 0{,}5, P_m = 0{,}2$ $P_{dp} = 0{,}15, P_i = 0{,}15, Torneo = 2, w_1 = 0{,}8, w_2 = 0{,}05$ $w_3 = 0{,}05\ y\ w_4 = 0{,}1$
	BD	*Prefijada e invariable*
GP-COACH + Ajuste Lateral	BR	$Eval = 20000, Pob = 200, \alpha = 0{,}7, P_c = 0{,}5, P_m = 0{,}2$ $P_{dp} = 0{,}15, P_i = 0{,}15, Torneo = 2, w_1 = 0{,}8, w_2 = 0{,}05$ $w_3 = 0{,}05\ y\ w_4 = 0{,}1$
	BD	$Eval = 50000, Pob = 50, BITSGENE = 30, \delta = 0{,}25$
GP-CO^{2}ACH	BR	$Eval = 20000, Pob = 200, \alpha = 0{,}7, P_c = 0{,}5, P_m = 0{,}2$ $P_{dp} = 0{,}15, P_i = 0{,}15, Torneo = 2, w_1 = 0{,}8, w_2 = 0{,}05$ $w_3 = 0{,}05\ y\ w_4 = 0{,}1$
	BD	$Eval = 50000, Pob = 50, BITSGENE = 30, \delta = 0{,}25$

Se ha usado la técnica de la validación cruzada con 10 particiones, y dado que todos los métodos utilizados en la comparativa son no deterministas, en nuestros experimentos hemos usado 3 semillas distintas para cada partición. Por lo tanto, para cada conjunto de datos se mostrarán los resultados medios obtenidos en 30 ejecuciones.

Finalmente, también hemos considerado el uso del test de ranking de signos de Wilcoxon para analizar estadísticamente de los resultados obtenidos por los distintos algoritmos en estudio.

3.5. Análisis comparativo de resultados

En esta sección analizamos los resultados obtenidos por los distintos métodos en el estudio experimental detallado anteriormente, tanto desde el punto de vista de la precisión como de la compacticidad de las BCs aprendidas.

Es importante señalar que en nuestros experimentos sólo hemos hecho uso del MRD de la suma normalizada, dado que en el capítulo anterior se detectó que existían diferencias significativas en el uso de dicho MRD para el algoritmo GP-COACH.

3.5.1. Análisis de la precisión

Los resultados de precisión obtenidos para cada uno de los conjuntos de datos y métodos considerados en nuestro estudio experimental se muestran en la Tabla 3.2, donde $\overline{\%Entr}$ y $\overline{\%Pru}$ representan el porcentaje de acierto en entrenamiento y prueba, respectivamente.

Tabla 3.2: Resultados de precisión de los algoritmos en estudio

	GP-COACH$_{Sum}$		**GP-COACH**$_{Sum}$ **+ Ajuste Lateral**		**GP-CO^{2}ACH-P**$_{Sum}$		**GP-CO^{2}ACH-S**$_{Sum}$	
Conj. Datos	$\overline{\%\textbf{Entr}}$	$\overline{\%\textbf{Pru}}$	$\overline{\%\textbf{Entr}}$	$\overline{\%\textbf{Pru}}$	$\overline{\%\textbf{Entr}}$	$\overline{\%\textbf{Pru}}$	$\overline{\%\textbf{Entr}}$	$\overline{\%\textbf{Pru}}$
Bupa	69,04	63,63	72,88	63,66	70,68	61,59	72,31	63,65
Cleveland	64,93	55,23	69,53	55,58	67,17	57,04	66,42	56,48
Ecoli	83,58	77,72	87,13	78,39	87,47	80,55	85,75	79,28
Flare	67,70	67,45	67,77	67,17	67,67	67,10	67,72	67,17
Glass	71,26	65,33	75,34	65,23	74,75	64,19	75,08	64,43
HillValley1	53,96	52,89	56,17	52,28	53,69	51,70	53,42	52,39
HillValley2	55,68	53,99	58,04	53,63	54,91	52,94	55,53	52,97
Iris	97,78	97,56	98,05	96,67	98,42	94,89	98,57	95,56
Libras Mov.	74,23	45,56	80,06	45,46	51,48	30,46	77,03	50,46
Magic	79,78	79,82	80,67	80,40	81,58	81,41	80,90	80,89
Page-blocks	91,30	91,23	92,78	92,37	94,70	94,46	94,80	94,56
Parkinsons	89,74	86,48	91,83	86,13	93,85	85,66	92,82	87,50
Pen-based	82,29	82,20	87,25	85,94	85,47	84,91	85,38	85,18
Pima	77,02	74,37	78,88	74,28	79,44	73,98	78,43	74,94
Quadruped	100,00	100,00	100,00	100,00	99,24	99,21	100,00	100,00
Ringnorm	91,24	91,13	95,92	94,67	90,58	89,88	92,82	92,59
Satimage	72,83	72,50	78,76	77,67	78,81	78,71	78,16	78,10
Segment	86,55	85,96	90,75	88,98	89,95	89,16	90,21	89,49
Sonar	80,25	67,46	86,88	71,01	89,87	73,94	86,00	74,56
Spambase	83,17	82,80	85,13	84,29	89,03	88,22	87,43	86,31
Twonorm	85,42	84,83	92,68	90,70	94,91	93,74	93,01	92,59
Wdbc	95,09	93,90	96,04	94,02	96,94	94,85	97,02	94,84
Wine	98,96	95,10	99,27	93,44	99,77	93,58	99,71	94,53
Yeast	49,74	48,56	55,75	52,43	59,40	56,03	58,65	56,36
MEDIA	79,23	75,65	82,40	76,85	81,24	76,59	81,97	77,70

En la Tabla 3.3 se muestra el análisis estadistico realizado mediante el test de Wilcoxon (considerando el porcentaje de acierto en prueba). El mejor método (resaltado en negrita) es aquel que presenta el mayor valor de ranking.

Tabla 3.3: Test de Wilcoxon para el análisis de la precisión, $p = 0{,}05$

GP-COACH$_{Sum}$ vs	**R$^+$**	**R$^-$**	**Valor crítico**	**Dif. sig.?**
GP-COACH$_{Sum}$ + Ajuste Lateral	81,5	**218,5**	81	No
GP-CO^{2}ACH-P$_{Sum}$	101,0	**199,0**	81	No
GP-CO^{2}ACH-S$_{Sum}$	45,5	**254,5**	81	Si
GP-COACH$_{Sum}$ + Ajuste Lateral vs	**R$^+$**	**R$^-$**	**Valor crítico**	**Dif. sig.?**
GP-CO^{2}ACH-P$_{Sum}$	133,0	**167,0**	81	No
GP-CO^{2}ACH-S$_{Sum}$	71,5	**228,5**	81	Si

Dicho análisis estadístico muestra que no existen diferencias significativas entre el algoritmo GP-COACH y el algoritmo GP-COACH + Ajuste Lateral, para un valor de $p = 0{,}05$. Tampoco existen estas diferencias con respecto a nuestra propuesta coevolutiva paralela (GP-CO^{2}ACH-P). Sin embargo, el estudio estadístico muestra que nuestra propuesta coevolutiva secuencial (GP-CO^{2}ACH-S) mejora estadísticamente tanto al algoritmo GP-COACH como al algoritmo GP-COACH + Ajuste Lateral, con un valor de $p = 0{,}05$.

3.5.2. Análisis de la compacticidad

Una vez que hemos analizado el rendimiento de los diferentes algoritmos en estudio en términos de precisión, en este segundo apartado nos disponemos a analizar su compacticidad e interpretabilidad.

Los resultados de compaticidad se muestran en la Tabla 3.4, donde $\overline{Reg}$ es el número medio de reglas, $\overline{Var}$ el número medio de variables por regla, $\overline{Cond}$ el número medio de condiciones (etiquetas) por regla, y IC es el índice de compacticidad (calculado según se indica en la ecuación 2.10). Como se puede ver, en dicha tabla no se han incluido los resultados de compacticidad del algoritmo GP-COACH + Ajuste Lateral, ya que estos son exactamente los mismos que los obtenidos por el algoritmo GP-COACH, dado que el proceso de ajuste lateral no altera la BR previamente aprendida.

Tabla 3.4: Resultados de compacticidad de los algoritmos en estudio

	GP-COACH$_{Sum}$				**GP-CO^{2}ACH-P**$_{Sum}$				**GP-CO^{2}ACH-S**$_{Sum}$			
Conj. Datos	$\overline{\textbf{Reg}}$	$\overline{\textbf{Var}}$	$\overline{\textbf{Cond}}$	**IC**	$\overline{\textbf{Reg}}$	$\overline{\textbf{Var}}$	$\overline{\textbf{Cond}}$	**IC**	$\overline{\textbf{Reg}}$	$\overline{\textbf{Var}}$	$\overline{\textbf{Cond}}$	**IC**
Bupa	10,07	1,38	2,49	0,16	5,17	1,18	3,22	0,15	4,67	1,14	2,98	0,12
Cleveland	23,83	3,05	7,44	0,35	20,17	2,50	5,25	0,23	24,37	4,24	6,17	0,37
Ecoli	25,57	2,88	6,21	0,57	25,10	2,13	5,07	0,40	19,67	2,07	4,87	0,36
Flare	8,13	1,80	4,11	0,19	4,27	1,00	1,82	0,03	4,43	1,02	1,75	0,03
Glass	17,43	2,64	5,56	0,39	13,37	1,87	4,18	0,24	12,97	1,81	4,13	0,23
HillValley1	7,27	3,18	7,65	0,04	3,63	1,90	4,95	0,02	3,40	1,25	3,94	0,01
HillValley2	6,90	3,20	8,16	0,04	4,53	1,65	4,35	0,02	4,27	1,40	4,32	0,01
Iris	3,23	1,22	1,75	0,13	3,83	1,03	1,39	0,04	3,97	1,02	1,48	0,04
Libras Mov.	113,93	53,36	76,03	1,07	157,77	59,41	78,10	1,31	113,37	50,98	76,27	1,05
Magic	9,33	1,71	4,33	0,16	21,57	1,88	5,95	0,20	10,80	1,51	3,66	0,12
Page-blocks	14,97	1,61	3,51	0,13	19,93	1,33	3,90	0,11	17,10	1,31	3,44	0,10
Parkinsons	6,40	1,67	3,77	0,09	6,10	1,69	4,27	0,09	4,87	1,38	3,45	0,06
Pen-based	89,70	4,27	9,35	0,35	222,80	6,23	12,40	0,52	76,40	4,45	8,34	0,34
Pima	17,23	2,46	5,15	0,36	12,80	1,32	3,61	0,13	8,03	1,25	2,98	0,10
Quadruped	4,57	1,24	1,55	0,01	20,77	32,03	33,23	0,04	4,00	1,00	1,00	0,01
Ringnorm	17,50	5,45	9,90	0,35	238,97	4,08	9,92	0,28	27,10	1,66	3,80	0,09
Satimage	27,53	5,82	13,29	0,22	68,10	13,30	20,61	0,44	28,07	16,01	17,94	0,47
Segment	23,30	3,28	6,85	0,21	15,93	1,81	4,39	0,09	14,07	1,67	3,74	0,08
Sonar	14,03	2,78	6,35	0,12	11,73	2,81	7,46	0,09	8,63	3,10	5,35	0,08
Spambase	10,27	3,77	7,48	0,08	27,47	1,67	5,71	0,04	9,77	3,06	5,40	0,04
Twonorm	51,67	4,11	9,15	0,27	515,43	6,49	14,38	0,52	102,00	4,55	12,33	0,34
Wdbc	4,90	1,17	3,03	0,03	4,77	1,08	2,99	0,02	4,53	1,10	2,91	0,02
Wine	7,57	1,90	4,65	0,18	5,50	1,44	3,45	0,10	5,27	1,41	3,60	0,10
Yeast	32,20	2,99	6,44	0,47	29,40	1,62	4,05	0,20	25,80	1,46	3,60	0,16

En la Tabla 3.5 se muestra el análisis estadistico realizado mediante el test de Wilcoxon (considerando el índice de compacticidad IC). El mejor método (resaltado en negrita) es aquel que presenta el mayor valor de ranking.

Tabla 3.5: Test de Wilcoxon para el análisis de la compacticidad, $p = 0{,}05$

GP-COACH$_{Sum}$ vs	**R$^+$**	**R$^-$**	**Valor crítico**	**Dif. sig.?**
GP-CO^{2}ACH-P$_{Sum}$	103,0	**197,0**	81	No
GP-CO^{2}ACH-S$_{Sum}$	39,0	**261,0**	81	Si

Dicho análisis estadístico muestra que no existen diferencias significativas en compacticidad entre el algoritmo GP-COACH y nuestra propuesta coevolutiva paralela (GP-CO^{2}ACH-P). Sin embargo, el estudio estadístico muestra que nuestra propuesta coevolutiva secuencial (GP-CO^{2}ACH-S) mejora estadísticamente al algoritmo GP-COACH (y por tanto también al algoritmo GP-COACH + Ajuste Lateral) en cuanto a la compacticidad de las BCs obtenidas, con un valor de $p = 0{,}05$.

3.6. Conclusiones

En este capítulo hemos presentado una propuesta coevolutiva para el aprendizaje de BCs compactas y precisas para problemas con alta dimensionalidad: el algoritmo GP-CO^{2}ACH. Se trata de un algoritmo en el que dos especies distintas cooperan entre sí y evolucionan simultaneamente, en parelelo (GP-CO^{2}ACH-P) y secuencialmente (GP-CO^{2}ACH-S), para aprender toda la BC:

- *Especie 1:* Contiene una población de reglas difusas DNF que formarán la BR del problema a resolver. Esta especie hace uso del algoritmo GP-COACH (descrito en el capítulo anterior) para derivar una BR compacta y precisa.

- *Especie 2:* Una población de cromosomas que hace uso del algoritmo CHC, para obtener la mejor definición para la BD mediante el esquema de representación de 2-tuplas lingüísticas, el cual permite mantener la forma original de las funciones de pertenencia (simétricas), modificando lateralmente la localización del soporte, y por lo tanto manteniendo la interpretabilidad de la BC aprendida.

Para analizar nuestra propuesta, se ha considerando el uso de varios conjuntos de datos de alta dimensionalidad en un estudio experimental en el que los resultados de nuestra propuesta coevolutiva (en sus dos versiones) se han comparado con los obtenidos por GP-COACH (que sólo aprende BRs) y los resultados obtenidos por un nuevo algoritmo, denominado GP-COACH + Ajuste Lateral, resultante de aplicar un proceso a posteriori de ajuste lateral (usando la representación de 2-tuplas lingüísticas) de la BD inicialmente definida, una vez que ha terminado la ejecución del algoritmo GP-COACH.

Por otra parte, también se han utilizado tests estadísticos no paramétricos, concretamente el test de Wilcoxon, para comparar y analizar la compacticidad y precisión de las BCs obtenidas. En este sentido, se han derivado las siguientes conclusiones:

1. Nuestra propuesta coevolutiva secuencial (GP-CO^{2}ACH-S) es capaz de mejorar los resultados en precisión obtenidos por los algoritmos GP-COACH y GP-COACH + Ajuste Lateral. Por lo tanto se demuestra estadísticamente que GP-CO^{2}ACH-S es capaz de aprender BCs que presentan una alta capacidad de generalización para problemas con alta dimensionalidad.

2. Dicha capacidad de mejora en la precisión además se consigue junto a una mejora en la interpretabilidad y compacticidad de las BCs aprendidas, ya que se ha demostrado estadísticamente que GP-CO^{2}ACH-S mejora la compacticidad de los SCBRDs obtenidos por los algoritmos GP-COACH y GP-COACH + Ajuste Lateral.

Capítulo 4

Conclusiones

En este capítulo, se resumen brevemente los resultados obtenidos y se destacan las conclusiones principales obtenidas en esta memoria. Se presentan también las publicaciones asociadas a esta memoria y se comentan algunos aspectos relacionados con los trabajos futuros que siguen la línea aquí desarrollada y sobre otras líneas de investigación que se pueden derivar.

4.1. Resultados Obtenidos

Hemos estudiado el problema del aprendizaje de SCBRDs que muestren un buen equilibrio entre interpretabilidad y precisión en problemas que presentan una alta dimensionalidad. Una vez analizado el problema y determinados los principales aspectos a resolver en el mismo, hemos estudiado la aplicación de los AEs, más concretamente de la PG y de los algoritmos coevolutivos, y la lógica difusa al mismo. Como consecuencia de esto, en esta memoria hemos presentado distintos algoritmos para el aprendizaje de SCBRDs compactos y precisos en problemas con una alta dimensionalidad. Los siguientes apartados resumen brevemente los resultados obtenidos, presentando las principales conclusiones sobre los mismos.

4.1.1. Un modelo basado en PG para el aprendizaje de SCBRDs compactos y precisos para problemas con alta dimensionalidad

Hemos propuesto un modelo evolutivo basado en PG para el aprendizaje de SCBRDs que presenten un buen nivel de equilibrio entre interpretabilidad y precisión para problemas que presenten una alta dimensionalidad (con un elevado número de variables de entrada), GP-COACH, que se refleja en el Capítulo 2 de esta memoria.

Este modelo representa el conocimiento extraido mediante reglas difusas en notación DNF. Para aprender dichas reglas hace uso de una gramática libre de contexto que también permite la ausencia de alguna de las variables de entrada en las reglas, lo que nos lleva a obtener reglas que tienen pocas variables y etiquetas lingüísticas en sus antecedentes. Este modelo sigue el enfoque *GCCL* de codificación de reglas, el cual codifica una única regla por individuo, estando la BR formada por todos los individuos (reglas) de la población. Esto hace necesario el incluir un un mecanismo para incrementar la diversidad de la población, la *Competición de Tokens*, el cual hace que las reglas compitan entre si durante el proceso evolutivo. Esto permite obtener conjuntos de reglas compactos que también presentan una alta capacidad de generalización. Finalmente, este modelo utiliza un proceso de inferencia jerárquico con dos niveles, el cual nos permite mejorar la precisión de las BR obtenidas, al evitar errores de clasificación generados por el uso de reglas muy específicas (*reglas secundarias*) que cubren pocos ejemplos, cuando otras reglas más generales (*reglas primarias*) se pueden usar.

Para analizar este modelo se ha considerando el uso de distintos conjuntos de datos de alta dimensionalidad en un estudio experimental dividido en dos partes distintas:

1. El análisis de la adecuación de varios componentes de nuestro método, como el uso de la competición de tokens, el uso de los operadores genéticos específicos o el uso de un algoritmo basado en PG en vez de en un AG tradicional.

   Para mostrar la validez de los resultados obtenidos se ha considerado el uso de tests estadísticos no paramétricos, los cuales han mostrado la robustez de GP-COACH.

2. Un estudio comparativo de los resultados obtenidos por GP-COACH y los

obtenidos por otros cinco métodos de aprendizaje de SCBRDs bien conocidos en la literatura especializada: PCA-Ravi, 2SLAVE, PG-PITT-Tsakonas, GCCL-Ishibuchi y FRBCS_GP.

Al igual que en el apartado anterior, también hemos utilizado tests estadísticos no paramétricos para comparar y analizar la compacticidad y precisión de los SCBRDs obtenidos, derivando las siguientes conclusiones:

a) Nuestro modelo mejora en precisión al resto de los algoritmos de aprendizaje de SCBRDs estudiados. Se ha demostrado estadísticamente que GP-COACH es capaz de aprender SCBRDs que tienen una alta capacidad de generación para problemas que presentan una alta dimensionalidad.

b) GP-COACH es capaz de aprender SCBRDs compactos e interpretables para problemas de alta dimensionalidad, mejorando a los resultados obtenidos por otros métodos que también aprenden reglas difusas en notación DNF: 2SLAVE y FRBCS_GP.

4.1.2. Un modelo coevolutivo para el aprendizaje de SCBRDs compactos y precisos para problemas con alta dimensionalidad

En el Capítulo 3 de esta memoria se recoge la propuesta desarrollada de algoritmo coevolutivo para el aprendizaje de SCBRDs que presenten un buen nivel de equilibrio entre interpretabilidad y precisión para problemas que presenten una alta dimensionalidad, GP-CO^{2}ACH.

Se trata de un algoritmo en el que dos especies distintas cooperan entre sí y evolucionan simultaneamente, en parelelo (GP-CO^{2}ACH-P) y secuencialmente (GP-CO^{2}ACH-S), para aprender toda la BC. La primera de las especies hace uso del algoritmo GP-COACH para derivar una BR compacta y precisa. Mientras, la segunda especie hace uso del algoritmo CHC para evolucionar una población de cromosomas, cada uno de ellos representando una definición de la BD mediante el esquema de representación de 2-tuplas lingüísticas, el cual mantiene la forma original de las funciones de pertenencia al modificar lateralmente sólo la localización del soporte, lo que mantiene la interpretabilidad de la BC.

Para analizar esta propuesta coevolutiva (en sus dos versiones), se ha llevado a cabo un estudio experimental considerando varios problemas con alta dimen-

sionalidad. Los resultados obtenidos por GP-CO^{2}ACH se han comparado con los obtenidos por GP-COACH (que sólo aprende BRs) y con los resultados obtenidos por un nuevo algoritmo, denominado GP-COACH + Ajuste Lateral, resultante de aplicar un proceso a posteriori de ajuste lateral (usando la representación de 2-tuplas lingüísticas) de la BD inicialmente definida, una vez que ha terminado la ejecución del algoritmo GP-COACH.

Hemos considerado el uso de tests estadísticos no paramétricos para comparar y analizar la compacticidad y precisión de los SCBRDs obtenidos, derivándose las siguientes conclusiones:

1. Nuestra propuesta coevolutiva secuencial (GP-CO^{2}ACH-S), es capaz de mejorar en precisión a los resultados obtenidos por los algoritmos GP-COACH y GP-COACH + Ajuste Lateral, demostrándose estadísticamente que GP-CO^{2}ACH-S es capaz de aprender SCBRDs que presentan una alta capacidad de generalización para problemas con alta dimensionalidad.

2. Dicha mejora en la precisión además se consigue junto a una mejora en la interpretabilidad y compacticidad de los SCBRDs aprendidos, pues también se demuestra estadísticamente que GP-CO^{2}ACH-S es capaz de mejorar la compacticidad de los SCBRDs obtenidos por los algoritmos GP-COACH y GP-COACH + Ajuste Lateral.

4.2. Publicaciones Asociadas a la Tesis

A continuación se presenta un listado de las publicaciones asociadas a la tesis.

- Publicaciones en revistas internacionales:

  1. Berlanga F.J., Rivera A.J., del Jesus M.J. y Herrera F. (2010) GP-COACH: Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems. Information Sciences 180(8): 1183-1200.

- Publicaciones en congresos internacionales:

  1. Berlanga F.J., del Jesus M.J. y Herrera F. (2005) Learning fuzzy rules using genetic programming: Context-free grammar definition for high-

dimensionality problems. *I* International Workshop on Genetic Fuzzy Systems (GFS05), páginas 136-141. Granada, Spain.

2. Berlanga F.J., del Jesus M.J. y Herrera F. (2005) Learning compact fuzzy rule-based classification systems with genetic programming. Proc. 4*th* Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT05), páginas 1027-1032. Barcelona, Spain.
3. Berlanga F.J., del Jesus M.J., Gacto M.J. y Herrera F. (2006) A Genetic-Programming-Based Approach for the Learning of Compact Fuzzy Rule-Based Classification Systems. Proc. 8*th* International Conference on Artificial Intelligence and Soft Computing (ICAISC06), number 4029 in Lecture Notes in Computer Science, páginas 182-191. Springer-Verlag, Berlin.
4. Berlanga F.J., del Jesus M.J. y Herrera F. (2008) A Novel Genetic Cooperative-Competitive Fuzzy Rule Based Learning Method using Genetic Programming for High Dimensional Problems. 3*rd* International Workshop on Genetic and Evolving Fuzzy Systems (GEFS08), páginas 101-106. WittenBommerholz, Germany.

- Publicaciones en congresos nacionales:

1. Berlanga F.J., del Jesus M.J. y Herrera F. (2005) Aprendizaje de reglas difusas mediante programación genética en problemas con alta dimensionalidad. *I* Simposio sobre Lógica Fuzzy y Soft Computing (LFSC05), páginas 93-100. Granada, España.

4.3. Líneas de Investigación Futuras

A continuación presentamos algunas líneas de trabajo que quedan abiertas relacionadas con los temas tratados en la memoria, además de las extensiones sobre las propuestas presentadas que serán objeto de estudios posteriores.

- En esta memoria, se han presentado modelos evolutivos para abordar el aprendizaje de SCBRDs compactos y precisos en problemas que presentan una alta dimensionalidad con respecto al número de variables de entrada. Otra posibilidad por la que un problema puede presentar una alta dimensionalidad es cuando este contiene un número elevado de instancias o ejemplos

de entrenamiento. En este tipo de problemas el modelo evolutivo presenta algunas dificultades al tener que procesar tal cantidad de información, lo que puede afectar al equilibrio entre la interpretabilidad y la precisión del conocimiento extraido. Para solucionar este problema, en la literatura especializada se ha propuesto el uso de técnicas de preprocesamiento que seleccionan un grupo reducido y significativo de instancias o ejemplos de entrenamiento, manteniendo así intactas las propiedades del problema a tratar pero facilitando la tarea del modelo evolutivo. Por este motivo sería interesante realizar un estudio sobre el funcionamiento de los modelos evolutivos propuestos en esta memoria junto con el uso de diversas técnicas de preprocesamiento para selección de instancias en problemas que presentan este otro tipo de alta dimensionalidad.

- En el capítulo 3 de esta memoria se ha propuesto un modelo coevolutivo para el aprendizaje simultaneo de un conjunto de reglas difusas compactas y precisas, y la mejor definición para los conjuntos difusos asociados a esas reglas, con el objetivo de obtener SCBRDs aun más compactos y precisos que los obtenidos por nuestro modelo evolutivo GP-COACH. Otra posible alternativa que también puede permitirnos obtener sistemas aun más compactos y precisos consiste en el uso de particiones difusas jerarquicas. En este tipo de aproximación extiende la extructura del SBCRD permitiendo que existan reglas difusas que están definidas sobre particiones difusas con distinta granularidad, lo cual puede hacer que el algoritmo mejore su funcionamiento en aquellos problemas en los que nuestros modelos tradicionales no obtenían un rendimiento. El uso de estas particiones jerarquicas se puede utilizar conjuntamente con la coevolución, complementando así sus buenos resultados.

- En el campo de la clasificación, es posible encontrar problemas en los que las distintas clases presentes en el problema se encuentran definidas por porcentajes de patrones o ejemplos de entramiento muy diferentes entre sí. Este tipo de problemas reciben el nombre de problemas de clasificación no balanceados. Sería de interés aplicar los algoritmos propuestos en esta memoria a la tarea del aprendizaje de SCBRDs compactos y precisos en problemas de clasificación no balanceados, estudiando así su viabilidad en este tipo de problemas y analizando las posibles mejoras a introducir en los mismos con el objetivo de ser aplicables a este tipo de problemas.

Apéndice A

Tests no paramétricos para el análisis estadístico de los resultados

En este apéndice se explican con mayor detalle los distintos tests estadísticos no parametricos que se han usado en esta memoria para el análisis de los resultados obtenidos por los distintos métodos en estudio.

A.1. Test de Friedman

El test de Friedman [She03] es un test no paramétrico equivalente al test ANOVA de medidas repetidas. Bajo la hipótesis nula, este test indica que todos los algoritmos son equivalentes. Por lo tanto, el rechazo de dicha hipótesis implica la existencia de diferencias en el rendimiento de todos los algoritmos estudiados.

El test de Friedman trabaja de la siguiente forma: Hace un ranking de cada algoritmo sobre cada uno de los conjuntos de datos, de forma que el mejor de los algoritmos obtiene un valor de ranking de 1, el segundo mejor un valor de 2, y así sucesivamente. Si se produce un empate entre dos algoritmos, el test asigna el ranking medio a ambos.

Supongamos que r_i^j es el ranking del j-ésimo algoritmo (siendo k el número to-

tal de algoritmos) sobre el i-ésimo conjunto de datos (con un total de N conjuntos de datos). El test de Friedman lleva a cabo una comparación del ranking medio de los algoritmos, $R_j = \frac{1}{N}\sum_i r_i^j$. Bajo la hipótesis nula, que como hemos dicho anteriormente afirma que todos los algoritmos son equivalentes y por lo tanto sus rankings R_j también deberían ser igual, el estadístico de Friedman

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{A.1}$$

se distribuye según una distribución χ_F^2 con $k-1$ grados de libertad, siempre que N y k sean lo suficientemente grandes (como regla general, $N_{ds} > 10$ y $k > 5$).

A.2. Test de Iman-Davenport

El test de Iman y Davenport [ID80] es un test no paramétrico derivado del test de Friedman, que presenta un comportamiento menos conservativo que su predecesor:

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

el cual se distribuye de acuerdo a una distribución F con $k-1$ y $(k-1)(N-1)$ grados de libertad. En [She03], se pueden encontrar las tablas estadísticas para los valores críticos.

A.3. Test de Holm

El método de Holm [Hol79], es un test a posteriori para el estadístico de Friedman. Este método lleva a cabo un proceso de comparación multiple que trabaja con un algoritmo de control (normalmente, se elige el mejor de los algoritmos), el cual es comparado con el resto. El test estadístico utilizado para comparar el i-ésimo y j-ésimo algoritmos según este método es el siguiente:

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6N}} \tag{A.3}$$

El valor z se utiliza para encontrar la probabilidad correspondiente dentro de la tabla de la distribución normal, el cual se compara entonces con un valor apropiado de α. En la comparativa de Bonferroni-Dunn, este valor α es siempre igual a $\alpha/(k-1)$. Sin embargo, el test de Holm ajusta el valor de α con el objetivo de compensar el error producido por las multiples comparaciones, controlando así la tasa de error *family-wise*.

El test de Holm es un procedimiento incremental, el cual testea secuencialmente las hipótesis ordenadas por su valor de significancia. Los p-values $(p_1, p_2, \ldots, p_{k-1})$ se muestran ordenados, de manera que $p_1 \leq p_2 \leq \ldots \leq p_{k-1}$. El test de Holm compara cada p_i con $\alpha/(k-i)$, empezando por el p-value más significativo. Si p_1 está por debajo de $\alpha/(k-1)$, entonces la correspondiente hipótesis se rechaza y pasamos a comparar p_2 con $\alpha/(k-2)$. Si la segunda hipótesis también se rechaza, el test pasa a comprobar la tercera, y así consecutivamente. En el momento, en que una cierta hipótesis nula no puede ser rechazada, el test acaba las comprobaciones aceptando la hipótesis nula para el resto de hipótesis que aun quedaban por comprobar.

A.4. Test de Wilcoxon

El test de ranking de signos de Wilcoxon [She03] es un test no paramétrico análogo al t-test. Por lo tanto, se trata de un test por parejas cuyo objetivo es detectar difirencias significativas en el comportamiento de dos algoritmos.

Supongamos que d_i es la diferencia entre los rendimientos de los algoritmos sobre el i-ésimo conjunto de datos (de un total de N). Estas diferencias son clasificadas según sus valores absolutos (en caso de empates se asignan los rankings medios). Supongamos que R^+ es la suma de los rankings para los conjuntos de datos en los que el segundo de los algoritmos mejora al primero, y que R^- es la suma de los rankings en el caso contrario. Los rankings en los que $d_i = 0$ se distribuyen de forma equitativa en las dos sumas (si hay un número impar de rakings con $d_i = 0$, entonces, se ignora uno de ellos):

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{A.4}$$

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{A.5}$$

Supongamos que T es el valor más pequeño de ambas sumas, $T = \min(R^+, R^-)$. Si T es menor o igual que el valor de la distribución de Wilcoxon con N grados de libertad (tabla B.12 en [Zar99]), entonces se rechaza la hipótesis de igualdad de medias.

Bibliografía

[AAFHO07] Alcalá R., Alcalá-Fdez J., Herrera F. y Otero J. (2007) Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. *International Journal of Approximate Reasoning* 44(1): 45–64.

[AC98] Au W. H. y Chan K. C. C. (1998) An effective algorithm for discovering fuzzy rules in relational databases. En *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'98)*, páginas 1314–1319. Anchorage, USA.

[AC99] Au W. H. y Chan K. C. C. (1999) FARM: a data mining system for discovering fuzzy association rules. En *Proceedings of the 8th IEEE International Conference on Fuzzy Systems*, páginas 1217–1222. Seoul, South Korea.

[ACT99] Alba E., Cotta C. y Troya J. M. (1999) Evolutionary design of fuzzy logic controllers using strongly-typed GP. *Mathware & Soft Computing* 6: 109–124.

[AIS93] Agrawal R., Imielinski T. y Swami A.Ñ. (1993) Mining association rules between sets of items in large databases. En *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, páginas 207–216. ACM Press, Washington D.C., USA.

[AKTJ00] Akbarzadeh-T M. R., Kumbla K., Tunstel E. y Jamshidi M. (2000) Soft computing for autonomous robotic systems. *Computers and Electrical Engineering* 26(1): 5–32.

[AL95] Abe S. y Lan M.-S. (1995) A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. *IEEE Transactions on Fuzzy Systems* 3(1): 18–28.

[AMS+96] Agrawal R., Mannila H., Srikant R., Toivonen H. y Verkamo I. (1996) Fast discovery of association rules. En *Advances in Knowledge Discovery and Data Mining*, páginas 307–328. AAAI Press, Menlo Park, USA.

[AN07] Asuncion A. y Newman D. J. (2007) UCI machine learning repository, *http://www.ics.uci.edu/∼mlearn/MLRepository.html*, University of California, Department of Information and Computer Science, Irvine, CA.

[ARS03] Abonyi J., Roubos J. A. y Szeifert F. (2003) Data-driven generation of compact, accurate, and linguistically sound fuzzy classifiers based on a decision-tree initialization. *International Journal of Approximate Reasoning* 32(1): 1–21.

[AS95] Agrawal R. y Srikant R. (1995) Mining sequential patterns. En *Proceedings of the 11th International Conference on Data Engineering*, páginas 3–14. Taipei, Taiwan.

[ASD08] Akbarzadeh V., Sadeghian A. y Dos Santos M. V. (2008) Derivation of relational fuzzy classification rules using evolutionary computation. En *Proceedings of the 2008 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'08)*, páginas 1689–1693. Hong Kong, China.

[AT97] Abe S. y Thawonmas R. (1997) A fuzzy classifier with ellipsoidal regions. *IEEE Transactions on fuzzy systems* 5(3): 358–368.

[Bäc96] Bäck T. (1996) *Evolutionary algorithms in theory and practice*. Oxford University Press.

[Bak87] Baker J. E. (1987) Reducing bias and inefficiency in the selection algorithm. En Grefenstette J. J. (Ed.) *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA'87)*, páginas 14–21. Lawrence Erlbaum Associates, Hillsdale, NJ, EE.UU.

[Bal96] Baldwin J. F. (1996) Knowledge from data using fuzzy methods. *Pattern Recognition Letters* 17(6): 593–600.

[Bas00] Bastian A. (2000) Identifying fuzzy models utilizing genetic programming. *Fuzzy Sets and Systems* 113(3): 333–350.

[BC96] Berndt D. y Clifford J. (1996) Finding patterns in time series: a dynamic programming approach. En *Advances in Knowledge Discovery and Data Mining*, páginas 229–248. AAAI Press, Menlo Park, USA.

[BD96] Brockwell P. J. y Davis R. A. (1996) *Introduction to time series and forecasting*. Springer-Verlag, New York, USA.

[BF99] Buckley J. J. y Feuring T. (Eds.) (1999) *Fuzzy and Neural: Interactions and Applications*. Studies in Fuzziness and Soft Computing. Heidelberg: Physica-Verlag.

[BFM97] Bäck T., Fogel D. y Michalewicz Z. (1997) *Handbook of evolutionary computation*. Oxford University Press.

[BFOS84] Breiman L., Friedman J. H., Olshen R. A. y Stone C. J. (1984) *Classification and Regression Trees*. Wadsworth, Belmont, CA.

[BG05] Bhatt R. B. y Gopal M. (2005) On fuzzy-rough sets approach to feature selection. *Pattern Recognition Letters* 26(7): 965–975.

[BLF00] Bojarczuk C. C., Lopes H. S. y Freitas A. A. (2000) Genetic programming for knowledge discovery in chest-pain diagnosis: Exploring a promising data mining approach. *IEEE Engineering in Medicine and Biology Magazine* 19(4): 38–44.

[BM07] Bouchachia A. y Mittermeir R. (2007) Towards incremental fuzzy classifiers. *Soft Computing* 11(2): 193–207.

[BN93] Basseville M. y Nikiforov I. V. (1993) *Detection of abrupt changes: theory and application*. Prentice Hall.

[Bon97] Bonissone P. (1997) Soft computing: the convergence of emerging reasoning technologies. *Soft Computing* 1(1): 6–18.

[BPU99] Bosc P., Pivert O. y Ughetto L. (1999) Database mining for the discovery of extended functional dependencies. En *Proc. of the International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99)*, páginas 580–584. New York, USA.

[BS91] Bäck T. y Schwefel H. P. (1991) Extended selection mechanisms in genetic algorithms. En *Proc. Fourth International Conference on Genetic Algorithms (ICGA'91)*, páginas 2–9. San Diego, EE.UU.

[CA98] Combs W. E. y Andrews J. E. (1998) Combinatorial rule explosion eliminated by a fuzzy rule configuration. *IEEE Transactions on Fuzzy Systems* 6(1): 1–11.

[CAAFR07] Cordón O., Alcalá R., Alcalá-Fdez J. y Rojas I. (2007) Genetic fuzzy systems: What's next? An introduction to the special section. *IEEE Transactions on Fuzzy Systems* 15(4): 533–535.

[CC04] Castro P. A. D. y Camargo H. A. (2004) Learning and optimization of fuzzy rule base by means of self-adaptive genetic algorithm. En *Proc. 2004 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004)*, páginas 1037– 1042. Budapest, Hungary.

[CC05] Castro P. A. D. y Camargo H. A. (2005) Learning and optimization of fuzzy rule base by means of self-adaptive genetic algorithm. En *Proc. 2005 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2005)*, páginas 696–701. Reno, USA.

[CC09] Casillas J. y Carse B. (2009) Special issue on genetic fuzzy systems: Recent developments and future directions. *Soft Computing* 13(5): 417–418.

[CCB07] Casillas J., Carse B. y Bull L. (2007) Fuzzy-XCS: a michigan genetic fuzzy system. *IEEE Transactions on Fuzzy Systems* 15(4): 536–550.

[CCdJH01] Casillas J., Cordón O., del Jesus M. J. y Herrera F. (2001) Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. *Information Sciences* 136(1-4): 135–157.

[CCdJH05] Casillas J., Cordón O., del Jesus M. J. y Herrera F. (2005) Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems* 13(1): 13–29.

[CCFM05] Castellano G., Castiello C., Fanelli A. M. y Mencar C. (2005) Knowledge discovery by a neuro-fuzzy modeling framework. *Fuzzy Sets and Systems* 149(1): 187–207.

[CCHM03a] Casillas J., Cordón O., Herrera F. y Magdalena L. (Eds.) (2003) *Accuracy Improvements in Linguistic Fuzzy Modeling*, volumen 129 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag.

[CCHM03b] Casillas J., Cordón O., Herrera F. y Magdalena L. (Eds.) (2003) *Interpretability Issues in Fuzzy Modeling*, volumen 128 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag.

[CCW00] Chiang D. A., Chow L. R. y Wang Y. F. (2000) Mining time series data by a fuzzy linguistic summary system. *Fuzzy Sets and Systems* 112: 419–432.

[CdJH98] Cordón O., del Jesus M. J. y Herrera F. (1998) Genetic learning of fuzzy rule-based classification systems co-operating with fuzzy reasoning methods. *International Journal of Intelligent Systems* 13(10-11): 1025–1053.

[CdJH99] Cordón O., del Jesus M. J. y Herrera F. (1999) A proposal on reasoning methods in fuzzy rule-based classification systems. *International Journal of Approximate Reasoning* 20: 21–45.

[CdJHL99] Cordón O., del Jesus M. J., Herrera F. y Lozano M. (1999) MOGUL: A Methodology to Obtain Genetic fuzzy rule-based systems Under the iterative rule Learning approach. *International Journal of Intelligent Systems* 14(11): 1123–1153.

[CF00] Castellano G. y Fanelli A. M. (2000) A staged approach for generation and compression of fuzzy classification rules. En *Proc. 2000 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2000)*, páginas 42–47. San Antonio, USA.

[CFM02] Castellano G., Fanelli A. M. y Mencar C. (2002) A neuro-fuzzy network to generate human-understandable knowledge from data. *Cognitive Systems Research* 3(2): 125–144.

[CGH+04] Cordón O., Gomide F., Herrera F., Hoffmann F. y Magdalena L. (2004) Ten years of genetic systems: current framework and new trends. *Fuzzy Sets and Systems* 141(1): 5–31.

[CHHM01] Cordón O., Herrera F., Hoffmann F. y Magdalena L. (2001) *Genetic Fuzzy Systems. Evolutionary tuning and learning of fuzzy knowledge bases*, volumen 19 of *Advances in Fuzzy Systems - Applications and Theory*. World Scientific, Singapur.

[CHP+07] Casillas J., Herrera F., Pérez R., del Jesus M. y Villar P. (2007) Special issue on genetic fuzzy systems and the interpretability-accuracy trade-off. *International Journal of Approximate Reasoning* 44(1): 1–3.

[CL04] Chen M.-Y. y Linkens D. A. (2004) Rule-base self-generation and simplification for data-driven fuzzy models. *Fuzzy Sets and Systems* 142(2): 243–265.

[CLH02] Chien B.-C., Lin J. Y. y Hong T.-P. (2002) Learning discriminant functions with fuzzy attributes for classification using genetic programming. *Expert Systems with Applications* 23(1): 31–37.

[CM07] Cherkassky V. y Mulier F. M. (2007) *Learning from data: Concepts, theory and methods*. Wiley-IEEE Press.

[CN89] Clark P. y Niblett T. (1989) The CN2 induction algorithm. *Machine Learning* 3(4): 261–283.

[Coh95] Cohen W. W. (1995) Fast effective rule induction. En *Proc. of the Twelfth International Conference on Machine Learning*, páginas 115–123. Morgan Kaufmann.

[CP04] Chakraborty D. y Pal N. R. (2004) A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification. *IEEE Transactions on Neural Networks* 15(1): 110–123.

[CP07] Carse B. y Pipe A. (2007) Introduction: Genetic fuzzy systems. *International Journal of Intelligent Systems* 22(9): 905–907.

[Cra85] Cramer N. L. (1985) A representation for the adaptive generation of simple sequential programs. En *Proceedings of an International Conference on Genetic Algorithms and the Applications, (ICGA'85)*, páginas 183–187. Pittsburgh, USA.

[CW02] Chen G. y Wei Q. (2002) Fuzzy association rules and the extended mining algorithms. *Information Sciences* 147(1-4): 201–228.

[CY96] Chi Z. y Yan H. (1996) ID3-Derived fuzzy rules and optimized defuzzification for handwritten numeral recognition. *IEEE Transactions on Fuzzy Systems* 4(1): 24–31.

[CYP96] Chi Z., Yan H. y Pham T. (1996) *Fuzzy algorithms with applications to image processing and pattern recognition.* World Scientific.

[Dem06] Demšar J. (2006) Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research* 7: 1–30.

[dJHNS04] del Jesus M. J., Hoffmann F., Navascués L. J. y Sánchez L. (2004) Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE Transactions on Fuzzy Systems* 12(3): 296–308.

[ES93] Eshelman L. J. y Schaffer J. D. (1993) Real-coded genetic algorithms and interval-schemata. En Whitley L. (Ed.) *Foundations of genetic algorithms*, volumen 2, páginas 187–202. Morgan Kaufman.

[ES03] Eiben A. E. y Smith J. E. (2003) *Introduction to evolutionary computation.* Springer Verlag, Berlin.

[Esh91] Eshelman L. J. (1991) The CHC adaptive search algorithm: How to have safe serach when engaging in nontraditional genetic recombination. En Rawlin G. (Ed.) *Foundations of genetic Algorithms*, volumen 1, páginas 265–283. Morgan Kaufman.

[FH51] Fix E. y Hodges Jr. J. (1951) Discriminatory analysis: Nonparametric discrimination: Consistency properties. Technical Report Report Number 4, USAF School of Aviation Medicine, Randolf Field, Texas.

[Fog88] Fogel D. B. (1988) An evolutionary approach to the travelling salesman problem. *Biological Cybernetics* 60(2): 139–144.

[FPS96a] Fayyad U., Piatetsky-Shapiro G. y Smyth P. (1996) From data mining to knowledge discovery: An overview. En *Advances in Knowledge Discovery and Data Mining*, páginas 1–34. AAAI Press, Menlo Park, USA.

[FPS96b] Fayyad U., Piatetsky-Shapiro G. y Smyth P. (1996) The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM* 39: 27–34.

[Fre99] Freitas A. A. (1999) On rule interestingness measures. *Knowledge-Based Systems* 12(5-6): 309–315.

[Fre02] Freitas A. A. (2002) *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer.

[FWS+98] Fu A. W.-C., Wong M. H., Sze S. C., Wong W. C., Wong W. L. y Yu W. K. (1998) Finding fuzzy sets for the mining of fuzzy association rules for numerical attributes. En *Proceedings of the First International Symposium on Intelligent Data Engineering and Learning, (IDEAL'98)*, páginas 263–268.

[GE08] Gunal S. y Edizkan R. (2008) Subspace based feature selection for pattern recognition. *Information Sciences* 178(19): 3716–3726.

[GGS99] García S., González F. y Sánchez L. (1999) Evolving fuzzy based classifiers with GA-P: A grammatical approach. En *2nd European Workshop on Genetic Programming (EuroGP'99)*, LNCS vol. 1598, páginas 203–210. Springer-Verlag.

[GMV96] Guyon O., Matic N. y Vapnik N. (1996) Discovering informative patterns and data cleaning. En *Advances in Knowledge Discovery and Data Mining*, páginas 181–204. AAAI Press, Menlo Park, USA.

[Gol89] Goldberg D. E. (1989) *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, MA.

[Gol96] Golden R. M. (1996) *Mathematical Methods for Neural Network Analysis and Design*. The MIT Press, Cambridge MA, USA.

[GP01] González A. y Pérez R. (2001) Selection of relevant features in a fuzzy genetic learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 31(3): 417–425.

[GRP+07] González J., Rojas I., Pomares H., Herrera L., Guillén A., Palomares J. y Rojas F. (2007) Improving the accuracy while preserving the interpretability of fuzzy function approximators by means of multi-objective evolutionary algorithms. *International Journal of Approximate Reasoning* 44(1): 32–44.

[GS95] Geyer-Schulz A. (1995) *Fuzzy rule-based expert systems and genetic machine learning*. Heidelberg: Physica-Verlag.

[Han81] Hand D. J. (Ed.) (1981) *Discrimination and Classification.* Wiley, Chichester, U.K.

[HBT99] Homaifar A., Battle D. y Tunstel E. (1999) Soft computing-based design and control for mobile robot path tracking. En *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, (CIRA'99)*, páginas 35–40. Monterey, USA.

[HCHC04] Ho S. Y., Chen H. M., Ho S. J. y Chen T. K. (2004) Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 34(2): 1031–1044.

[HD95] Howard L. M. y D'Angelo D. J. (1995) The GA-P: A genetic algorithm and genetic programming hybrid. *IEEE Expert: Intelligent Systems and Their Applications* 10(3): 11–15.

[Her08] Herrera F. (2008) Genetic fuzzy systems: Taxonomy, current research trends and prospects. *Evolutionary Intelligence* 1: 27–46.

[HKC99] Hong T.-P., Kuo C. S. y Chi S. C. (1999) Mining association rules from quantitative data. *Intelligent Data Analysis* 3(5): 363–376.

[HLS03] Herrera F., Lozano M. y Sánchez A. M. (2003) A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems* 18: 309–338.

[HLV97] Herrera F., Lozano M. y Verdegay J. L. (1997) Fuzzy connectives based crossover operators to model genetic algorithms population diversity. *Fuzzy Sets and Systems* 92(1): 21–30.

[HLV98] Herrera F., Lozano M. y Verdegay J. L. (1998) Tackling real-coded genetic algorithms: operators and tools for behavioural analysis. *Artificial Intelligence Review* 12: 265–319.

[HM00] Herrera F. y Martínez L. (2000) A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems* 8(6): 746–752.

[HN01] Hoffmann F. y Nelles O. (2001) Genetic programming for model selection of TSK-fuzzy systems. *Information Sciences* 136(1-4): 7–28.

[Hol75] Holland J. H. (1975) *Adaptation in natural and artificial systems.* Ann arbor: The University of Michigan Press.

[Hol79] Holm S. (1979) A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* 6: 65–70.

[HRF04] Hernández J., Ramírez M. J. y Ferri C. (Eds.) (2004) *Introducción a la minería de datos.* Pearson.

[HYLW08] Hu Q., Yu D., Liu J. y Wu C. (2008) Neighborhood rough set based heterogeneous feature subset selection. *Information Sciences* 178(18): 3577–3594.

[ID80] Iman R. L. y Davenport J. M. (1980) Approximations of the critical region of the friedman statistic. *Communications in Statistics* 18: 571–595.

[IN07] Ishibuchi H. y Nojima Y. (2007) Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning* 44(1): 4–31.

[INM99a] Ishibuchi H., Nakashima T. y Morisawa T. (1999) Voting in fuzzy rule-based systems for pattern classification problems. *Fuzzy Sets and Systems* 103(2): 223–238.

[INM99b] Ishibuchi H., Nakashima T. y Murata T. (1999) Design of accurate classifiers with a compact fuzzy-rule base using an evolutionary scatter partition of feature space. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 29(5): 601–618.

[INN04] Ishibuchi H., Nakashima T. y Nii M. (2004) *Classification And Modeling With Linguistic Information Granules: Advanced Approaches To Linguistic Data Mining.* Springer Verlag.

[INT92] Ishibuchi H., Nozaki K. y Tanaka H. (1992) Distributed representation of fuzzy rules and its application to pattern classification. *Fuzzy Sets and Systems* 52: 21–32.

[INT94] Ishibuchi H., Nozaki K. y Tanaka H. (1994) Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorithms. *Fuzzy Sets and Systems* 65: 237–253.

[INYT94] Ishibuchi H., Nozaki K., Yamamoto N. y Tanaka H. (1994) Construction of fuzzy classification systems with rectangular fuzzy rules using genetic algorihtms. *Fuzzy Sets and Systems* 65(2-3): 237–253.

[INYT95] Ishibuchi H., Nozaki K., Yamamoto N. y Tanaka H. (1995) Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems* 3(3): 260–270.

[IY04] Ishibuchi H. y Yamamoto T. (2004) Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy Sets and Systems* 141(1): 59–88.

[IY05] Ishibuchi H. y Yamamoto T. (2005) Rule weight specification in fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems* 13(4): 428–435.

[IYN05] Ishibuchi H., Yamamoto T. y Nakashima T. (2005) Hybridization of fuzzy gbml approaches for pattern classification problems. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics* 35(2): 359–365.

[Jan98] Janikow C. Z. (1998) Fuzzy decision tress: issues and methods. *IEEE Transactions on Systems, Man and Cybernetics* 28(1): 1–14.

[JD88] Jain A. K. y Dubes R. C. (1988) *Algorithms for Clustering Data*. Prentice-Hall.

[JM97] Jang J. R. y Mizutani C. S. E. (1997) *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall.

[KBA+97] Koza J. R., Bennett III F. H., Andre D., Keane M. A. y Dunlap F. (1997) Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation* 1(2): 109–128.

[KJ97] Kohavi R. y John G. H. (1997) Wrappers for feature subset selection. *Artificial Intelligence* 97(1-2): 273–324.

[Klö96] Klösgen W. (1996) Explora: A multipattern and multistrategy discovery assistant. En *Advances in Knowledge Discovery and Data Mining*, páginas 249–271. AAAI Press, Menlo Park, USA.

[Kov04] Kovacs T. (2004) *Strength or Accuracy: Credit Assignment in Learning Classifier Systems*. Springer-Verlag.

[Koz89] Koza J. R. (1989) Hierarchical genetic algorithms operating on populations of computer programs. En *Proceedings of the 11th International Joint Conference on Artificial Intelligence, (IJCAI'89)*, volumen 1, páginas 768–774. Morgan Kaufmann, Detroit, USA.

[Koz92] Koza J. R. (1992) *Genetic Programming: On the programming of computers by means of natural selection*. The MIT Press, Cambridge MA, USA.

[Koz94a] Koza J. R. (1994) Genetic programming as a means for programming computers by natural selection. *Statistics and Computing* 4(2): 87–112.

[Koz94b] Koza J. R. (1994) *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press, Cambridge MA, USA.

[Koz94c] Koza J. R. (1994) Recognizing patterns in protein sequences using iteration-performing calculations in genetic programming. En *Proceedings of the First IEEE Conference on Evolutionary Computation*, páginas 244–249.

[KR92] Koza J. R. y Rice J. P. (1992) Automatic programming of robots using genetic programming. En *Proceedings of the Tenth National Conference on Artificial Intelligence*, páginas 194–201. The MIT Press.

[Kun00] Kuncheva L. I. (2000) *Fuzzy classifier design*. Springer, Berlin.

[KY95] Klir G. y Yuan B. (1995) *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall.

[KZ02] Klösgen W. y Zytkow J. (Eds.) (2002) *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, New York, USA.

[KZB07] Kouchakpour P., Zaknich A. y Bräunl T. (2007) Population variation in genetic programming. *Information Sciences* 177(17): 3438–3452.

[KZB09] Kouchakpour P., Zaknich A. y Bräunl T. (2009) Dynamic population variation in genetic programming. *Information Sciences* 179(8): 1078–1091.

[LL00] Lee R. S. T. y Liu J. N. K. (2000) Tropical cyclone identification and tracking system using integrated neural oscillatory leastic graph matching and hybrid RBF network track mining techniques. *IEEE Transactions on Neural Networks* 11: 680–689.

[LL04] Lin M.-Y. y Lee S.-Y. (2004) Interactive sequence discovery by incremental mining. *Information Sciences* 165(3-4): 187–205.

[LM98] Liu H. y Motoda H. (1998) *Feature selection for knowledge discovery and data mining.* Kluwer Academic Publishers.

[LMB02] Li R. P., Mukaidono M. y Burhan I. (2002) A fuzzy neural network for pattern classification and feature selection. *Fuzzy Sets and Systems* 130(1): 101–108.

[LY05] Liu H. y Yu L. (2005) Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering* 17(3): 491–502.

[MA75] Mamdani E. H. y Assilian S. (1975) An experiment in linguistic systhesis with a fuzzy logic controller. *International Journal of Man-Machine Studies* 7: 1–13.

[McL92] McLachlan G. J. (1992) *Discriminant analysis and statistical pattern recognition.* Wiley-Interscience.

[MdBVFN01] Mendes R. R. F., de B. Voznika F., Freitas A. A. y Nievola J. C. (2001) A genetic-programming-based approach for the learning of compact fuzzy rule-based classification systems. En *Principles of Data Mining and Knowledge Discovery: 5th European Conference (PKDD'01)*, LNCS vol. 2168, páginas 314–325. Springer-Verlag.

[MG95] Miller B. L. y Goldberg D. E. (1995) Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems* 9: 193–212.

[Mic92] Michalewicz Z. (1992) *Genetic algorithms + data structures = evolution programs*. Springer-Verlag.

[Mic96] Michalewicz Z. (1996) *Genetic algorithms + data structures = evolution programs*. Springer-Verlag.

[Mit97] Mitchell T. M. (Ed.) (1997) *Machine learning*. McGraw-Hill.

[MJG05] Mikut R., Jäkel J. y Gröll L. (2005) Interpretability issues in data-based learning of fuzzy systems. *Fuzzy Sets and Systems* 150(2): 179–197.

[MM06] Moguerza J. y Muñoz A. (2006) Support vector machines with applications. *Statistical Science* 21(3): 322–336.

[MMHL86] Michalski R. S., Mozetic I., Hong J. y Lavrac N. (1986) The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. En *Proceedings of Fifth National Conference on Artificial Intelligence, (AAAI'86)*, páginas 1041–1047. Morgan Kaufmann, Philadelphia, USA.

[MP96] Mitra S. y Pal S. K. (1996) Fuzzy self organization, inferencing and rule generation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 26: 608–620.

[MST94] Michie D., Spiegelhalter D. J. y Taylor C. C. (Eds.) (1994) *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited.

[MVBL09] Mucientes M., Vidal J. C., Bugarín A. y Lama M. (2009) Processing time estimations by variable structure TSK rules learned through genetic programming. *Soft Computing* 13(5): 497–509.

[NK97] Nauck D. y Kruse R. (1997) A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems* 89: 277–288.

[Par95] Paredis J. (1995) Coevolutionary computation. *Artificial Life* 2: 355–375.

[PD00] Potter M. A. y De Jong K. A. (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. *Evolutionary Computation* 8(1): 1–29.

[PD06] Popovici E. y De Jong K. (2006) The effects of interaction frequency on the optimization performance of cooperative coevolution. En *Proc. of the 8th annual conference on Genetic and Evolutionary Computation (GECCO'06)*, páginas 353–360. ACM, New York, NY, USA.

[PDH97] Palm R., Driankov D. y Hellendoorn H. (1997) *Model based fuzzy control*. Springer-Verlag, Berlin.

[Pea88] Pearl J. (1988) *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, Palo Alto, USA.

[Ped96a] Pedrycz W. (1996) Conditional fuzzy c-means. *Pattern Recognition Letters* 17(6): 625–632.

[Ped96b] Pedrycz W. (1996) *Fuzzy Modelling: Paradigms and Practices*. Kluwer Academic Publishers, Norwell, USA.

[Ped98] Pedrycz W. (1998) Fuzzy set technology in knowledge discovery. *Fuzzy Sets and Systems* 98(3): 279–290.

[PF01] Provost F. J. y Fawcett T. (2001) Robust classification for imprecise environments. *Machine Learning* 42(3): 203–231.

[PHH03] Pérez E., Herrera F. y Hernández C. (2003) Finding multiple solutions in job shop scheduling by niching genetic algorithms. *Journal of Intelligent Manufacturing* 14(3-4): 323–339.

[Pia91] Piatetsky-Shapiro G. (1991) Knowledge discovery in real databases: A report on the IJCAI-89 workshop. *AI Magazine* 11(5): 68–70.

[PRS01] Peña-Reyes C.-A. y Sipper M. (2001) Fuzzy CoCo: A cooperative-coevolutionary approach to fuzzy modeling. *IEEE Transactions on Fuzzy Systems* 9(5): 727–737.

[PT06] Papadakis S. E. y Theocharis J. B. (2006) A genetic method for designing TSK models based on objective weighting: application to classification problems. *Soft Computing* 10(9): 805–824.

[Qui93] Quinlan J. R. (1993) *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

[RB97] Rosin C. y Belew R. (1997) New methods for competitive coevolution. *Evolutionary Computation* 5(1): 1–29.

[RL99] Russell S. y Lodwick W. (1999) Fuzzy clustering in data mining for telco database marketing campaigns. En *Proc. of the 8th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99)*, páginas 720–726. New York, USA.

[RRZ00] Ravi V., Reddy P. J. y Zimmermann H. J. (2000) Pattern classification with principal component analysis and fuzzy rule bases. *European Journal of Operational Research* 126(3): 526–533.

[RSA03] Roubos J. A., Setnes M. y Abonyi J. (2003) Learning fuzzy classification rules from labeled data. *Information Sciences* 150(1-2): 77–93.

[RZ00] Ravi V. y Zimmermann H. J. (2000) Fuzzy rule based classification with FeatureSelector and modified threshold accepting. *European Journal of Operational Research* 123(1): 16–28.

[RZ01] Ravi V. y Zimmermann H. J. (2001) A neural network and fuzzy rule base hybrid for pattern classification. *Soft Computing* 5(2): 152–159.

[SB00] Silipo R. y Berthold M. R. (2000) Input features' impact on fuzzy decision processes. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 30(6): 821–834.

[SB01] Setnes M. y Babuška R. (2001) Rule base reduction: Some comments on the use of orthogonal transforms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 31: 199–206.

[SCC01] Sánchez L., Couso I. y Corrales J. A. (2001) Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Information Sciences* 136(1-4): 175–191.

[Sch95] Schwefel H. P. (1995) *Evolution and Optimum Seeding.* Wiley Inc.

[Seb84] Seber G. A. F. (1984) *Multivariate Observations*. John Wiley and Sons.

[She03] Sheskin D. (2003) *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC.

[SJ04] Shen Q. y Jensen R. (2004) Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring. *Pattern Recognition* 37(7): 1351–1363.

[Smi80] Smith S. F. (1980) *A learning system based on genetic algorithms*. PhD thesis, University of Pittsburgh.

[SR00] Setnes M. y Roubos H. (2000) GA-fuzzy modeling and classification: complexity and performance. *IEEE Transactions on Fuzzy Systems* 8(5): 509–522.

[ST95] Silberschatz A. y Tuzhilin A. (1995) On subjective measures of interestingness in knowledge discovery. En *Proc. of the 1st International Conference on Knowledge Discovery and Data Mining (KDD'95)*, páginas 275–281. Menlo Park, USA.

[TAKJ96] Tunstel E., Akbarzadeh-T M.-R., Kumbla K. y Jamshidi M. (1996) Soft computing paradigms for learning fuzzy controllers with applications to robotics. En *Proceedings of the 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS'96)*, páginas 335–359. Berkely, USA.

[Thr91] Thrift P. (1991) Fuzzy logic synthesis with genetic algorithms. En *Proc. of the 4th International Conference on Genetic Algorithms (ICGA'91)*, páginas 509–513. San Diego, USA.

[TS85] Takagi T. y Sugeno M. (1985) Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics* 15(1): 116–132.

[Tsa06] Tsakonas A. (2006) A comparison of classification accuracy of four genetic programming-evolved intelligent structures. *Information Sciences* 176(6): 691–724.

[TSK06] Tan P.-N., Steinbach M. y Kumar V. (2006) *Introduction to Data Mining*. Pearson Education, Boston, USA.

[TSM85] Titterington D. M., Smith A. F. M. y Makov U. E. (1985) *Statistical Analysis of Finite-Mixture Distributions.* Wiley, Chichester, U.K.

[TT01] Tettamanzi A. y Tomassini M. (2001) *Soft Computing: Integrating Evolutionary, Neural and Fuzzy Systems.* Springer.

[VAD07] Van Broekhoven E., Adriaenssens V. y De Baets B. (2007) Interpretability-preserving genetic optimization of linguistic terms in fuzzy models for fuzzy ordered classification: An ecological case study. *International Journal of Approximate Reasoning* 44(1): 65–90.

[Vap95] Vapnik V. (1995) *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, USA.

[WC99] Wei Q. y Chen G. (1999) Mining generalized association rules with fuzzy taxonomic structures. En *Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99)*, páginas 477–481. New York, USA.

[Wie04] Wiegand R. P. (2004) *An Analysis of Cooperative Coevolutionary Algorithms.* PhD thesis, George Mason University, Fairfax, Virginia.

[Wil95] Wilson S. (1995) Classifier fitness based on accuracy. *Evolutionary Computation* 3(2): 149–175.

[WK88] Whitley L. D. y Kauth J. (1988) GENITOR: A different genetic algorithm. En *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, páginas 118–130. Denver, USA.

[WK91] Weiss S. M. y Kulikowski C. A. (1991) *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Networks, Machine Learning, and Expert Systems.* Morgan Kaufmann, San Francisco, CA.

[WL00] Wong M. L. y Leung K. S. (2000) *Data Mining using grammar based genetic programming and applications.* Kluwer Academics Publishers.

[WL02] Wang J.-S. y Lee C. S. G. (2002) Self-adaptive neuro-fuzzy inference systems for classification applications. *IEEE Transactions on Fuzzy Systems* 10(6): 790–802.

[WLD01] Wiegand R. P., Liles W. C. y De Jong K. A. (2001) An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. En *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'01)*, páginas 1235–1242. San Francisco, USA. Errata disponible en *http://www.tesseract.org/paul/papers/gecco01-cca-errata.pdf*.

[WM92] Wang L. X. y Mendel J. M. (1992) Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics* 22(6): 1414–1427.

[Wol75] Wold H. (1975) Soft modelling by latent variables: the nonlinear iterative partial least squares (NIPALS) approach. En Gani J. (Ed.) *Perspectives in Probability and Statistics: Papers in Honor of M.S. Bartlett*, páginas 117–144. Academic Press, London, U.K.

[Wro97] Wrobel S. (1997) An algorithm for multi-relational discovery of subgroups. En *Principles Of Data Mining And Knowledge Discovery*, páginas 78–87. Springer.

[Yag96] Yager R. R. (1996) Database discovery using fuzzy sets. *International Journal of Intelligent Systems* 11: 691–712.

[YBS02] Yu S., Backer S. D. y Scheunders P. (2002) Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. *Pattern Recognition Letters* 23(1-3): 183–190.

[YBY99] Yam Y., Baranyi P. y Yang C. (1999) Reduction of fuzzy rule base via singular value decomposition. *IEEE Transactions on Fuzzy Systems* 7(2): 120–132.

[YW99] Yen J. y Wang L. (1999) Simplifying fuzzy rule-based models using orthogonal transformation methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 29: 13–24.

[Zad65] Zadeh L. A. (1965) Fuzzy sets. *Information Control* 8: 338–353.

[Zad75] Zadeh L. A. (1975) The concept of a linguistic variable and its applications to approximate reasoning, parts I, II, III. *Information Sciences* 8-9: 199–249, 301–357, 43–80.

[Zad77] Zadeh L. A. (1977) Fuzzy sets and their applications to classification and clustering. En Ryzin J. V. (Ed.) *Classification and Clustering*, páginas 251–299. Academic Press, New York.

[Zad92] Zadeh L. A. (1992) Knowledge representation in fuzzy logic. En Yager R. R. y Zadeh L. A. (Eds.) *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, páginas 1–25. Kluwer, Boston.

[Zad94] Zadeh L. A. (1994) Soft computing and fuzzy logic. *IEEE Software* 11(6): 48–56.

[Zar99] Zar J. H. (1999) *Biostatistical Analysis*. Prentice Hall.

[ZPR09] Zhang M.-L., Peña J. M. y Robles V. (2009) Feature selection for multi-label naive bayes classification. *Information Sciences* 179(19): 3218–3229.

[Zur92] Zurada J. M. (1992) *Introduction to Artificial Neural Systems*. West Publishing Company.

[ZZ96] Zembowicz R. y Zytkow J. (1996) From contingency tables to various forms of knowledge in databases. En *Advances in Knowledge Discovery and Data Mining*, páginas 329–351. AAAI Press, Menlo Park, USA.