UNIVERSIDAD DE CÓRDOBA

Departamento de Informática y Análisis Numérico

Modelos de Aprendizaje Basados en Programación Genética para Clasificación Multi-Etiqueta

Memoria de Tesis presentada por

José Luis Ávila Jiménez

como requisito para optar al grado de

Doctor Ingeniero en Informática

Directores

Dr. Sebastián Ventura Soto

Dra. Eva Gibaja Galindo

Córdoba Abril de 2013

TITULO: *MODELOS DE APRENDIZAJE BASADOS EN PROGRAMACIÓN GENÉTICA PARA CLASIFICACIÓN MULTI-ETIQUETA*

AUTOR: *JOSÉ LUIS ÁVILA JIMÉNEZ*

© Edita: Servicio de Publicaciones de la Universidad de Córdoba.
Campus de Rabanales
Ctra. Nacional IV, Km. 396 A
14071 Córdoba

www.uco.es/publicaciones
publicaciones@uco.es

La memoria titulada "*Modelos de aprendizaje basados en programación genética para clasificación multi-etiqueta*", que presenta José Luis Ávila Jiménez para optar al grado de doctor, ha sido realizada dentro del programa de doctorado "*Ingeniería y tecnología*"de la Universidad de Córdoba, bajo la dirección de los doctores Sebastián Ventura Soto y Eva Gibaja Galindo, reuniendo, en su opinión, los requisitos exigidos a este tipo de trabajos.

Córdoba, Abril de 2013

El Director, La Directora,

Fdo. Sebastián Ventura Soto Fdo. Eva Gibaja Galindo

El Doctorando,

Fdo. José Luis Ávila Jiménez

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología (CICYT) con el proyecto **TIN2008-06681-C06-03** y por la Junta de Andalucía con fondos FEDER con el proyecto de excelencia **P08-TIC-3720.**

El hombre más irremediablemente estúpido

es aquel que ignora su sabiduría.

Isaac Asimov

Agradecimientos

Llevo bastante tiempo pensando en como escribir esto, ya que resulta difícil resumir en unas pocas palabras todo lo que siento al finalizar este trabajo. Es un buen momento para echar la vista atrás y reconocer que es un logro que en el fondo no me pertenece, sino que es en gran medida prestado o regalado, ya que han sido propiciado por todas las personas que me rodean.

En primer lugar tengo que agradecer a mis directores todo el gran trabajo que han hecho en esta tesis, ya que sin ellos no hubiera sido posible. Especialmente tengo que agradecer a Eva su infinita más una paciencia, y a Sebastián haber confiado sin dudar que lo lograría cuando ni yo mismo lo creía. A ambos tengo que reconocerles que sin su gran calidad humana nunca lo hubiera conseguido.

También tengo que agradecer su apoyo a mis compañeros del grupo de investigación KDIS y a todos los que me han ayudado en estos años que he estado trabajando en la Universidad de Córdoba. Gracias a todos ellos este trabajo ha sido posible.

Este agradecimiento se hace extensible a mis compañeros en el Instituto, y en los centros por los que he pasado anteriormente. Y por supuesto a mis alumnos, de todos los cursos y niveles con los que he tenido la suerte de trabajar, no sé si les habré enseñado mucho, pero desde luego he aprendido mucho de ellos.

Por supuesto tengo que dar las gracias a toda mi familia, a mi madre y a mi padre, a mi hermana y a mis abuelos, los que están y los que se fueron. Gracias a ellos soy como soy, con todos mis defectos y virtudes, y mis logros son en el fondo suyos. Y no me olvido de mi familia 'política', este agradecimiento también es para Paqui, Antonio y Vanesa por su apoyo.

Especialmente tengo que agradecer a Yolanda toda su comprensión y apoyo mientras he estado haciendo este trabajo, y que espero me perdone todo el tiempo que le he robado. El esfuerzo de esta tesis es tan suyo como mío.

Y para acabar el agradecimiento más importante es para mi hijo Isaac, por la forma en la que ha llenado mi vida de ilusión desde que nació. A ti especialmente va dedicado este trabajo.

A todos muchas gracias de todo corazón.

Indice de Contenidos

Indice de Figuras

Indice de Tablas

1

Planteamiento

El volumen de información almacenada en el mundo ha ido creciendo exponencialmente en las últimas décadas, y este crecimiento no tiene visos de detenerse. El hecho de que hoy en día haya memorias de gran capacidad a un precio por *byte* bastante reducido, hace que sea muy sencillo conservar información que hasta hace poco hubiera sido descartada sin más. Por esta razón en los últimos años, y en una amplia variedad de situaciones, se toma la decisión de almacenar sistemáticamente todos los datos disponibles en lugar de decidir qué merece la pena conservar y qué no.

En este contexto, en el que se calcula que cada 20 meses se duplica el volumen de la información almacenada en bases de datos digitales [2] y en el que dispositivos electrónicos almacenan nuestras decisiones, sea en el supermercado, navegando por Internet o solicitando un crédito, es necesario tener mecanismos que permitan no sólo almacenar y recuperar eficientemente los datos, sino también procedimientos que permitan obtener información y conocimiento útil a partir de estos datos en bruto.

El hecho de que las bases de datos estén en muchas ocasiones saturadas de información, trae consigo que aumente la brecha entre el almacenamiento de

esta información y su utilización real, es decir, la proporción entre datos e información útil decrece. Además, encerrada entre esta gran cantidad de datos sin duda hay mucho conocimiento útil, desaprovechado, y que simplemente está ahí, rodeado entre *terabytes* de ruido, en espera de ser hallado.

La *Minería de Datos* (*Data Mining*, *DM*) trata de resolver estos problemas mediante el análisis de toda esta información que ya está almacenada en bases de datos y, se puede definir como un proceso de búsqueda y descubrimiento de patrones en datos. El proceso idealmente debe ser automático y el objetivo principal es que los patrones descubiertos deben tener algún significado en el sentido de que puedan ser utilizados de alguna forma como conocimiento útil [2] (actualmente suele significar económicamente útil).

Estos patrones, o dicho de manera más adecuada, estos *conceptos* o *modelos* nos permitirán hacer predicciones sobre nuevos datos que se almacenen. Además, idealmente estos modelos deben estar expresados de manera inteligible, para que nos permitan no solamente hacer predicciones, sino también comprender por qué los datos están estructurados de dicha manera y no de otra.

Aunque la disciplina de la minería de datos abarca un amplísimo rango de tareas, la mayor parte de las técnicas de la minería de datos caen dentro de lo que se denomina *Aprendizaje Automático* (*Machine Learning*, ML) [2].

Dentro del aprendizaje automático debemos destacar el *Aprendizaje Inductivo*, en el que se basan la mayoría de las tareas de la minería de datos [3]. El objetivo del aprendizaje inductivo consiste en descubrir descripciones y conceptos generales a partir de un grupo de ejemplos, y por tanto lo que trata es de determinar de algún modo las similitudes entre ejemplos y suponer que estas similitudes pueden generalizarse a todos los ejemplos del mismo concepto. En definitiva el aprendizaje inductivo obtiene nuevo conocimiento en forma de conclusiones generales a partir de datos específicos. Dentro del

aprendizaje inductivo se suelen distinguir tres paradigmas clásicos de aprendizaje: el aprendizaje supervisado, el no supervisado y el aprendizaje con refuerzo[1] [4].

Las técnicas de ML se han utilizado en minería de datos para resolver una gran cantidad de tareas. Estas tareas se suelen agrupar en dos grandes categorías: tareas descriptivas y predictivas.

Las tareas descriptivas son las que tratan de obtener información acerca de la estructura de los datos almacenados, entre ellas encontramos las tareas de *Clustering* y *Asociación*:

- **Clustering:** La tarea de *clustering* es una tarea de aprendizaje supervisado que trata de agrupar un conjunto de objetos de acuerdo con un criterio de cercanía definido utilizando alguna función de distancia. En definitiva lo que hace la tarea es de agrupar objetos asumiendo que los patrones de un mismo grupo (o *cluster*) comparten propiedades comunes.

- **Asociación**: Esta tarea generalmente es de aprendizaje no supervisado y el objetivo es descubrir relaciones o asociaciones entre los distintos atributos del conjunto de datos. El resultado del proceso de aprendizaje es un conjunto de reglas de asociación, que determinan que si un nuevo patrón contiene determinados valores entre sus características, esto determinará otras características con un cierto grado de confianza.

[1]El aprendizaje supervisado se realiza cuando junto con los ejemplos, se suministra la salida o concepto que el sistema debe aprender; sin embargo en el aprendizaje no supervisado es el propio sistema el que debe de aprender la propia estructura del nuevo conocimiento que genera. El aprendizaje con refuerzo es una combinación de ambos paradigmas en el que conforme se van obteniendo resultados se indica si los conceptos aprendidos son correctos o no.

Por otro lado, las tareas predictivas tratan de predecir un comportamiento a partir de información ya almacenada. Entre las tareas predictivas encontramos las de *Regresión* y *Clasificación*:

- **Regresión:** Es una tarea que utiliza el aprendizaje supervisado y en la que los ejemplos se presentan como un conjunto de pares de elementos, que representan las entradas y la salida esperada del clasificador. La salida es un valor real y el objetivo es aprender una función que represente la correspondencia existente en los ejemplos.

- **Clasificación:** Esta tarea se caracteriza porque los ejemplos se presentan, del mismo modo que en regresión, como un conjunto de pares de elementos que representan las entradas y la salida esperada, siendo el objetivo de la tarea aprender una función mediante aprendizaje supervisado. Esta función se denomina clasificador y almacena la correspondencia existente en los ejemplos y clases, de modo que para cada valor de entrada tenemos un único valor de salida. La principal diferencia entre clasificación y regresión es que la salida es nominal, es decir, puede tomar un conjunto de valores discretos o clases que identifican si el ejemplo pertenece a un determinado concepto o no.

  Dentro de la tarea de clasificación, podemos encontrar varios tipos de subtareas con sus características específicas. Así por ejemplo si en los problemas a resolver los patrones están asociados a solamente dos clases se habla de clasificación binaria y por contra si el objetivo es aprender más de dos clases simultáneamente se habla de clasificación multiclase [5].

1.1. Clasificación multi-etiqueta

La tarea de clasificación se definió originalmente considerando que la correspondencia entre patrones y clases es unívoca, sin embargo, hay muchos problemas en los que la restricción de una etiqueta por patrón no se cumple. Dentro de estos problemas merece especial atención el paradigma en el que un patrón puede estar asociado simultáneamente con más de una clase, la llamada *Clasificación Multi-Etiqueta* (*Multi-Label Classification, MLC*) [6; 7].

La presente memoria trabaja sobre el campo de la clasificación multi-etiqueta ya que se trata de un área de creciente interés dentro del campo del aprendizaje automático, debido a la multitud de problemas que pueden ser abordados desde esta perspectiva.

De estos problemas, merece la pena citar aquí la clasificación semántica de imágenes [8] y textos [9; 10], problemas de bioinformática relacionados con la selección de la función de proteínas y genes [6], asociados con el diagnóstico médico [11] e interacciones entre medicamentos [12] o la categorización de sonidos [13].

Todo lo anteriormente comentado nos hace pensar que dentro del campo del aprendizaje automático, el problema de la clasificación multi-etiqueta es de un gran interés y elevada actualidad, y por este motivo la presente tesis tratará de, además de realizar un exhaustivo estudio del estado de la materia, proponer una serie de novedosos algoritmos que nos permitan modelar este tipo de problemas, usando para ello el paradigma de la programación genética.

1.2. Objetivos

El objetivo de la presente tesis es presentar una serie de avances en el modo de construir modelos de representación del conocimiento dentro del campo relacionado con el aprendizaje en clasificación multi-etiqueta. Para ello, en primer lugar, se hará un estudio detallado del estado del arte en la materia, revisando los principales modelos que hasta ahora se han propuesto para esta tarea bajo distintos paradigmas, lo que será el punto de partida para nuestras propuestas.

Dentro de este estudio se tratará de hacer una síntesis de las métricas más comúnmente utilizadas para medir el rendimiento de algoritmos de clasificación multi-etiqueta, ya que debido a sus características, distintas del problema de la clasificación clásica, el rendimiento de los algoritmos en la tarea ha de ser determinando de forma diferente [9]. Además también se hará un estudio de los conjuntos de datos que se han utilizado en distintos trabajos, revisando los métodos descritos en la literatura para determinar cómo de multi-etiqueta es un conjunto de datos [9].

A partir de esta revisión, se propondrán nuevos modelos de aprendizaje para clasificación multi-etiqueta. En concreto, esta tesis va a presentar la utilización de métodos de programación genética [14] aplicada al problema de la clasificación multi-etiqueta. Por un lado los algoritmos evolutivos en general suelen constituir una aproximación bastante buena a problemas de aprendizaje supervisado, y suelen además constituirse en una alternativa bastante eficaz a otros modelos de aprendizaje. Por otro lado los algoritmos de programación genética no han sido aplicados a los problemas de clasificación multi-etiqueta, a pesar de haber sido ámpliamente utilizados en la resolución de problemas de clasificación clásica.

En concreto nos apoyaremos en la propuesta de la *Gene Expression Programming* [15], un paradigma de programación genética que aunque fue originalmente diseñado para resolver problemas de regresión simbólica, ha sido muy utilizado en tareas de clasificación.

Los objetivos científicos que se persiguen en la presente tesis son los siguientes:

1. Revisión del estado del arte en relación al aprendizaje multi-etiqueta. En este trabajo se propone llevar a cabo un repaso general a la mayoría de los modelos y paradigmas que se han utilizado para solucionar problemas basados en el aprendizaje multi-etiqueta así como las aplicaciones prácticas que se han abordado mediante este. Se propondrá una notación unificada para el problema y además dentro de esta revisión se incluirán las métricas más comúnmente utilizadas para determinar el rendimiento de las propuestas de clasificación multi-etiqueta, así como los conjuntos de datos que se utilizan con más frecuencia en la comparativa de algoritmos.

2. Propuesta de un modelo de clasificación multi-etiqueta basado en funciones discriminantes. Las funciones discriminantes representan un modo de abordar problemas de clasificación utilizando funciones matemáticas que permitan separar a los patrones. El modelo aprende una función matemática que devuelve una salida real y dicha salida es discretizada según uno o varios umbrales que nos indican la clase a la que pertenece el ejemplo que se le pase como parámetro.

3. Propuestas de un modelo de clasificación multi-etiqueta basados en reglas. Las reglas de clasificación constituyen una forma más elaborada de representar el conocimiento, en el sentido de ser mucho más interpretables, que las funciones discriminantes. Por este motivo se va a

proponer un modelo que utilice reglas de clasificación. De esta manera se va a almacenar la información obtenida en forma de estructuras condicionales que permitan determinar las etiquetas asociadas a los nuevos patrones en función de los valores de sus atributos.

4. Validación experimental de los modelos propuestos. Todos los modelos que se desarrollen se probarán con conjuntos de datos multi-etiqueta de los existentes en la literatura, y además su rendimiento se comparará con otros enfoques que se han utilizado y descrito en la literatura para resolver problemas de clasificación multi-etiqueta. Este rendimiento se cuantificará utilizando varias de las métricas que están ya firmemente asentadas en otros trabajos previos sobre el problema y además se realizará un estudio estadístico de los resultados que nos permita comparar si los modelos que se proponen aquí son adecuados en referencia a otros modelos ya publicados.

1.3. Estructura de la memoria de tesis

Esta memoria se compone de seis capítulos. Además de la presente introducción, el resto de capítulos están estructurados de la siguiente manera:

El capítulo 2 introduce el problema de la clasificación multi-etiqueta, y sus principales áreas de aplicación. En el se presenta una nomenclatura uniforme para el problema. Posteriormente se presenta una taxonomía de los métodos que se han utilizado para abordar este tipo de problemas. A continuación se hace una enumeración de las métricas que se han utilizado con mayor frecuencia para comparar el rendimiento de este tipo de algoritmos, y por último se introducen los conjuntos de datos que se han utilizado en los estudios experimentales de este trabajo.

El capítulo 3 trata sobre programación genética. Se hace una introducción a los algoritmos evolutivos, a la programación genética y especialmente nos

centramos en la *Gene Expression Programming* (GEP), un paradigma de programación genética muy utilizado para resolver problemas de regresión y clasificación. Se introduce la codificación de los individuos en GEP, así como sus operadores genéticos. Posteriormente se hace una exposición del paradigma de la GEPCLASS [16], otra propuesta de programación genética, desarrollada a partir de la GEP y específicamente concebida para desarrollar clasificadores basados en reglas. Por último, se hace un repaso de los principales trabajos que han aplicado programación genética y GEP a problemas de clasificación.

En el capítulo 4 se expone el modelo GEP-MLC, un nuevo modelo desarrollado para abordar problemas de clasificación multi-etiqueta utilizando funciones discriminantes. Este algoritmo aprende al menos una función para cada etiqueta, constituyendo el clasificador final un conjunto de funciones discriminantes. Se detalla en el capítulo cómo se codifican las funciones discriminantes en los individuos, cómo se realiza la evaluación de estos, y además se expone un novedoso modelo de evaluación basado en nichos que garantiza que haya funciones asociadas a cada clase. Se finaliza detallando los experimentos realizados, los resultados experimentales así como la comparativa con otras propuestas multi-etiqueta, y el análisis estadístico de estos resultados.

El capítulo 5 presenta otro nuevo modelo para clasificación multi-etiqueta mediante reglas de clasificación, denominado GC. Este modelo se presenta en dos versiones cuya principal diferencia se produce en el consecuente de las reglas. El capítulo describe la codificación de los individuos, la evaluación de estos, junto con el modelo de nichos que se propone para la evaluación de reglas multi-etiqueta, el marco experimental, los valores de rendimiento de la propuesta frente a otras alternativas, junto con un estudio estadístico de dichos resultados y la discusión de estos resultados.

El capítulo 6 sirve, a modo de cierre, para presentar un resumen de las principales aportaciones que realiza la presente tesis, además se exponen los trabajos publicados fruto del desarrollo de la misma así como un esbozo de las futuras líneas de investigación que se seguirán a partir de los resultados y las conclusiones a las que se ha llegado tras la elaboración de esta memoria de tesis.

2

Clasificación multi-etiqueta

Como ya se ha descrito en el capítulo de introducción, se puede considerar la tarea de clasificación como una de las principales áreas de interés del campo del aprendizaje automático. Esta tarea esencialmente consiste en hallar una función, denominada clasificador, que reciba como entrada los atributos de un determinado patrón y devuelva como salida una clase, de entre una serie de *clases* predefinidas en el problema, que indique la clase con la que se considera que está asociado dicho patrón.

De forma clásica, este problema se ha formulado asumiendo la restricción de que cada patrón ha de estar asociado a solamente una clase de las definidas en el problema. En esta formulación clásica se habla de clasificación binaria cuando el número de posibles etiquetas es dos, y de clasificación multiclase cuando el número de etiquetas es mayor que dos [5].

No obstante, existen numerosos problemas de creciente actualidad en los que el requisito de que cada patrón pertenezca a una y sólo una clase es difícil de cumplir. Por ejemplo si se trata de modelar un problema de clasificación semántica de escenas como el presentado en [8], una imagen como la de la figura 2.1 que muestre el mar y una montaña o bien es asociada con la

Figura 2.1: Ejemplo de imagen multi-etiqueta

clase *mar* o con la clase *montaña*, pero no se puede asociar con las dos simultáneamente.

Para poder tratar adecuadamente este tipo de problemas surge el paradigma de aprendizaje denominado *Clasificación Multi-Etiqueta* (*Multi-Label Classification*, *MLC*). La clasificación multi-etiqueta se caracteriza porque los conjuntos de clases no son excluyentes entre sí, y por tanto puede haber patrones a los que se les asocie más de una etiqueta a la vez. De esta manera, si se utilizan técnicas de clasificación multi-etiqueta en lugar de técnicas de clasificación clásicas la imagen de la figura 2.1 puede ser asociada a las dos categorías de *mar* y *montaña* simultáneamente. En MLC las etiquetas asociadas a cada patrón son etiquetas binarias, que indican si dicho patrón está asociado no a la etiqueta.

En la tabla 2.1 se puede ver un conjunto de una sola etiqueta y multi-etiqueta respectivamente. De esta manera se observa que, mientras que el primero solo permite que un patrón esté asociado a una sola clase, en el segundo cada patrón puede estar asociado a varias categorías simultáneamente.

Ejemplo	Atributos	Etiqueta
1	X_1	Mar
2	X_2	Bosque
3	X_3	Mar
4	X_4	Ciudad

(a) Conjunto de una sola etiqueta

Ejemplo	Atributos	Etiquetas			
		Mar	Mont.	Bosque	Ciudad
1	X_1	T	T	F	F
2	X_2	F	T	T	F
3	X_3	T	F	F	F
4	X_4	F	T	T	T

(b) Conjunto multi-etiqueta

Tabla 2.1: Ejemplos de conjuntos de una y varias etiquetas

2.1. Tareas relacionadas

La clasificación multi-etiqueta se enmarca dentro de un área mucho más ámplia, el denominado *Aprendizaje Multi-Etiqueta*, (*Multi-Label Learning*, MLL). El aprendizaje multi-etiqueta abarca una gran variedad de tareas, entre las que podemos destacar, además de la clasificación multi-etiqueta, el *ranking*, la clasificación jerárquica multi-etiqueta, y clasificación multi-dimensional. En esta sección se van a comentar estas tareas brevemente ya que esto nos permitirá encuadrar a la MLC dentro del marco más ámplio que representa el MLL.

Los problemas de *ranking* [17] comparten bastantes aspectos con la clasificación multi-etiqueta. En el *ranking* lo que se busca es que el clasificador devuelva, para cada patrón, una lista ordenada con todas las etiquetas presentes en el conjunto de etiquetas definido en el problema, indicando dicho

orden el grado de afinidad del patrón con la etiqueta, es decir, las etiquetas con una posición superior en el *ranking* se consideran que están más asociadas al patrón que las que tienen una posición *ranking* inferior.

Las técnicas de *ranking* se pueden utilizar para realizar clasificación multi-etiqueta, lo que se denomina *Multi-Label Ranking*. Para ello a la técnica de *ranking* se le añade algún método de *bipartición*, que permita distinguir entre las etiquetas relevantes e irrelevantes en el patrón [18]. Cuando se utiliza *ranking* para resolver problemas multi-etiqueta surge el concepto de consistencia entre *ranking* y clasificación multi-etiqueta: un clasificador de *ranking* se considera consistente con un clasificador multi-etiqueta si las etiquetas predichas como positivas tienen siempre un *ranking* mejor que las no predichas.

La *Clasificación Multi-Etiqueta Jerárquica* [19], también llamada *Predicción Estructurada* [20] es la tarea de aprendizaje en la que se busca, de manera similar a la clasificación multi-etiqueta, predecir una serie de variables de etiqueta, pudiendo estar un patrón asociado a varias etiquetas. La diferencia con la clasificación multi-etiqueta estriba en que las etiquetas están jerárquicamente relacionadas (mediante una estructura de árbol o grafo dirigido acíclico), de tal manera que si un patrón está asociado con una etiqueta, estará también asociado con todas las etiquetas conectadas con ella. Además, esta estructura es un parámetro del problema, de tal manera que se conoce a priori la estructura jerárquica subyacente que agrupa las etiquetas.

La *Clasificación Multi-Dimensional* [21] trata también con problemas en los que un patrón puede estar asociado con varias etiquetas a la vez, pero en este caso las etiquetas no tienen valores binarios, sino multiclase.

Además de las tareas citadas, que se encuentran englobadas dentro del aprendizaje multi-etiqueta, también hay otras tareas de clasificación, que comparten ciertas similitudes con esta, como los problemas con múltiples etiquetas o múltiples instancias.

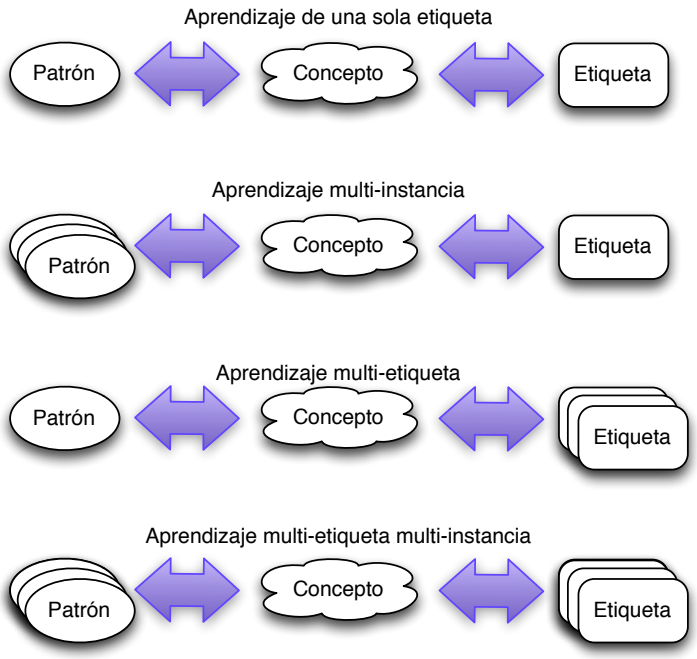


Figura 2.2: Taxonomía de las tareas de clasificación [1].

Los problemas de clasificación denominados de *Múltiples Etiquetas* (*Multiple Label Classification*) [22] son problemas de aprendizaje semi-supervisado en los que los ejemplos de entrenamiento están asociados con varias etiquetas pero solamente una de las etiquetas es la etiqueta válida. Básicamente el objetivo es generar un clasificador de una sola etiqueta, pero partiendo de información de entrenamiento de tipo multi-etiqueta.

En el aprendizaje *Multi-Instancia* [23], el conjunto de entrenamiento está formado por una serie de *bolsas* o conjuntos de patrones, y son las bolsas, en vez de los patrones individuales, las que se asocian con una clase.

En algunos trabajos se han desarrollado modelos para clasificación multi-instancia multi-etiqueta [1; 24], en los cuales cada bolsa está asociada a varias etiquetas simultáneamente. De hecho, estos mismos autores proponen una taxonomía en la que la clasificación clásica, multi-etiqueta y multi-instancia son diversas facetas o variantes del mismo problema (Figura 2.2).

Y por último cabe destacar el *Aprendizaje Multitarea* (*Multitask Learning*), donde se trata de resolver un problema de aprendizaje supervisado simultaneando su resolución con la de otros problemas relacionados, con el objetivo de mejorar el modelo para el problema que se trate permitiendo al procedimiento usar los elementos comunes entre las tareas. Estos elementos comunes en ocasiones se utilizan de forma similar a las distintas etiquetas en un problema multi-etiqueta, por lo que el aprendizaje multitarea, aun siendo un enfoque diferente, puede solaparse en algunos con el aprendizaje multi-etiqueta [25].

2.2. Notación

A continuación se define una notación para el problema de la clasificación multi-etiqueta. Esta notación será la que se utilice durante todo el documento y además trata de sintetizar la que se viene utilizando en la mayor parte de trabajos publicados.

Para una descripción formal de la tarea de clasificación multi-etiqueta se considerará al conjunto de etiquetas presente en un problema determinado $\boldsymbol{L} = \{\lambda_i : i = 1..n\}$ siendo n el número máximo de etiquetas que puede tener asociadas un patrón. Por otro lado, se denominará al conjunto $\boldsymbol{D} = \{\boldsymbol{D}_j : j = 1..m\}$ conjunto de datos multi-etiqueta. Cada elemento del conjunto de datos $\boldsymbol{D}$, también denominado patrón multi-etiqueta, estará compuesto por un par de elementos $\boldsymbol{D}_j = (\boldsymbol{X}_j, \boldsymbol{Y}_j)$ siendo $\boldsymbol{X}_j$ el vector de características asociado al patrón e $\boldsymbol{Y}_j \subseteq \boldsymbol{L}$ el subconjunto de etiquetas asociadas con el patrón j, por tanto $|\boldsymbol{Y}_j|$ será el número de etiquetas asociado al patrón j.

Debido a que en MLC las etiquetas son binarias, los conjuntos de etiquetas se pueden representar de dos maneras distintas, se puede considerar $\boldsymbol{Y}_j$ o bien como un vector binario, cuyos elementos positivos indican qué etiquetas

del conjunto $\boldsymbol{L}$ se encuentran asociadas con el patrón $\boldsymbol{D}_j$, o bien como una lista de etiquetas, que indican qué etiquetas de $\boldsymbol{L}$ tiene asociado $\boldsymbol{D}_j$.

La tarea de aprendizaje automático consistirá en encontrar una función o clasificador multi-etiqueta $h : \boldsymbol{X} \rightarrow \boldsymbol{P} \subseteq \boldsymbol{L}$ que para cada patrón j realice una bipartición del conjunto de etiquetas $\boldsymbol{L}$ que minimice la diferencia entre el conjunto de etiquetas predicho, $\boldsymbol{P}_j$, y el conjunto de etiquetas que realmente tiene asociado el patrón, $\boldsymbol{Y}_j$.

2.3. Métricas para la evaluación de modelos

El hecho de que la clasificación multi-etiqueta tenga características propias hace que sea necesario definir métricas específicas para determinar el rendimiento de los clasificadores multi-etiqueta. Se han propuesto diversas formas de determinar el rendimiento de un clasificador multi-etiqueta. Básicamente las métricas que existen se agrupan en métricas basadas en ejemplos (*example-based*) y métricas basadas en etiquetas (*label-based*) [26].

Las métricas basadas en etiquetas son métricas de clasificación binaria clásicas y se calculan para cada etiqueta, siendo posteriormente promediadas para obtener un valor único. Por contra las métricas basadas en ejemplos están específicamente concebidas para problemas multi-etiqueta, y por esta razón se calculan para cada ejemplo teniendo en cuenta todo el conjunto de etiquetas predicho y real.

En esta sección se definen las principales métricas, tanto basadas en etiquetas como basadas en ejemplos, que se pueden utilizar para medir el rendimiento de un algoritmo de aprendizaje multi-etiqueta. Para ello se hace uso de la notación introducida en el apartado 2.2.

		Valores predichos	
		Positivo	Negativo
Valores reales	Positivo	t_p	f_n
	Negativo	f_p	t_n

Figura 2.3: Matriz de confusión

2.3.1. Métricas basadas en etiquetas

Cualquier métrica de evaluación binaria, B, puede ser utilizada como base de una métrica basada en etiquetas. Estas medidas toman como base la matriz de confusión (Figura 2.3), en la que se representa, para una determinada clase, la salida del clasificador frente al valor esperado. Así t_p indica el número de patrones positivos clasificados correctamente, f_p los patrones incorrectamente asignados o falsos positivos, t_n los patrones negativos correctamente asignados o verdaderos negativos y f_n los patrones no asignados o falsos negativos.

Debido a que en MLC hay varias etiquetas presentes en el problema, para promediar se pueden utilizar dos aproximaciones, los denominados enfoques *macro* y *micro* [9]. Bajo el enfoque *macro* primero se calcula la métrica para cada etiqueta, y se hace la media por etiquetas para obtener el valor final (Ecuación 2.1), mientras que en el enfoque *micro* se agregan los valores de la matriz de confusión para las etiquetas, y posteriormente se calcula la métrica con estos valores (Ecuación 2.2).

$$B_{macro} = \frac{1}{n}\sum_{i=1}^{n} B(tp_i, fp_i, tn_i, fn_i) \tag{2.1}$$

$$B_{micro} = B(\sum_{i=1}^{n} tp_i, \sum_{i=1}^{n} fp_i, \sum_{i=1}^{n} tn_i, \sum_{i=1}^{n} fn_i) \tag{2.2}$$

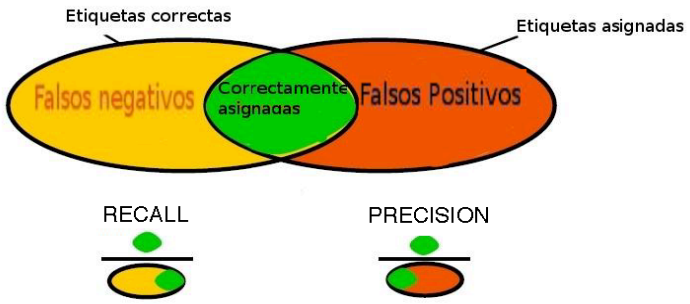


Figura 2.4: Representación gráfica de *precision* y *recall*

Aunque se puede utilizar cualquier métrica binaria, las dos medidas basadas en etiquetas utilizadas con más frecuencia para medir rendimiento son la *precision* y el *recall* del clasificador [9]. La *precision* de un clasificador se define como la fracción de etiquetas correctamente asignadas entre todas las etiquetas asignadas por el clasificador (Ecuación 2.3).

$$precision = \frac{t_p}{t_p + f_p} \tag{2.3}$$

El *recall* de un clasificador es la fracción de etiquetas correctamente clasificadas entre todas las etiquetas realmente positivas (Ecuación 2.4).

$$recall = \frac{t_p}{t_p + f_n} \tag{2.4}$$

La figura 2.4 expresa la relación entre *precision* y *recall* en función de la matriz de confusión de manera gráfica.

Ambas métricas pueden ser promediadas mediante las aproximaciones macro (Ecuación 2.1) y micro (Ecuación 2.2), siendo el resultado de su cálculo distinto según la aproximación que se use.

Debido al hecho de que en ocasiones se puede aumentar el *recall* a costa de disminuir la *precision* y viceversa, cuando se pretende maximizar ambas se

utiliza como métrica la media armónica entre la *precision* y el *recall*, también denominada *F-score*:

$$F\text{-}score = \frac{2 \times precision \times recall}{precision + recall} \tag{2.5}$$

Otra métrica frecuentemente utilizada, y también referida a la matriz de confusión, es la *accuracy* del clasificador. La *accuracy* es la fracción de etiquetas correctamente clasificadas (Ecuación 2.6).

$$accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{2.6}$$

2.3.2. Métricas basadas en ejemplos

Las métricas basadas en ejemplos toman como entrada la respuesta completa del clasificador para cada instancia. Posteriormente se promedia el valor de la métrica para obtener el valor medio de esta sobre todo el conjunto de entrenamiento.

Dentro de las métricas basadas en ejemplos encontramos las denominadas *métricas binarias*, que son utilizadas para medir cualquier algoritmo de clasificación multi-etiqueta y las llamadas *métricas de ranking* que se utilizan en el contexto de algoritmos de *ranking*.

2.3.2.1. Métricas binarias

La métrica más sencilla de calcular es la denominada *subset accuracy*, la cual calcula el porcentaje de ejemplos que el clasificador ha acertado en todas sus etiquetas asociadas (Ecuación 2.7). Esta métrica tiene la característica de que es muy estricta, ya que en el momento en que el conjunto real y el predicho no sean exactamente iguales, el patrón no puntúa en absoluto. Por esta razón no suele ser utilizada para evaluar clasificadores multi-etiqueta.

$$subset\,accuracy = \frac{1}{m}\sum_{i=1}^{m}\delta(\boldsymbol{P}_i = \boldsymbol{Y}_i) \tag{2.7}$$

Siendo:

$$\delta(x) = \begin{cases} 1 & si\ x = verdadero \\ 0 & si\ x = falso \end{cases}$$

Una métrica basada en ejemplos muy usada para evaluar clasificadores multi-etiqueta es la denominada pérdida de Hamming (*Hamming loss*). Esta métrica considera tanto los errores de clasificación (el hecho que que una etiqueta incorrecta sea predicha) como los errores por omisión (cuando una etiqueta que debería estar presente en la salida del clasificador no lo está). La *Hamming loss* es una medida de la diferencia entre el conjunto de etiquetas predicha y el conjunto de etiquetas real, siendo por tanto más cercana a 0 cuanto mejor sea el resultado del clasificador. En la ecuación 2.8 aparece la expresión de la métrica, siendo $|\boldsymbol{P}_i \Delta \boldsymbol{Y}_i|$ la diferencia simétrica entre los dos conjuntos de etiquetas predichas y reales.

$$Hamming\,loss = \frac{1}{m}\sum_{i=1}^{m}\frac{|\boldsymbol{P}_i \Delta \boldsymbol{Y}_i|}{n} \tag{2.8}$$

Godbole y Sarawagi, en [27], han redefinido las métricas *accuracy*, la *precision* y el *recall*, para que tuvieran en cuenta la respuesta completa del clasificador, y por tanto han de ser consideradas como métricas basadas en ejemplos. Estos autores definen la *precision* (Ecuación 2.9) como el promedio de etiquetas acertadas frente a las predichas, el *recall* (Ecuación 2.10) como el promedio de etiquetas acertadas frente a las reales y la *accuracy* (Ecuación 2.11) como el promedio de la fracción de aciertos del clasificador frente a las unión de etiquetas reales y predichas.

$$precision = \frac{1}{m}\sum_{i=1}^{m}\frac{|\boldsymbol{Y}_i \cap \boldsymbol{P}_i|}{|\boldsymbol{P}_i|} \tag{2.9}$$

$$recall = \frac{1}{m}\sum_{i=1}^{n}\frac{|\boldsymbol{Y}_i \cap \boldsymbol{P}_i|}{|\boldsymbol{Y}_i|} \tag{2.10}$$

$$accuracy = \frac{1}{m}\sum_{i=1}^{m}\frac{|\boldsymbol{Y}_i \cap \boldsymbol{P}_i|}{|\boldsymbol{Y}_i \cup \boldsymbol{P}_i|} \tag{2.11}$$

Otra forma, basada en ejemplos, de evaluar el rendimiento de un clasificador multi-etiqueta es la α-*evaluation*, propuesta por Boutell et al. [8]. En esta evaluación se puntúa cada predicción realizada por el clasificador según la ecuación 2.12.

$$score\left(\boldsymbol{P}_i\right) = \left(\frac{|\boldsymbol{Y}_i \cap \boldsymbol{P}_i|}{|\boldsymbol{Y}_i \cup \boldsymbol{P}_i|}\right)^{\alpha} (\alpha \geq 0) \tag{2.12}$$

Al parámetro α de la ecuación se de denomina *forgiveness rate* porque refleja el modo en el que la métrica penaliza errores en la clasificación. Un α pequeño implica que la medida es menos sensible a errores de clasificación, y un α grande hace que la medida sea muy sensible a fallos del clasificador. Llevado al extremo, cuando $\alpha = 0$, basta con que el clasificador asigne una etiqueta correcta para obtener la máxima puntuación, y en el caso opuesto, cuando $\alpha = \infty$ el clasificador solamente obtiene puntuación si predice correctamente todas las etiquetas.

Utilizando la puntuación de la ecuación 2.12, se definen las medidas de *precision*, *recall* y *accuracy* para un subconjunto de clases $C \subseteq L$ según las ecuaciones 2.13, 2.14 y 2.15 respectivamente.

$$precision_C = \frac{1}{|\boldsymbol{D}_C|}\sum_{i \in \boldsymbol{D_C}} score\left(\boldsymbol{P}_i\right), \tag{2.13}$$

Siendo $\boldsymbol{D}_C = \{\boldsymbol{D}_j | C = \boldsymbol{P}_i\}$

$$recall_C = \frac{1}{|\boldsymbol{D}_C|} \sum_{i \in \boldsymbol{D}_C} score\left(\boldsymbol{P}_i\right), \tag{2.14}$$

Siendo $\boldsymbol{D}_C = \{\boldsymbol{D}_j | C = \boldsymbol{Y}_i\}$

$$accuracy_D = \frac{1}{|\boldsymbol{D}|} \sum_{i \in \boldsymbol{D}} score\left(\boldsymbol{P}_i\right) \tag{2.15}$$

La *precision*, mide la fracción de patrones que contienen realmente el subconjunto de etiquetas C frente los que el clasificador le ha asociado el subconjunto C, y el *recall* la fracción de patrones que son clasificados como pertenecientes al subconjunto C frente a los que realmente lo contienen[1]. La *accuracy* proporciona una medida del rendimiento global del clasificador, en función del *forgiveness rate*.

2.3.2.2. Métricas de Ranking

Las métricas utilizadas para medir el rendimiento de un algoritmo de *ranking* también pueden utilizarse para determinar el rendimiento de un clasificador multi-etiqueta que esté basado en técnicas de *ranking*. Para su definición consideraremos que $r(\lambda)$ es la posición en el *ranking* de la etiqueta λ, es decir, en la etiqueta con mejor posición en el *ranking*, $r(\lambda) = 1$, la siguiente, $r(\lambda) = 2$ y así sucesivamente hasta la última, $r(\lambda) = n$.

La métrica *one error* mide el promedio de cuantas veces la etiqueta que el clasificador considera como más relevante no se encuentra en el conjunto de etiquetas asociado al patrón (Ecuación 2.16).

$$one\,error = \frac{1}{m} \sum_{i=1}^{m} \delta(\arg \min_{\lambda \in L} r_i(\lambda)) \tag{2.16}$$

[1]Con la α-*evaluation* también se pueden calcular ambas métricas para una sola etiqueta λ_i haciendo $C = \{\lambda_i\}$

Siendo:

$$\delta(x) = \begin{cases} 1 & si\ \lambda \notin Y_i \\ 0 & si\ no \end{cases}$$

La métrica de *cobertura* (*coverage*) evalúa hasta que punto hay que avanzar en la lista de *ranking* devuelta por el clasificador para cubrir todas las etiquetas asociadas a un ejemplo (Ecuación 2.17).

$$coverage = \frac{1}{m}\sum_{i=1}^{m}(\max_{\lambda \in Y_i} r_i(\lambda) - 1) \tag{2.17}$$

Ranking loss es una medida que nos indica el número promedio de veces en que las etiquetas irrelevantes en el ejemplo aparecen en el *ranking* antes que etiquetas relevantes (Ecuación 2.18)

$$rloss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\boldsymbol{Y}_i||\overline{\boldsymbol{Y}_i}|}|\{(\lambda_a, \lambda_b) : r_i(\lambda_a) > r_i(\lambda_b), (\lambda_a, \lambda_b) \in \boldsymbol{Y}_i \times \overline{\boldsymbol{Y}_i}\} \tag{2.18}$$

Y por último reseñar la *precisión media* (*average precision*) que nos indica el promedio de etiquetas con un *ranking* superior a una etiqueta $\lambda \in \boldsymbol{Y}_i$ (Ecuación 2.19), es decir, nos indica cuanto hay que avanzar en promedio en el *ranking* hasta encontrar una etiqueta determinada.

$$avgprec = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\boldsymbol{Y}_i|}\sum_{\lambda \in \boldsymbol{Y}_i}\frac{|\{\lambda' \in \boldsymbol{Y}_i : r_i(\lambda') \leq r_i(\lambda)\}|}{r_i(\lambda)} \tag{2.19}$$

2.4. Técnicas de clasificación multi-etiqueta

En este apartado se exponen las principales técnicas que se han utilizado para resolver problemas de clasificación multi-etiqueta.

Esencialmente se ha tratado de resolver problemas de clasificación multi-etiqueta desde dos puntos de vista, por un lado se ha tratado de modificar

los datos de entrada multi-etiqueta para convertirlos en datos de una sola etiqueta, a los que aplicar una técnica de clasificación clásica, y por otro lado, se han tratado de crear nuevas técnicas que puedan trabajar directamente con los datos multi-etiqueta [9].

Los métodos agrupados en la primera opción se denominan métodos de *transformación de problemas* ya que transforman un problema multi-etiqueta para obtener uno o varios conjuntos de datos de una sola etiqueta. Esta perspectiva puede tener algunos inconvenientes ya que en primer lugar, puede implicar una pérdida de información, bien en forma de etiquetas que se desprecian, o sencillamente por el hecho de que se pierdan las correlaciones entre etiquetas al generarse nuevos conjuntos de datos de una sola etiqueta. Además puede llegar a ser poco práctica para determinados conjuntos de datos por el coste que supone su transformación. Como contrapartida, este enfoque permite utilizar técnicas de clasificación clásica sobre estos datos, más ámpliamente probadas y aceptadas que sus equivalentes multi-etiqueta.

Al segundo grupo se le denomina métodos de *adaptación de algoritmos*, ya que adaptan técnicas de clasificación clásica a la clasificación multi-etiqueta. La mayor parte de las técnicas utilizadas en clasificación clásica han sido adaptadas para tratar con datos multi-etiqueta directamente. La principal diferencia entre los métodos de transformación de problemas y los de adaptación de algoritmos es que los primeros son independientes del paradigma de clasificación que se utilice posteriormente, pero los segundos no, aunque algunos métodos, internamente, modifiquen los datos de entrada de manera similar a alguna técnica de transformación.

A continuación se hace un repaso las diferentes propuestas que se han desarrollado recientemente dentro de ambos enfoques:

Libros	**Etiquetas**			
	Historia	**Religión**	**Filosofía**	**Ciencia**
El código Da Vinci	X	X		
La mente errabunda	X	X		X
Momentos estelares de la ciencia	X			X
Historia de la energía nuclear	X			X
Más Platón y menos prozac			X	X
La República Romana	X			
Historia del tiempo				X

Tabla 2.2: Ejemplo de conjunto multi-etiqueta: libros asociados con etiquetas en función de su temática

2.4.1. Métodos de transformación de problemas

Se han propuesto diversas formas de convertir conjuntos multi-etiqueta en conjuntos de una sola etiqueta, y todos estos métodos son independientes del algoritmo de clasificación que posteriormente se utilice para generar el modelo. En la figura 2.5 se muestra una taxonomía de los principales métodos y propuestas de transformación de problemas, de los cuales se comentan los principales rasgos en sucesivas secciones.

La tabla 2.2 muestra un conjunto de datos multi-etiqueta, formado por patrones que representan a varios libros que nos servirá de ejemplo al que aplicar las transformaciones. Las etiquetas asociadas a cada libro son las marcas binarias que nos indican si el libro pertenece o no a una serie de áreas temáticas (Historia, Religión, Filosofía o Ciencia).

2.4.1.1. Binary Relevance

Uno de los métodos de transformación de problemas más comúnmente utilizado es el método *Binary Relevance* (*BR*) [9; 28]. Éste método genera una serie de conjuntos de datos binarios, uno por cada etiqueta presente en el problema. Cada uno de los conjuntos generados contiene todos los patrones

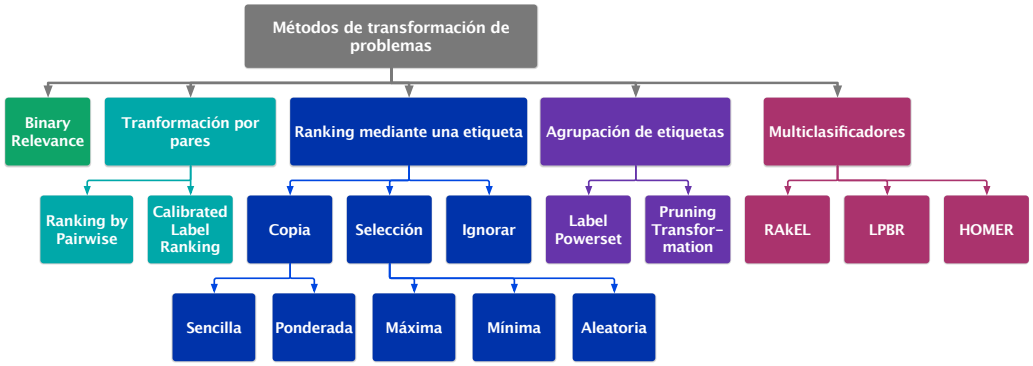


Figura 2.5: Taxonomía de los principales métodos de transformación de problemas

del conjunto original y en cada uno de ellos, están marcados como positivos los patrones que estaban asociados a la etiqueta, y como negativos el resto (Tabla 2.3).

Una vez realizada la transformación ha de utilizarse un algoritmo de clasificación binaria clásica que permita aprender un clasificador por cada nuevo conjunto generado. El clasificador multi-etiqueta se formará a partir de la agregación de todos estos clasificadores binarios, utilizando las respuestas de cada uno de los clasificadores aprendidos para determinar el conjunto de etiquetas a las que pertenece el patrón.

El método BR es una adaptación de la técnica *uno-contra-todos* (*one-versus-all, OVA*) utilizada en clasificación clásica para transformar un problema multiclase en varios binarios, entrenando un clasificador binario para cada clase con los los patrones de la clase marcados como positivos y el resto como negativos.

El método BR presenta como ventaja que es una transformación muy sencilla. Además la transformación aplicada es reversible, sin embargo presenta dos inconvenientes, por un lado asume que los conjuntos de etiquetas son independientes, y esto no siempre es cierto, y por tanto su utilización implica la pérdida de dependencia entre etiquetas que pudiera existir en el conjunto

Libros	Hist.	Rel.	Fil.	Cien.
El código Da Vinci	T	T	F	F
La mente errabunda	T	T	F	T
Momentos estelares de la ciencia	T	F	F	T
Historia de la energía nuclear	T	F	F	T
Más Platón y menos prozac	F	F	T	T
La República Romana	T	F	F	F
Historia del tiempo	F	F	F	T

Tabla 2.3: Ejemplo de transformación BR, aplicado al *dataset* de la tabla 2.2

de datos original [29; 30], y por otro lado, los conjuntos que genera suelen estar desbalanceados.

2.4.1.2. Ranking mediante una sola etiqueta

Otro conjunto de técnicas muy utilizadas son las conocidas como *Ranking Mediante una sola Etiqueta* (*Ranking Via Single Label*). Estas técnicas en primer lugar aplican una transformación sobre el conjunto de datos de entrada, que genere un conjunto de datos de una sola etiqueta. Posteriormente se ha de utilizar una técnica de clasificación clásica, que devuelve un valor que indique el valor de pertenencia de un patrón a cada clase (por ejemplo la probabilidad de pertenencia), con el fin de obtener un *ranking* de etiquetas según la transformación utilizada [8].

Se han propuesto varias formas de transformar el conjunto de datos, siendo estos métodos clasificados como métodos de *copia*, métodos de *selección* y el método de *ignorar* [9].

Los métodos de copia transforman cada patrón multi-etiqueta en varios patrones de una sola etiqueta. Para ello copian varias veces el patrón con cada una de las etiquetas que contiene.

Libros	**Clase**	**Peso**
El código Da Vinci	Historia	1/2
El código Da Vinci	Religión	1/2
La mente errabunda	Historia	1/3
La mente errabunda	Religión	1/3
La mente errabunda	Ciencia	1/3
Momentos estelares de la ciencia	Historia	1/2
Momentos estelares de la ciencia	Ciencia	1/2
Historia de la energía nuclear	Historia	1/2
Historia de la energía nuclear	Ciencia	1/2
Más Platón y menos prozac	Filosofía	1/2
Más Platón y menos prozac	Ciencia	1/2
La República Romana	Historia	1
Historia del tiempo	Ciencia	1

Tabla 2.4: Ejemplo de transformación de copia sencilla y ponderada, aplicado al *dataset* de la tabla 2.2

La ventaja de la copia es que no se pierde información sobre las clases, pero presenta el inconveniente que aumenta el número de patrones, siendo aquellos con más etiquetas los que tendrán más relevancia en el clasificador final. Esto se puede solventar aplicando el método de *copia ponderada*, en el que tras realizar la copia, a cada patrón generado se le asigna un peso en razón inversa al número de etiquetas que tuviera el patrón multi-etiqueta original. De esta manera se evita que los patrones con muchas etiquetas adquieran mayor relevancia en el clasificador final (Tabla 2.4). Sin embargo, el clasificador base debe permitir dar un peso a las instancias de entrenamiento, lo que limita las técnicas a utilizar.

Los métodos de selección transforman cada patrón multi-etiqueta en un patrón con una sola etiqueta, en contraposición con los de copia. El método debe seleccionar de algún modo la etiqueta que se le asigna al patrón de

Libros	**Clase(máx)**	**Clase(min)**
El código Da Vinci	Historia	Religión
La mente errabunda	Ciencia[1]	Religión
Momentos estelares de la ciencia	Ciencia[1]	Historia[2]
Historia de la energía nuclear	Ciencia[1]	Historia[2]
Más Platón y menos prozac	Ciencia[1]	Filosofía
La República Romana	Historia	Historia
Historia del tiempo	Ciencia	Ciencia

[1]Se podría seleccionar también *Historia*.
[2]Se podría seleccionar también *Ciencia*.

Tabla 2.5: Ejemplo de transformación de selección máxima y mínima, aplicado al *dataset* de la tabla 2.2

entre todas las que contiene el patrón multi-etiqueta. Dependiendo de como se realice esta selección se pueden diferenciar las siguientes técnicas:

- *Selección máxima:* se selecciona la etiqueta que aparece con más frecuencia en el conjunto de datos (Tabla 2.5).
- *Selección mínima:* se selecciona aquella etiqueta que menos ocurrencias tiene en el conjunto de datos (Tabla 2.5).
- *Selección aleatoria*: se selecciona aleatoriamente la etiqueta que se le asigna al patrón.

Estos métodos permiten trasformar de manera sencilla un *dataset* multi-etiqueta en uno de una sola etiqueta, sin aumentar su tamaño, sin embargo el principal inconveniente es que, además de eliminar información sobre etiquetas, eliminan toda la información sobre las relaciones entre etiquetas. Además es una transformación irreversible que implica una cierta pérdida de la información presente en el conjunto de datos original.

El método de *ignorar* elimina del conjunto de datos los patrones multi-etiqueta, quedando dicho conjunto constituido solamente por los patrones

Libros	**Clase**
La República Romana	Historia
Historia del tiempo	Ciencia

Tabla 2.6: Ejemplo de transformación de ignorar, aplicado al *dataset* de la tabla 2.2

no multi-etiqueta del conjunto de datos original (Tabla 2.6). Este método, aunque es muy simple de implementar, presenta los mismos inconvenientes de los métodos de selección, pero agravados, ya que hace que desaparezca de manera irreversible toda la información multi-etiqueta del conjunto original.

Por otro lado, en Boutell et al. [8] se muestra otra taxonomía de estos métodos de trasformación, agrupándolos en:

- Transformación *Model-s* (*single label*): A cada ejemplo se le asigna una sola etiqueta de las que posee, o bien aleatoriamente, o bien siguiendo algún criterio determinado. Similar a las trasformaciones de *selección*.
- Transformación *Model-i* (*ignore*): Se ignoran, y por tanto se eliminan los ejemplos multi-etiqueta. Esta es similar al método *ignorar*.
- Transformación *Model-n* (*new class*): Se crea una nueva clase para cada una de las posibles combinaciones de clases que existen en el conjunto de datos. Es similar a la que posteriormente veremos en la sección 2.4.1.4 denominada *Label Powerset* o LP.
- Transformación *Model-x* (*cross trainning*), la cual es similar al BR. La idea es que el algoritmo aprenda varios clasificadores binarios, uno para cada etiqueta, y el clasificador final se construya en base a estos clasificadores, que devuelven si un patrón pertenece o no a cada una de las clases existentes.

Libros	Conjuntos
	Historia-Religión
Momentos estelares de la ciencia	T
Historia de la energía nuclear	T
La República Romana	T
	Historia-Filosofía
El código Da Vinci	T
La mente errabunda	T
Momentos estelares de la ciencia	T
Historia de la energía nuclear	T
Más Platón y menos prozac	F
La República Romana	T
	Historia-Ciencia
El código Da Vinci	T
Más Platón y menos prozac	F
La República Romana	T
Historia del tiempo	F
	Religión-Filosofía
El código Da Vinci	T
La mente errabunda	T
Más Platón y menos prozac	F
	Religión-Ciencia
El código Da Vinci	T
Momentos estelares de la ciencia	F
Historia de la energía nuclear	F
Más Platón y menos prozac	F
Historia del tiempo	F
	Filosofía-Ciencia
La mente errabunda	F
Momentos estelares de la ciencia	F
Historia de la energía nuclear	F
Historia del tiempo	F

Tabla 2.7: Ejemplo de transformación por pares RPC, aplicado al *dataset* de la tabla 2.2

2.4.1.3. Métodos de transformación por pares

Los métodos de trasformación *por pares* (*pairwise methods*) [31] convierten un conjunto multi-etiqueta en varios conjuntos de datos binarios, uno por cada par de etiquetas. Posteriormente por cada uno de ellos se aprenderá un clasificador binario, utilizando cualquier técnica de clasificación binaria clásica.

Por ejemplo el método *Ranking by Pairwise Comparison (RPC)* [32] entrena cada uno de los modelos utilizando los patrones que están etiquetados con alguna de las dos etiquetas del par, pero omitiendo los patrones que está asociados con ambas etiquetas. Los patrones de una de las etiquetas actúan como patrones positivos y los de la segunda como patrones negativos, siendo por tanto otra manera de adaptar el método OVA a MLC (Tabla 2.7). Cada uno de estos conjuntos se utilizan para entrenar un clasificador binario, estando el modelo resultante constituido por todos los clasificadores entrenados. Las nuevas instancias son ordenadas mediante un *ranking* obtenido tras llamar a cada uno de estos clasificadores, y contando los votos que recibe cada etiqueta por cada clasificador.

El método *Calibrated Label Ranking (CLR)* [33] extiende el método anterior. La dinámica es similar a éste, pero añade una etiqueta virtual que se utiliza como punto de corte entre las etiquetas que se considerarán positivas y las que se considerarán negativas. A la hora de construir los conjuntos con la etiqueta virtual, los patrones que tiene asociada una determinada etiqueta se consideran positivos para esta y negativos para la virtual, y por el contrario los que no estén asociados a la etiqueta se consideran negativos para esta y positivos para la virtual. Por ejemplo, la salida final podría ser algo similar a *(Historia, Religión, Filosofía,* ***Virtual****, Ciencia)*, indicando la etiqueta virtual que el patrón pertenece solamente a las tres primeras.

2.4.1.4. Métodos de agrupación de etiquetas

Además de los anteriores hay una serie de técnicas que transforman los conjuntos multi-etiqueta agrupando las etiquetas entre sí, convirtiendo posteriormente cada grupo de etiquetas en una sola clase que pueda ser manejada como una tarea de clasificación de una sola etiqueta.

El método *Label Powerset (LP)* [9] revisa el conjunto de datos original y detecta todas las combinaciones de etiquetas que se dan en cada uno de los

Libros	**Clase**
El código Da Vinci	Historia-Religión
La mente errabunda	Historia-Religión-Ciencia
Momentos estelares de la ciencia	Historia-Ciencia
Historia de la energía nuclear	Historia-Ciencia
Más Platón y menos prozac	Filosofía-Ciencia
La República Romana	Historia
Historia del tiempo	Ciencia

Tabla 2.8: Ejemplo de transformación LP, aplicado al *dataset* de la tabla 2.2

patrones. Posteriormente genera un solo conjunto de datos multiclase, con tantas clases como combinaciones haya en el conjunto original, estando cada patrón asociado con la clase que representa a la combinación que tenía originalmente (Tabla 2.8). Es un método simple y efectivo que además tiene en cuenta las relaciones entre etiquetas. No obstante presenta ciertos problemas: tras la trasformación puede haber pocos patrones de entrenamiento para las combinaciones menos frecuentes y por otro lado solamente tendrá en cuenta las combinaciones presentes en el conjunto de datos, por lo que no se pueden predecir combinaciones de etiquetas que no estén en el conjunto de entrenamiento.

El método *Pruning Transformation (PT)* [34] es similar al LP pero está específicamente concebido para problemas con una gran cantidad de combinaciones de etiquetas. PT elimina los patrones asociados con combinaciones de etiquetas menos relevantes para el problema. Para ello elimina los patrones cuyas combinaciones aparecen menos veces que un determinado umbral definido por el usuario. Estos ejemplos eliminados no son descartados sin más sino que se pueden reintroducir en el conjunto de entrenamiento si sus etiquetas pueden ser divididas en subconjuntos con frecuencia superior al umbral. Por ejemplo, supongamos que un patrón de entrenamiento está asociado con la combinación descartable *(Historia, Filosofía ,Ciencia)*, en este

caso el patrón podría descomponerse en varias combinaciones, como *(Historia, Ciencia)* o *(Filosofía,Ciencia)*. Según el método PT, si alguna de estas combinaciones está por encima del umbral, se reintroducirá en el conjunto de entrenamiento un patrón asociado a esa combinación.

2.4.1.5. Métodos multiclasificadores

El último grupo de métodos de transformación son los denominados multiclasificadores (*ensemble methods*), los cuales combinan las respuestas de varios clasificadores, desarrollados mediante la misma o distinta técnica, para obtener una respuesta en general más adecuada que la de cada uno de los clasificadores por separado [35].

El algoritmo *Random k Labelset* (RAkEL), propuesto en [26], aprende una serie de clasificadores de tipo LP cada uno de ellos especializado en clasificar subconjuntos de etiquetas presentes en el problema. Para ello genera aleatoriamente una serie de subconjuntos de k etiquetas (*k-labelset*), entrenando un clasificador multi-etiqueta para cada uno de los subconjuntos. En el trabajo se propone que se entrene mediante clasificadores LP, pero el algoritmo puede aplicarse con cualquier clasificador multi-etiqueta. El clasificador final esta formado por la agregación de todos los clasificadores entrenados, y las etiquetas se asignan a los nuevos patrones por votación. Los mismos autores proponen además en [36] una exploración previa del espacio de etiquetas con el fin de encontrar los valores óptimos tanto para el número de subconjuntos como para el valor de k.

La propuesta LPBR [37] pretende llegar a un compromiso entre las transformaciones BR y LP, desde la primera en la que no se tienen en cuenta las relaciones entre etiquetas, hasta la segunda donde se tienen en cuenta todas las relaciones pudiendo llegar a un coste computacional inasequible. Para ello el *ensemble* actúa de la siguiente manera: genera un modelo BR, posteriormente genera un modelo que incluya la combinación de etiquetas más

relevante y evalúa si el rendimiento es mayor que el primero, si es mayor continúa con la siguiente combinación, incluyendo combinaciones hasta que se la agregación de nuevos modelos no aumenta el rendimiento del clasificador.

El algoritmo *Hierarchy Of Multi-labEl leaRners* (HOMER) [38] ha sido especialmente concebido para manejar conjuntos de datos de gran tamaño. Para ello se construye una jerarquía de clasificadores multi-etiqueta, encargándose cada uno de ellos de manejar un subconjunto de etiquetas. HOMER además trata de balancear el número de etiquetas que tendrá que manipular cada clasificador, tratando de crear subconjuntos de etiquetas que tengan subconjuntos similares de patrones. Por otro lado, en [39] se hace un estudio de los parámetros de HOMER, así como del impacto del uso de distintos clasificadores base en varios problemas, estudiando el coste computacional de las diversas opciones.

La propuesta EPS (*Ensemble of Pruned Set*) [40] construye un multiclasificador utilizando como base la transformación PT anteriormente comentada en la sección 2.4.1.4. Para ello entrena una serie de modelos usando PT, cada uno con una muestra del conjunto de entrenamiento proporcional al número de modelos, es decir, cuantos más modelos se decida utilizar, más pequeña será la muestra del conjunto de entrenamiento. Posteriormente los nuevos patrones son presentados a todos los modelos y clasificados por un mecanismo de votación. El *ensemble* es capaz de generar nuevas combinaciones de etiquetas combinando las predicciones por votación, mejorando en este aspecto a PT.

2.4.2. Métodos de adaptación de algoritmos

Los métodos de adaptación emplean una técnica de clasificación clásica adaptada para trabajar directamente con datos multi-etiqueta. Casi todas las técnicas de clasificación se han tratado de adaptar para solventar problemas multi-etiqueta. En la tabla 2.9 se describe un breve resumen de las técnicas,

Técnica	Referencias
Máquinas de vectores soporte	[6][41][42][43][44][45][46][47][48][49] [50]
Árboles de decisión	[51] [52] [53] [54] [55] [56] [57] [58]
Redes neuronales	[1] [59] [60] [61] [62] [63] [64] [65] [66]
kNN	[7] [67] [68] [69]
Clasificación asociativa	[70] [71] [72] [73] [74]
Métodos probabilísticos	[75] [76] [77] [78] [79]
Algoritmos bioinspirados	[80] [81]
Métodos multiclasificadores	[82] [83] [84] [85]

Tabla 2.9: Principales técnicas de adaptación de algoritmos, junto con sus referencias

así como las referencias más relevantes que las han descrito. A continuación se hace un breve repaso a los principales paradigmas adaptados y se comentan los principales trabajos que los han desarrollado:

2.4.2.1. Máquinas de vectores soporte

Las máquinas de vectores soporte, (*Support Vector Machines, SVM*) constituyen una técnica de clasificación lineal que busca una serie de hiperplanos en el espacio de características que maximicen la separación entre clases. En principio se concibieron para clasificación binaria, aunque posteriormente se han extendido para solucionar problemas como la clasificación multi-etiqueta. Podemos destacar las siguientes contribuciones:

En [6] Elisseeff y Weston plantean el algoritmo, denominado *Rank-SVM*, que utiliza SVM para tratar directamente con problemas multi-etiqueta mediante *ranking*. Para ello se construye un clasificador por cada etiqueta y se define un margen de separación multi-etiqueta para el todo el conjunto de entrenamiento, obteniendo buenos resultados en la clasificación funcional de proteínas.

En el trabajo de Wan y Xu [42], se proponen las llamadas *Triple Class SVM* que aplica la técnica *uno-contra-todos* y a los patrones con dos etiquetas los considera como pertenecientes a una clase intermedia entre las dos. La máquina de vectores soporte busca en este caso dos hiperplanos que separen las dos clases del área común donde están los patrones multi-etiqueta. Estos mismos autores proponen en [86] las *Double Label SVM*, similares a las anteriores, pero optimizadas para que su ejecución sea más rápida.

Jiang et al. proponen en [43] un método que, utilizando máquinas de vectores soporte, permite resolver problemas de clasificación y *ranking*. El clasificador devuelve un *ranking* de etiquetas que incluye una etiqueta virtual, al modo del algoritmo CLR, que sirve para discriminar entre las etiquetas relevantes e irrelevantes. De esta manera el algoritmo aprende simultáneamente el umbral más adecuado para cada problema multi-etiqueta.

El trabajo de Rousu et al. [44] se resuelven problemas de MLC jerárquica mediante SVM haciendo uso de cadenas de Markov. Una cadena de Markov modela un proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende del evento inmediatamente anterior, en este caso la cadena se utiliza para modelar relaciones entre atributos.

2.4.2.2. Árboles de decisión

Los árboles de decisión son una estructura muy utilizada en tareas de clasificación, y por ello este paradigma ha sido frecuentemente adaptado a problemas multi-etiqueta.

El trabajo de Clare y King [54] presenta una adaptación del algoritmo C4.5 donde se modifica la forma de construir el árbol para que éste permita la utilización de datos multi-etiqueta. En concreto se presenta una definición de la entropía multi-etiqueta que permite distinguir qué atributos se utilizarán en la construcción del árbol, además permite que haya múltiples etiquetas en las hojas de árbol y adapta el método de poda.

A partir del trabajo anterior trabajo surgen otras propuestas, como [55] donde se adapta a la clasificación multi-etiqueta el algoritmo J48, basado en el C4.5, para ello se adapta la definición de la entropía utilizada por el J48 de una sola etiqueta a las características de los datos multi-etiqueta, y además se proponen novedosos métodos de poda que tienen en cuenta que las clases objetivo se pueden solapar.

Por otro lado el trabajo de Noh et al. [57] se centra en un método de clasificación basado en árboles de decisión multi-etiqueta, en los que el criterio de decisión almacenado en cada nodo del árbol se calcula utilizando un test estadístico.

También subrayar los trabajos [51; 52] en los que se utilizan árboles de decisión para hacer clasificación multi-etiqueta jerárquica. Estos estudios comparan la utilización de un árbol sencillo para realizar predicciones simultaneas de todas las etiquetas frente a la opción de aprender varios árboles, uno por etiqueta. Por otro lado en el trabajo [53] se propone un *ensemble* basado en árboles de decisión, que utiliza el paradigma denominado *Predictive Clustering Trees (PCT)* [87], en el que se considera que un árbol de decisión realiza una tarea de *clustering* sobre los patrones de tal manera que todos patrones en el mismo *cluster* tienen propiedades similares.

2.4.2.3. Redes neuronales

El aprendizaje mediante redes neuronales es una de las técnicas más utilizadas en aprendizaje automático, y por este motivo se han propuesto múltiples adaptaciones de redes neuronales para ser utilizadas con problemas multi-etiqueta, de las cuales conviene destacar las siguientes:

Por un lado la propuesta de Crammer y Singer [59] en la que se expone el algoritmo *Multi-label Multi-class Perceptron (MMP)*, que es una generalización del perceptrón multicapa que permite trabajar con datos multi-etiqueta, y en el que la salida de la red neuronal es un *ranking* de etiquetas.

También destacar el trabajo de Zhang y Zhou [1] el el que se propone el algoritmo *Back Propagation Multi-Label Layer (BP-MLL)* el cual es un algoritmo neuronal de retropropagación que permite acomodar la información de patrones multi-etiqueta modificando la función de error.

Las redes neuronales de tipo *Adaptative Resonance Theory (ART)* [88], específicamente diseñadas para resolver problemas de reconocimiento de patrones, también han sido adaptadas a la clasificación multi-etiqueta [60], haciendo que la salida de la red sea un vector de etiquetas.

Por otro lado, también se han adaptado las redes neuronales *Radial Basis Network (RBF)* [61] las cuales utilizan una función de base radial como función de transferencia. Básicamente la adaptación consiste en añadir una nueva capa a la red de tal forma que se encargue de separar la información que corresponda a cada etiqueta.

Finalmente citar las redes neuronales probabilísticas, donde la red devuelve la probabilidad de pertenencia a una clase, que también han sido adaptadas a la clasificación multi-etiqueta, también alterando el número de capas de la red, en los trabajos [62; 63].

2.4.2.4. Algoritmo kNN

El algoritmo de los k vecinos más cercanos (k *Nearest Neighbour*, kNN), es uno de los más ámpliamente utilizados en clasificación clásica. Este algoritmo determina si un patrón pertenece a una clase determinando l los k patrones más cercanos del conjunto de entrenamiento y examinando la clase a la que pertenecen estos.

En [67] Zhang y Zhou proponen una modificación del algoritmo kNN, denominada *ML-*kNN, que adapta este algoritmo a la utilización de patrones multi-etiqueta. Para ello se le asignan etiquetas a los patrones teniendo cuenta las etiquetas que tienen asociados los k patrones más cercanos. Estos mismos autores en [7] hacen un estudio del rendimiento de este algoritmo

mostrando que para determinadas aplicaciones típicamente multi-etiqueta, como categorización de imágenes o determinación de la función de proteínas, tiene un rendimiento superior a otras propuestas basadas en *boosting* o en SVM.

2.4.2.5. Clasificación asociativa

En la tarea de asociación las clases no están predefinidas de antemano, sino que se busca agrupar patrones que presentan características similares [89]. Se han utilizado frecuentemente técnicas de asociación para resolver problemas de clasificación, utilizando estas técnicas para descubrir reglas relevantes, y posteriormente tratando de determinar si estas permiten asociar a conjuntos de patrones con las clases definidas en el problema [90].

Merece la pena destacar los trabajos [71; 72] donde se propone la *Multi-class Multi-label Associative Classification (MMAC)* que permite resolver problemas de clasificación, tanto clásica como multi-etiqueta enfocándolos como un problema de clasificación asociativa. El algoritmo va aprendiendo reglas sucesivamente mediante técnicas asociativas, y en cada iteración elimina del conjunto de entrada los patrones cubiertos por dichas reglas.

Estos mismos autores han planteado también utilizar la técnica MMAC para implementar un algoritmo de *ranking* denominado *Ranked Multi-label Rule (RMR)* [73; 74]. Este algoritmo funciona de manera similar al anterior pero incluye una etapa de poda para eliminar patrones de entrenamiento redundantes compartidos por las reglas.

Por otro lado, Rak *et al.* en [70] han propuesto un algoritmo que genera reglas asociativas multi-etiqueta utilizando el algoritmo *tree projection* [91] de minería de patrones frecuentes modificado para que permita acomodar la información multi-etiqueta.

2.4.2.6. Métodos probabilísticos

Los métodos probabilísticos tratan de aplicar alguna técnica estadística a problemas multi-etiqueta. Por ejemplo, en [75] se aplica un enfoque bayesiano a la clasificación de textos, que trata de aprender un modelo de probabilidades que determine si la presencia o no de determinados términos indica que el texto pertenece a una categoría semántica. Para ello se proponen tres enfoques, el enfoque *Single Label*, en el que cada término contribuye a la probabilidad de una sola etiqueta, el enfoque *Collective Multi-Label (CML)* en el que se tienen en cuenta las relaciones entre etiquetas y el enfoque *Collective Multi-Label with Features classifier (CMLF)* en el cual, además de considerarse las relaciones entre etiquetas, cada término puede contribuir a la probabilidad de más de una etiqueta.

Las redes bayesianas también se utilizan en el trabajo [76], mediante las denominadas *Multi-Dimensional Bayesian Network Classifiers (MBC)*, las cuales abordan la MLC como si un problema de clasificación multi-dimensional se tratara. Para ello se encadenan una serie de clasificadores bayesianos (uno por etiqueta), con el fin de capturar las correlaciones entre etiquetas, de tal manera que la salida de cada uno sirve de entrada al siguiente.

En [77] se ha desarrollado un clasificador probabilístico basado en el denominado *Latent Independent Component Analysis (LICA)*. Esta técnica considera un problema de MLC como un problema de *Multitask Learning* en el que hay que aprender simultáneamente varios clasificadores, uno por etiqueta, y se basa en utilizar una distribución de Laplace para hallar correlaciones ocultas entre las diversas etiquetas.

La búsqueda de correlaciones entre etiquetas se plantea también en [78], donde se describe el modelo *Correlated Label Model (CoL Model)*. Este modelo se utiliza para capturar las correlaciones estadísticas entre categorías y palabras de un texto en problemas de clasificación textual.

También merece la pena destacar que en [79] se propone que la distribución estadística de etiquetas en un conjunto de datos multi-etiqueta es la suma de las distribuciones de varios conjuntos de una sola etiqueta, y propone una técnica de *deconvolución* que pretende hallar la contribución de cada fuente individual a la distribución de datos multi-etiqueta.

Por último citar el trabajo [92] en el que se expone el método *Mixture Discriminant Analysis (MDA)*, que busca mediante técnicas estadísticas las fronteras entre clases.

2.4.2.7. Algoritmos bioinspirados

Por lo que sabemos, los enfoques bioinspirados apenas se han utilizado para resolver problemas multi-etiqueta. Entre estas propuestas merece la pena destacar dos:

En [80] Freitas propone el algoritmo *MULAM*, basado en colonias de hormigas. Cada hormiga descubre un conjunto de reglas candidatas para resolver el problema planteado, descubriendo en cada iteración, como mínimo, una sola regla y, como máximo, una regla por clase. En cada iteración se calcula la ganancia de información de los términos y la matriz de feromonas, para posteriormente generar las reglas de clasificación. Cada vez que se generan las reglas se eliminan del conjunto de datos los patrones cubiertos por las reglas descubiertas. El algoritmo finaliza cuando el número de patrones sin cubrir es menor que un parámetro predeterminado.

En segundo lugar destacar el trabajo de Vallim et al. [81] en el que se propone un algoritmo genético para resolver problemas de clasificación multi-etiqueta. El algoritmo modela un conjunto de reglas ya conocidas como población y las utiliza para descubrir nuevas reglas que se organizan jerárquicamente.

2.4.2.8. Métodos multiclasificadores (Ensemble methods)

El *boosting* es una técnica en la que se combinan las respuestas de varios clasificadores débiles (es decir, clasificadores que presentan una exactitud no muy elevada) para formar uno fuerte, y ha sido de las que más se ha utilizado para construir *ensembles* multi-etiqueta.

Utilizando *boosting* destacan los algoritmos *AdaBoost.MH* y *Adaboost.MR* [82] que realizan un mapeo del conjunto de datos multi-etiqueta en varios problemas de clasificación binaria. Para ello cada patrón multi-etiqueta se convierte en tantos patrones binarios como etiquetas haya definidas en el problema mediante la adición de un nuevo atributo que identifica a la etiqueta. Posteriormente se resuelve el problema binario mediante el tradicional algoritmo *AdaBoost* [93], utilizando como clasificadores base árboles de decisión. La diferencia entre ambas propuestas es que *AdaBoost.MH* está diseñado para minimizar la *Hamming loss* y *Adaboost.MR* para minimizar la *ranking loss* (métricas descritas en la sección 2.3).

Johnson y Cipolla [94] presentan una modificación del *AdaBoost.MH*, llamada *ML Boost*, que se diferencia del original en la forma en que se calculan los pesos de los clasificadores base de tal manera que se pueda adaptar con mayor flexibilidad al problema a resolver.

Por otro lado en [83] se propone el algoritmo *MP Boost*, que es otra modificación del algoritmo *AdaBoost*. En este caso el propio algoritmo es modificado, permitiendo la existencia de varios *pivotes* (valores de atributos que examina el clasificador para tomar una decisión), en lugar de uno, para probar las hipótesis de los clasificadores débiles.

También conviene destacar el algoritmo *ADTBoost* [58] donde se adapta el paradigma denominado *Alternating Decision Trees (ADT)* [95] a la clasificación multi-etiqueta. Los ADT son árboles de decisión con una estructura en la que se intercalan nodos denominados *partición* con nodos *decisión*, con

lo que la asociación de un patrón a una clase no se hace solo en las hojas del árbol. De esta forma a lo largo de un camino del árbol puede haber varios nodos *decisión*, cada uno de ellos asociado a una etiqueta.

Aparte del *boosting*, se han adaptado otros multiclasificadores al aprendizaje multi-etiqueta. La combinación de clasificadores, independientemente del clasificador base, también es abordada en [96] utilizando una red neuronal que permite aprender cual es el modo adecuado para combinar los clasificadores aprendidos. También hay trabajos en los que se utilizan SVM, como en [97] donde se entrenan varias SVM similares que varían en sus parámetros iniciales, y posteriormente el clasificador final es generado combinando los resultados de cada una de ellas mediante votación.

Por último comentar el trabajo [98] donde se propone un algoritmo genético para construir un multiclasificador que permita resolver problemas multi-etiqueta, utilizando el genético para aprender los pesos que deben tener cada uno de los clasificadores base en el *ensemble*.

2.5. Aplicaciones de la clasificación multi-etiqueta

La clasificación multi-etiqueta se ha aplicado para resolver múltiples problemas reales, en múltiples dominios, como categorización de documentos, bioinformática, clasificación de imágenes, o medicina entre otras. En esta sección se abordan las principales particularidades de cada una de estas aplicaciones, además de hacer un repaso a los trabajos que se han enfocado en estos problemas. En la tabla 2.10 se incluye un resumen las principales aplicaciones junto con las aportaciones que han contribuido a su resolución.

Problema	**Subproblema**	**Referencias**
Cat. de textos	Noticias	[47] [59] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108] [109]
	Doc. jurídicos	[110] [111]
	E-mail	[85] [112] [113] [114] [115]
	Clasif. de sent.	[116] [117] [118] [119]
	Otros	[111] [120] [121] [122] [123] [124] [125]
Bioinformática	Clasif. de proteínas	[26] [92] [126] [127] [128] [129] [130]
	Clasif. de genes	[6] [131] [132]
	Otros	[133] [134] [135]
Imágenes y video	Clasif. de Imágenes	[43] [136] [137] [138] [139] [140] [141] [142] [143] [144] [145] [146] [147] [148]
	Clasif. de vídeos	[149] [150] [151]
	Anotado de imágenes	[94] [146] [152] [153] [154] [155] [156] [157] [158] [159] [160] [161] [162] [163] [164] [165]
	Anotado de video	[166] [167]
	Rec. de patrones	[53] [168] [169] [170] [171] [172] [173] [174] [175] [176]
	Otros	[113] [177]
Medicina	Diagnosis	[178] [179] [180] [181]
	Medicamentos	[182] [183] [184] [185]
	Textos médicos	[186] [187] [188] [189]
	Imágenes médicas	[65] [190] [191] [192]
Cat. de sonidos	Géneros	[193] [194]
	Instrumentos	[195]
	tagging	[196] [197]
	Emociones	[49] [198] [199]
Otras aplicaciones	E-learning	[200] [201]
	Ingeniería	[98] [202] [203]
	Economía	[204] [205] [206]

Tabla 2.10: Problemas abordados mediante clasificación multi-etiqueta junto con sus referencias

2.5.1. Categorización de textos

Los primeros trabajos aplicados de clasificación multi-etiqueta se centraron en tratar de manejar la ambigüedad que se produce cuando se aborda la clasificación de documentos mediante técnicas de clasificación clásicas [21]. Este problema consiste en, dado un corpus documental, clasificar éste por

categorías semánticas, por ejemplo temáticas, estando estas categorías previamente definidas. La ambigüedad surge debido a que es razonable que un documento esté en dos o más categorías simultáneamente, y por tanto debiera estar asociado con todas estas etiquetas. Por ejemplo, no es difícil encontrar un artículo científico en el área de Ciencias de la Computación que pueda ser categorizado como de *machine learning* y *bases de datos* simultáneamente.

El hecho de que fuera una de las primeras aplicaciones del aprendizaje multi-etiqueta, ha generado la existencia de varios corpus documentales que se suelen utilizar frecuentemente para evaluar nuevas propuestas. Un ejemplo es el corpus textual de la agencia Reuters (RCV1) [109], que ha sido ámpliamente usado en numerosos trabajos bajo muy diversos paradigmas [47; 59; 99; 100; 101; 102; 103; 104; 105; 106; 107; 108]. Además de este, también cabe destacar el corpus de la ACM [120], que contiene información sobre artículos científicos, multi-etiquetados por categorías semánticas.

La clasificación de documentos jurídicos es un subproblema dentro de la clasificación de textos que tiene sus propias características particulares debido a la complejidad que entraña la búsqueda conceptual dentro de textos legislativos y jurídicos. También se abordó en los primeros trabajos [110], y, sobre esta temática uno de los corpus más utilizados es el conjunto de textos de legislación europea denominado EUR-LEX [111] el cual contiene documentación legislativa producida por la Unión Europea, incluyendo regulaciones, directivas comunitarias, decisiones y acuerdos estando todos ellos organizados jerárquicamente con más de 3000 etiquetas.

Otra base de datos de tipo textual es la del *Aviation Safety Reporting System (ASRS)*[2]. Esta contiene más de 7000 documentos, obtenidos a partir de los formularios que se deben rellenar cada vez que ocurre algún incidente

[2]Accesible on-line en `http://asrs.arc.nasa.gov`

o accidente aéreo, estando estos documentos clasificados en 60 etiquetas asociadas a tipos de problemas que pueden aparecer durante el vuelo [121].

El análisis y clasificación automática de correo electrónico es otro problema de clasificación de texto que se ha abordado mediante técnicas de clasificación multi-etiqueta [85; 112; 113; 114; 115]. La principal fuente de datos proviene del proyecto *Enron Email Analysis* [115] de la UC Berckeley, el cual ha clasificado manualmente mas de 1800 correos electrónicos asociados a 53 temáticas. El objetivo de este proyecto ha sido crear un conjunto de datos de *e-mails* reales que esté disponible para toda la comunidad científica. Estos correos reales provienen de altos directivos de la empresa Enron, y fueron hechos públicos en el transcurso de la investigación judicial producida tras la quiebra de la empresa.

Otro enfoque en cuanto a la clasificación textual, en el que se han aplicado técnicas de aprendizaje multi-etiqueta es lo que se ha llamado la *clasificación de sentimientos* (*sentiment classification*). El problema consiste en detectar estados afectivos, a través de texto [116; 117] o en otra variante, identificar opiniones mostradas en un escrito [118; 119].

También se ha utilizado la MLC en aplicaciones relacionadas con la web,como en el trabajo [123] en el que se propone su uso para el *tagging* automático de páginas, o [124] donde estas técnicas se emplean en la clasificación automática de artículos de Wikipedia. También se han propuesto técnicas multi-etiqueta para optimizar motores de búsqueda [125].

Y por último en esta sección nombrar otro problema modelado como MLC, la clasificación automática de patentes [122] en la que el documento en el que se describe esta se trata de clasificar automáticamente en varias categorías.

2.5.2. Bioinformática

El campo de la bioinformática es uno de los que más se ha beneficiado de la utilización de técnicas multi-etiqueta, ya que estas se han aplicado satisfactoriamente a problemas como la clasificación de proteínas o de genes, la determinación de la estructura de las proteínas o la multi-localización de proteínas, que se describe a continuación.

La clasificación de la función de proteínas es un problema que consiste en asociar cada proteína con un conjunto de clases que indiquen su función en los organismos vivos. El problema es multi-etiqueta porque hay proteínas que desempeñan varias funciones simultáneamente, por ejemplo la proteína *IL-6* tiene una actividad tanto proinflamatoria como antiinflamatoria en el cuerpo humano [207]. Ha sido ámpliamente tratado en numerosos trabajos mediante técnicas multi-etiqueta [26; 92; 126; 127; 128; 129] y de hecho existe el conjunto de datos genbase [130] que se suele utilizar para comparar propuestas sobre este dominio de aplicación.

Similar a este problema es el problema de la clasificación funcional de genes [6], en el que estos se clasifican también dependiendo de la funcionalidad que desarrollan las proteínas que sintetizan. Ya que una proteína puede estar asociada a varias funciones metabólicas, el problema ha sido tratado desde el enfoque multi-etiqueta [6; 131; 132].

Otro problema al que se ha aplicado la clasificación multi-etiqueta es el problema de la multi-localización de las proteínas. A la hora de determinar la función de una determinada proteína, uno de los principales aspectos que hay que tener en cuenta es donde se puede localizar una proteína dentro de una célula viva. Este problema por sí solo tiene suficiente envergadura como para ser considerado de manera separada a la simple clasificación funcional, y es modelado como multi-etiqueta ya que una misma proteína puede hallarse en varios puntos de una célula a la vez [133; 134].

Por último comentar un cuarto problema, la predicción de la estructura tridimensional de proteínas. Este problema de manera clásica se ha abordado como un problema de clasificación, categorizando las proteínas en dos niveles, según su estructura y según su plegamiento (*fold*). Sin embargo en [135] se plantea un modelo que aborda ambos modos de clasificación simultáneamente, presentando el algoritmo planteado características de clasificación multi-etiqueta jerárquica.

2.5.3. Tratamiento de imágenes y vídeo

La clasificación de escenas es un problema en el que se trata de asociar automáticamente a cada imagen la clase conceptual a la que pertenece [8]. Como se comentó al comienzo del presente capítulo (Figura 2.1), una imagen puede pertenecer a más de una clase conceptual simultáneamente, por ejemplo *mar* y *montaña*. Este problema ha sido de los más ámpliamente abordados utilizando técnicas multi-etiqueta, siendo enfocado desde numerosas ópticas y paradigmas [43; 136; 137; 138; 139; 140; 141; 142; 143; 144; 145; 146; 147; 148]. Incluso en varios trabajos se han utilizado las mismas técnicas para clasificar vídeos [149; 150; 151].

Otra aplicación directamente relacionada con la categorización de imágenes es el *tagging* o anotado automático de imágenes. En la web, y desde que surgió el concepto de web 2.0, es muy común que los usuarios de determinadas aplicaciones web etiqueten las imágenes que comparten a través de Internet; se han utilizado técnicas multi-etiqueta para automatizar o semiautomatizar este trabajo, sugiriéndole al usuario los *tags* que se consideran más adecuados para sus imágenes [94; 146; 152; 153; 154; 155; 156; 157; 158; 159; 160; 161; 162; 163; 164; 165; 208] y similares enfoques se han usado también para etiquetar vídeos [166; 167].

La clasificación de escenas mediante técnicas multi-etiqueta también se ha utilizado para realizar segmentación sobre imágenes y reconocimiento de

objetos. En este enfoque las distintas etiquetas con las que se asocia una imagen son los diversos objetos que aparecen en ella, siendo los objetos reconocidos conforme se le asocian etiquetas [53; 168; 169; 170; 171; 172; 173]. En otros trabajos el reconocimiento de objetos se ha llevado un poco más allá, utilizándolo para reconocer rasgos faciales [174; 175; 176], para realizar descripciones textuales automaticas de imágenes [177] o para filtrar imágenes de *spam* en correos electrónicos [113].

2.5.4. Medicina

Dentro del campo de la medicina, la clasificación multi-etiqueta se ha utilizado para solucionar varios problemas. El más inmediato de ellos es diagnóstico médico, este problema se puede modelar como multi-etiqueta ya que hay enfermedades muy distintas que comparten síntomas. Tratando el problema de la diagnosis, hay varios trabajos que aplican técnicas multi-etiqueta, por ejemplo en la medicina clásica china [179], o a determinar el diagnóstico tipos de cáncer de pulmón [181].

Otro problema que se ha modelado mediante técnicas de aprendizaje multi-etiqueta es el de las interacciones entre medicamentos, siendo las diversas etiquetas las substancias que presentan interacciones [182; 183; 184]. También se ha aplicado a problemas como la acción de medicamentos anti-cancerígenos sobre distintos tipos de células tumorales [185].

Las aplicaciones clásicas también se pueden extrapolar al campo de la medicina, como la categorización de textos, específicamente aplicada a artículos de medicina [186], o a historiales clínicos [187; 188; 189] o por otro lado la categorización de imágenes, aplicada específicamente a radiografías y otras imágenes médicas [65; 190] o el *tagging* automático de imágenes médicas [191; 192].

2.5.5. Categorización de sonidos

El problema de la categorización de sonidos consiste en clasificar sonidos, sobre todo musicales, según un criterio determinado. De la misma manera que ocurría con las imágenes, con mucha frecuencia es necesario modelar este problema con aprendizaje multi-etiqueta, porque las categorías en las que se agrupan las piezas musicales suelen solaparse entre sí. El problema de la categorización musical se ha abordado al menos desde tres puntos de vista distintos:

Se puede clasificar la música en géneros musicales, habiendo multitud de composiciones que se pueden considerar asociadas a varios géneros simultáneamente. De esta manera se enfoca el problema en los trabajos [193; 194]. Otra manera de clasificar las composiciones es según los instrumentos musicales que intervienen [195] siendo obvio que en una pieza pueden intervenir varios instrumentos.

Un segundo enfoque es tratar la clasificación musical como un problema de *tagging* automático, similar al *tagging* en imágenes y enfocado a aplicaciones web en la que los usuarios organizan o clasifican música según sus gustos o preferencias [196; 197].

Finalmente, un tercer enfoque, sobre el que también se ha trabajado es la clasificación de canciones y piezas musicales según la emoción que evocan en quien las oye. Una misma pieza puede causar dos sentimientos, como sorpresa y agresividad, simultáneamente, y es por esta razón que se usan técnicas de clasificación multi-etiqueta [49; 198; 199].

2.5.6. Otras aplicaciones

Además de las principales aplicaciones que se han descrito en las secciones precedentes, el aprendizaje multi-etiqueta se ha aplicado, y se está aplicando en la actualidad, a un enorme número de problemas de muy diversa

naturaleza, ente los que, además de los ya expuestos, conviene destacar los siguientes:

Se ha aplicado satisfactoriamente en entornos educativos y de *e-learning*, por ejemplo para hallar perfiles de aprendizaje del alumnado [200] o evaluar la manera en que los alumnos colaboran entre sí en un entorno de trabajo colaborativo utilizando *wikis* [201].

Dentro del campo de la ingeniería industrial se ha utilizado también para detectar anomalías en plantas de procesos químicos [202], para el diagnóstico de bombas y motores [98] o dentro de la ingeniería del software, para clasificar solicitudes de cambios dentro de la gestión de la configuración [203].

Las ciencias económicas también se han aprovechado del potencial de la clasificación multi-etiqueta, habiendo sido aplicada para predecir variables macro-económicas [204], para clasificar directorios de negocios [205] o para determinar el impacto económico en la planificación de programas de I+D [206].

2.6. Conjuntos de datos multi-etiqueta

Hay varios conjuntos de datos publicados que se suelen utilizar para comparar entre sí diversas propuestas de algoritmos que resuelvan el problema de clasificación multi-etiqueta. En esta sección en primer lugar se discuten qué métricas se pueden establecer sobre los conjuntos multi-etiqueta, y posteriormente se presentan los conjuntos multi-etiqueta que se han utilizado en la fase experimental de esta tesis.

2.6.1. Métricas para la descripción de conjuntos de datos

A la hora de plantear un desarrollo experimental sobre un modelo que genere clasificadores multi-etiqueta, así como de discutir los resultados obtenidos por éste, es importante determinar cómo de multi-etiqueta es el conjunto de datos con el que se valida. Para ello se han propuesto varias métricas.

La métrica más sencilla es el denominado *distinct* [9], el cual es el número de combinaciones de distintas etiquetas presente en un conjunto de datos.

Además del número de combinaciones, se han propuesto otras dos medidas, la *cardinalidad* y la *densidad* [9]. Si consideramos $|Y_i|$ en número de etiquetas del patrón i, la cardinalidad de un conjunto de datos D viene dada por el promedio de etiquetas por patrón (Ecuación 2.20).

$$cardinalidad(D) = \frac{1}{m}\sum_{i=1}^{m}|Y_i| \tag{2.20}$$

Sin embargo intuitivamente parece razonable considerar también el número total de etiquetas posible. Por ejemplo podemos tener dos conjuntos de cardinalidades similares, pero puede ocurrir que el número de posibles etiquetas máximo que pueda tener asignado un patrón varíe enormemente. Para evitar la distorsión que supone comparar cardinalidades entre conjuntos con diverso número de etiquetas se define la densidad como la cardinalidad entre el número total de etiquetas $|L|$ (Ecuación 2.21).

$$densidad(D) = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i|}{|L|} \tag{2.21}$$

Dataset	$\|D\|$	$\|L\|$	#atrib	Tipo	Card.	Dens.	Distinct
Bibtext	7395	159	1836	nominal	2.402	0.015	2856
Medical	978	45	1449	nominal	1.245	0.028	94
Rcv1	6000	101	47236	nominal	2.880	0.029	1028
Genbase	662	27	1186	nominal	1.252	0.046	32
Enron	1702	53	1001	nominal	3.378	0.064	753
Scene	2407	6	294	numérico	1.061	0.176	15
Yeast	2417	14	103	numérico	4.228	0.302	198
Emotions	593	6	72	numérico	1.868	0.311	27
TMC2007	28596	22	4960	nominal	2.158	0.098	1341
Mediamill	43907	101	120	numérico	4.376	0.043	655

Tabla 2.11: Características de los conjuntos de datos

2.6.2. Conjuntos de datos

En esta sección se describen brevemente los conjuntos de datos multi-etiqueta que se han utilizado en la fase experimental de esta tesis. Estos conjuntos pertenecen a gran cantidad de dominios, como clasificación de música y escenas, clasificación de proteínas y genes, diagnóstico médico o clasificación de textos, y han sido ámpliamente utilizados en otros trabajos relacionados con clasificación multi-etiqueta. Además presentan características muy dispares tanto en tipos de datos (nominales y numéricos) como en sus métricas de cardinalidad y densidad, así como en el número de combinaciones de etiquetas. La tabla 2.11 presenta las principales características y métricas de cada uno de estos conjuntos de datos.

El conjunto de datos *bibtext* [209] trata sobre clasificación textual, en este caso se trata de datos sobre etiquetado o *tagging*. La información presente es sobre numerosos documentos web etiquetados por usuarios. Se trata de un conjunto de datos nominal y sus características más reseñables son el

elevado número de etiquetas así como de combinaciones de etiquetas que presenta, ambos son los más elevados de los conjuntos presentados.

El conjunto de datos *medical* [187] es un conjunto de datos desarrollado para una competición internacional, el *2007 International Challenge: Classifying Clinical Free Text Using Natural Language Processing*, en la que se trataba de categorizar textos médicos por enfermedades. Es un conjunto nominal y destaca por el pequeño valor de densidad que posee.

Rcv1 [109] es el corpus documental de la agencia de noticias *Reuters*, y almacena datos sobre textos de noticias emitidos por la agencia y clasificados según la temática de la noticia o el artículo periodístico. Es un conjunto también nominal y destaca por su elevado número atributos.

El dataset *genbase* [130] contiene información sobre bioinformática, sobre proteínas y la función que desempeñan. *Genbase* es un conjunto pequeño en número de patrones, además de no presentar un elevado número de combinaciones de etiquetas.

El dataset *enron* [115] es un conjunto de datos también sobre textos, en concreto correos electrónicos hechos públicos a raíz del caso Enron y que fueron manualmente clasificados en categorías en la Universidad de Berkeley a lo largo del proyecto *Enron Email Analysis*. Es un conjunto nominal en el que cada etiqueta representa una de las categorías a las que se ha asociado el correo correspondiente.

El conjunto de datos *scene* [8] contiene una serie de patrones sobre imágenes de distintos tipos de paisajes. Cada imagen ha sido dividida en cuadrados, y sobre cada uno de ellos se han medido una serie de parámetros promedio del cuadro (color, brillo, saturación, etc.) y tiene asignada una o varias etiquetas en función del tipo de paisaje que contiene, por ejemplo playas, puestas de sol, montañas, campo, bosques y paisajes urbanos. Es uno de los dos conjuntos que presenta menor número de etiquetas siendo el que menos

combinaciones distintas contiene, además es el que menor valor de densidad posee de los mostrados.

El conjunto de datos *yeast* [6] contiene información sobre genes, y proteínas de la levadura. Clasifica cada gen dentro de en 14 posibles grupos funcionales dependiendo de la función metabólica que desempeñan las proteínas que genere. Destaca por ser el conjunto de mayor densidad de los estudiados.

El conjunto de datos *emotions* [210] contiene datos multi-etiqueta sobre piezas musicales. De cada pieza musical se extrajeron características de ritmo y de timbre, y dichas piezas fueron etiquetadas dependiendo de la emoción que evoca a quien la escucha, de entre seis posibles emociones (sorpresa, enfado, felicidad, tristeza, calma y relax). Es el más pequeño de los conjuntos, tanto en patrones como en atributos, además de ser junto con *scene* el que menos etiquetas presenta. Sin embargo también hay que destacar su elevada cardinalidad, la más alta de todos los conjuntos presentados.

Los datos de *TMC2007* [211] tratan sobre clasificación de textos, en concreto contiene información sobre informes técnicos de la industria aeroespacial y los clasifica en función de las anomalías que posteriormente ocurrieron y que pudieran haberse detectado a raíz del informe. Es un conjunto de tamaño elevado tanto en número de patrones como en número de atributos.

Por último, el conjunto *mediamill* [212] contiene información sobre ficheros de vídeo, y éstos son clasificados semánticamente de manera parecida a como se hace en clasificación semántica de escenas, con la intención de poder hacer búsquedas sobre ellos. Este conjunto de datos es el que tiene un mayor número de patrones de los aquí presentados.

3

Programación genética y GEP

El término *Computación Evolutiva* engloba un conjunto de técnicas de solución de problemas complejos de búsqueda y aprendizaje basadas en la emulación de modelos naturales de evolución. La computación evolutiva se inspira en la teoría de la selección natural propuesta por Charles Darwin en el siglo XIX, en la que postuló que los organismos vivos evolucionan y se adaptan al entorno en base a mutaciones aleatorias, cruce entre los miembros de la misma especie y selección y supervivencia de los más aptos [213]. La metodología de la computación evolutiva para la resolución de problemas consiste en el uso de mecanismos de selección de soluciones potenciales y de construcción de nuevos candidatos por recombinación de las características de otros ya presentes de modo parecido a como ocurre en los organismos naturales.

Dentro de la computación evolutiva se han desarrollado varios paradigmas fundamentales como la *Programación Evolutiva* [214], las *Estrategias Evolutivas* [215], los *Algoritmos Genéticos* [216] o la *Programación Genética* [14].

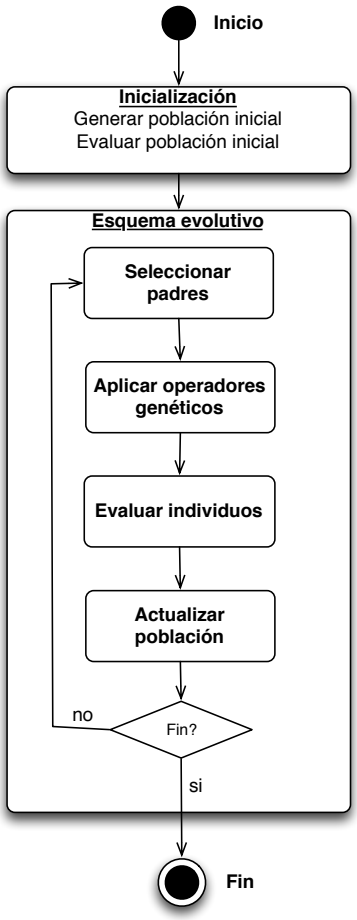


Figura 3.1: Estructura general de un algoritmo genético

3.1. Algoritmos Genéticos

Un *algoritmo genético* (AG) tiene como propósito genérico guiar una búsqueda estocástica haciendo evolucionar un conjunto de posibles soluciones a un problema y seleccionar de modo iterativo la más adecuada. La estructura general de un AG aparece en la figura 3.1.

Formalmente se pueden definir los AGs como algoritmos probabilísticos iterativos que mantienen una población de individuos $P(t)=\{x_1(t), x_2(t), ..., x_{\mathrm{n}}(t)\}$ para la iteración o *generación* t. Cada individuo representa de alguna manera una solución potencial (o parte de una solución potencial) al problema a

resolver, que es representada mediante una estructura de datos denominada *genotipo*. Para almacenar el genotipo se han utilizado múltiples estructuras de datos, como vectores de diversos tipos de datos, árboles, grafos, pilas, etc. Cada campo del genotipo representa una característica del problema, a la que se denomina *gen*. El significado de un gen particular se denomina *fenotipo* siendo este una solución o un faceta de la solución del problema a resolver [217].

Para guiar el proceso evolutivo es necesario evaluar cada solución $x_i(t)$. Para ello se utiliza una función de *aptitud* o *fitness* que proporciona un valor numérico que evalúa cuánto se aproxima esta a una solución óptima del problema.

En la iteración $t + 1$ se obtiene una nueva población mediante la selección de los mejores individuos de la población t. A los individuos seleccionados de la población se les aplica una serie de operadores genéticos, que los alteran generando nuevas soluciones candidatas. Después de algún numero de generaciones, el algoritmo converge hacia una solución, la cual estará razonablemente cerca de la solución óptima si el algoritmo está bien diseñado.

Un AG normalmente finaliza cuando se ha obtenido un individuo *adecuado*, es decir un individuo óptimo o cercano al óptimo, aunque se suele añadir como parámetro un número máximo de generaciones o bien se detiene el algoritmo cuando se observa que los individuos ya no evolucionan más.

3.1.1. Operadores Genéticos

Los operadores genéticos tienen dos finalidades, a veces contrapuestas: por un lado generar variabilidad en la población para ampliar el campo de búsqueda del algoritmo, y por otro tratar de conseguir individuos cada vez mejores que profundicen en los caminos de búsqueda que sean más satisfactorios.

Básicamente hay tres tipos de operadores genéticos, que describiremos en los siguientes apartados: los operadores de *cruce*, que se aplican a dos o mas individuos, los operadores de *mutación*, que se aplican a un solo individuo y los operadores de selección que, aplicados a una población, escogen a un subconjunto de individuos.

3.1.1.1. Cruce

El cruce tiene como finalidad obtener nuevos individuos a partir de las características de otros individuos presentes en la población. Independientemente de la técnica que se utilice tanto para la codificación como para el cruce el objetivo último de este operador es tratar de que las características adecuadas para resolver el problema se hereden, y a ser posible se combinen de manera que se potencien entre sí. Con este operador se generan uno varios nuevos individuos *hijos* a partir de las características de varios individuos preexistentes, *padres* o *progenitores*.

3.1.1.2. Mutación

Los operadores de mutación realizan una alteración aleatoria de la información contenida en el genotipo de un individuo, para obtener otro individuo denominado *mutante*. El operador de mutación hace que el algoritmo tenga mecanismos que permitan el desbloqueo del algoritmo si este ha alcanzado un mínimo local en el espacio de búsqueda, en este caso una mutación puede incorporar nuevos genotipos de otras zonas de dicho espacio. Además la implementación de operadores de mutación permite evitar que surjan poblaciones degeneradas, en las que todos o casi todos los individuos tienen el mismo genotipo.

Además de evitar que se bloquee el proceso evolutivo, el operador de mutación también permite incrementar el número de saltos evolutivos, es decir, la mutación permite explorar nuevos subespacios de soluciones, por lo que,

si el subespacio es bueno en términos de adaptación, se producirá un salto evolutivo después de la mutación que se expandirá por la población. Sin embargo, si la tasa de mutación es excesivamente alta aparecerá la llamada *deriva genética* (es decir, la población no converge hacia una solución, sin llegar nunca a un punto estable).

3.1.1.3. Selección

La selección se produce en dos instantes dentro de un AG, en primer lugar se han de seleccionar los individuos a los que aplicar los operadores genéticos (selección de padres) y en segundo lugar se han de seleccionar los individuos que formarán la siguiente generación, después de que cada individuo ha sido evaluado y se le ha asignado un *fitness*. Para llevar a cabo la selección se utilizan una serie de técnicas entre las que podemos destacar las siguientes:

- *Selección directa*: se seleccionan los individuos de acuerdo a un criterio objetivo, como puede ser los x mejores o eliminar los y peores.
- *Selección aleatoria*: en el criterio de selección influye de algún modo el azar. En ella podemos destacar dos técnicas:
  - *Selección por torneos*: Un torneo es una *competición* en la que participan un número predeterminado de individuos (necesariamente menor que el tamaño de la población) elegidos aleatoriamente, y el torneo es *ganado* por el individuo que posea una mayor aptitud. Se realizará un torneo por cada individuo que sea necesario seleccionar.
  - *Selección por ruleta*: Selecciona individuos utilizando una metáfora de ruleta en la que se encuentran todos los individuos, ocupando un sector circular con área proporcional a su aptitud. La ruleta se *lanzará* tantas veces como individuos haya que seleccionar.

3.2. Programación genética

El paradigma de la *Programación Genética* (PG) surge como una especialización de los AGs y plantea utilizar la metodología de la computación genética para encontrar en lugar de la mejor solución a un problema determinado, el mejor procedimiento para obtenerla. Dicho de otro modo, los individuos en la PG no codifican soluciones candidatas de un problema sino que codifican información procedimental sobre como resolver un problema determinado. Fue originalmente propuesta por varios autores [218] aunque destacan los trabajos de Koza [14] quien popularizó su uso en 1992.

La PG sigue el mismo enfoque que el resto de paradigmas evolutivos, sin embargo varía en que para la representación de los individuos utiliza un lenguaje complejo que permite definir un procedimiento candidato a solventar un problema a partir de un conjunto predeterminado de funciones, o acciones básicas, y de terminales, o datos de entrada [14].

La codificación de los individuos puede ser diversa [219], aunque predomina el uso de estructuras de datos de tipo árbol. En las representaciones de tipo árbol, los nodos no terminales del árbol almacenan funciones que representan operaciones a realizar, las cuales reciben una serie de parámetros y devuelven un valor, y por otro lado, los nodos terminales representan al conjunto de datos de entrada o valores constantes, sobre los que se aplicarán las operaciones. La figura 3.2 muestra una representación típica de un individuo en PG en forma de árbol en la que el genotipo del individuo es el árbol de expresión y el fenotipo una expresión matemática.

Idealmente, para poder solucionar un problema mediante un método basado en PG, se deben cumplir dos propiedades, *suficiencia* y *cierre* [220].

La propiedad de suficiencia nos indica que el conjunto de funciones y de terminales ha de ser lo suficientemente variado como para que se pueda representar la solución al problema. Por otro lado, la propiedad de cierre

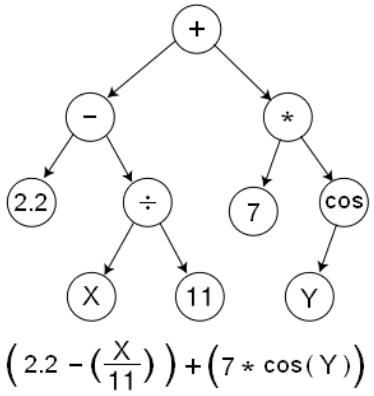


Figura 3.2: Ejemplo de representación de un individuo en PG

consiste en que todas las funciones que puedan aparecer en el árbol deben poder recibir cualquiera de los posibles parámetros que aparezcan en el.

La solución clásica para garantizar el cierre es forzar que todas las funciones tengan esta propiedad, es decir, que el tipo de datos que recibe y devuelve todo el conjunto de funciones esté definido en el mismo dominio. Sin embargo esta restricción es difícil de mantener sobre todo en situaciones en las que se deba utilizar funciones que empleen distintos tipos de argumentos para poder mantener la propiedad de suficiencia. Hay que tener en cuenta que si esta propiedad no se cumple, se generarán durante el proceso evolutivo individuos no válidos con la consiguiente pérdida de eficiencia de este.

Para resolver el problema anterior se han desarrollado variantes de la PG que evitan que se generen individuos no válidos durante el proceso evolutivo, entre las que podemos destacar la PG *fuertemente tipada*, donde se especifican un conjunto de reglas sintácticas para cada nodo que indican qué tipo de nodos pueden descender de el [221] o la *PG basada en gramáticas*, en la que una gramática formal especifica una estructura que los individuos deben respetar [222; 223].

3.3. Programación Genética aplicada a clasificación

Las características de la PG hacen que sea especialmente adecuada para tareas de aprendizaje supervisado [224; 225; 226; 227] y por tanto ofrece numerosas ventajas para abordar tareas de clasificación, entre ellas la flexibilidad de poder adaptarse a un gran número de problemas y de representaciones del conocimiento utilizadas en clasificación, como reglas, funciones discriminantes, y distintos tipos de árboles de decisión [220].

La idea principal en la que se basa la utilización de programación genética para la obtención de clasificadores es que cada individuo represente un clasificador o parte de un clasificador, y utilizar como función de *fitness* una medida de su rendimiento al clasificar los patrones de entrenamiento, para tratar de que el propio proceso evolutivo obtenga finalmente un clasificador adecuado.

Además la PG tiene como principal característica que la representación de los individuos no solamente se puede usar para almacenar conocimiento factual, sino que también permite almacenar conocimiento procedimental, en forma de algoritmos, lo cual es muy útil si se desea resolver un problema de clasificación.

En las siguientes subsecciones se describen los principales formalismos de representación del conocimiento que se han utilizado al aplicar PG a tareas de clasificación, así como las principales contribuciones desarrolladas sobre estos.

3.3.1. Árboles de decisión

Los árboles de decisión son una de las técnicas más utilizadas para representar clasificadores [228], además teniendo en cuenta que la representación en

forma de árbol es uno de los esquemas de codificación más utilizados para los individuos en PG, su utilización para hacer evolucionar clasificadores parece bastante obvia.

Por ejemplo destacan los trabajos [229; 230; 231] donde de una manera bastante sencilla se hacen evolucionar árboles de decisión utilizado como *fitness* varias funciones que permiten optimizar el clasificador aprendido para adaptarlo a diferentes tipos de problemas.

Un problema de este enfoque es que en ocasiones los árboles aprendidos son bastante grandes. Este fenómeno suele aparecer con conjuntos de entrenamiento de cierto tamaño. Para evitar esto en ocasiones se utilizan subconjuntos del conjunto de entrenamiento para inducir árboles más pequeños, que posteriormente se combinan [232].

3.3.2. Reglas de clasificación

Las reglas de clasificación son otro de los formalismos más utilizados para la representación del conocimiento en problemas de clasificación. Constan de un antecedente, formado por una condición o conjunto de condiciones, y un consecuente, que suele ser una clase de las presentes en el problema. Si un patrón cumple con las condiciones del antecedente se le considerará asociado con la clase del consecuente.

A la hora de hacer evolucionar mediante PG clasificadores basados en reglas se puede optar por dos tipos de representaciones de los individuos. O bien cada individuo codifica una regla [233; 234; 235] o cada individuo codifica un conjunto de reglas [236; 237]. Cuando cada individuo codifica una regla el clasificador obtenido estará formado por un conjunto de individuos de la población final. Si cada individuo codifica un conjunto de reglas será un sólo individuo (presumiblemente el de mejor *fitness*) el clasificador inducido.

También se han planteado modelos híbridos en los que un individuo puede codificar una regla o varias [238].

Otro aspecto a tener en cuenta a la hora de utilizar programación genética en clasificación es si los problemas con los que se va a tratar son problemas de clasificación binaria o multiclase. En el primer caso basta con que los individuos almacenen el antecedente de la regla, pero si se trata de resolver problemas con varias clases, los algoritmos tienden a ser mucho más complejos. En cualquier caso como un problema multiclase puede dividirse en tantos problemas binarios como clases tenga, en muchas ocasiones se utilizan enfoques binarios para resolver problemas multiclase [220].

Las reglas de clasificación se han utilizado para resolver problemas de clasificación binarios en los trabajos [239; 240]. El clasificador final, en este caso, contiene varias reglas, indicando estas que el patrón pertenece a la case, no perteneciendo en caso contrario.

Para clasificación multiclase, además del enfoque de dividir el problema en varios problemas binarios [241; 242], se han utilizado enfoques en los que un individuo almacena reglas para cada clase [238], o bien enfoques coevolutivos en los que los consecuentes evolucionan a la misma vez que el antecedente [243; 244].

3.3.3. Funciones discriminantes

Las funciones discriminantes son un tercer formalismo que se utiliza para representar el conocimiento de un clasificador. Una función discriminante es una expresión matemática en la que los valores de entrada son los diferentes atributos del patrón a clasificar. La salida de una función disciminante es un valor numérico, y en función de uno o varios umbrales se determinará la clase a la que pertenece el patrón.

En el caso de problemas de clasificación binaria, una sola función con un umbral es suficiente para representar un clasificador, valores por encima del umbral indican que el patrón pertenece a la clase y valores por debajo del umbral nos indican que no pertenece. Normalmente el umbral se establece en cero, de tal manera que valores por encima de cero nos indican que la clase es positiva y por debajo de cero que es negativa. Este enfoque se ha utilizado en los trabajos [245; 246; 247; 248; 249; 250].

En el caso de problemas multiclase se puede optar por dos opciones. Como ya se ha comentado anteriormente, un problema multiclase de n clases puede ser dividido en $n-1$ problemas de una sola clase. Así, si se opta por este enfoque, serán necesarias $n-1$ funciones binarias, cada una de ellas con su propio umbral. De esta manera se han hecho evolucionar clasificadores en [251; 252; 253; 254].

Una segunda opción es utilizar solamente una función discriminante y definir $n-1$ umbrales. De esta forma se determinarán n intervalos, cada uno de los cuales será asociado a una clase, asociándose el patrón a la clase si la salida está en el intervalo correspondiente. Desde este punto de vista se ha enfocado el uso de funciones discriminantes en [255; 256; 257; 258].

Por otra parte también se han planteado algunos modelos híbridos en los que se utilizan simultáneamente reglas de clasificación y funciones discriminantes, como el que se presenta en [259], en el que se usa programación genética en dos fases distintas, primero para hacer evolucionar reglas de clasificación y posteriormente para incorporar funciones discriminantes a las reglas ya generadas.

3.3.4. Otras representaciones

La programación genética, debido a su potencia expresiva, se ha utilizado combinada con muchos otros modelos de representación del conocimiento para clasificación, entre los que podemos destacar redes neuronales, máquinas de vectores soporte, consultas SQL o k vecinos mas cercanos. A continuación se citan los principales trabajos.

La programación genética se ha utilizado para hacer evolucionar la estructura de redes neuronales en [236; 260; 261]. Normalmente cada individuo almacena un árbol que representa la arquitectura una red neuronal o una parte de ella, siendo la función de *fitness* la exactitud de la red neuronal al clasificar el conjunto de entrenamiento.

Por otro lado las máquinas de vectores soporte son una estructura muy utilizada en clasificación [262; 263]. Esta técnica trata de hallar un hiperplano que separe linealmente los datos de entrenamiento. En caso de que estos no sean linealmente separables se busca elevar la dimensionalidad del espacio de los datos del conjunto de entrenamiento, a un nuevo espacio en el que dicho conjunto si que sea linealmente separable. En la construcción de estos hiperplanos es fundamental utilizar una función núcleo (*kernel function*). Para hallar estas funciones núcleo se ha utilizado programación genética en varios trabajos [264; 265; 266].

La programación genética también se ha utilizado para hacer evolucionar consultas SQL. El resultado de ejecutar esta consulta sobre una base de datos relacional, en la que están los patrones almacenados en forma de tuplas, es una lista de patrones que pertenecen a una clase. Este es el caso del trabajo [267] en el que cada individuo almacena un árbol que codifica una consulta sobre un a base de datos relacional. En el trabajo [268] se utiliza el enfoque de construir un clasificador en base a $n-1$ consultas para solventar problemas multiclase con n clases.

El algoritmo de los k vecinos mas cercanos es un algoritmo de clasificación en el que el conjunto de entrenamiento se utiliza como referencia para clasificar nuevas instancias calculándose la cercanía de las nuevas instancias a las ya conocidas. La clase se adjudica en base a la clase dominante entre las k instancias más cercanas. En ciertas ocasiones conviene no dar el mismo peso a cada instancia del conjunto de entrenamiento, y es aquí donde se ha utilizado la programación genética, para calcular dichos pesos [269], codificando cada individuo una expresión matemática que calcula el peso de cada instancia.

3.4. Gene Expression Programming

Uno de los problemas de la programación genética es la dificultad de aunar sencillez y potencia expresiva en la representación de los individuos. Así se observa que si la forma de representación es sencilla de manipular, pierde complejidad funcional, no siendo apta para resolver determinados problemas, y si la representación permite una gran complejidad funcional, es difícil hacerla evolucionar hacia soluciones viables [15].

Para solventar este dilema se han propuesto varias alternativas, entre las que destaca *Gene Expression Programming (GEP)* desarrollada por Cândida Ferreira [15] que propone una representación que trata de combinar sencillez con potencia expresiva.

En GEP cada individuo tiene una codificación dual. El genotipo está compuesto por un conjunto de genes que se representan mediante una cadena simple de elementos, y el fenotipo correspondiente a cada cadena es un árbol de expresión, formado por funciones en sus nodos no terminales y por valores de entrada o constantes en sus nodos terminales. Además en GEP se propone un modo de transformar las cadenas que representan a los genes en árboles

de tal manera que cualquier cadena válida genere un árbol sintácticamente correcto.

En las siguientes secciones se describe la representación de los individuos, tanto de su genotipo como de su fenotipo, así como el proceso de transformación del genotipo al fenotipo. Posteriormente se muestran los operadores genéticos que se definen en GEP.

3.4.1. Estructura de los individuos

En *Gene Expression Programming*, tanto el genotipo como el fenotipo de los individuos está compuesto de los mismos elementos, siendo diferente la estructura en la que se organizan. Los elementos que aparecen en un individuo son los mismos que aparecen en cualquier otro paradigma de PG, esto es, elementos no terminales y elementos terminales:

- Conjunto de no-terminales: son funciones que solamente pueden aparecer en los nodos no terminales del árbol de expresión. La aridad de las funciones de cada nodo representará el número de hijos que tendrá el nodo. Como se verá a continuación es un factor a tener en cuenta durante el proceso de construcción de árboles válidos.
- Conjunto de terminales: es el conjunto de elementos que solamente pueden aparecer en las hojas del árbol. Dentro de este conjunto encontramos tanto valores constantes como parámetros del entrada que reciba el árbol.

Como se comentaba anteriormente, el genotipo es una cadena lineal y está formado por varios genes. Cada uno de estos genes se divide en dos partes: la *cabeza* y la *cola*. La cabeza del gen tendrá un tamaño elegido a *a priori* para cada problema, y puede contener elementos terminales y no terminales, pero

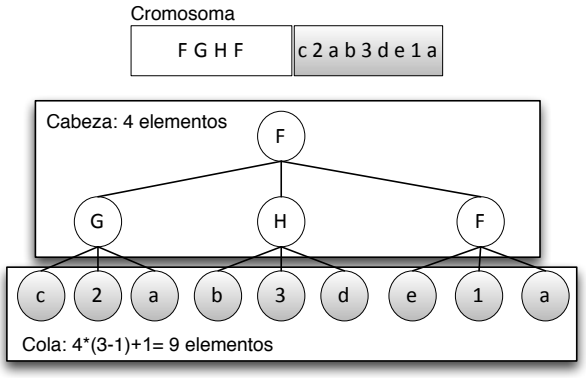


Figura 3.3: Ejemplo de genotipo y fenotipo en GEP

el tamaño de la cola, que solamente puede contener elementos no terminales, estará determinado por la siguiente expresión:

$$t = h(n-1) + 1 \tag{3.1}$$

siendo t el tamaño de la cola, h el tamaño de la cabeza, y n[1] la aridad máxima presente en los nodos no terminales. De esta forma se garantiza que la cola contendrá elementos terminales suficientes como para rellenar el último nivel del árbol de expresión en el peor caso posible. Gráficamente se puede observar en la figura 3.3, donde se representa un individuo GEP genérico, siendo las mayúsculas funciones, que consideraremos con aridad 3, los valores en minúscula los parámetros y los números que aparecen valores constantes. Como se puede observar, la cola ha de tener un elemento más que el tamaño de la cabeza multiplicado por la aridad máxima menos uno. El objetivo de la limitación de elementos junto con el tamaño de la cola es permitir que cualquier gen pueda ser transformado en un árbol válido, como se aprecia en la figura.

[1]No confundir con $n = |L|$, o sea número de etiquetas de un problema de MLC. Se ha optado por utilizar también n en esta expresión para respetar la notación propuesta por Ferreira.

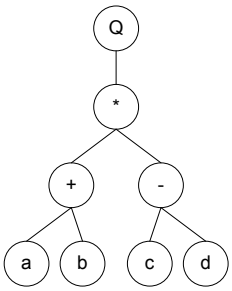


Figura 3.4: Árbol que representa una expresión sencilla

El formato en el que se almacena en la cadena-genotipo el árbol de expresión se denomina *K-Expression*[2]. El modo de construir un árbol válido a partir de la *K-Expression* que lo representa consiste en ir rellenándolo por niveles. El primer elemento, será la raíz del árbol. Los siguientes elementos del gen (tantos como indique la aridad del primero) corresponderán a los hijos del nodo raíz, posteriormente se genera el siguiente nivel, asignando los hijos que necesite cada nodo según su aridad hasta que todas las hojas de árbol tengan elementos terminales.

Al construir el árbol según este procedimiento se garantiza que todos los árboles generados serán válidos, ya que en el peor caso posible habrá elementos terminales en la cola suficientes para completar el árbol.

Por ejemplo consideremos la *K-Expression* `Q*+-abcd`. El nodo raíz etiquetado con `Q` representa a la función raíz cuadrada. El árbol que representa dicha *K-Expression* es el presentado en la figura 3.4, y la expresión matemática la de la ecuación 3.2.

$$\sqrt{(a+b)\times(c-d)} \tag{3.2}$$

Hay que tener en cuenta que las *K-expressions* que representan a un árbol pueden ser de tamaño más pequeño que el del gen que las almacena, de hecho

[2]La *K-Expression* que representa un árbol determinado no se corresponde con ningún recorrido del árbol, ni *prefijo* ni *infijo* ni *postfijo*.

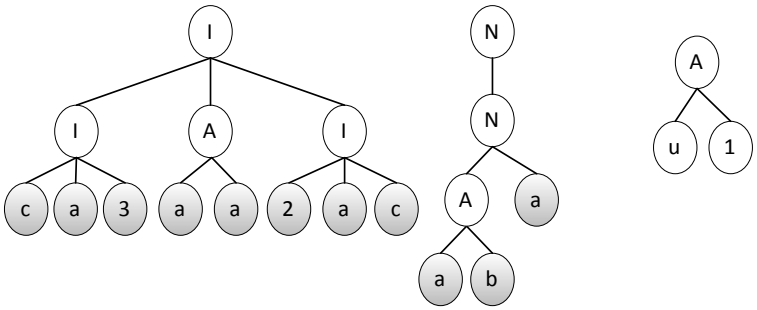
puede haber genes distintos que representen la misma expresión matemática. Por ejemplo, si consideramos $h = 5$ y $n = 2$, cadenas como `Q*+-abcd11b` o `Q*+-abcd12a` pueden ser convertidas en el mismo árbol de expresión que la *K-Expression* del ejemplo anterior. Todos los genes cuya *K-Expression* siga el patrón `Q*+-abcd`$\_\_\_$ almacenan la misma expresión, pudiendo tener en los subrayados (_) cualquier elemento del conjunto de terminales.

Como se deduce del párrafo anterior, el hecho de que un elemento esté en el gen, no implica que aparezca en el árbol. Estos elementos, según Ferreira, sirven para aumentar la diversidad genética de la población sin que esto implique pérdidas de buenos individuos. Además, estos elementos permiten aumentar la tasa de mutación sin que se produzcan fenómenos de deriva genética [270].

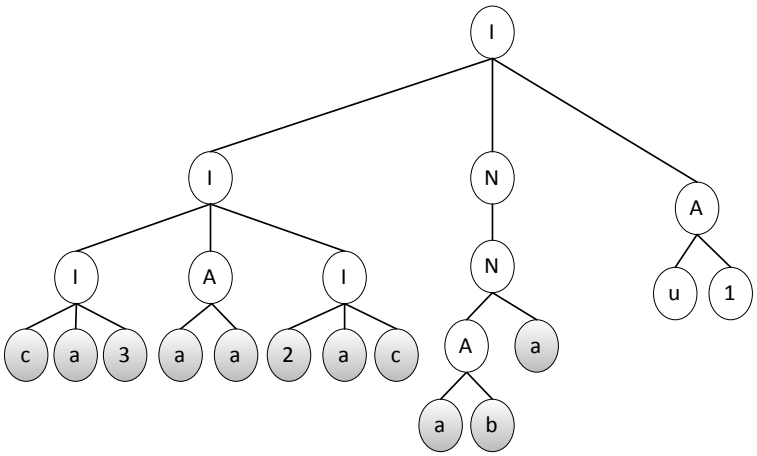
También hacer notar que, en caso de que el individuo esté representado por varios genes, su fenotipo consistirá en varios árboles. Para determinados problemas esto puede ser adecuado, pero si se desea que cada individuo genere un valor numérico concreto, todos estos árboles deben de conectarse mediante la denominada *función de conexión*, que recibirá tantos parámetros como genes existan. La función de conexión puede ser un parámetro predeterminado del algoritmo, o bien puede incluirse al principio o al final de los individuos para que evolucione junto con éstos.

El proceso de transformación en este caso se realiza en dos pasos: en primer lugar se transforma cada gen en un árbol, y en segundo lugar se unen todos los subárboles mediente la función de conexión. Por ejemplo, la figura 3.5, representa a un individuo con tres genes, $h = 4$ (por tanto $t = 4*(3-1)+1) = 9$ según ecuación 3.1) y aridad máxima = 3 (la aridad de las funciones es $aridad(I) = 3$, $aridad(A) = 2$ y $aridad(N) = 1$). En primer lugar cada gen del individuo se transforma en un árbol. Posteriormente los árboles se conectan con una función de conexión I para constituir un único árbol. Todo este proceso aparece recogido en la figura 3.5

(a) Genotipo



(b) Árboles de expresión

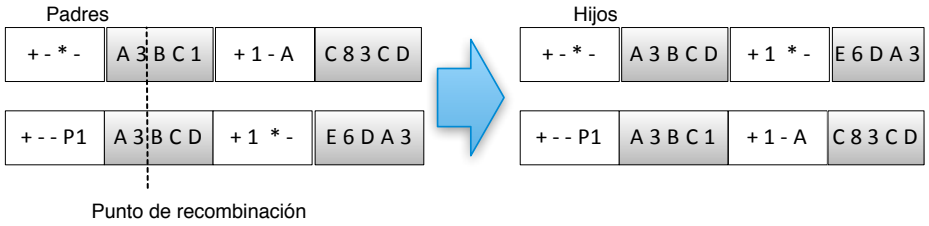


(c) Árboles conectados mediante función de conexión

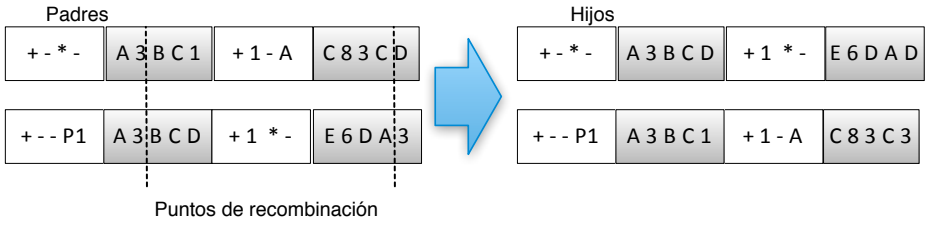
Figura 3.5: Transformación de un individuo con varios genes. Tamaño del gen 13 y número de genes 3

3.4.2. Operadores genéticos

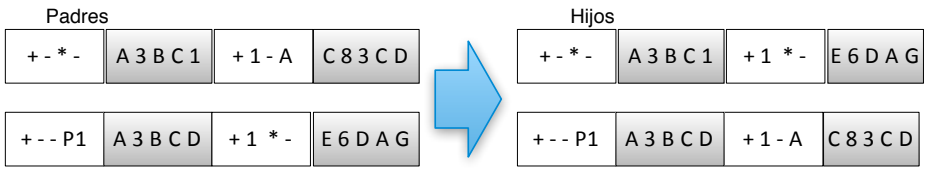
El paradigma de la GEP incluye la definición de una serie de operadores genéticos adaptados a la representación dual de los individuos, y formulados de manera que respeten la estructura de los genes. A continuación se comentan los tres tipos de operadores definidos: cruce, transposición y mutación.



(a) Operador de recombinación en un punto



(b) Operador de recombinación en dos puntos



(c) Operador de recombinación de genes

Figura 3.6: Operadores de cruce en GEP

3.4.2.1. *Operadores de cruce*

En GEP hay tres tipos de operadores de cruce: recombinación en un punto, recombinación en dos puntos y recombinación de genes. En todos los casos se escogen aleatoriamente dos padres, y entre ellos intercambian parte de su genotipo para generar dos nuevos hijos, con genes válidos:

- *Recombinación en un punto*: en el cruce en un punto se escoge aleatoriamante una posición del genotipo de los padres. Los hijos resultantes tendrán como genotipo, hasta el punto escogido, el genotipo de un progenitor, y a partir de éste, el genotipo del otro progenitor (Figura 3.6a).

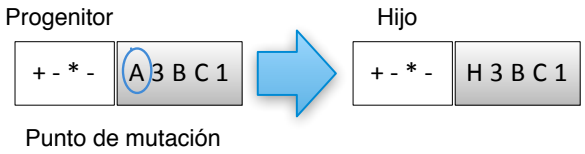
- *Recombinación en dos puntos*: en esta variante del cruce se escogen dos puntos del genotipo. El genotipo de cada uno de los hijos será alternativamente una copia del cromosoma de un progenitor, hasta el primer punto, una copia de otro progenitor hasta el segundo punto y por último de nuevo copia del primer progenitor (Figura 3.6b).

- *Recombinación de genes*: se escoge aleatoriamente un gen del genotipo, y los padres intercambian éste para generar los hijos (Figura 3.6c). Solamente se puede aplicar a individuos con más de un gen.

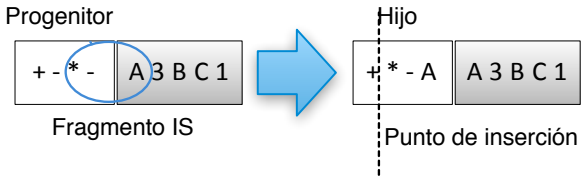
3.4.2.2. Operadores de mutación y transposición

El operador de mutación consiste en un cambio aleatorio de un elemento en una posición aleatoria del genotipo, sin embargo, hay que tener en cuenta a la hora de su implementación, que la estructura interna del genotipo ha de mantenerse intacta, esto es, un elemento que pertenezca a la cola del gen solamente cambiará a elementos que pertenezcan al conjunto de terminales, y un elemento que pertenezca a la cabeza del gen podrá alterarse por cualquier elemento del conjunto de terminales y funciones. En la figura 3.7a se muestra un ejemplo. La mutación de un terminal en una función, o de una función en otra con aridad superior pueden representar un cambio drástico en el fenotipo del individuo, aunque solamente afecte a un elemento. También pueden ocurrir mutaciones *neutras* que no alteren el fenotipo del individuo.

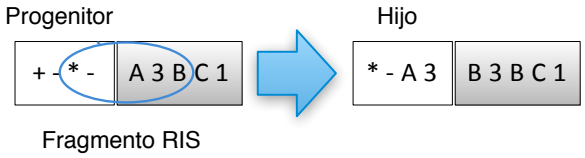
En la terminología de la GEP aparece un nuevo tipo de operadores, los operadores de *transposición*, que se corresponden con operadores de mutación modificados para actuar sobre los genes. Los operadores de transposición son un tipo de operadores que seleccionan un fragmento del cromosoma del individuo y lo transfieren a otra posición de éste, por tanto podrían catalogarse como operadores de cruce interno. En GEP se definen tres operadores de transposición:



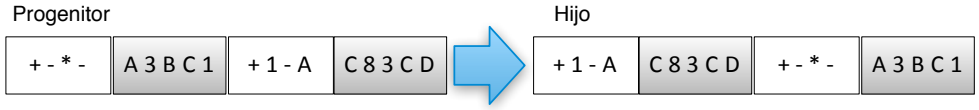
(a) Operador de mutación



(b) Operador de transposición IS



(c) Operador de transposición RIS



(d) Operador de transposición de genes

Figura 3.7: Operadores de mutación y transposición en GEP

- La *transposición de secuencia de inserción*, o transposición *IS*, consiste en seleccionar aleatoriamente un fragmento del cromosoma del individuo, de tamaño también aleatorio, que es insertado en cualquier parte de la cabeza de un gen, salvo en el primer elemento de la cabeza (la raíz del árbol). El fragmento original se copia (se mantiene también en su posición original) y el tamaño del gen no varía (Figura 3.7b).

- La *transposición de secuencia de inserción en la raíz*, o transposición RIS, consiste en seleccionar aleatoriamente un fragmento del cromosoma del individuo, de tamaño también aleatorio, pero con la salvedad de que debe comenzar por una función, el cual es insertado en el primer elemento de la cabeza de un gen, o sea, la raíz del árbol (Figura 3.7c).

- La *transposición de genes* es aplicada a individuos con más de un gen y consiste en la modificación de la posición que ocupa un gen en el cromosoma. En esta transposición se escogen aleatoriamente dos genes del individuo, que intercambian sus posiciones en el cromosoma (Figura 3.7d).

3.5. GEP aplicada a clasificación

Desde su propuesta original GEP ha sido muy utilizada para resolver problemas de clasificación. La propia Ferreira en el trabajo en el que planteaba la GEP [15], apuntaba la utilización de esta técnica en tareas de clasificación. Posteriormente, en [270] se sientan las bases para utilizar GEP en problemas de clasificación, utilizando para ello conjuntos de funciones lógicas y esta misma autora resuelve en [271] un problema de clasificación relacionado con el comportamiento de autómatas celulares.

Básicamente se han utilizado dos enfoques distintos a la hora de abordar problemas de clasificación utilizando GEP, o bien mediante el aprendizaje de reglas o mediante el empleo de funciones discriminantes.

como hemos comentado anteriormente, una función discriminante es una función matemática que separa las clases entre sí haciendo uso de un umbral. De esta manera la función recibe como entrada los atributos de un patrón y devuelve una salida numérica. Esta salida se interpreta como la clase a la que pertenece el patrón haciendo uso de uno o varios umbrales. Las funciones discriminantes son bastante robustas y eficientes pero presentan el problema de que es difícil interpretar el conocimiento que generan. En cuanto a su uso en clasificación mediante GEP merece la pena destacar en esta revisión varios estudios:

Uno de los primeros fue el trabajo de Zhou et al. [272] donde se abordaron varios problemas de clasificación binaria del repositorio UCI con GEP

y conjuntos de funciones aritméticas interpretadas como funciones discriminantes. Posteriormente estos mismos autores en [273] trataron de buscar un compromiso entre la exactitud del clasificador y la simplicidad de la función, medida ésta en función del número de atributos que recibe la función.

Las funciones discriminantes se han utilizado también para tratar con problemas multiclase, en [274], utilizando el enfoque de *uno-contra-todos*. Posteriormente se han aplicado usando solamente una función discriminante y varios umbrales en [275], donde se trata de buscar los centroides de cada clase utilizando para ello los autovalores de cada clase. El trabajo [276] codifica en un solo individuo varias funciones discriminantes, cada una de ellas asociada a una clase.

También conviene reseñar otras propuestas como como el trabajo [277] en el que se comparan las técnicas de *bagging* y de *boosting* a la hora de construir un multiclasificador basado en GEP o el artículo [278] en el que se utilizan funciones discriminantes evolucionadas mediante GEP para minería de asociaciones.

Un problema que han tratado de abordar algunos trabajos es la presencia de un gran número de atributos en el espacio de entrada, como en el trabajo [279] en el que los patrones de entrada proceden de *DNA microarrays*, que se utilizan para la clasificación de moléculas orgánicas.

Merece la pena citar también el trabajo [280], en el que se propone una modificación del paradigma de la GEP, haciendo un híbrido con un algoritmo de selección clonal. Esta técnica también busca emular un proceso biológico, en este caso el modo en el que el sistema inmunitario clasifica y recuerda las amenazas ante las que se ha enfrentado.

En cuanto al uso de reglas de clasificación, los primeros trabajos que tratan de aprender reglas con GEP utilizaron arboles formados solamente por funciones lógicas. Destaca el trabajo ya mencionado [271] en el que los patrones son un autómata celular, y se busca clasificar este según su estado final. Otra

perspectiva más elaborada es plantear la utilización de funciones lógicas difusas. De esta manera, en [281] se abordan varios problemas de clasificación. Este mismo enfoque se usa en [282] donde cada individuo representa una regla difusa de clasificación, y además se proponen algunos operadores de mutación sobre los atributos difusos.

El principal problema del uso de funciones lógicas es que solamente se pueden abordar problemas en que los conjuntos de terminales sean binarios. Para poder trabajar con datos numéricos se han buscado algunas soluciones, como en el trabajo [283] donde cada terminal es un triplete formado por un atributo de entrada, una función relacional y un valor constante. Estos mismos autores proponen en [284] integrar la GEP con un algoritmo celular evolutivo y utilizar ambos combinados como clasificadores débiles en un *ensemble* mediante la técnica del *boosting*.

Finalmente otra forma muy efectiva de abordar la utilización de reglas clasificación mediante GEP y funciones lógicas ha sido el paradigma GEPCLASS, propuesto por Weinert y Lopes [16], en el que se modifica la GEP para que se pueda establecer una jerarquía entre las funciones relacionales y lógicas sin que esto implique restricciones a la hora de plantear el proceso evolutivo. La GEPCLASS ha sido empleada en varios trabajos para realizar clasificación con conjuntos tanto nominales como numéricos, entre los que podemos citar [275; 285]. Dado el interés de este paradigma se describe en más detalle en la siguiente sección.

3.6. GEPCLASS

Un enfoque para hacer evolucionar reglas de clasificación mediante GEP es modificar el paradigma para que pueda utilizar reglas de clasificación, abstracciones muy utilizadas en problemas de clasificación, que permiten aunar en una estructura robustez con interpretabilidad.

Formalmente, una regla de clasificación es una estructura del tipo *Si A Entonces C* siendo A el antecedente, esto es, una serie de condiciones que debe cumplir un patrón para estar asociado al consecuente C. El antecedente es un conjunto de condiciones del tipo (T_i **op** V_j) donde T_i es el i-ésimo atributo del patrón, V_j un valor constante, **op** un operador relacional del tipo $=, >, <$ o $\neq$. En el caso en que el dominio de los datos del problema sea nominal, los únicos operadores relacionales válidos son $=$ y $\neq$. Todo el conjunto de condiciones está relacionado con operadores lógicos (como Y, O, NO) para constituir una única regla de clasificación.

La propuesta de la GEPCLASS [16] surge para permitir que los individuos GEP codifiquen reglas de clasificación en una estructura árbol. El principal problema que presenta GEP es que no permite que haya un orden de funciones dentro de fenotipo, que genere una estructura ordenada en el árbol resultante. Debido a que las reglas de clasificación necesitan distinguir entre funciones relacionales y funciones lógicas, es esencial que las funciones puedan quedar ordenadas dentro del árbol, estando en el penúltimo nivel las funciones relacionales y en los superiores funciones lógicas.

En GEPCLASS, del mismo modo que en la GEP clásica, cada individuo tiene tanto genotipo y fenotipo, formados por los mismos elementos, funciones y terminales. En el caso que nos ocupa, el conjunto de funciones estará formado por dos subconjuntos, los mismos que aparecen en las reglas de clasificación, el subconjunto de las funciones relacionales y el subconjunto de las funciones lógicas.

El conjunto de terminales, en lugar de estar formado por la agregación de los valores constantes y los parámetros de entrada, estará formado por pares *atributo-valor_constante*, con la intención de garantizar que todas las funciones relacionales reciban como parámetro un atributo y una constante. Cada parámetro de entrada estará asociado con un atributo del conjunto de datos, y los valores constantes serán valores fijos en el dominio de estos atributos.

El genotipo, de la misma manera en que la GEP, está formado por una cadena lineal, compuesta de uno o varios genes de la misma longitud, estando estos divididos en cabeza y cola. La cabeza contendrá tanto funciones como elementos terminales pero la cola solamente contendrá terminales.

La cabeza tiene un tamaño fijo, y el tamaño de la cola es calculado de tal manera que garantice que haya elementos suficientes como para generar un árbol válido, y por ello su tamaño se calcula según la expresión siguiente:

$$t = int([h * (n - 1) + 1]/2) \tag{3.3}$$

donde *int* devuelve la parte entera de su argumento, h es el tamaño de la cabeza y n la aridad máxima. Esta expresión surge de dividir entre dos la ecuación 3.2, ya que en GEPCLASS, al estar cada elemento de la cola constituido por un par *atributo-valor_constante*, solo es necesaria la mitad de estos en la cola. Esto se debe a que las funciones relacionales deben recibir como parámetros un par *atributo-valor_constante* (es decir, un solo elemento del conjunto de terminales), lo que hace que cada terminal realmente represente a dos elementos del árbol.

La figura 3.8 ilustra esta situación con un ejemplo sencillo, en el que podemos ver que cada elemento de la cola realmente se corresponde con dos nodos terminales del árbol generado.

De la misma manera que en la GEP, los individuos pueden tener más de un gen. Es este caso se puede optar por o bien unirlos mediante una función de conexión lógica, o bien que cada individuo represente una regla por separado.

El fenotipo es construido a partir del genotipo de la misma manera que en la GEP, rellenando el árbol capa a capa. El primer elemento será la raíz del árbol, del que colgarán tantos elementos como indique su aridad. Una vez relleno este segundo nivel se procede iterativamente de la misma manera con

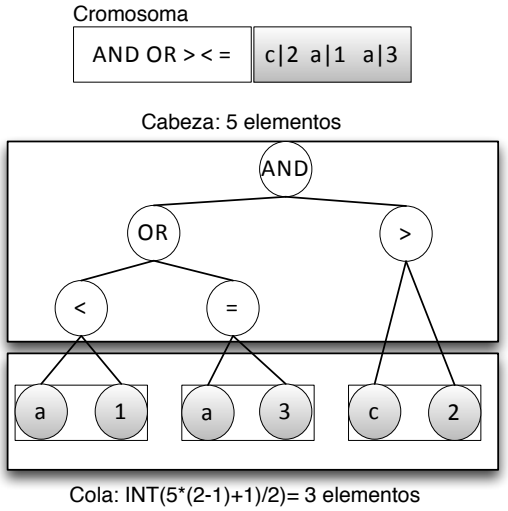


Figura 3.8: Fenotipo y genotipo en un individuo GEPCLASS

el resto, rellenando cada nodo con los hijos que sea necesario y hasta que todas las hojas de árbol tengan elementos terminales.

El hecho de que los individuos representen clasificadores basados en reglas impone una serie de restricciones a los árboles generados: en primer lugar la función de la raíz del árbol debe ser una función lógica, en segundo lugar los hijos de una función lógica han de ser funciones también (sean éstas lógicas o relacionales) y la tercera restricción versa sobre las funciones relacionales, que han de tener forzosamente dos hijos, un atributo y un valor constante.

Para implementar las restricciones en la precedencia entre los operadores lógicos y relacionales, los elementos en la cabeza deben de estar ordenados por precedencia, es decir, estando situados en primer lugar las funciones lógicas y en segundo lugar las relacionales, y el número de funciones lógicas ha de estar en el intervalo $[1, (h/2) - 1]$. De esta manera al menos la segunda mitad de la cabeza estará ocupada por funciones relacionales, lo que permite la existencia de suficientes funciones relacionales para completar el penúltimo nivel del árbol.

Todas estas restricciones anteriormente comentadas han de ser implementadas a nivel del genotipo, y durante la creación de la población inicial, si se

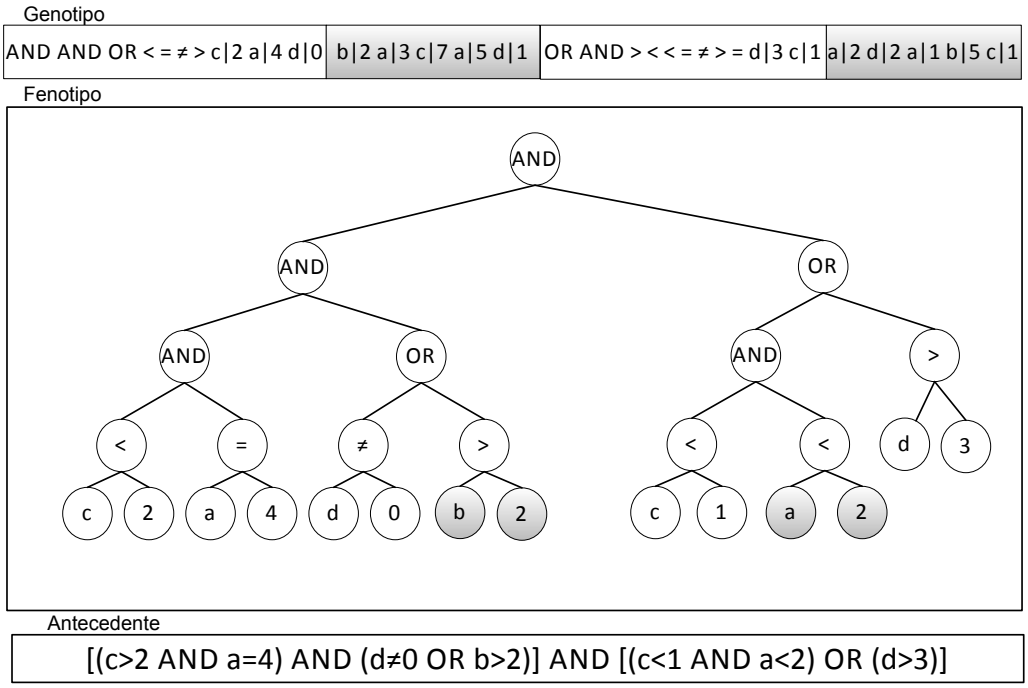


Figura 3.9: Ejemplo de individuo en GEPCLASS.

genera un individuo que no cumpla con las restricciones, debe ser eliminado de la población ya que no puede generar una regla válida.

En la figura 3.9 se muestra un individuo con dos genes, que cumple con las restricciones comentadas anteriormente, el árbol que se genera como fenotipo (considerando AND la función de conexión), y la regla que éste individuo representa. La construcción del fenotipo a partir del fenotipo es similar a como se realiza en GEP, aunque presenta algunas características específicas en GEPCLASS. Para ello el primer elemento del gen pasa a ser la raíz del árbol, y de este se van colgando tantos elementos como indique su aridad. Se va procediendo por niveles hasta que todas las funciones lógicas están colocadas. Tras las lógicas se colocan las funciones relacionales, colgando de cada función lógica tantas relacionales como aridad tenga. Una vez que de todas las funciones lógicas cuelgan funciones relacionales, se procede con

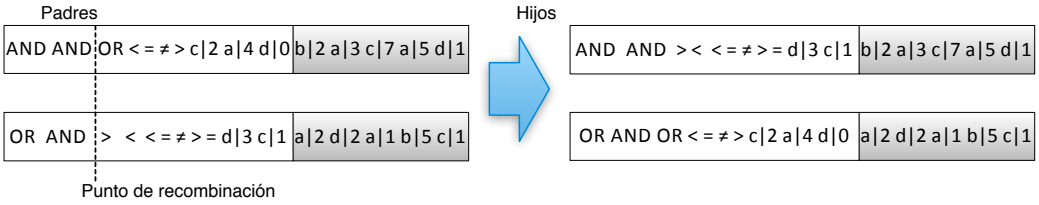


Figura 3.10: Ejemplo del operador de recombinación en GEPCLASS

los terminales[3], colgando de cada función relacional un sólo terminal, que posteriormente se expandirá para representar a dos elementos del árbol.

El consecuente de la regla no se almacena en los individuos, sino que debe asignarse durante el proceso de evaluación, como se discutirá en capítulos posteriores.

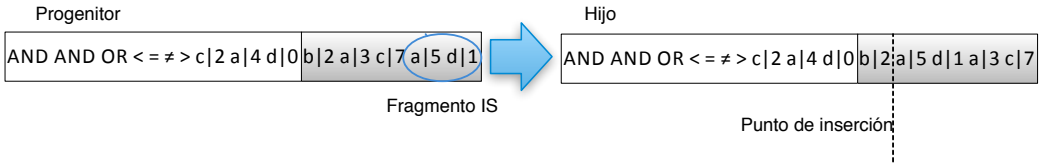
3.6.1. *Operadores genéticos*

Los operadores genéticos de la GEP han sido adaptados en la GEPCLASS a la codificación de individuos planteada en el apartado anterior para asegurar que tras la aplicación de un operador, los individuos que se generen sigan siendo individuos válidos.

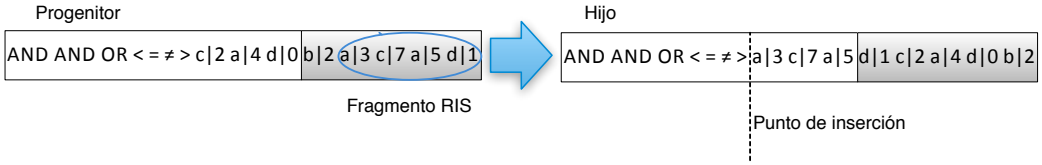
El operador de recombinación seleccionará una posición aleatoria en la cabeza de un gen aleatorio e intercambiará el material genético desde esta posición hasta el final de la cabeza para generar dos nuevos hijos a partir de dos padres, como se muestra en la figura 3.10. No existen operadores de recombinación en dos puntos ni de recombinación de genes ya que estos podrían generar individuos incorrectos.

El operador de mutación sigue trabajando de la misma manera que el de la GEP, realizando un cambio aleatorio en una posición aleatoria del genotipo,

[3]Si quedaran funciones relacionales pendientes de colgar, estas se ignoran.



(a) Operador de transposición IS



(b) Operador de transposición RIS

Figura 3.11: Operadores de transposición en GEPCLASS

pero teniendo en cuenta que si el elemento seleccionado es una función lógica, relacional o elemento terminal, solamente puede ser cambiado por otra función lógica, relacional o terminal respectivamente.

Los operadores de transposición, que como ya se ha dicho, son operadores de mutación específicamente concebidos para trabajar con genes, van a operar solamente con los elementos del conjunto de terminales. La transposición de secuencia de inserción (*IS*) moverá una cadena de material genético de una posición a otra de la cola (Figura 3.11a) y la transposición de secuencia de inserción en la raíz (*RIS*) moverá una cadena aleatoria desde la cola del gen hasta el primer terminal que aparezca en el genotipo del individuo, como aparece en la figura 3.11b.

4

Clasificación multi-etiqueta mediante funciones discriminantes

4.1. Introducción

Las funciones discriminantes son un mecanismo ámpliamente utilizado en minería de datos para representar el conocimiento de un clasificador. Una función discriminante es una función matemática cuya entrada es un subconjunto de los atributos del patrón a clasificar y la salida un valor numérico que se interpretará, haciendo uso de un umbral o de un conjunto de umbrales, como la clase a la que pertenece el patrón.

Formalmente un clasificador formado por funciones discriminantes, está constituido por una serie de estructuras condicionales cuyo antecedente es una función matemática que recibe el vector de características de cada patrón y devuelve un valor real. Este valor se compara con un umbral para determinar la clase. Generalmente se toma como umbral el valor 0 (Ecuación 4.1).

$$Si(f(\mathbf{X}) > 0)\ entonces\ \mathbf{X} \in clase\ sino\ \mathbf{X} \notin clase \tag{4.1}$$

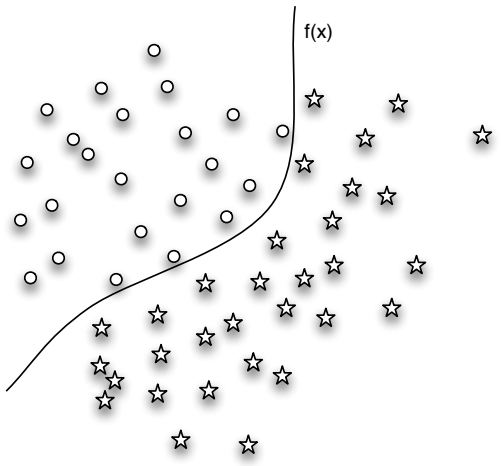


Figura 4.1: Una función discriminante es una función matemática cuya salida permite separar los patrones

Como se ha comentado en la sección 3.3.3 cuando se trabaja con clasificación binaria, una sola función con un umbral es suficiente para representar un clasificador. El umbral suele ser cero, considerando que valores por encima indican que el patrón pertenece a la clase y por debajo no pertenece [247; 246]. En el caso de problemas multiclase se puede optar por dos alternativas. Por un lado se pueden utilizar $n-1$ funciones discriminantes, siendo n el número de clases, cada una con un solo umbral que distinga si un patrón pertenece a cada clase o no [252; 251] o un segundo enfoque es utilizar solamente una función discriminante pero definir $n-1$ umbrales que determinan n intervalos, cada uno de los cuales está asociado a una clase [255; 256].

En trabajos previos se han utilizado funciones discriminantes para aprender clasificadores de una sola etiqueta utilizando PG [273] y GEP [272]. Los clasificadores generados por estos modelos tienen como principal característica su robustez, y además son especialmente adecuados para tratar con problemas cuyos datos de entrada sean numéricos. Debido a que hay numerosos conjuntos de datos multi-etiqueta numéricos (ver sección 2.6), y al buen

rendimiento obtenido por estos modelos de una sola etiqueta, en esta tesis se ha considerado apropiado diseñar y probar un modelo que utiliza GEP para hacer evolucionar funciones discriminantes con el propósito de resolver problemas multi-etiqueta.

En el presente capítulo se describen los aspectos más relevantes del diseño del algoritmo desarrollado, que hemos llamado *GEP-MLC*, el cual aprende clasificadores multi-etiqueta basados en funciones discriminantes, utilizando para ello el paradigma evolutivo de la GEP ya expuesto en el capítulo 3.

El resto del capítulo está estructurado de la siguiente manera: En primer lugar nos centraremos en la representación de los individuos, esto es, cómo se codifican en el genotipo las funciones discriminantes. En la segunda sección del capítulo se pasa a comentar cómo se evalúan estos individuos, y cómo se calcula su aptitud como parte del modelo de clasificación final, para pasar a hacer una descripción global de la dinámica del algoritmo de aprendizaje propuesto. Posteriormente se detalla el marco experimental al que se ha sometido la propuesta que aquí se hace para comparar su rendimiento con otros enfoques de la bibliografía, ya expuesta en el capítulo 2. Tras esta exposición se continúa mostrando resultados experimentales obtenidos, así como una discusión de estos y de su análisis estadístico. El capítulo termina detallando las conclusiones a las que se ha llegado tras el desarrollo y prueba experimental de este modelo.

4.2. Representación de los individuos

El algoritmo GEP-MLC va a tratar de aprender un clasificador formado por un conjunto de funciones discriminantes para cada etiqueta presente en el problema multi-etiqueta que se quiera abordar.

Como ya comentábamos en la introducción, en el caso de problemas con varias clases (siendo el número de clases $n > 1$) una posibilidad es establecer

Conj. de no terminales	$+, -, \times, \div$(protegida)
Conj. terminales	Constantes, Parám. de entrada

Tabla 4.1: Conjuntos de terminales y no terminales en GEP-MLC

$n-1$ umbrales que definan n intervalos y en función del intervalo del valor que devuelva la función discriminante, el patrón se asocia a una clase u otra. Sin embargo este enfoque nos nos permitiría tener clasificadores en los que las clases se solaparan entre sí, como ha de ocurrir en clasificación multi-etiqueta. Este es el motivo por el que los modelos que genera nuestra propuesta están formados por un conjunto de funciones con un solo umbral (en este caso 0) y se asocia la pertenencia o no a una etiqueta del modo que se hace en problemas de clasificación binaria.

En GEP-MLC los individuos tienen una codificación dual y cada individuo va a codificar en su genotipo la expresión matemática correspondiente a una función discriminante, siendo el fenotipo un árbol sintáctico cuya evaluación nos dé el valor de la función discriminante. Genotipo y fenotipo están formados por los mismos elementos, perteneciendo estos elementos al conjunto de terminales y no terminales (Tabla 4.1):

- Conjunto de no terminales: Constituido por las funciones elementales que constituirán la función discriminante. En el conjunto de funciones se han incluido solamente las funciones aritméticas básicas, esto es, suma, resta, multiplicación, y división. Esta última se ha incluido en su modalidad protegida, es decir $x \div 0 = 0$, para evitar errores durante la evaluación de los individuos. No se han incluido operadores más complejos, como la potencia, raíz cuadrada o razones trigonométricas, debido a que las pruebas para afinar el modelo determinaron que su inclusión no mejoraba el rendimiento global del clasificador.

- Conjunto de terminales: El conjunto de terminales está formado por los valores constantes así como por elementos de entrada que corresponden con los atributos del patrón a utilizar. Para los valores constantes se han utilizado veinte elementos distribuidos uniformemente entre el valor máximo y el valor mínimo de los presentes en el conjunto de datos. De cualquier forma, si fueran necesarios otros valores para generar correctamente las funciones discriminantes estos se producirán durante el proceso evolutivo, por lo que la elección de las constantes no es un factor decisivo que influya en el rendimiento del modelo.

La evaluación posterior del árbol sintáctico generado por el genotipo de un individuo sobre un patrón determinará el valor de la función discriminante representada por el individuo para dicho patrón. Para ello cada gen se convertirá en un subárbol el cual se unirá utilizando la función de conexión como se vio en la sección 3.4. El consecuente de la regla, es decir la etiqueta asociada a cada función discriminante no está codificada en el genotipo del individuo, sino que se asigna durante la evaluación del individuo como se verá posteriormente.

El clasificador final estará compuesto por un conjunto de funciones discriminantes, habiendo una o varias por cada una de las etiquetas del problema. Durante el proceso de clasificación siempre que alguna de éstas funciones determine que el patrón pertenece a la clase, el clasificador considerará que pertenece independientemente del resultado arrojado por el resto. Para que el clasificador considere que el patrón no pertenece a una etiqueta, todas las funciones discriminantes asociadas a ella han de devolver un resultado no positivo, como se puede ver en la figura 4.2.

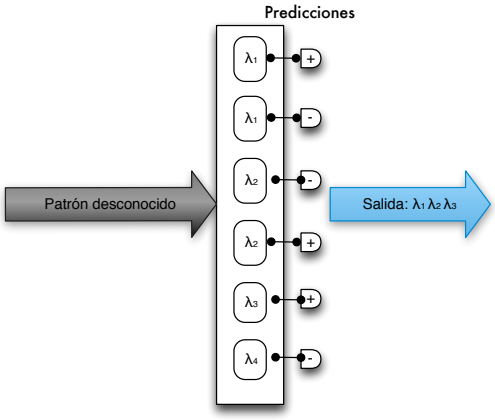


Figura 4.2: Predicciones realizadas mediante GEP-MLC

4.3. Operadores genéticos

Los operadores genéticos que utiliza el algoritmo desarrollado son los ya comentados en el capítulo 3 de recombinación, mutación y transposición.

En cuanto a los operadores de recombinación se han utilizado, la recombinación en un punto, en dos puntos y la recombinación de genes. El principal motivo por el que se han implementado operadores de recombinación es que durante el proceso evolutivo los individuos intercambien entre sí fragmentos de árboles que forman parte de las funciones discriminantes, y que estos operadores fomenten el que se combinen entre sí las funciones que parezcan más adecuadas.

La recombinación en un punto, en el contexto del presente algoritmo intercambia entre individuos todos los niveles del árbol fenotipo a partir del punto determinado que se seleccione. De esta manera un progenitor dona a su descendencia la parte superior del árbol y otro la parte inferior, con los que este operador facilita que se prueben funciones discriminantes que hayan funcionado correctamente con nuevas variables de entrada, además de generar nuevas funciones y combinaciones de entrada. Por ejemplo en la

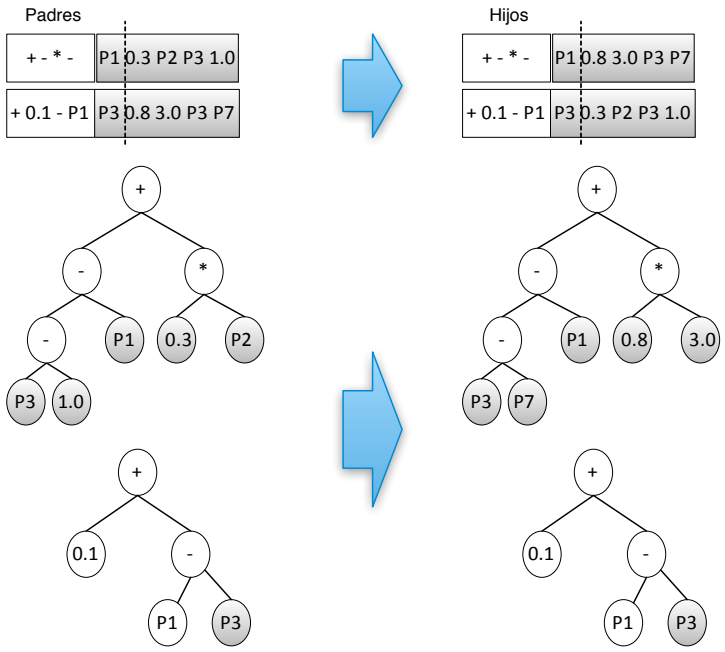


Figura 4.3: La recombinación en un punto suele modificar el conjunto de datos de entrada de la función discriminante

figura 4.3 se puede observar como la recombinación en un punto hace que una función discriminante se pruebe con otro conjunto de datos de entrada.

La recombinación en dos puntos actúa intercambiando partes intermedias de los individuos, y por tanto hace que se modifique tanto la función discriminante como las variables a considerar en el modelo generado. De esta forma contribuye a que se generen nuevas funciones discriminantes distintas de las ya presentes en la población, así como a que se prueben las ya existentes con nuevas variables. Sin embargo las nuevas funciones y conjuntos no se generan de manera aleatoria, sino a partir de bloques de funciones y conjuntos que ya hayan mostrado su validez. En la figura 4.4 se puede observar como, tras aplicar el operador de recombinación, cambia tanto la función discriminante como el conjunto de datos de entrada.

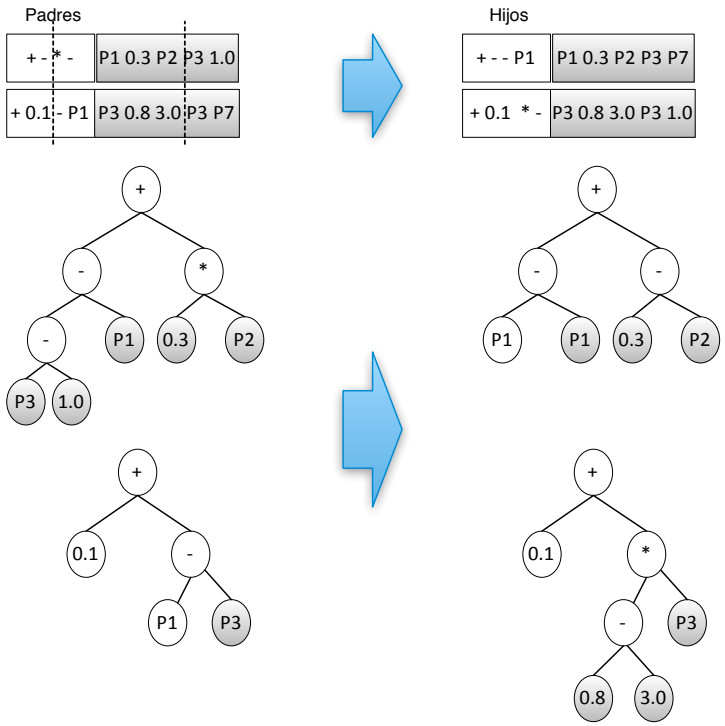


Figura 4.4: La recombinación en dos puntos modifica el conjunto de datos de entrada y la función

La recombinación de genes intercambia entre individuos el subárbol completo asociado a un gen. En este sentido actúa de manera similar a la recombinación en dos puntos ya que genera nuevas funciones discriminantes con nuevos datos de entrada, pero no de manera aleatoria, sino partiendo de funciones ya presentes en la población (Figura 4.5).

El operador de mutación, así como los operadores de transposición se han utilizado con una probabilidad relativamente moderada, debido a que los cambios que producen en el genotipo pueden traducirse en cambios muy bruscos en el fenotipo. La idea del operador de mutación es que se puedan ir explorando nuevos caminos de búsqueda sin que se produzcan fenómenos de deriva genética.

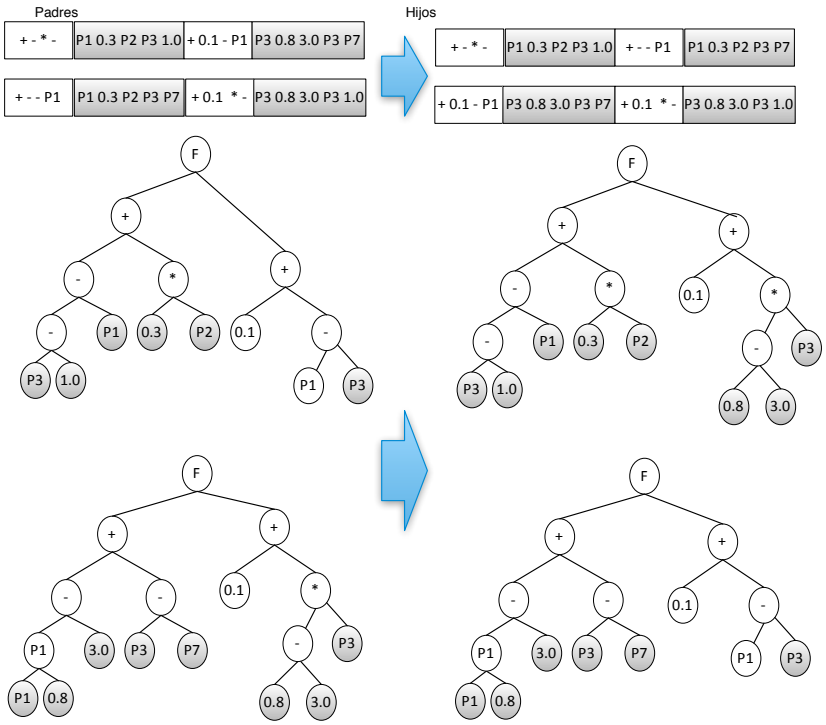


Figura 4.5: Ejemplo de recombinación de genes

Los operadores de transposición generan en el fenotipo cambios parecidos a los del operador de recombinación en dos puntos. Es decir, generan nuevas funciones discriminantes y nuevos conjuntos de datos de entrada.

El operador de transposición RIS genera una función discriminante totalmente distinta a la que tuviera el individuo antes de ser sometido al operador (Figura 4.6), sin embargo esta nueva función no es completamente aleatoria, sino que es parte de uno de los subárboles que tuviera el individuo.

Por otro lado el operador de transposición IS lo que hace es crear una nueva función discriminante, o en menor medida un nuevo conjunto de variables a considerar, a partir de intercambio de posición de partes del árbol fenotípico. De esta manera la función no es tampoco completamente aleatoria ya que permite probar si algunas partes partes del árbol tendrán mejor rendimiento en otra posición. Por ejemplo en la figura 4.7 se puede observar como la

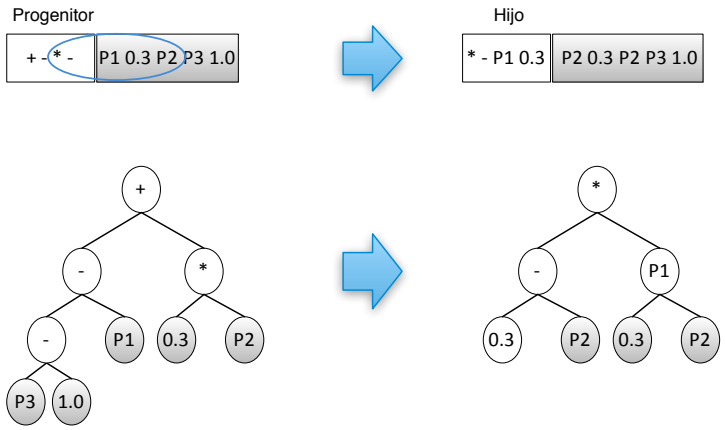


Figura 4.6: La transposición RIS implica un cambio brusco en el genotipo

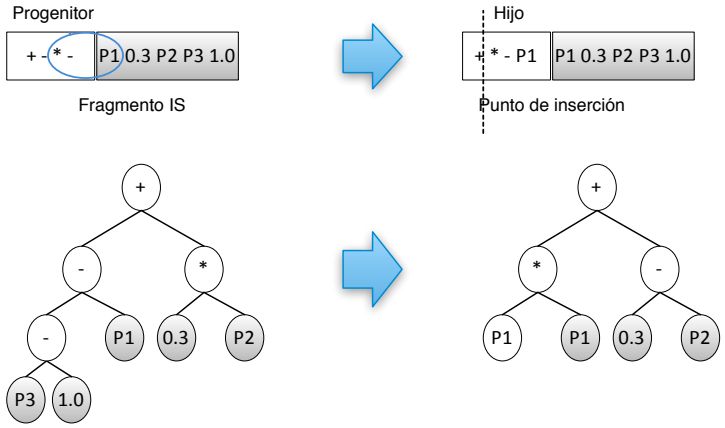


Figura 4.7: La transposición IS modifica la función y en menor medida las variables a considerar

aplicación de la transposición cambia la función discriminante haciendo que partes del árbol se muevan de posición y, en menor medida, hace que cambie el conjunto de terminales a considerar en el modelo.

En último lugar comentar que el operador de transposición de genes no se ha utilizado debido a que no cambia la función discriminante asociada a un genotipo, y por tanto apenas afecta al proceso evolutivo, conllevando una ralentizando de su ejecución.

4.4. Evaluación de los individuos

Dentro de la evaluación de los individuos se han de distinguir dos fases diferenciadas, por un lado el cálculo en sí de la aptitud de cada individuo, que se hace de la manera clásica de los algoritmos evolutivos, mediante una función de *fitness*, y por otro lado la aplicación de una técnica de nichos que trata de potenciar la diversidad en la población de tal manera que haya individuos relacionados con cada una de las etiquetas con las que está trabajando en un determinado problema.

4.4.1. Función de fitness

Nuestra propuesta, GEP-MLC, mantiene una población de individuos con las características mencionadas en la sección 4.2. Una vez hallada la función discriminante asociada a cada genotipo cada uno de los individuos es evaluado mediante una función de *fitness* para cada etiqueta del conjunto de aprendizaje.

La función de *fitness* utilizada es la media armónica entre la *precision* y el *recall*, o *F-score*:

$$fitness = \frac{2 \times precision \times recall}{precision + recall} \tag{4.2}$$

El hecho de que cada patrón de entrenamiento tenga asociadas varias etiquetas implica que el *fitness* de cada individuo depende de la etiqueta para la que se calcule. Por esta razón para cada individuo se almacenará un vector de *fitness*, donde se guardarán todos los valores calculados, uno por cada etiqueta, pero teniendo en cuenta que para aplicar el operador de selección solamente se considerará el mayor de éstos valores.

4.4.2. Algoritmo de nichos multi-etiqueta

Tras la evaluación de los individuos se aplica una técnica de nichos para corregir el *fitness* de los individuos, esta buscará que haya subpoblaciones dentro de la población, formadas individuos especializados en clasificar cada una de las etiquetas presentes en el problema. Esta técnica esta basada en la técnica del *token competition*, propuesta por Tan [286] que es ámpliamente utilizada en algoritmos genéticos aplicados a problemas de clasificación [287; 288].

El *token competition* se utiliza en algoritmos de clasificación clásica para tratar de emular el efecto de nichos presente en los ecosistemas naturales: normalmente cuando una especie encuentra un lugar cómodo donde vivir, no suele evolucionar bruscamente, sino que se adapta a este entorno benigno y trata de impedir que otras especies lo colonicen.

La técnica de nichos aquí propuesta busca que haya conjuntos de individuos especializados en clasificar subconjuntos de patrones, y para ello por cada patrón de entrenamiento positivo entra en juego un *token*, el cual es ganado por el individuo con *fitness* más alto que lo clasifica correctamente. Una vez que se han repartido todos los *tokens*, se procede a corregir el *fitness* de cada uno de los individuos según la siguiente fórmula:

$$nuevo\_fitness = \frac{fitness\_original \times tokens\_ganados}{tokens\_totales} \tag{4.3}$$

De esta manera resultan penalizados los individuos que, aunque clasifiquen un número de patrones razonable, y por tanto tengan un *fitness* adecuado, aporten poco o nada al clasificador porque los patrones que clasifican correctamente también son clasificados por mejores individuos. Por el contrario resultan favorecidos los individuos con un buen *fitness* y que clasifican correctamente muchos patrones, y los individuos que se han especializado

	Individuo_1	Individuo_2	Individuo_3
Patrón_1	x		
Patrón_2	x	x	
Patrón_3	x	x	
Patrón_4			x
fitness	3	2	1
fitness corregido	$(3 * 3)/4 = 2.25$	$(2 * 0)/4 = 0$	$(1 * 1)/4 = 0.25$

Tabla 4.2: Ejemplo de algoritmo de nichos

en clasificar patrones *raros* que se le puedan pasar por alto a los mejores individuos.

Un ejemplo del funcionamiento de esa técnica se puede observar en la tabla 4.2. Consideramos el *fitness* el número de patrones clasificado correctamente, y además que el $individuo_1$ clasifica correctamente los tres primeros patrones, el $individuo_2$, el segundo y tercer patrón y el $individuo_3$ solamente el cuarto patrón. El efecto de nichos se produce ya que el *fitness* corregido favorece al $individuo_3$, que, a pesar de solamente clasificar un patrón, se ha especializado en un patrón raro, y por otro lado perjudica al $individuo_2$, que no aporta nada a la población ya que los patrones que clasifica también son clasificados por el $individuo_1$.

En nuestro caso se realizan tantos *token competition* como etiquetas haya en el conjunto de datos, utilizando como *tokens* en cada caso solamente los patrones que poseen la etiqueta. En cada competición participan todos los individuos con el *fitness* que tuvieran asignado para la etiqueta correspondiente, como se ha comentado en 4.4.1 siendo este valor del vector de *fitness* del individuo el que resulta corregido.

4.5. Algoritmo evolutivo

El algoritmo desarrollado tiene en cuenta que ha de calcularse el vector de *fitness* para cada individuo, y además implementa la técnica de nichos multi-etiqueta ya comentada para corregir la aptitud de los individuos una vez evaluados. El resto de fases son las comunes en cualquier otro algoritmo basado en GEP [15]. Un esquema en seudocódigo del algoritmo propuesto puede observarse en el algoritmo 1.

Una vez generada la población inicial, mientras no se llegue al máximo de generaciones se realizan la siguientes acciones para cada generación:

1. Para cada etiqueta presente en el problema, se evalúan todos los individuos sobre dicha etiqueta, y se obtiene el vector de *fitness*.

2. Posteriormente para cada etiqueta se lleva a cabo un *token competition*, para ello se llevan a cabo las siguientes acciones:

   a) El número de *tokens* ganado por cada individuo comienza siendo 0, y el número total de *tokens* que entran en juego igual al número total de patrones que tienen la etiqueta.

   b) Cada *token* es ganado por el individuo con mayor *fitness* que lo clasifica correctamente.

   c) Una vez jugados todos los *tokens*, se actualiza el *fitness* de los individuos según la ecuación 5.3.

3. Una vez acabada la competición, y con el mayor *fitness* de cada individuo, se realiza la selección de los individuos que constituirán la población de la siguiente generación, y se aplican los operadores genéticos.

Al final de la ejecución del algoritmo es sencillo localizar aquellos individuos que deben formar parte del clasificador aprendido, solamente aquellos

Algorithm 1 Seudocódigo para GEP-MLC

Generar población inicial $P(0)$
$g_{count} \leftarrow 0$
while $g_{count} < g_{max}$ **do**
  for all $\lambda_i \in L$ (Conjunto de etiquetas) **do**
    for all Individuos Ind_j en $P(g_{count})$ **do**
      Evaluar Ind_j con λ_i como consecuente
      Almacenar $fitness_i$ en el vector de *fitness* de Ind_j
    end for
  end for
  for all $\lambda_i \in L$ **do**
    $tokens\_totales$ = número de patrones pertenecientes a λ_i
    for all Individual Ind_j in $P(g_{count})$ **do**
      $tokens\_ganados^{\lambda_i}_{Ind_j} = 0$
    end for
    for all patrones positivos ($token$) **do**
      Seleccionar Ind_j con mayor *fitness* que clasifica el patrón
      $tokens\_ganados^{\lambda_i}_{Ind_j} + +$
    end for
    for all Individuo Ind_j en $P(g_{count})$ **do**
      Actualizar $fitness$ según formula 4.3
    end for
  end for
  Realizar selección considerando mayor $fitness$ del vector
  Aplicar operador de cruce
  Aplicar operadores de mutación y transposición RIS e IS
  Actualizar población
  $g_{count} + +$
end while
for all $Ind_j \in$P(g$_{max}$)(Generar clasificador) **do**
  for all $\lambda_i \in L$ **do**
    if $tokens\_ganados^{\lambda_i}_{Ind_j} > 0$ **then**
      consecuente$_{Ind_j} \leftarrow consecuente_{Ind_j} \cup \lambda_i$
    end if
  end for
  if consecuente$_{Ind_j} \neq \emptyset$ **then**
    clasificador $\leftarrow clasificador +$ Ind$_j$
  end if
end for

Poblacion	1000 individuos
Núm. Genes	6
Longitud cabeza	35
Núm. Max. Generaciones	60
Selección	Torneos tamaño 2
Prob. mutación	0.2
Prob. cruce	0.7
Prob. Transposición RIS e IS	0.4
Tamaño fragmento IS	6
Tamaño fragmento RIS	10
Conjunto de no terminales	+,-,*,/
Función de conexión	+

Tabla 4.3: Parámetros del algoritmo GEP-MLC

que hayan ganado algún *token* son relevantes para el clasificador, pudiendo despreciarse el resto.

4.6. Marco experimental

El objetivo de la fase experimental es comparar el rendimiento del algoritmo desarrollado frente a otras implementaciones de algoritmos multi-etiqueta.

Previo a abordar el grueso la experimentación se han realizado una serie de pruebas con el fin de determinar los parámetros óptimos del algoritmo. Estos parámetros, con los que se ha trabajado en el resto de los experimentos aparecen detallados en la tabla 4.3.

El desempeño del algoritmo GEP-MLC se ha comparado con otros seis métodos para clasificación multi-etiqueta: dos de trasformación de problemas, el método de *Binary Relevance* (BR) y el método *Label Powerset* (LP).

Por otro lado, también se ha comparado con los métodos de adaptación ML-kNN[67], CLR [1] y BPML[1]. Y en último lugar se ha probado con el *ensemble* RAkEL[26]. Todos los métodos con los que se ha comparado están descritos en la sección 2.4.

En cuanto a los parámetros de los algoritmos de comparación reseñar que los métodos *Binary Relevance (BR)* y *Label Powerset (LP)* utilizan como algoritmo de clasificación base el algoritmo basado en árboles C4.5 en su implementación J48 de *Weka* [26]. El algoritmo RAkEL se ha utilizado en su configuración original, es decir, utilizando LP con C4.5 como clasificador base, un tamaño de subconjunto igual a 3, un número de modelos igual al doble del número de etiquetas y 0.5 como valor umbral [26]. Por otro lado, el algoritmo ML-kNN se ha utilizado con un número de vecinos igual a 10 y un factor de *smoothing* igual a 1, como se recomienda en [7]. Los parámetros de BPML utilizados son un valor 0.05 de ratio de aprendizaje, 100 épocas y un número de unidades ocultas igual al 20 % de las unidades de entrada, configuración recomendada en [289]. Por último *Calibrated Label Ranking (CLR)* utiliza como clasificador base también C4.5 en su versión J48.

Las implementaciones de los métodos que se han utilizado en la comparativa son los desarrollados sobre la librería MULAN[1]. Esta es una biblioteca de código abierto realizada Java para el aprendizaje a partir de conjuntos de datos multi-etiqueta. En este momento, la biblioteca incluye una gran variedad de algoritmos de aprendizaje multi-etiqueta tanto métodos de adaptación de algoritmos como de transformación de problemas. Los algoritmos incluidos abarcan un gran cantidad de tareas de MLL, entre las que se incluyen MLC, *ranking* y MLC mediante *ranking*.

[1]Disponible en línea en `http://mulan.sourceforge.com`

4.6.1. Conjuntos de datos

Los seis conjuntos de datos, ya introducidos en el capítulo 2, utilizados para realizar la fase experimental son los denominados *scene*, *emotions*, *yeast*, *TMC2007*, *genbase* y *mediamill*.

Los atributos nominales, presentes en los datasets *genbase* y *TMC2007*, han sido binarizados de la manera que se describe en [290] convirtiendo cada atributo en los valores binarios cero y uno respectivamente si los atributos tenían un valor booleano. Este proceso es necesario cuando se utilizan funciones discriminantes, porque estas funciones solamente pueden aceptar valores numéricos como entrada.

Todos los algoritmos se han probado sobre los conjuntos descritos utilizando una validación cruzada con $k = 10$ (*10-fold cross validation*).

4.7. Resultados y discusión

4.7.1. Resultados experimentales

Con el fin de comparar el algoritmo propuesto, GEP-MLC, con los algoritmos BR, LP, ML-kNN, RAkEL, CLR y BPML se ha llevado a cabo una validación cruzada con 10-*fold* utilizando medidas comentadas en el capítulo 2 de *accuracy* (Ecuación 2.6), *precision* (Ecuación 2.3), y *recall* (Ecuación 2.4). La tabla 4.4 muestra los resultados siguiendo la aproximación micro, mostrada también en el capítulo 2 (Ecuación 2.2), para cada métrica y conjunto de datos.

El modelo propuesto, como se puede observar, obtiene en general buenos resultados para cada uno de los *dataset* y métricas, comparando con los

	BPML	CLR	ML-kNN	BR	LP	RAkEL	GEP-MLC
Mediamill	0.512	0.592	0.634	0.416	0.485	0.457	**0.703**
Yeast	0.515	0.469	0.508	0.406	0.347	0.444	**0.738**
TMC2007	**0.748**	0.747	0.688	0.541	0.525	0.587	0.543
Emotions	0.549	0.477	0.341	0.462	0.438	0.509	**0.755**
Genbase	0.043	**0.986**	0.941	0.985	0.983	**0.986**	0.903
Scene	0.499	0.527	0.673	0.536	0.588	0.622	**0.709**

(a) *Accuracy*

	BPML	CLR	ML-kNN	BR	LP	RAkEL	GEP-MLC
Mediamill	0.472	0.540	**0.698**	0.487	0.584	0.524	0.669
Yeast	0.604	0.652	**0.729**	0.621	0.441	0.524	0.715
TMC2007	**0.757**	0.710	0.584	0.642	0.545	0.687	0.618
Emotions	0.661	0.604	0.639	0.606	0.557	0.649	**0.724**
Genbase	0.010	0.992	**0.989**	0.980	0.974	**0.989**	0.650
Scene	0.597	0.606	**0.820**	0.606	0.590	0.705	0.746

(b) *Precision*

	BPML	CLR	ML-kNN	BR	LP	RAkEL	GEP-MLC
Mediamill	0.503	**0.592**	0.503	0.431	0.511	0.472	0.581
Yeast	0.698	0.584	0.570	0.479	0.428	0.628	**0.749**
TMC2007	0.679	**0.698**	0.693	0.632	0.629	0.602	0.540
Emotions	0.691	0.657	0.372	0.600	0.544	0.614	**0.695**
Genbase	0.341	**0.983**	0.901	0.946	0.897	0.983	0.977
Scene	0.670	0.652	0.672	0.624	0.592	0.645	**0.744**

(c) *Recall*

Tabla 4.4: Resultados experimentales para GEP-MLC

obtenidos por el resto de algoritmos. Los resultados de GEP-MLC son similares a los de otras propuestas en algunos conjuntos de datos. No obstante, en otros casos los resultados de GEP-MLC son superiores que los del resto.

Si analizamos los resultados obtenidos para la *accuracy*, mostrados gráficamente en la figura 4.8, observamos que nuestro modelo obtiene los mejores

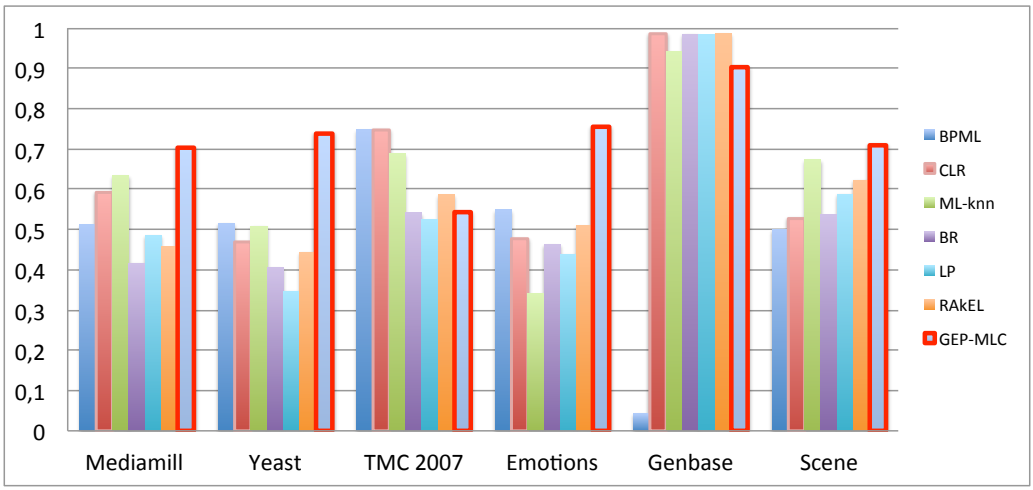


Figura 4.8: Resultados experimentales para *accuracy*

resultados para todos los conjuntos excepto para *genbase* y *TMC2007*. Los resultados obtenidos en los conjuntos *scene* y *genbase* son bastante similares a los del resto e las propuestas, pero destaca que los conjuntos con mayor cardinalidad, esto es *mediamill* y *yeast* son en los que GEP-MLC obtiene un resultado apreciablemente mejor que el resto.

Los resultados para *precision* y *recall* se muestran en las figuras 4.9 y 4.10 respectivamente. Debido a que ambas métricas están interelacionadas como ya se comentó en capítulo 2 es conveniente interpretar los resultados de ambas conjuntamente.

Lo primero que merece la pena destacar es que los resultados para *precision* y *recall* de GEP-MLC en todos los conjuntos de datos son bastante homogéneos, esto es, para el mismo conjunto se obtienen valores similares de ambas métricas, lo que indica que el algoritmo no maximiza una a costa de penalizar la otra. Sin embargo hay que hacer ciertas salvedades a esta tendencia, los resultados de ML-kNN superan en varios conjuntos a los de GEP-MLC en *precision* y lo mismo ocurre con los resultados de CLR para el *recall*. La explicación de estos resultados es la misma en ambos casos, dichos algoritmos aumentan el valor de una métrica a costa de reducir el de la

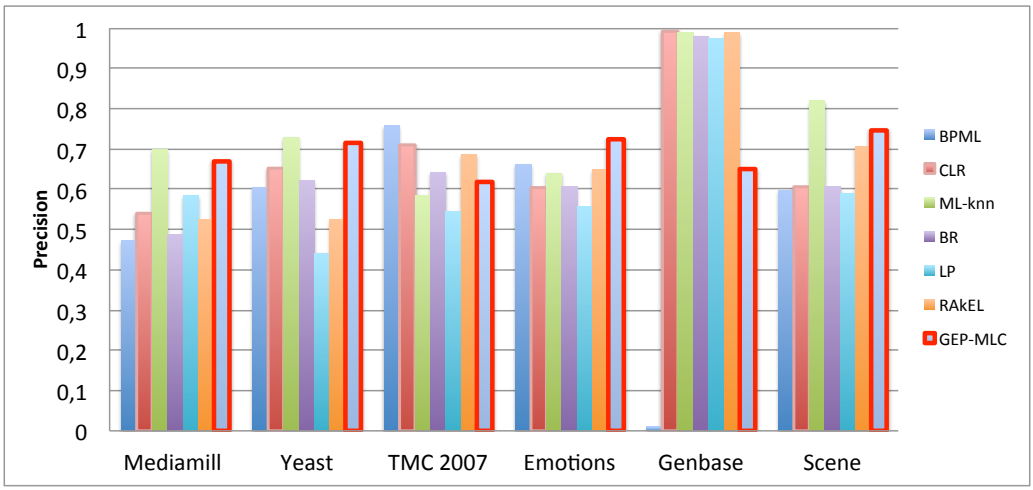


Figura 4.9: Resultados experimentales para *precision*

otra, ya que como se puede observar en las gráficas, ML-kNN obtiene unos resultados modestos en *recall* y CLR relativamente moderados en *precision*.

También hay que reseñar que los resultados que se obtienen siguen la misma tendencia que los comentados para la *accuracy*, esto es, resultados similares a los de resto de propuestas para conjuntos con cardinalidad baja y resultados mejores para conjuntos con cardinalidad alta, exceptuando de esa tenencia

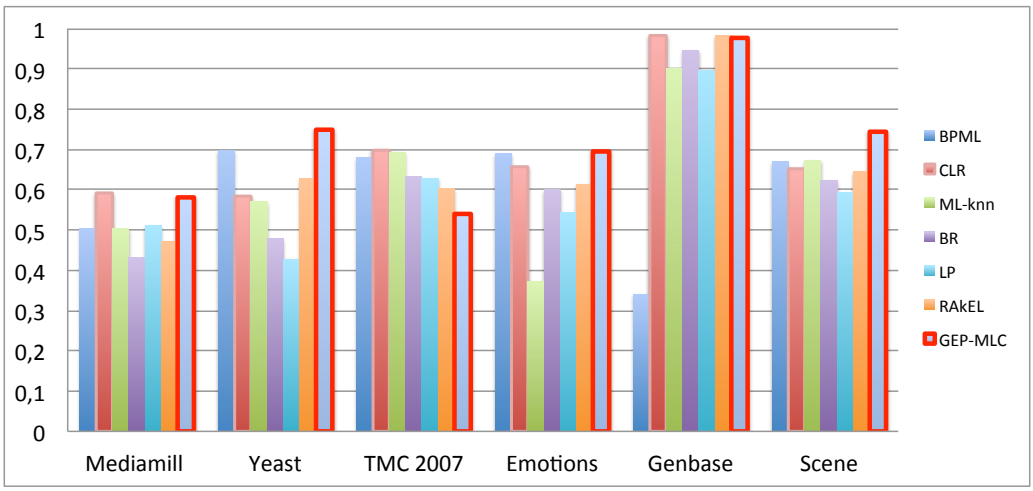


Figura 4.10: Resultados experimentales para *recall*

Algoritmo	*Accuracy*	*Precision*	*Recall*
BPML	6.10	7.00	4.85
RAkEL	5.35	5.20	5.35
CLR	5.05	5.25	3.25
ML-kNN	3.67	2.58	3.67
BR	2.83	2.83	2.58
LP	2.33	2.83	2.51
GEP-MLC	1.16	2.26	1.83

Tabla 4.5: *Ranking* medio de los algoritmos para las tres métricas

los conjuntos *genbase* y *TMC2007*. Con respecto a los conjuntos de datos *TMC2007* y *genbase* en los que GEP-MLC no obtiene tan buenos resultados como se podría inferir de la cardinalidad, hay que destacar que los dos son, a diferencia del resto de *datasets*, conjuntos de datos nominales booleanos que han sido convertidos en numéricos para utilizar sobre ellos funciones discriminantes. Este hecho sugiere que la propuesta aquí formulada no es especialmente adecuada para manejar atributos nominales, hipótesis bastante razonable ya que las funciones discriminantes no pueden manejar este tipo de datos directamente.

Otro aspecto que conviene destacar es que GEP-MLC siempre gana en todas las métricas y conjuntos a los algoritmos de transformación BR y LP.

4.7.2. Análisis Estadístico de los resultados

Los algoritmos probados se han comparado estadísticamente utilizando un test de Friedman [291; 292]. Las puntuaciones promedio de los algoritmos se encuentran en la tabla 4.5.

El estadístico de Friedman se compara posteriormente con una distribución χ^2 con tantos grados de libertad como conjuntos de datos se hayan utilizado menos uno, lo que nos lleva a aceptar la hipótesis nula (los resultados son

Métrica	Estad. Friedman	χ^2 $(p = 0.1)$	Conclusión
Accuracy	11.8	10.64	Rechaza Hipótesis Nula
Precision	8.79		Acepta Hipótesis Nula
Recall	9.79		Acepta Hipótesis Nula

Tabla 4.6: Resultados del test de Friedman ($\alpha = 0.1$) para GEP-MLC

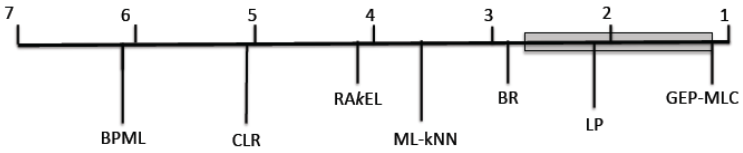


Figura 4.11: Test de Bonferroni-Dunn para *accuracy* ($\alpha = 0.1$)

estadísticamente similares) o a rechazarla (hay diferencias significativas entre las propuestas). Los resultados para el test de Friedman se pueden observar en la tabla 4.6. Estos resultados nos indican que hay diferencias significativas entre los algoritmos solamente para la *accuracy*, siendo las diferencias no significativas en los casos de *precision* y *recall*. De cualquier manera hay que destacar que nuestra propuesta, GEP-MLC, obtiene para todas las métricas los mejores valores de *ranking*.

Una vez realizado el test de Friedman, para determinar cual es el mejor algoritmo se ha de realizar el post test de Bonferroni-Dunn. En este caso solamente se ha de utilizar para la *accuracy*, ya que para el resto no se ha determinado que haya diferencias significativas. El valor crítico del post test de Bonferroni-Dunn con $p = 0.1$ es 1.58, lo que nos indica que hay diferencias significativas entre dos algoritmos si los valores de *ranking* superan este umbral. De los valores de *ranking* se observa que GEP-MLC es significativamente mejor que todas las demás excepto LP. El resultado de este test se puede observar gráficamente en la figura 4.11.

4.8. Conclusiones

En este capítulo se ha presentado el algoritmo GEP-MLC, el cual es un algoritmo evolutivo que permite realizar clasificación multi-etiqueta. Está basado en el paradigma de la *Gene Expression Programming*, y codifica funciones discriminantes en el individuo con un genotipo formado por cadenas lineales que se interpreta como un árbol sintáctico.

Estas funciones discriminantes nos indican si un patrón pertenece o no a una clase determinada en base a un valor numérico. El clasificador final es construido combinando varias de las funciones presentes en la población final.

El algoritmo propuesto utiliza una novedosa técnica de nichos multi-etiqueta para asegurar que hay individuos en la población de todas las clases presentes en el problema, de tal manera que se constituyan nichos de individuos en la población especializados en clasificar subconjuntos de patrones para una determinada clase.

Los estudios que se han llevado a cabo para medir el rendimiento del algoritmo con respecto a otras alternativas tanto de transformación de problemas como de adaptación de algoritmos nos indican que la propuesta hecha es razonablemente mejor que las anteriores mostradas en la bibliografía, siendo adecuada para trabajar con datos numéricos.

5

Clasificadores multi-etiqueta evolutivos basados en reglas

5.1. Introducción

El modelo desarrollado en el capítulo anterior es un modelo robusto, y como se puso de manifiesto en los experimentos a los que se sometió, resulta especialmente adecuado para tratar con conjuntos de datos numéricos. Por otro lado esos mismos resultados experimentales pusieron de manifiesto que el modelo anterior no resulta tan potente a la hora de tratar con conjuntos de datos categóricos.

Sin embargo, como se planteó en la introducción, los conjuntos de datos sobre muchos de los problemas tratados mediante aprendizaje muti-etiqueta son nominales, como la categorización de textos o problemas de bioinformática. Estos problemas, aunque pueden ser abordados utilizando funciones discriminantes, parece más adecuado tratar de buscar un formalismo que pueda aceptar como entrada directamente este tipo de datos.

Por otro lado al trabajar con funciones discriminantes se generan clasificadores bastante robustos, pero el resultado del clasificador es difícilmente

interpretable como conocimiento útil, ya que este realiza un conjunto de operaciones aritméticas con el vector de características, devolviendo un valor que se asociará con la clase mediante un umbral.

Estos son los motivos por los que en el presente trabajo se ha planteado diseñar una serie de modelos de clasificación que nos permitan por un lado trabajar tanto con datos numéricos como nominales sin necesitar un procesado previo, y por otro lado, obtener conocimiento interpretable a partir del clasificador desarrollado.

Un formalismo que permite evolucionar clasificadores con estas características son las reglas de clasificación [220]. Como ya se comentó en la sección 3.6 una regla es una estructura condicional: $Si\ A\ Entonces\ C$, siendo A el antecedente, o sea, el conjunto de condiciones que debe cumplir un patrón para ser asociado con el consecuente C. El consecuente de una regla, en problemas de clasificación clásica, suele ser la clase que se asocia al patrón que *cubra* la regla, es decir, aquel que cumpla el antecedente.

En este capítulo se introduce un algoritmo de programación genética, llamado GC. La principal característica de este algoritmo es que induce un clasificador basado en reglas, de esta manera cada individuo de la población codifica una regla candidata a formar parte del clasificador final.

El resto del capítulo está organizado de la siguiente manera: primero se comentan los aspectos más relevantes sobre la codificación de los individuos, para posteriormente comentar los operadores genéticos utilizados. A continuación se detalla como se realiza la evaluación de los individuos en dos variantes que se proponen del algoritmo, denominadas GC-1 y GC-k, para pasar a hablar de la dinámica global del algoritmo. Tras esto se plantea el marco experimental. El capítulo finaliza con la presentación de los resultados experimentales, su estudio y discusión así como las conclusiones a las que se ha llegado tras el desarrollo y prueba de los modelos planteados.

5.2. Representación de los individuos

El objetivo del modelo que se ha desarrollado es aprender clasificadores multi-etiqueta formados por un conjunto de reglas de clasificación que estarán asociadas con uno o varios consecuentes.

Cada individuo del algoritmo evolutivo representará el antecedente de una regla. Los individuos del algoritmo GC hacen uso de una codificación dual, por esta razón el genotipo de los individuos será una cadena lineal y el fenotipo será un árbol de expresiones que representará el conjunto de condiciones del antecedente de una sola regla de clasificación.

El consecuente de la regla no aparece codificado en los individuos sino que es asignado durante el proceso evolutivo, en su fase de evaluación, cuando a cada individuo se le asigna el consecuente que maximiza su *fitness*.

Debido a que el fenotipo del individuo representa una regla de clasificación, se han utilizado los mecanismos definidos en la GEPCLASS [16], mostrada en el capítulo 3, que permite que los individuos codifiquen reglas de clasificación.

De este modo, el genotipo y fenotipo están formados, como ya se ha visto anteriormente, por los mismos elementos, funciones y terminales:

- El conjunto de funciones estará dividido en dos subconjuntos: el subconjunto de funciones relacionales y el subconjunto de funciones lógicas. Las características de los dos subconjuntos son las siguientes:
  - Conjunto de funciones lógicas: Este conjunto de funciones se utiliza para los elementos presentes en los niveles entre la raíz y el antepenúltimo del árbol. Se han utilizado las funciones booleanas clásicas, Y, O y NO.
  - Conjunto de funciones relacionales: Este conjunto constituye el nivel penúltimo del árbol generado. Cuando se están manejando datos nominales, el conjunto de relacionales solamente contiene

Conj. de funciones lógicas	Y, O NO
Conj. de funciones relacionales	$=$,$\neq$, $>$,$<$
Conj. no terminales	pares *atributo-valor_constante*

Tabla 5.1: Conjuntos de terminales y no terminales en GC

dos, las funciones igual y distinta ($=$, $\neq$), pero si se trabaja con datos numéricos se han utilizado ($=$,$\neq$, $>$,$<$).

- Conjunto de terminales: A diferencia de la GEP está formado por pares *atributo-valor_constante*, siendo los atributos valores de entrada del conjunto de datos, y los valores, parámetros constantes. Cuando se trabaja con valores nominales se utilizan como constantes las categorías que aparezcan en el *dataset*, pero en el caso de tratar con valores reales, las constantes han se ser escogidas de manera más cuidadosa que en el anterior modelo basado en funciones discriminantes, ya que el propio algoritmo no podrá generar los valores que necesite durante el proceso de búsqueda con la misma facilidad que el modelo GEP-MLC[1]. Cada valor constante se asocia con una serie de valores repartidos uniformemente entre el máximo y el mínimo de cada atributo.

En la tabla 5.1 se resumen los conjuntos de elementos que constituyen los individuos.

El genotipo de los individuos está constituido por cabeza y cola. En la primera mitad de la cabeza se permite cualquier elemento del conjunto de funciones, sin embargo en la segunda mitad solamente se permiten funciones relacionales y en la cola solamente pueden aparecer terminales. De esta

[1]El algoritmo GEP-MLC puede generar los valores constantes que necesite durante el propio proceso evolutivo, ya que el fenotipo de un individuo puede realizar operaciones aritméticas con los valores que se le suministren. Sin embargo las propuestas GC no pueden realizar operaciones aritméticas con las constantes, siendo por tanto más delicado el proceso de escogerlas al inicio.

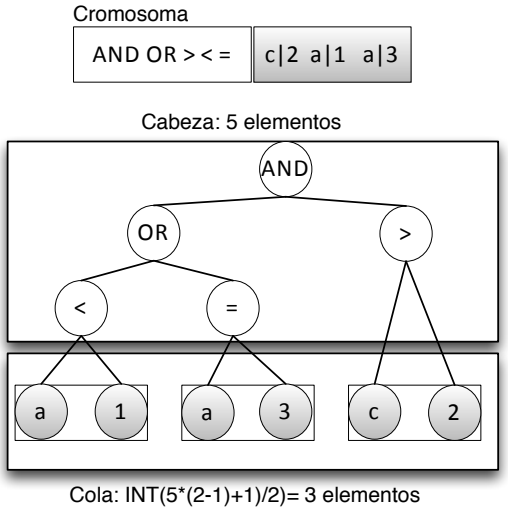


Figura 5.1: Cabeza y cola en un individuo GC

manera se garantiza que el último nivel del árbol este formado por elementos terminales, y el penúltimo nivel por funciones relacionales.

El modo de calcular el tamaño de la cola también cambia ya que cada elemento de la cola almacena un par de elementos del árbol sintáctico, y por tanto el tamaño de cola necesario para rellenar el último nivel del árbol se reduce a la mitad, como ya se discutió en el capítulo 3 (Figura 5.1) y se calcula según la ecuación 5.1 donde *int* devuelve la parte entera de su argumento, h es el tamaño de la cabeza y n la aridad máxima de conjunto de funciones.

$$t = int([h * (n - 1) + 1]/2) \tag{5.1}$$

El fenotipo del individuo se genera construyendo un árbol de arriba a abajo como se vio en el capítulo 3. El primer elemento pasa a ser la raíz del árbol, los siguientes (tantos como aridad tenga la raíz) pasan a colgar de este, y así sucesivamente hasta que en cada hoja sólo hay elementos del conjunto de terminales, como también se puede observar en la figura 5.1. Este fenotipo

generado a partir del genotipo es un árbol sintáctico que representa el antecedente de una regla de clasificación candidata a pertenecer al clasificador final.

El modelo muti-etiqueta que resulta de la ejecución de ambas versiones del algoritmo sobre un conjunto de entrenamiento está compuesto por un conjunto de reglas de clasificación. En la versión GC-1 el clasificador final tendrá una o varias reglas por cada etiqueta presente en el problema, pero sin embargo los clasificadores generados por la versión GC-k están compuestos por reglas cuyo consecuente es una combinación de etiquetas.

5.3. Operadores genéticos

Los operadores genéticos de cruce, transposición, y mutación utilizados son los presentados en el capítulo 3.

El operador de cruce (Figura 3.10 del capítulo 3) intercambia fragmentos de la cabeza de cada gen, desde el punto seleccionado aleatoriamente hasta el final de esta. De esta forma el operador de cruce genera nuevas reglas de clasificación a partir de las reglas seleccionadas como padres. También conviene tener en cuenta que, por la estructura de los individuos, puede actuar en muchas ocasiones solo sobre los operadores relacionales (ya que estos conforman la segunda mitad de la cabeza). Esto puede tener cierta relevancia en los conjuntos de datos nominales, en los que solo hay uno o dos operadores relacionales, ya que el cruce puede dejar prácticamente inalterado al individuo, y por tanto el proceso evolutivo puede ser un poco más lento.

Los operadores de transposición (Figura 3.11 del capítulo 3) mueven de una posición a otra fragmentos de la cola. De esta manera lo que se está haciendo es probar la misma regla de clasificación codificada en la cabeza con distintos conjuntos de atributos y constantes. Así, las reglas que han demostrado tener

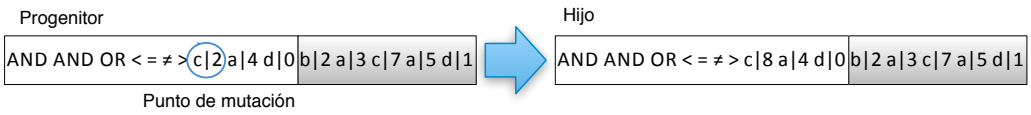


Figura 5.2: Ejemplo del operador de mutación de constantes

valor como clasificador se afinan buscando el conjunto de terminales más adecuado para ellas.

El operador de mutación simple (Figura 3.7a del capítulo 3), en este contexto lo que hace es introducir diversidad en la población, ya que mientras que los dos operadores anteriores van combinando las características de los individuos más prometedores, la mutación sencilla genera la diversidad necesaria para explorar nuevos caminos de búsqueda, ya que cambia aleatoriamente elementos del individuo por otros de los conjuntos de elementos que puedan aparecer en su posición.

Además de estos operadores se ha añadido un nuevo operador de mutación, que no había sido previamente utilizado en otros algoritmos basados en GEPCLASS, que se ha denominado *operador de mutación de constantes*. Este operador es, como se indica, un operador de mutación que, por tanto, actúa sobre un único individuo, y modifica a este del siguiente modo: se selecciona un elemento terminal de árbol (formado por un par *atributo-valor_constante*) y se cambia el valor constante por un valor aleatorio dentro del rango de valores del atributo (Figura 5.2).

El objetivo de este operador es introducir diversidad en las constantes durante el proceso evolutivo de búsqueda, de tal manera que los pares *atributo-valor_constante* vayan cambiando durante dicho proceso, no estando limitados a los pares que se hayan generado al comienzo de la ejecución del algoritmo.

5.4. Evaluación de los individuos

Después de la generación del fenotipo, cada individuo debe ser evaluado según una función de *fitness*. Puesto que el consecuente no se almacena en los individuos, antes de proceder a la evaluación ha de serle asignado un consecuente para la regla de clasificación que codifican, ya que obviamente el *fitness* del individuo depende del valor de dicho consecuente.

La principal diferencia entre las dos versiones del algoritmo aquí propuestas, GC-1 y GC-k, aparece en los tipos de consecuente de la regla. Las reglas encontradas por GC-1 han de tener como consecuente una etiqueta del conjunto de etiquetas, pero las de GC-k pueden tener como consecuente varias etiquetas, siempre que estas sean una combinación de etiquetas de las presentes en el conjunto de entrenamiento.

En ambos casos, el algoritmo almacena un vector con todos los posibles *fitness* de cada individuo, guardando un *fitness* para cada posible consecuente, pero solamente considera el mayor de todos ellos a la hora de aplicar el operador de selección.

La función de *fitness* utilizada es la *F-Score*, la media armónica entre la *precision* y el *recall* del clasificador, es mostrada en la ecuación 5.2.

$$fitness = \frac{2 \times precision \times recall}{precision + recall} \tag{5.2}$$

Debido a que es necesario que en la población haya individuos que estén asociados con todos los consecuentes presentes en el problema, los algoritmos aplican la técnica de nichos multi-etiqueta descrita en el capítulo 4. Para ello, para cada posible consecuente se llevará a cabo una competición en la que, por cada patrón positivo asociado a este consecuente, se juega un *token* que es ganado por el individuo con mayor *fitness* que lo clasifica correctamente.

Una vez estén distribuidos todos los *tokens*, el valor de *fitness* de toda la población es corregido utilizando la ecuación 5.3.

$$nuevo\_fitness = \frac{fitness\_original \times tokens\_ganados}{tokens\_totales} \tag{5.3}$$

De esta manera la técnica penaliza a aquellos individuos, que, a pesar de tener un *fitness* adecuado, apenas contribuyen al clasificador final porque los patrones que clasifican correctamente son también clasificados por individuos con mejor *fitness*. Por otro lado favorece tanto a individuos que clasifican muchos patrones como a individuos especializados en clasificar patrones raros, que suelen ser bloqueados por los mejores individuos.

5.5. Algoritmo evolutivo

El algoritmo propuesto se presenta en dos versiones, como ya se comentó en la introducción. La versión GC-1 en la que el consecuente de los individuos está asociado a una sola etiqueta y la versión GC-k que permite que el consecuente sea un conjunto de etiquetas. A continuación en esta sección se discuten los detalles de ambas versiones del algoritmo evolutivo.

5.5.1. Versión GC-1

En el algoritmo 2 puede observarse una descripción en seudocódigo del algoritmo GC-1. La dinámica de este es similar al de cualquier otro algoritmo evolutivo, con una fase de inicialización, varias iteraciones evolutivas en las que se seleccionan los mejores individuos, se evalúa y se aplican los operadores genéticos y una fase final de construcción del clasificador una vez se ha alcanzado el límite de iteraciones.

Algorithm 2 Algoritmo evolutivo. Versión GC-1

Generar población inicial $P(0)$
$g_{count} \leftarrow 0$
while $g_{count} < g_{max}$ **do**
  for all $\lambda_i \in L$ (Conjunto de etiquetas) **do**
    for all Individuos Ind_j en $P(g_{count})$ **do**
      Evaluar Ind_j con λ_i como consecuente
      Almacenar $fitness_i$ en el vector de *fitness* de Ind_j
    end for
  end for
  for all $\lambda_i \in L$ **do**
    $tokens\_totales$ = número de patrones pertenecientes a λ_i
    for all Individuo Ind_j en $P(g_{count})$ **do**
      $tokens\_ganados^{\lambda_i}_{Ind_j} = 0$
    end for
    for all Patrones positivos ($tokens$) **do**
      Seleccionar Ind_j con mayor *fitness* que clasifica el patrón
      $tokens\_ganados^{\lambda_i}_{Ind_j} + +$
    end for
    for all Individuo Ind_j en $P(g_{count})$ **do**
      Actualizar $fitness$ según formula 5.3
    end for
  end for
  Realizar selección considerando el mayor *fitness* del vector
  Aplicar operadores de cruce
  Aplicar operadores de mutación y transposición RIS e IS
  Actualizar población
  $g_{count} + +$
end while
for all $Ind_j \in$P(g$_{max}$)(Generar clasificador) **do**
  for all $\lambda_i \in L$ **do**
    if $tokens\_ganados^{\lambda_i}_{Ind_j} > 0$ **then**
      consecuente$_{Ind_j} \leftarrow consecuente_{Ind_j} \cup \lambda_i$
    end if
  end for
  if consecuente$_{Ind_j} \neq \emptyset$ **then**
    clasificador $\leftarrow clasificador+$ Ind$_j$
  end if
end for

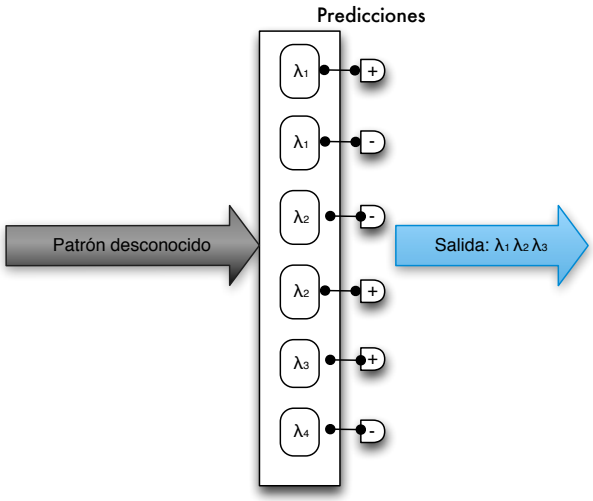


Figura 5.3: Ejemplo de clasificador inducido mediante GC-1

El primer proceso que se lleva a cabo es generar aleatoriamente la población inicial. Para esto han de construirse los conjuntos de terminales y no terminales a partir de los parámetros del algoritmo y tras examinar el *dataset* para hallar el número de atributos de entrada y los valores constantes dentro del rango de los atributos.

Tras la inicialización, la primera acción que se lleva a cabo en cada generación es construir el árbol fenotípico de cada individuo, y en segundo lugar se evalúa cada regla generada, considerando como consecuente cada una de las etiquetas presentes en el problema, y almacenando este valor en un vector de *fitness*.

Tras la fase de evaluación, se aplica la técnica de nichos multi-etiqueta, jugándose para ello un *token* por cada etiqueta y patrón positivo asociada a ella, y una vez acabadas todas las competiciones, se modifica el vector *fitness* de cada individuo, dependiendo de los *tokens* que ha ganado cada uno.

Por último, en cada generación se aplican los operadores genéticos de cruce, mutación y transposición, y se seleccionan los individuos que constituirán la

siguiente generación en base a una selección por torneos, considerando para el torneo el mayor *fitness* del vector.

Una vez alcanzada la última generación, y antes de que el algoritmo termine, se buscan los individuos que deben de constituir el clasificador final. Este será construido en base a los individuos que han ganado algún *token*, y que por tanto tienen reglas relevantes para el clasificador, despreciándose el resto de la población. Se añadirá al clasificador una regla por cada *token* ganado, siendo el antecedente de la regla el fenotipo del individuo y el consecuente la etiqueta asociada al *token*.

En la figura 5.3 se observa como un clasificador inducido por GC-1 predice la combinación de etiquetas de un patrón desconocido, utilizando para ello reglas con una sola etiqueta en el consecuente. Para ello se evalúan sobre el patrón desconocido todas las reglas que constituyen el clasificador, y en el momento que una de ellas determina que el patrón está asociado a la etiqueta almacenada en su consecuente, esta etiqueta será devuelta como salida, independientemente del resultado del resto.

5.5.2. Versión GC-k

En el algoritmo 3 se muestra en seudocódigo la versión GC-k que que presenta algunas diferencias significativas respecto a GC-1 que a continuación se comentan.

Cuando la población es inicializada aleatoriamente, GC-k realiza una lectura del conjunto de datos de entrenamiento para buscar todas las combinaciones de etiquetas presentes en el. Después de esto, las acciones que lleva a cabo son similares a las del GC-1, pero considerando como consecuentes las combinaciones de etiquetas presentes en el *dataset* en lugar de las etiquetas de forma individual a la hora de llevar a cabo la evaluación de los individuos, así como la fase en la que se aplica la técnica de nichos.

Algorithm 3 Algoritmo evolutivo. Versión GC-k

Generar población inicial $P(0)$
Leer combinaciones del conj. de entrenamiento C
$g_{count} \leftarrow 0$
while $g_{count} < g_{max}$ **do**
  for all $c_i \in C$ (conjunto de combinaciones) **do**
    for all Individuo Ind_j en $P(g_{count})$ **do**
      Evaluar $Inds_j$ con c_i como consecuente
      Almacenar $fitness_i$ en el vector de *fitness* de $Inds_j$
    end for
  end for
  for all $c_i \in C$ **do**
    $tokens\_totales$ = numero de patrones pertenecientes a c_i
    for all Individuo Ind_j en $P(g_{count})$ **do**
      $tokens\_ganados^{c_i}_{Ind_j} = 0$
    end for
    for all Patrones positivos (tokens) **do**
      Seleccionar Ind_j con mayor *fitness* que clasifica el patrón
      $tokens\_ganados^{c_i}_{Ind_j} + +$
    end for
    for all Individuo Ind_j en $P(g_{count})$ **do**
      Actualizar fitness según formula 5.3
    end for
  end for
  Realizar selección considerando el mayor $fitness$ del vector
  Aplicar operadores de cruce
  Aplicar operadores de mutación y transposición RIS e IS
  Actualizar población
  $g_{count} + +$
end while
for all $Ind_j \in$P(g$_{max}$)(Generar clasificador) **do**
  for all $c_i \in C$ **do**
    if $tokens\_ganados^{c_i}_{Ind_j} > 0$ **then**
      consecuente$_{Ind_j} \leftarrow consecuente_{Ind_j} \cup c_i$
    end if
  end for
  if consecuente$_{Ind_j} \neq \emptyset$ **then**
    clasificador $\leftarrow clasificador +$ Ind$_j$
  end if
end for

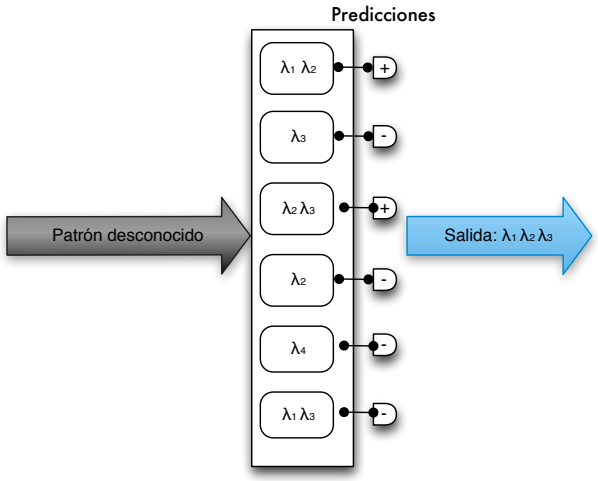


Figura 5.4: Ejemplo de clasificador inducido mediante GC-k

Por otro lado, en el momento en que las iteraciones del algoritmo genético finalizan, de la misma forma que el anterior, GC-k busca los individuos que formarán parte del clasificador aprendido. Solamente aquellos individuos que han ganado algún *token* en alguna combinación de etiquetas resultan relevantes para el clasificador final, pudiendo despreciarse el resto de la población. Para ello se añade al clasificador final una regla por cada *token* ganado, cuyo antecedente, igual que en GC-1, es el propio individuo, pero cuyo consecuente es la combinación de etiquetas asociada con el *token*.

La figura 5.4 muestra como se desarrolla la clasificación de un patrón desconocido en un clasificador inducido mediante la versión GC-k, realizándose en ese caso la predicción de la combinación asociada al patrón a partir de las combinaciones presentes en el *dataset* de entrenamiento. Conviene destacar que el clasificador inducido puede asociar una combinación de etiquetas que no esté presente en el conjunto de datos original, como se aprecia en la figura 5.4 la combinación $(\lambda_1, \lambda_2, \lambda_3)$ no aparece como consecuente en el clasificador inducido, sino que es generada a partir de las combinaciones (λ_1, λ_2) y (λ_2, λ_3).

Población	1000 individuos
Número de Genes	6
Longitud de cabeza	35
Max. generaciones	60
Selección	Torneos tamaño 2
Prob. mutacion	0.2
Prob. mutación constantes	0.2
Prob. cruce	0.7
Prob. transposición IS	0.5
Prob. transposición RIS	0.3
Tamaño fragmento IS	3
Tamaño fragmento RIS	12
Función de conexión	AND

Tabla 5.2: Parámetros del algoritmo GC

5.6. Marco experimental

El objetivo de los experimentos llevados a cabo es doble. Por un lado, se busca contrastar el rendimiento de las dos propuestas GC-1 y GC-k para determinar cual de estos enfoques es mejor. Y por otro, lado se va a comparar la mejor propuesta con otros algoritmos multi-etiqueta descritos en la literatura.

Antes de proceder con el grueso de la ejecución de las pruebas se han llevado a cabo una serie de test con el fin de determinar los parámetros óptimos de ejecución de los algoritmos. Estos parámetros aparecen en la tabla 5.2 y son los que posteriormente se han utilizado en el resto de experimentos llevados a cabo.

El rendimiento del mejor de los algoritmos propuestos se ha comparado con varios métodos de clasificación multi-etiqueta, que abarcan muchos de los paradigmas más ámpliamente utilizados para manejar este tipo de problemas,

entre los que podemos destacar árboles de decisión, kNN, redes neuronales y métodos de *ranking*. La implementación de todos estos algoritmos utilizada es la de la librería MULAN[2].

Estos algoritmos ya se han comentado ámpliamente en el capítulo 2. El método *Binary Relevance (BR)* y *Label Powerset (LP)* son de transformación de problemas, y utilizan como algoritmo de clasificación base el algoritmo basado en árboles C4.5 en su implementación J48 de *Weka* [26]. El algoritmo RAkEL es un método de multiclasificación que integra las respuestas de varios clasificadores multi-etiqueta, cada uno de ellos entrenado con un subconjunto de etiquetas del conjunto original [26], y se ha utilizado en su configuración original, es decir, utilizando LP con C4.5 como clasificador base, un tamaño de subconjunto igual a 3, un número de modelos igual al doble del número de etiquetas y 0.5 como valor umbral.

Por otro lado, el algoritmo ML-kNN es una adaptación del popular algoritmo de los k vecinos más cercanos al problema de la clasificación multi-etiqueta [67] y se ha utiliada con un número de vecinos igual a 10 y un factor de *smoothing* igual a 1, como se recomienda en [7]. BPML es un algoritmo de redes neuronales multi-etiqueta [1] y se ha utilizado con un valor 0.05 de ratio de aprendizaje, 100 épocas y un número de unidades ocultas igual al 20 % de las unidades de entrada, configuración recomendada en [289]. Por último *Calibrated Label Ranking (CLR)* [33] es una propuesta de *ranking* específicamente adaptada para atrabajar con problemas multi-etiqueta y utiliza como clasificador base también C4.5 en su versión J48.

Todos estos algoritmos han sido considerados contra la mejor versión de las dos propuestas. Además, los algoritmos han sido probados y comparados con la propuesta original de la GEPCLASS, sin ningún tipo de adaptación multi-etiqueta, pero aplicando sobre los datos las trasformaciones BR y LP

[2]Están disponibles en línea en `http://mulan.sourceforge.com`

antes de proceder a la ejecución del algoritmo, a las que se ha denominado respectivamente BR_{GCL} y LP_{GCL}.

Además se ha hecho una revisión cualitativa de los clasificadores inducidos por GC-1 y GC-k, con el objetivo de determinar cualitativamente la calidad del conocimiento obtenido por estos.

Por último también se ha comparado el algoritmo aquí propuesto, con el algoritmo basado en funciones discriminantes mostrado en el capítulo 4, tanto utilizando datos categóritos como numéricos con el propósito de determinar qué propuesta es más adecuada para cada tipo de datos.

En cuanto a las métricas utilizadas para medir el rendimiento de los diferentes algoritmos multi-etiqueta probados, se han utilizado las comentadas en el capítulo 3 de *precision* y *recall*, en su versión *micro* (Ecuación 2.2) además de la *accuracy* (Ecuación 2.6) y *Hamming loss* (Ecuación 2.8).

Referente a los conjuntos de datos se han utilizado diez de los expuestos en el capítulo 2 para llevar a cabo la experimentación: *bibtext* [209], *rcv1* [109], *scene* [8], *emotions* [210], *yeast* [6], *genbase* [130], *mediamill* [212], *TMC2007* [211], *enron* [115] y *medical* [187].

Todos los algoritmos se han probado sobre los conjuntos descritos utilizando una validación cruzada con $k = 10$ (10-*fold cross validation*).

5.7. Resultados experimentales y discusión

En la presente sección se presenta el rendimiento de la propuesta que en este capítulo se hace. En primer lugar, se va a determinar si hay alguna diferencia en el rendimiento entre las dos versiones del algoritmo propuesto, GC-1 y GC-k, y posteriormente la mejor de estas propuestas va a ser comparada con los algoritmos anteriormente citados, para pasar a comentar las reglas obtenidas por el algoritmo. Se finaliza esta sección comparando

	Hamming loss		*Accuracy*		*Precision*		*Recall*	
	GC-1	GC-k	GC-1	GC-k	GC-1	GC-k	GC-1	GC-k
Mediamill	0.017	**0.016**	0.504	**0.537**	0.593	**0.611**	0.531	**0.583**
Yeast	0.158	**0.149**	0.438	**0.509**	**0.702**	0.683	0.462	**0.632**
Enron	0.040	**0.038**	0.534	**0.613**	0.689	**0.702**	0.593	**0.652**
Rcv1	0.039	**0.031**	0.400	**0.413**	0.418	**0.425**	0.391	**0.410**
Bibtext	0.024	**0.022**	0.335	**0.362**	**0.442**	0.432	0.379	**0.391**
TMC 2007	0.045	**0.031**	0.598	**0.650**	0.676	**0.692**	0.701	**0.729**
Emotions	0.183	**0.182**	0.562	**0.569**	0.691	**0.696**	0.679	**0.686**
Genbase	**0.001**	**0.001**	0.986	**0.986**	0.990	**0.991**	0.977	**0.987**
Medical	**0.009**	**0.009**	0.750	**0.792**	**0.853**	0.849	0.796	**0.817**
Scene	0.101	**0.100**	0.682	**0.691**	0.771	**0.774**	0.683	**0.684**

Tabla 5.3: Resultados de GC-1 y GC-k

el rendimiento de esta propuesta con la hecha en el capítulo 4 basada en funciones discriminantes.

5.7.1. Comparativa entre GC-1 y GC-k

Las dos propuestas GC-1 y GC-k han sido comparadas utilizando para ello un test estadístico de rangos con signo de Wilcoxon [291]. Este test es no paramétrico y está especialmente indicado para determinar si dos propuestas probadas sobre varios conjuntos de datos son significativamente diferentes o no. Para ello, la hipótesis nula del test estadístico presupone que no hay diferencias significativas entre los valores de rendimiento de las dos propuestas comparadas, y por contra, la hipótesis alternativa supone que sí los hay.

La tabla 5.3 muestra los resultados de rendimiento para las dos propuestas, utilizando para ello las métricas que se expusieron en el capítulo 2 de *Hamming loss*, *accuracy*, *precision* y *recall*. Hay que tener en cuenta que *Hamming loss* mide la diferencia entre los conjuntos de etiquetas predichos

Métrica	R+	R-	*p-value*
Hamming loss	53.5	1.5	0.004883
Accuracy	45.0	0.0	0.003906
Precision	36.0	19.0	$\leq$0.02
Recall	55.0	0.0	0.001953

Tabla 5.4: Resultados del test de Wilcoxon, GC-k comparado GC-1

y real (Ecuación 2.8), y valores menores indican un mejor rendimiento. Por otro lado para las métricas de *accuracy*, *precision* y *recall* el rendimiento es máximo conforme aumenta el valor.

La tabla 5.4 muestra los rangos y los *p-valores* de las dos alternativas, para las cuatro métricas. R+ Es la suma de rangos (diferencias entre los dos valores de la métrica) en los que el segundo algoritmo (GC-k) gana al primero (GC-1) y R- la suma de rangos en los que ocurre lo contrario. De esta tabla podemos observar que el algoritmo GC-1 obtiene una puntuación media menor para todas las métricas que el GC-k, lo que nos dice que a priori, el rendimiento promedio de GC-k es mayor que el de GC-1. Esto puede ser debido al hecho que GC-k considera las combinaciones de etiquetas como consecuente, y por tanto permite aprender con mayor facilidad las relaciones entre etiquetas.

De acuerdo a los *p-valores* obtenidos tras la aplicación del test estadístico, se concluye que existen diferencias significativas entre las propuestas GC-1 y GC-k en términos de *Hamming loss*, *precision* y *recall*, con un nivel de significancia del 95 % (*p-value*$\leq$ 0.005). Conviene destacar que aunque no haya diferencias significativas en términos de *precision*, el valor de rango favorece GC-k, y por otro lado GC-k supera en *precision*, a GC-1 en siete de los diez conjuntos de datos sobre los que se ha probado.

A la luz de estos resultados, podemos concluir que la propuesta GC-k obtiene mejores resultados que GC-1 para las métricas y los conjuntos de datos

	BPML	CLR	ML-kNN	BR$_{GCL}$	LP$_{GCL}$	BR$_{C4.5}$	LP$_{C4.5}$	RAkEL	GC-k
Mediamill	0.043	0.057	0.018	0.031	0.083	0.047	0.024	0.047	**0.016**
Yeast	0.228	0.195	0.222	0.157	0.169	0.152	0.171	0.226	**0.149**
Enron	0.250	0.052	0.047	0.050	0.054	0.042	0.048	0.041	**0.038**
Rcv1	0.189	0.097	0.088	0.045	0.027	0.056	0.069	**0.024**	0.031
Bibtext	0.026	0.026	0.030	0.032	0.092	0.095	0.025	0.028	**0.022**
TMC 2007	0.063	0.074	0.075	0.078	0.099	0.084	0.076	0.033	**0.031**
Emotions	0.207	0.242	0.262	0.239	0.269	0.238	0.201	0.185	**0.182**
Genbase	0.266	**0.001**	0.005	**0.001**	**0.001**	**0.001**	0.002	**0.001**	**0.001**
Medical	0.312	0.015	0.010	0.011	0.012	0.010	0.010	**0.009**	**0.009**
Scene	0.141	0.138	**0.085**	0.156	0.127	0.137	0.143	0.102	0.100

Tabla 5.5: Resultados para *Hamming loss*

que se han considerado en la fase experimental, siendo por tanto un mejor enfoque para abordar problemas de tipo multi-etiqueta. A continuación compararemos la mejor de nuestras propuestas, GC-k, con otros algoritmos de los presentes en la literatura.

5.7.2. Comparativa GC-k con el estado del arte

En esta sección se compara GC-k con un conjunto de algoritmos de los descritos en el capítulo 2. El algoritmo propuesto obtiene, en general, mejores resultados que el resto de algoritmos para el conjunto de todos los *datasets*, en términos de de *Hamming loss*, *accuracy*, *precision* y *recall*.

La tabla 5.5 nos muestra los resultados experimentales para la métrica de *Hamming loss*, que como ya se ha comentado, es una medida que tiende a cero conforme el conjunto de etiquetas predicha y real es más similar. El algoritmo GC-k obtiene el valor mínimo de *Hamming loss* para 8 de los 10 conjuntos de datos estudiados, empatando en valor en los conjuntos *genbase* y *medical*. Esto ha de considerarse como un resultado bastante positivo ya

	BPML	CLR	ML-kNN	BR$_{GCL}$	LP$_{GCL}$	BR$_{C4.5}$	LP$_{C4.5}$	RAkEL	GC-k
Mediamill	0.512	0.592	**0.634**	0.404	0.555	0.416	0.485	0.457	0.537
Yeast	**0.515**	0.469	0.508	0.431	0.455	0.406	0.347	0.444	0.509
Enron	0.179	0.423	0.307	0.523	0.553	0.469	0.403	0.526	**0.613**
Rcv1	0.072	0.328	0.213	**0.443**	0.342	0.379	0.298	0.399	0.413
Bibtext	**0.474**	0.342	0.214	0.389	0.269	0.385	0.301	0.323	0.362
TMC 2007	**0.748**	0.747	0.688	0.472	0.516	0.541	0.525	0.587	0.650
Emotions	0.549	0.477	0.341	0.422	0.524	0.462	0.438	0.509	**0.569**
Genbase	0.043	**0.986**	0.941	0.822	0.970	0.985	0.983	**0.986**	**0.986**
Medical	0.068	0.731	0.565	0.693	0.687	0.744	0.735	0.733	**0.792**
Scene	0.499	0.527	0.673	0.662	**0.710**	0.536	0.588	0.622	0.691

Tabla 5.6: Resultados de *accuracy*

que esta métrica está especialmente diseñada para medir el resultado de clasificadores multi-etiqueta, tomado en consideración tanto los errores por omisión como los errores de asignación incorrecta de etiquetas.

Con el conjunto de datos genbase ocurre que varios algoritmos como CLR, BR$_{C4.5}$, RAkEL, LP$_{GCL}$ y nuestra propuesta, GC-k, obtienen el mismo valor para la métrica (0.001) y de hecho el resto de algoritmos obtienen un valor casi idéntico (0.002). Este fenómeno ocurre también para el resto de métricas, lo que lleva a concluir que este conjunto de datos es bastante fácil de aprender.

Los resultados en términos de *accuracy* aparecen detallados en la tabla 5.6 y en ello se puede observar que la propuesta GC-k gana en cuatro de los *datasets*, teniendo valores muy próximos al mejor en el resto. Aunque no se comporta mejor que otras propuestas en la mayoría de los conjuntos de datos, hay que mencionar que no hay ningún otro algoritmo que obtenga de manera clara y consistente mejores resultados que GC-k para todos y cada uno de los conjuntos de datos experimentales. El comportamiento para *precision* es similar, y puede verse en la tabla 5.7.

	BPML	CLR	ML-kNN	BR$_{GCL}$	LP$_{GCL}$	BR$_{C4.5}$	LP$_{C4.5}$	RAkEL	GC-k
Mediamill	0.472	0.540	**0.698**	0.579	0.659	0.487	0.584	0.524	0.611
Yeast	0.604	0.652	**0.729**	0.598	0.531	0.621	0.441	0.524	0.683
Enron	0.184	0.677	0.660	0.593	0.602	0.483	0.429	0.614	**0.702**
Rcv1	0.046	0.256	0.159	0.403	0.363	0.388	0.286	0.412	**0.425**
Bibtext	**0.485**	0.257	0.153	0.415	0.373	0.379	0.277	0.432	0.432
TMC 2007	**0.757**	0.710	0.584	0.681	0.654	0.642	0.545	0.687	0.692
Emotions	0.661	0.604	0.639	0.652	0.559	0.606	0.557	0.649	**0.696**
Genbase	0.010	**0.992**	0.989	0.923	0.983	0.980	0.974	0.989	0.991
Medical	0.069	0.836	0.812	0.755	0.803	0.802	0.771	0.823	**0.849**
Scene	0.597	0.606	0.820	0.729	0.707	0.606	0.590	0.705	**0.774**

Tabla 5.7: Resultados en *precision*

En la tabla 5.8 se pueden observar los resultados para la métrica *recall*. Estos resultados muestran que GC-k es la mejor propuesta para la mayoría de los conjuntos de datos, lo cual es significativo en el sentido que de dicha métrica indica el porcentaje de etiquetas relevantes recuperadas por el clasificador.

Los resultados obtenidos por la GEPCLASS combinada con BR y LP se muestran en las columnas BR$_{GCL}$ y LP$_{GCL}$. Estos resultados son similares a los que se obtiene con la misma trasformación previa a la utilización de C4.5. Sin embargo los dos algoritmos multi-etiqueta aquí presentados logran mejores resultados para cada métrica y conjunto de datos demostrando que la propuesta de adaptación aquí hecha obtiene un rendimiento superior a la propia GEPCLASS sin adaptación.

5.7.3. Análisis estadístico

Los algoritmos han sido estadísticamente comparados mediante el test de Friedman [291; 292], que es un test no paramétrico que compara los valores de *ranking* medios de los algoritmos que estudia. Los valores de *ranking* para

	BPML	CLR	ML-kNN	BR$_{GCL}$	LP$_{GCL}$	BR$_{C4.5}$	LP$_{C4.5}$	RAkEL	GC-k
Mediamill	0.503	**0.592**	0.503	0.458	0.475	0.431	0.511	0.472	0.583
Yeast	**0.698**	0.584	0.570	0.424	0.537	0.479	0.428	0.628	0.632
Enron	**0.809**	0.489	0.366	0.436	0.43	0.549	0.408	0.531	0.652
Rcv1	0.132	**0.437**	0.278	0.335	0.303	0.365	0.301	0.372	0.410
Bibtext	0.432	**0.456**	0.317	0.303	0.264	0.412	0.318	0.373	0.391
TMC 2007	0.679	0.698	0.693	0.501	0.513	0.632	0.629	0.602	**0.729**
Emotions	**0.691**	0.657	0.372	0.481	0.609	0.600	0.544	0.614	0.686
Genbase	0.341	0.983	0.901	0.886	0.959	0.946	0.897	0.983	**0.987**
Medical	0.688	0.778	0.573	0.692	0.787	0.782	0.740	0.763	**0.817**
Scene	0.670	0.652	0.672	0.498	0.565	0.624	0.592	0.645	**0.684**

Tabla 5.8: Resultados de *recall*

Algoritmo	*Hamming loss*	*Accuracy*	*Precision*	*Recall*
BPML	6.75	5.50	6.20	4.25
LP$_{GCL}$	6.35	4.80	5.30	6.10
CLR	6.10	4.50	4.45	2.75
BR$_{GCL}$	5.55	5.80	5.10	7.70
ML-kNN	5.20	5.90	4.35	6.25
BR$_{C4.5}$	5.10	5.30	5.95	5.10
LP$_{C4.5}$	4.90	6.40	7.50	6.50
RAkEL	3.45	4.50	4.30	4.55
GC-k	1.60	2.30	1.85	1.80

Tabla 5.9: *Ranking* medio de los algoritmos

los algoritmos que se han obtenido en el test de Friedman se detallan en la tabla 5.9.

Los valores del estadístico de Friedman, que siguen una distribución de tipo χ^2 con ocho grados de libertad, tantos como algoritmos probados menos uno, se pueden observar en la tabla 5.10. De acuerdo con los valores obtenidos se puede rechazar la hipótesis nula (todos los algoritmos tienen igual rendimiento), con una confianza del 95 %, para los resultados de *Hamming loss*,

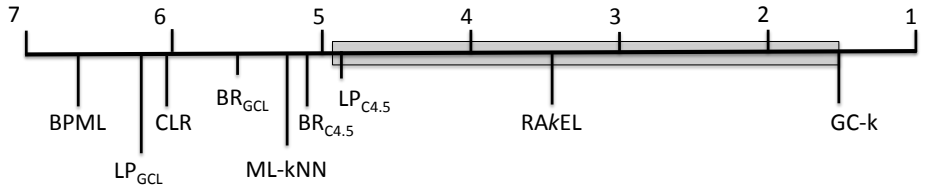
Métrica	Estad. Friedman $\chi^2(p = 0.05)$		Conclusión
Hamming loss	27.23	15.51	Rechaza Hipótesis nula
Accuracy	15.44		Acepta Hipótesis nula
Precision	26.44		Rechaza Hipótesis nula
Recall	37.85		Rechaza Hipótesis nula

Tabla 5.10: Resultados del test de Friedman ($\alpha = 0.05$) para GC

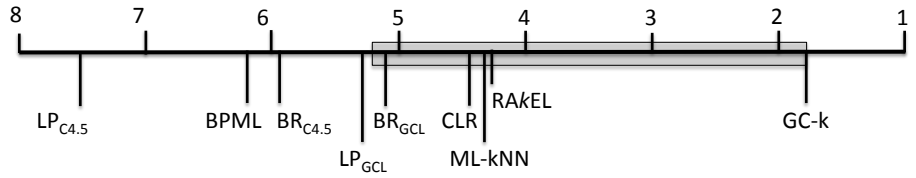
precision y *recall*. Conviene resaltar que la propuesta GC-k obtiene siempre la mejor puntuación de *ranking* para todas las métricas y valores. Para la *accuracy*, aunque no se puede rechazar la hipótesis nula, conviene destacar que nuestro algoritmo obtiene los mejores resultados de *ranking*.

Para determinar que algoritmos presentan diferencias significativas es necesario realizar un post-test. En el caso de los valores de *accuracy* no es necesario ya que ninguno de los algoritmos es significativamente mejor que el resto según el test. El valor de diferencia crítico para el post-test de Bonferroni-Dunn considerando $\alpha = 0.05$ es 3.33. Es decir, que la diferencia en *ranking* entre los dos algoritmos debe ser mayor que este valor para considerarlos significativamente diferentes.

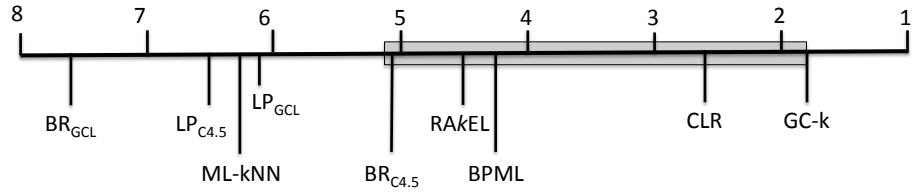
La figura 5.5a muestra la aplicación de dicho test para la métrica *Hamming loss*. En dicha figura se representan los *ranking* obtenidos por los diferentes algoritmos, así como el intervalo en el que no se consideran significativamente distintos a GC-k. Si al mejor algoritmo (el que aparece mas cerca de uno) se le añade la diferencia crítica, obtenemos una barra horizontal que nos define un umbral. Todos los algoritmos fuera de este umbral son significativamente peores que el mejor. En esta figura se puede observar que solo hay dos algoritmos que no son significativamente peores que GC-k: $\text{LP}_{C4.5}$ y RAkEL. No obstante hay que recalcar la gran diferencia que se puede observar entre nuestra propuesta y la segunda en el *ranking*. También destacar que LP_{GCL} y BR_{GCL} quedan fuera del umbral.



(a) Hamming loss



(b) Precision



(c) Recall

Figura 5.5: Test de Bonferroni-Dunn ($\alpha = 0.05$)

En la figura 5.5b se muestra la aplicación del test con respecto a la medida de *precision*. En este caso el umbral para considerar dos propuestas significativamente distintas es 5.18. Aquí se observa que hay más algoritmos que no exceden el umbral, sin embargo de nuevo la propuesta aquí hecha obtiene el mejor valor de *ranking*, además de una considerable diferencia con la segunda mejor propuesta. De nuevo LP_{GCL} y BR_{GCL} quedan fuera del umbral.

Por ultimo la figura 5.5c muestra los resultados para *recall*, pudiéndose observar el mismo comportamiento para nuestro algoritmo que el comentado en las figuras anteriores. De nuevo LP_{GCL} y BR_{GCL} se muestra como significativamente peores.

Algoritmo	*Hamming loss*	*Precision*	*Recall*
BPML	+	+	-
LP_{GCL}	+	+	+
CLR	+	-	-
BR_{GCL}	+	-	+
ML-kNN	+	-	+
$\text{BR}_{C4.5}$	+	+	-
$\text{LP}_{C4.5}$	-	+	+
RAkEL	-	-	-

Tabla 5.11: Test de Bonferroni-Dunn ($\alpha = 0.05$)

En la tabla 5.11 se hace un resumen de los resultados obtenidos por los algoritmos para las tres métricas. Las celdas marcadas con un + indican que hay diferencias significativas y las marcadas con un - que no las hay con respecto a GC-k. De ahí podemos observar que hay diferencias significativas entre nuestra propuesta y las basadas en GEPCLASS tras aplicarle una trasformación de etiquetas, de tal forma que se puede concluir que nuestro algoritmo mejora los algoritmos basados en técnicas de una sola etiqueta combinadas con una transformación. Además también destacar que RAkEL, el único que no es significativamente peor en todas las métricas, obtiene una puntuación de *ranking* inferior en todas ellas.

5.7.4. Discusión sobre el conocimiento obtenido

La propuesta GC es capaz de obtener un modelo consistente en un conjunto de reglas de clasificación. Las reglas de clasificación se suelen utilizar una vez obtenidas en sistemas de apoyo a la decisión (*Decision Support System, DSS*). Por tanto el modelo es adecuado para obtener información representativa sobre el conocimiento descubierto que puede ser utilizado por expertos humanos para incrementar su comprensión del dominio del problema, ya

que las reglas pueden ayudar a determinar qué atributos e intervalos son relevantes para clasificar nuevos patrones.

Los siguientes ejemplos incluyen una selección de varias reglas, obtenidas de conjuntos de datos numéricos y categóricos tanto por la propuesta GC-1 como por la propuesta GC-k

Las siguientes reglas han sido obtenidas del conjunto *scene* utilizando GC-1. El *dataset* contiene información sobre fotografías clasificadas según el tipo de escena que representan. Cada imagen se parte en 49 rectángulos y para cada uno se determina el histograma de 6 colores. Se puede observar del antecedente obtenido qué rectángulos y qué colores son importantes para determinar una clase. Por ejemplo el atributo 230 se refiere al rectángulo 38 y al histograma 2. Este antecedente solamente tiene en cuenta un pequeño subconjunto de los 294 posibles atributos, y devuelve como resultado 2 reglas, debido al hecho de que el individuo que la representa ganó *tokens* durante la aplicación de la técnica de nichos asociados a las etiquetas *Beach* y *Field* respectivamente.

IF$((((Attr_{230} = 0.02) AND (Attr_{254} > 0.39)) OR ((Attr_{108} > 0.95) AND$
$(Attr_{218} > 0.78))) OR (NOT((Attr_{217} > 0.47) OR (Attr_{284}! = 0.9)))) AND$
$(((NOT(Attr_{177} > 0.45)) OR (Attr_{147}! = 0.66)) OR ((Attr_{231} > 0.61) AND$
$(Attr_{214} < 0.48)))$
THEN $Label = Beach$
THEN $Label = Field$

En la propuesta GC-k las reglas participan en un *token competition* por cada conjunto de etiquetas presente en el *dataset*. Las siguientes reglas han sido obtenidas por GC-k también del conjunto *scene*, y el antecedente muestra las mismas características que el anterior. Sin embargo el consecuente está asociado a un conjunto de etiquetas. En el ejemplo, el nuevo patrón que cumpla las reglas será asociado con las categorías *sunset*, *fall foliage* y *field*.

IF$(NOT((NOT(((Attr_{264} = 0.55)OR(Attr_{41}! = 0.09))AND((Attr_{204} < 0.93)OR$
$(Attr_{91} = 0.69))))OR(NOT(Attr_{240} = 0.5)OR(Attr_{141} < 0.65)))))AND$
$((Attr_{255}! = 0.05)AND(Attr_{253} < 0.07))$
THEN$Labelset = Sunset, fallfoliage$
THEN$Labelset = Field, fallfoliage$

Las últimas reglas mostradas han sido obtenidas del *dataset genbase* utilizando la propuesta GC-1. Este conjunto almacena información sobre proteínas y las características que exhiben estas. Es booleano ya que para cada proteína se indica si tiene una determinada característica (YES) o no la tiene (NO). En este caso el antecedente solamente contiene dos funciones relacionales (igual y distinto) pero el consecuente muestra las mismas características que los anteriores. En este caso las proteínas asociadas a la regla se clasificarán como pertenecientes a las clases 1, 16 y 21.

IF$((((Attr_{582}! = NO)AND(Attr_{1008} = YES))AND((Attr_{304} = NO)$
$OR(Attr_{100} = NO)))AND((NOT(Attr_{228} = YES))AND$
$(NOT(Attr_{272} = YES))))AND(((Attr_{673} = YES)OR(Attr_{928}! = YES))$
$AND((Attr_{922} = NO)OR(Attr_{1093} = NO)))$
THEN$Label = class_1$
THEN$Label = class_{16}$
THEN$Label = class_{21}$

5.7.5. Comparación entre GC-k y GEP-MLC

La tabla 5.12 muestra una comparativa en términos de *accuracy, precision* y *recall* de la propuesta aquí hecha, GC-k, con el algoritmo que se propuso el el capítulo 4 basado en funciones discriminantes. Como ya se concluyó en dicho capítulo el modelo GEP-MLC es bastante robusto, pero no es muy adecuado para tratar con datos nominales.

	Accuracy		*Precision*		*Recall*	
	GEP-MLC	GC-k	GEP-MLC	GC-k	GEP-MLC	GC-k
Mediamill	**0.703**	0.537	**0.669**	0.611	0.581	**0.583**
Yeast	**0.738**	0.509	**0.715**	0.683	**0.749**	0.632
TMC 2007	0.543	**0.650**	0.618	**0.692**	0.540	**0.729**
Emotions	**0.755**	0.569	**0.724**	0.696	**0.695**	0.686
Genbase	0.903	**0.986**	0.650	**0.991**	0.977	**0.987**
Scene	**0.709**	0.691	0.746	**0.774**	**0.744**	0.684

Tabla 5.12: Resultados comparados de GEP-MLC y GC-k

Los datos experimentales mostrados reafirman las conclusiones a las que se llegó en el capítulo 4 ya que el algoritmo GEP-MLC exhibe un comportamiento mucho mejor para todas las métricas cuando el problema trata con datos numéricos, como en los conjuntos *mediamill, yeast, emotions* y *scene*. Sin embargo, cuando se trata de conjuntos categóricos el algoritmo GC-k obtiene mejores resultados (*datasets TMC 2007* y *genbase*).

5.8. Conclusiones

Este capítulo nos ha permitido mostrar una propuesta evolutiva basadas en *Gene Expression Programing*, concretamente en GEPCLASS. La propuesta se presenta en dos versiones, GC-1 y GC-k. Ambas versiones del algoritmo codifican reglas de clasificación en cada individuo, permitiendo la primera reglas con una sola etiqueta en el consecuente y la segunda reglas con más de una etiqueta en el consecuente.

El clasificador resultante del proceso de aprendizaje se construye combinando varias de las reglas presentes en la población final y además, ambos algoritmos utilizan una técnica de nichos multi-etiqueta para asegurar diversidad en la población. Ambas versiones pueden predecir nuevas combinaciones de

etiquetas para patrones desconocidos a partir de las etiquetas presentes en el problema, lo que es un valor añadido frente a los algoritmos que solo pueden predecir combinaciones de las presentes en el conjunto de entrenamiento.

Los estudios que se han llevado a cabo para medir el rendimiento de GC-1 y GC-k indican que la mejor aproximación al problema es el utilizar reglas de clasificación con más de una etiqueta en el consecuente en la mayoría de los casos.

Además los estudios realizados para medir el rendimiento de la propuesta hecha nos muestra que es en general mejor que otras propuestas multi-etiqueta, tanto basadas en transformaciones como métodos de adaptación, de *ranking* o multiclasificadores. Se ha mejorado los resultados obtenidos por las trasformaciones LP y BR combinada con GEPCLASS mediante un enfoque multi-etiqueta directo.

El algoritmo propuesto en este capítulo además se ha confirmado como más adecuado que el enfoque basado en funciones discriminantes expuesto en el capítulo 4 para tratar con conjuntos de datos categóricos.

Por otro lado, las reglas obtenidas por nuestra propuesta son interpretables como conocimiento por un experto en el dominio del problema con el que se esté trabajando.

Todo ello nos lleva a conformar que nuestra propuesta es una técnica más fiable que las propuestas en la literatura con las que se ha comparado, especialmente adecuada para conjuntos categóricos o cuando se quiera obtener información interpretable sobre el dominio de un problema determinado.

6

Conclusiones

En este último capítulo se hace un breve resumen de los resultados alcanzados en la presente tesis, de las conclusiones a las que se ha llegado desde el punto de partida y tras el desarrollo de los distintos modelos expuestos en los capítulos anteriores. Posteriormente se detallan las publicaciones asociadas a este trabajo y se termina con un resumen de las principales líneas de trabajo por las que se continuará en el futuro.

6.1. Resultados obtenidos

En la presente tesis se han propuesto dos modelos de programación genética para solventar problemas de clasificación multi-etiqueta.

Una primera aproximación al problema ha consistido en el uso de funciones discriminantes para determinar si un patrón pertenece o no a cada una de las etiquetas del problema, y se ha desarrollado en el algoritmo GEP-MLC. Una segunda aproximación más elaborada en el aspecto de la comprensibilidad, así como en el manejo de conjuntos de datos categóricos es utilizar reglas de clasificación. Dentro de las reglas de clasificación se han desarrollado dos modelos, el denominado GC-1 en el que las reglas solamente tienen asociada

una etiqueta en el consecuente, y el modelo GC-k, que comparte rasgos similares con el anterior pero en el que las reglas pueden estar asociadas a más de una etiqueta en su consecuente.

Todos estos modelos han sido probados con varios de los conjuntos de datos que se describen en la literatura y su rendimiento de ha comparado con varios algoritmos de referencia en MLC utilizando métricas ámpliamente aceptadas. En las siguientes secciones se detallan las conclusiones a las que se ha llegado con cada uno de los modelos expuestos en esta memoria.

6.1.1. Algoritmo GEP-MLC: Funciones discriminantes para clasificación multi-etiqueta

En el capítulo 4 se ha presentado el algoritmo GEP-MLC, un algoritmo evolutivo para clasificación multi-etiqueta, el cual, basado en el paradigma de la *Gene Expression Programming*, codifica funciones discriminantes en el individuo, las cuales sirven para indicar si un patrón pertenece o no a una clase determinada, y construye el clasificador final combinando varias de las funciones presentes en la población.

El algoritmo propuesto utiliza una novedosa técnica de nichos de tal manera que se constituyen nichos de individuos en la población especializados en clasificar subconjuntos de patrones para una determinada clase.

Los estudios que se han llevado a cabo para medir el rendimiento muestran que este algoritmo presenta un mejor comportamiento respecto a los que realizan transformaciones del conjunto de entrada como BR, LP o CLR. Además se han encontrado evidencias estadísticas de la propuesta realizadas es mejor que otros algoritmos de adaptación como la implementación multi-etiqueta del algoritmo de los k vecinos más cercanos o el algoritmo basado en redes neuronales BPML.

Por ultimo también reseñar que los experimentos a los que se ha sometido GEP-MLC también indican que obtiene de manera consistente los mejores resultados, siendo más adecuada para trabajar con datos numéricos.

6.1.2. Algoritmos GC: Reglas de clasificación multi-etiqueta

El capítulo 5 sirve para exponer una propuesta evolutiva también basadas en GEPCLASS. El algoritmo se presenta en dos versiones, denominadas GC-1 y GC-k. Ambas versiones codifican reglas de clasificación en cada individuo, permitiendo GC-1 reglas con una sola etiqueta en el consecuente y GC-k reglas con más de una etiqueta en el consecuente.

El clasificador resultante del proceso de aprendizaje se construye combinando varias de las reglas presentes en la población final y además, ambos algoritmos utilizan una técnica de nichos para asegurar diversidad en la población y que haya reglas que representen las consecuentes más relevantes que existan en el conjunto de datos de entrenamiento.

Los estudios llevados a cabo para medir el rendimiento de ambos algoritmos propuestos indican que la mejor aproximación al problema es el utilizar reglas de clasificación con más de una etiqueta en el consecuente en la mayoría de los casos. Además los estudios realizados para medir el rendimiento de la propuesta hecha nos muestra que es en general mejor que otras propuestas multi-etiqueta, tanto basadas en transformaciones como métodos de adaptación, de *ranking* o multiclasificadores. Se han mejorado los resultados obtenidos por las trasformaciones LP y BR combinada con GEPCLASS mediante un enfoque multi-etiqueta directo.

Esta técnica además se ha confirmado como más adecuada que el enfoque GEP-MLC para tratar con conjuntos de datos categóricos.

Finalmente, las reglas obtenidas por nuestra propuesta son interpretables como conocimiento por un experto en el dominio del problema en el que se esté trabajando.

Todo ello nos lleva a conformar que nuestra propuesta es una técnica más fiable que las propuestas en la literatura con las que se ha comparado, especialmente adecuada para conjuntos categóricos o cuando se quiera obtener información interpretable sobre el dominio de un problema determinado.

6.2. Artículos asociados a la presente tesis

6.2.1. Publicados en revistas

- Ávila-Jiménez, J. L., Gibaja, E. L., Zafra, A. and Ventura, S. (2011). A niching algorithm to learn discriminant functions with multi-label patterns. Journal of multiple-valued logic and soft computing, 1-10.
- Ávila-Jiménez, J. L., Gibaja, E. L., and Ventura, S. (2013). An evolutionary proposal to generate multi-label classification rules. Knowledge Based System. (Enviado)

6.2.2. Publicados en congresos internacionales

- Ávila-Jiménez, J. L., Gibaja, E. L. and Ventura, S (2010). Evolving Multi-label Classification Rules with Gene Expression Programming: A Preliminary Study. En Proceedings of th 5th International Conference on Hybrid Artificial Intelligence Systems (HAIS'10), páginas 9-16.
- Ávila-Jiménez, J. L., Gibaja, E. L., Zafra, A. and Ventura, S (2009). A Niching Algorithm to Learn Discriminant Functions with Multi-Label

Patterns. En En Proceedings of th 10th Inteligent Data Engineering and Automated Learning (IDEAL'09), páginas 570-577. Springer.

- Ávila-Jiménez, J. L., Gibaja, E. L. and Ventura, S (2009). Multi-label Classification with Gene Expression Programming. En En Proceedings of th 4th International Conference on Hybrid Artificial Intelligence Systems (HAIS'09), páginas 629-637. Springer.

6.3. Trabajos futuros

Para cerrar la presente memoria en esta sección se hace un esbozo de las futuras líneas por las que se seguirá trabajando a partir de los resultados obtenidos en esta tesis.

- **Rebalanceo de conjuntos de datos:** Los conjuntos de datos multi-etiqueta suelen estar poco balanceados en el sentido de que hay disparidad en el número de patrones asociados a cada etiqueta. Si se examinan las combinaciones de etiquetas el desbalanceo de los conjuntos puede ser considerable. Por este motivo es posible que el rendimiento de los algoritmos multi-etiqueta pueda ser mejorado sustancialmente aplicando algunas de las técnicas de rebalanceo de datos ya usadas en clasificación clásica, pero adaptadas a las características de la clasificación multi-etiqueta.

- **Clasificación multi-etiqueta multi-instancia:** Como se expuso en el capítulo 2 la clasificación multi-etiqueta y la clasificación multi-instancia guardan ciertas similitudes. Se pueden plantear soluciones a problemas en los que haya bolsas de patrones asociadas a múltiples etiquetas simultáneamente, que apliquen las técnicas aquí expuestas adaptadas a las características de estos problemas.

- **Reducción de la dimensionalidad de datos:** Muchos de los conjuntos de datos que se utilizan en la experimentación multi-etiqueta tienen un peso bastante elevado tanto en número de atributos como en número de instancias. La aplicación de métodos de reducción de la dimensionalidad del problema, sea mediante selección de características o mediante selección de patrones, pueden permitir disminuir el coste computacional de la ejecución de los modelos aquí propuestos sin penalizar el rendimiento de los algoritmos.

- ***Ranking* de etiquetas:** Muchas propuestas para clasificación multi-etiqueta utilizan como base algoritmos de *ranking*. Por esta razón puede considerarse indicado plantear un algoritmo basado en programación genética que induzca un clasificador basado en *ranking* que posteriormente se utilice para resolvar problemas multi-etiqueta.

A

Detalles de implementación

A.1. Introducción

La codificación de los algoritmos diseñados en esta tesis se ha llevado a cabo utilizando el lenguaje de programación Java, orientado a objetos y que permite la ejecución en cualquier plataforma para la que se haya desarrollado una implementación de la *Java Virtual Machine*. El desarrollo de nuestros algoritmos se ha realizado utilizando como marco de trabajo la librería JCLEC [293], especialmente diseñada para desarrollar aplicaciones de computación evolutiva.

El resto de este capítulo está estructurado de la siguiente manera, en primer lugar se comentan las principales características de la librería JCLEC, para pasar a comentar los principales rasgos de la implementación base del paradigma GEP. Posteriormente se detalla la arquitectura de la implementación de los algoritmos propuestos en esta tesis, GEP-MLC y GC.

A.2. Librería JCLEC

JCLEC es un sistema software que se puede utilizar como entorno de trabajo para el desarrollo de aplicaciones de computación evolutiva implementadas en Java. La librería provee al programador de un conjunto de clases genéricas que abstraen las principales características presentes en los algoritmos evolutivos y permite el desarrollo rápido de aplicaciones para resolver muchos tipos de problemas sobre casi todos los paradigmas evolutivos.

El núcleo de la librería JCLEC está formado por un conjunto de interfaces que muestran el comportamiento que debe tener cualquier algoritmo evolutivo genérico (ver figura A.1). El objetivo de esta arquitectura es tener, por un lado una clase que nos represente el algoritmo genético, representada por la interfaz *IAlgorithm* y por otro lado el resto de características asociadas a este, como operadores, codificación de los individuos, selección, etc.

Debido a la amplia variedad de posibilidades que pueden presentarse en cuanto a operadores, los esquemas de codificación, modos de creación, de selección, etc. existen un conjunto de interfaces que representarán, de manera desacoplada, cada uno de estos pasos elementales del proceso evolutivo, siendo responsabilidad del algoritmo evolutivo mantener referencias y coordinar todas las acciones del proceso evolutivo.

A partir de esta arquitectura genérica, y siguiendo esta filosofía de diseño, en primer lugar se ha desarrollado el paradigma de la GEP dentro de la librería, que posteriormente ha sido utilizado para el diseño de los algoritmos que se proponen en esta tesis.

A.3. Implementación de GEP

El desarrollo del paradigma de la GEP sobre la librería JCLEC consta de 14 clases distintas, siendo cada una de ellas responsable de parte del proceso

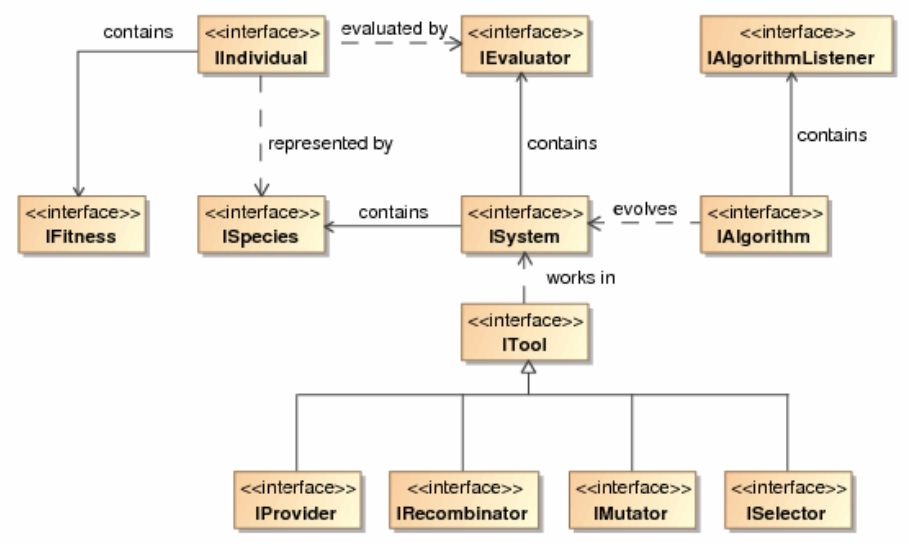


Figura A.1: Núcleo de la librería JCLEC (Obtenido de `http://jclec.sourceforge.com`)

evolutivo. La relación entre ellas se muestra en el diagrama de la figura A.3, y a continuación se comentan brevemente las responsabilidades de cada una de ellas:

- *GEPSpecies*: Implementa el interfaz del núcleo *ISpecies*. Se encarga de almacenar todas las características de los individuos, como el tamaño de la cabeza, de la cola, conjuntos de elementos (funciones y terminales), número de genes y función de conexión. Se encarga además de generar el genotipo, así como de comprobar que los genotipos son válidos.

- *GEPIndividualSpecies*: Hereda de la anterior y describe el interfaz que deben implementar los algoritmos que utilicen individuos cuyo fenotipo esté basado en árboles.

- *GEPCreator*: Implementa el interfaz del núcleo *IProvider* y actúa como una clase factoría, siendo responsable de la creación de los individuos, apoyándose para ello en la clase *GEPSpecies*.

- *GEPIndividual*: Implementa el interfaz del núcleo *IIndividual*. Almacena tanto el genotipo de los individuos, como un *array* de índices a los elementos de los conjuntos de funciones y terminales. Además guarda el *fitness* de los individuos.

- *ExecContext*: Esta clase proporciona el entorno de ejecución necesario para evaluar los árboles generados a partir del genotipo de los individuos.

- *GEPRecombinator*: Representa un interfaz común para todas las clases que implementan operadores de cruce. De ella heredan las clases *GEPOnePointRecombinator*, *GEPTwoPointsRecombinator* y *GEPGeneRecombinator* que implementan los operadores de recombinación en un punto, dos puntos y de genes respectivamente. Implementa el interfaz del núcleo *IRecombinator*

- *GEPMutator*: Del mismo modo que la anterior clase representa el interfaz común de los mutadores. De ella heredan las clases que implementan los diversos operadores de mutación y transposición, es decir, *GEPRISTranspositionMutator*, *GEPISTranspositionMutator*, *GEPGeneTranspositionMutator* y *GEPSimpleMutator*. Implementa el interfaz del núcleo *IMutator*

A.4. Implementación de GEP-MLC

Partiendo de la implementación base de GEP se han desarrollado una serie de clases que se encargan de implementar las diversas responsabilidades específicas del algoritmo diseñado en el capitulo 4. Esta implementación consta de 7 clases que se apoyan en las anteriores, además de en el núcleo de la librería, que se pasan a comentar a continuación. Sus interacciones se pueden observar en el diagrama A.3:

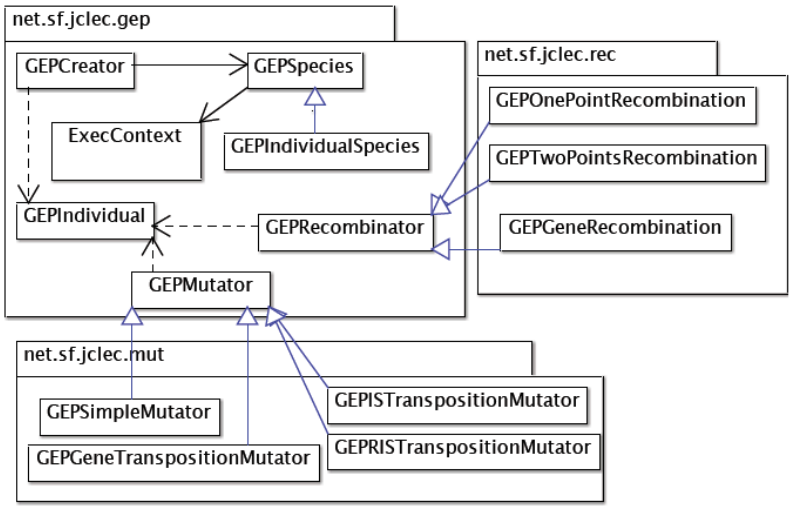


Figura A.2: Diagrama de clases del la implementación base de GEP

- *GEPBasedAlgorithm*: Implementa las características comunes de los algoritmos basados en GEP, además se encarga de ir secuenciando los pasos básicos de cada generación del proceso evolutivo.

- *GEPMLCAlgorithm*: Hereda de la anterior, y se encarga de implementar las características básicas del algoritmo GEP-MLC, apoyada por el creador, la especie y el evaluador.

- *GEPMLCCreator*: Hereda del creador de GEP, y se encarga de definir las características específicas con las que éste creará a los individuos.

- *GEPMLCSpecies*: Almacena las características específicas de los individuos GEPMLC y además se encarga de construir la función discriminante a partir del genotipo de un individuo.

- *GEPMLCEvaluator*: Implementa el evaluador, y por tanto se encarga, con la colaboración de la especie, de crear los fenotipos a partir de los individuos, evaluarlos para el conjunto de datos y para cada una de las etiquetas.Para ello genera los *fitness* necesarios, y realiza el *token*

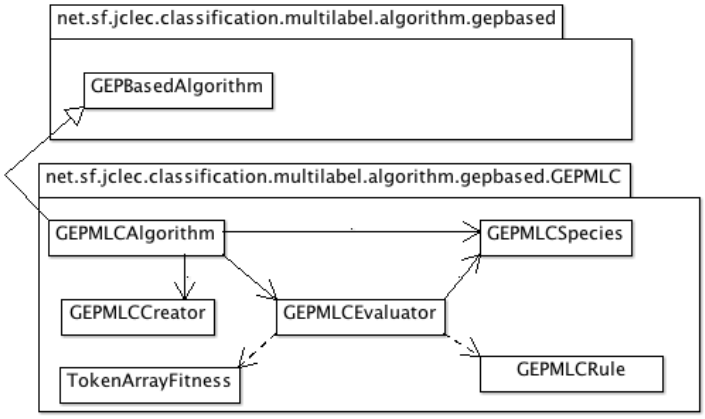


Figura A.3: Diagrama de clases del la implementación del algoritmo GEP-MLC

competition para cada etiqueta, para posteriormente corregir el *fitness* de los individuos.

- *GEPMLCRule*: Encapsula una función discriminante como una regla de clasificación, permitiendo que esta sea evaluada sobre un patrón y devuelve el resultado de la ejecución.

- *TokenArrayFitness*: Se encarga de almacenar el *fitness* multi-etiqueta, guardando el *array* de valores de *fitness*. Además almacena el número de *tokens* que gana cada individuo para posteriormente poder realizar el *token competition*.

A.5. Implementación de los algoritmos GC

El núcleo de los algoritmos GC-1 y CG-k está formado por 7 clases, basadas en el núcleo de JCLEC y en la implementación base de GEP, cuyas interrelaciones se muestran en la figura A.4. Además de estas clases, se han reutilizado algunas de las clases desarrolladas para el algoritmo GEP-MLC,

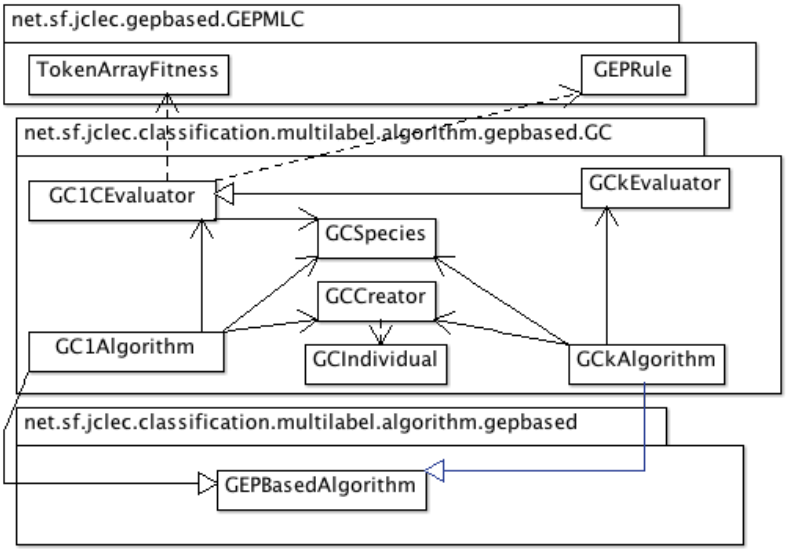


Figura A.4: Diagrama de clases de los algoritmos GC

mostradas en la sección anterior, en concreto las clases *TokenArrayFitness*, que almacena el vector de *tokens* de los individuos, y la clase *GEPMLCRule*, que nos permite encapsular una regla cuyo antecedente sea un árbol obtenido de un genotipo GEP.

Las responsabilidades de las clases se enumeran a continuación:

- *GCSpecies*: Almacena las características de los individuos GC, en concreto el tamaño de la cabeza, de la cola, de la sección de relacionales de la cabeza, y además se encarga de generar los genotipos de los individuos, comprobar la validez de los genotipos y construir el fenotipo a partir del genotipo.

- *GCCreator*: Clase factoría que hereda de *GEPCreator*, y se encarga de crear las clases que encapsulan los individuos con el genotipo característico de GEPCLASS.

- *GCIndividual*: Clase que hereda de *GEPIndividual*, y se encarga de almacenar el genotipo de un individuo, así como las características particulares de este, y en concreto un enlace a la clase que almacena el *fitness* y el *array* de *tokens*.

- *GC1Algoritmh*: Se encarga de implementar las características básicas del algoritmo GC-1, secuenciando cada uno de los pasos básicos de su ejecución, apoyándose para ello por el creador, la especie y el evaluador.

- *GC1Evaluator*: Esta clase se encarga de la evaluación de los individuos, por tanto se encarga de ejecutar cada individuo con cada patrón para calcular el *fitness* y de realizar los *tokens competitions* para cada etiqueta.

- *GCkAlgorithm*: Esta clase de manera similar a la correspondiente a la versión GC-1 implementa las características del algoritmo GC-k y se encarga de implementar los pasos básicos del proceso evolutivo, y almacena los enlaces al creador la especie y su evaluador específico.

- *GCkEvaluator*: Esta clase extiende su funcionalidad del evaluador del algoritmo GC-1, ya que lo que hace es modificar el proceso de evaluación para que este tenga en cuenta todas las combinaciones de etiquetas presentes en el problema.

Además, debido a que los operadores genéticos que utiliza el algoritmo actúan de manera distinta a los operadores definidos en GEP, ha sido necesario desarrollar una serie de clases que los implementen a partir de las clases genéricas definidas en el núcleo de la GEP. En concreto se han implementado clases para los diversos operadores de mutación definidos y para el operador de recombinación. Las clases responsables de los operadores son las siguientes (figura A.5):

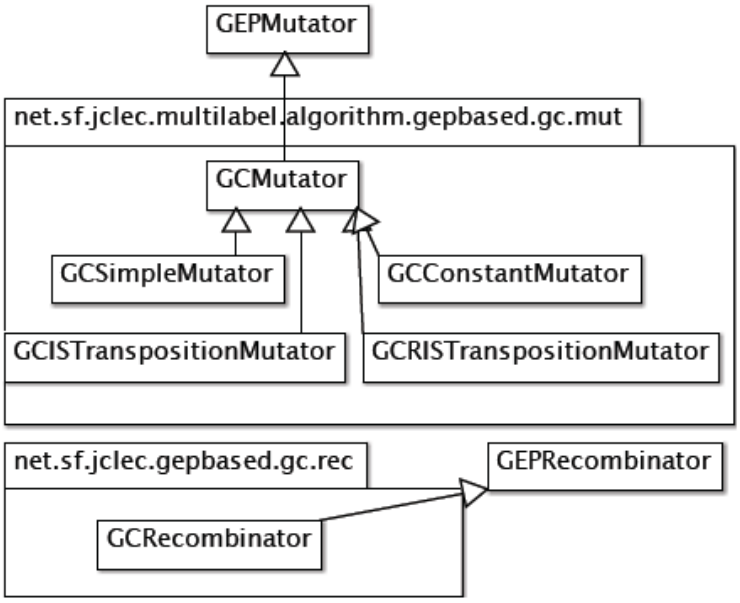


Figura A.5: Diagrama de clases de los operadores genéticos GC

- *GCMutator*: Esta clase es la raíz de la jerarquía de mutadores específicos para el algoritmo, y extiende de la clase *GEPMutator* del núcleo JCLEC.

- *GCSimpleMutator*: Implementa el mutador simple, cambiando aleatorioamente un elemento del individuo por otro similar.

- *GCConstantMutator*: Implementa el operador de mutación de constantes, cambiando la constante de uno de los elementos aleatorios de la cola.

- *GCISTranspositionMutator*: Realiza la transposición IS, pero tal y como se define en la GEPCLASS, es decir, afectando solamente a la cola.

- *GCRISTranspositionMutator*: Del mismo modo que la anterior, implementa la transposición RIS de GEPCLASS, llevando una cadena aleatoria de la cola al comienzo de esta.

- *GCRecombinator*: Esta clase hereda de *GEPRecombinator*, y se encarga del operador de recombinación de individuos, tal y como estaba definido en GEPCLASS, es decir, similar al operador de recombinación en un punto, pero afectando solamente a la cabeza de los individuos.

B

Test estadísticos

B.1. Introducción

En la presente tesis se han propuesto dos métodos novedosos de aprendizaje multi-etiqueta. En la fase experimental, estos métodos se han ejecutado sobre diversos conjuntos de datos, y simultáneamente, se han ejecutado varios algoritmos ya descritos en la literatura. Como resultado se han obtenido una serie de métricas de rendimiento de todas estas ejecuciones. Una vez reunidos todos estos datos experimentales resulta crucial establecer si estadísticamente existen diferencias significativas entre ellas. Para esto se han utilizado una serie de test estadísticos, comúnmente utilizados por la comunidad científica de aprendizaje automático [291] para comparar las nuevas propuestas con las ya establecidas.

En el presente anexo se describen los test estadísticos utilizados en esta tesis: el test de *Wilcoxon*, que nos permite comparar el rendimiento de dos algoritmos, el test de *Friedman* que se utiliza para comparar varias propuestas simultáneamente, y el post-test de *Bonferroni-Dunn* que se aplica tras el test de Friedman para realizar un estudio de la diferencia estadística en el rendimiento de los algoritmos utilizados.

B.2. Test de Wilcoxon

La prueba de los rangos con signo de Wilcoxon es una prueba no paramétrica para comparar la media de dos muestras relacionadas y determinar si existen diferencias entre ellas. Debe su nombre a Frank Wilcoxon, que la publicó en 1945 [294]. Se utiliza con variables continuas y no presupone ningún tipo de distribución particular, siendo por tanto un test estadístico no paramétrico.

Supongamos que se dispone de n pares de observaciones (x_i, y_i), el objetivo del test es comprobar si puede dictaminarse que los resultados son o no iguales. De esta forma el test nos permitirá comprobar si el rendimiento de dos algoritmos, que se ha probado sobre diversos conjuntos de datos, son esencialmente iguales.

La idea que subyace al test de Wilcoxon es que si restamos los valores del par, $d_i = x_i - y_i$, los valores de la diferencia, d_i deberán estar distribuidos uniformemente, es decir, existirán tantos valores de d_i positivos como negativos.

Para aplicar el test se calculan las diferencias entre las dos series de observaciones, se ordenan de mayor a menor y se compara los lugares que ocupan las diferencias a favor de una serie u otra. La hipótesis nula H_0 presupone que los resultados son iguales.

Formalmente, se denomina rango de d_i y se denota por $rang(d_i)$ a la posición que ocupa d_i en la ordenación. Sean R^+ y R^- las sumas respectivas de los rangos de las diferencias positivas y negativas. Los rangos de las diferencias nulas se reparten entre ambas. Por tanto R^+ y R^- se calculan según las ecuaciones B.1 y B.2.

$$R^+ = \sum_{d_i>0} rang(d_i) + \frac{1}{2}\sum_{d_i=0} rang(d_i) \tag{B.1}$$

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

Sea T la menor de las sumas, es decir, $T = min(R^+, R^-)$. Si la hipótesis nula es cierta, T debe aproximarse al valor $(n(n+1)/4$. Por otro lado cuanto más diferente sean los resultados de ambas series, los valores más se aproximarán a 0. Numerosos textos de estadística incluyen los valores críticos exactos (p-valores) para T con un número de observaciones menor a 25 ($n \leq 25$) Para valores superiores de n se puede obtener el estadístico z:

$$z = \frac{T - n(n+1)/4}{\sqrt{n(n+1)(2n+1)/24}} \tag{B.3}$$

El estadístico z para valores mayores que 25 tiene una distribución aproximadamente normal, por tanto para $\alpha = 0.05$ se puede rechazar la hipótesis nula si z es menor que -1.96.

B.3. Test de Friedman

El test de Friedman es también un test no paramétrico que fue desarrollado por el economista estadounidense Milton Friedman [295]. Es similar al test paramétrico ANOVA, pero sin considerar los datos normalmente distribuidos. Originalmente fue propuesto para detectar las diferencias entre diversos tratamientos suministrados a varios pacientes y determinar si estos tratamientos eran significativamente distintos o no. En nuestro caso los tratamientos serán distintos algoritmos de MLC y los pacientes los conjuntos de datos sobre los que se aplican los algoritmos. De esta forma el test nos permitirá determinar si hay diferencias significativas entre un conjunto de algoritmos.

Si suponemos que tenemos k algoritmos y n conjuntos de datos, lo primero que hace el algoritmo es calcular una serie de valores de *ranking*, r_{ij}, para cada algoritmo y conjunto de datos. Estos valores de *ranking* se obtienen asignándole una serie de puntuaciones a los algoritmos de la siguiente manera: el mejor algoritmo para un conjunto de datos se le asigna una puntuación de 1, al siguiente mejor un 2, y así sucesivamente hasta el valor k.

Posteriormente se promedian los *ranking* obtenidos por los algoritmos para todos los conjuntos de datos. De esta manera se obtiene un valor $R_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij}$. Estos valores permiten saber qué algoritmo obtiene los mejores resultados considerando globalmente todos los conjuntos de datos. De esta manera el algoritmo con el valor más próximo a uno indica que es el mejor algoritmo en la mayoría de los *datasets*.

Una vez obtenidos todos los valores de R_k el estadístico de Friedman se calcula de la siguiente manera:

$$\chi^2{}_F = \frac{12n}{k(k+1)}\left(\sum_{j=1}^{k} R_j{}^2 - \frac{k(k+1)^2}{4}\right) \tag{B.4}$$

Los valores del estadístico de Friedman, que siguen una distribución de tipo χ^2 con $k-1$ grados de libertad, tantos como algoritmos probados menos uno, si el número de algoritmos es lo suficientemente grande (como regla empírica se suele utilizar valores de $n \geq 10$ y $k \geq 5$). En cualquier caso para valores inferiores se pueden obtener los valores críticos exactos en cualquier texto de estadística.

B.4. Post-test de Bonferroni-Dunn

El test de Friedman permite determinar si el rendimiento de un conjunto de algoritmos es significativamente distinto o no, pero no nos permite identificar

entre qué algoritmos del conjunto se han producido las diferencias significativas. Para esto hay que realizar un post-test. Se han propuesto varios post-test en la literatura [291] pero entre ellos en esta tesis se ha utilizado el test de Bonferroni-Dunn ya que se considera más potente más potente que el test de Nemenyi [291], también muy utilizado.

La comparación entre las tablas de los test de Nemenyi y los de Bonferroni-Dunn muestran que este último es más potente si todos los algoritmos se comparan con uno solo, es decir, si se trata de comparar una propuesta novedosa con el estado del arte, como se hace en esta tesis. Esta es la razón por la que se ha utilizado este test y no otro.

En todos los post-test se calcula un valor diferencia crítica. Dicho valor nos indica que dos algoritmos son estadísticamente distintos si los rangos promedio obtenidos por ellos difieren al menos este valor crítico. El valor crítico se calcula utilizando la ecuación B.5 donde q_α son el valor para k y n del estadístico de rangos estudentizados dividido entre $\sqrt{2}$ [291].

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \tag{B.5}$$

Para comparar varios algoritmos con un algoritmo de control se utiliza el estadístico Z (Ecuación B.6). Este estadístico nos permite saber si el algoritmo i-ésimo y j-ésimo son estadísticamente distintos o no. Este valor Z se compara con el correspondiente p-valor usando la distribución normal. La principal diferencia del test de Bonferroni-Dunn con el resto de post-test se encuentra en la manera en que se ajusta el valor de α para compensar las comparaciones múltiples, ya que el test de Nemenyi está pensado para realizar todas las comparaciones posibles entre los k algoritmos, pero el test de Bonferruni-Dunn está concebido para comparar una serie de algoritmos con uno de control. Esta es la situación que generalmente se produce cuando probamos una nueva propuesta con otras ya establecidas en la literatura. De

esta forma en el test de Bonferroni-Dunn se utiliza para el cálculo de q_α el valor $\alpha/(k-1)$ en lugar de $\alpha/(k(k-1)/2)$.

$$Z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6n}}} \tag{B.6}$$

Bibliografía

[1] M.-L. Zhang and X.-H. Zhou, "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1338–1351, Oct. 2006.

[2] M. A. H. Ian H Written, Eibe Frank, *Data Mining: Practical machine learning tools and techniques*. 2011.

[3] J. Hernández, Ramírez, M. J., and C. Ferri, *Introducciǿn a la Minería de Datos*. Pearson Prentice Hall, 2004.

[4] N. J. Nilsson, *Introduction to Machine Learning*. Stanford University, 1996.

[5] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. 500 Sansome Street, Suite 400, San Francisco, CA 94111: Morgan Kaufmann Publishers, second ed., 2006.

[6] A. Elisseeff and J. Weston, "A Kernel Method for Multi-Labelled Classification," *Advances in Neural Information Processing Systems*, vol. 14, pp. 681–687, 2001.

[7] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[8] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification1," *Pattern Recognition*, vol. 37, pp. 1757–1771, Sept. 2004.

[9] G. Tsoumakas, I. Katakis, and I. Vlahavas, *Data Mining and Knowledge Discovery Handbook*, ch. Mining Multi-label Data. Springer, 2009.

[10] T. Li, C. Zhang, and S. Zhu, "Empirical studies on multi-label classification," in *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, (Washington, DC, USA), pp. 86–92, IEEE Computer Society, 2006.

[11] K. Kawai and Y. Takahashi, "Identification of the dual action antihypertensive drugs using tfs-based support vector machines," *Chem-Bio Informatics Journal*, vol. 9, pp. 41–51, 2009.

[12] J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch, "A shared task involving multi-label classification of clinical free text," in *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, BioNLP '07, (Stroudsburg, PA, USA), pp. 97–104, Association for Computational Linguistics, 2007.

[13] T. Pohle, *Automatic Characterization of Music for Intuitive Retrieval*. PhD thesis, Johannes Kepler University, Linz, Austria, January 2010.

[14] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, December 1992.

[15] C. Ferreira, "Gene expression programming:a new adaptative algorithm for solving problems.," *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[16] W. R. Weinert and H. S. Lopes, "GEPCLASS: A classification rule discovery tool using gene expression programming," in *Advanced Data Mining and Applications, Proceedings of the Second International Conference, ADMA* (X. Li, O. R. Zaïane, and Z. Li, eds.), vol. 4093 of *Lecture Notes in Computer Science*, (Xián, China), pp. 871–880, Springer, August 14-16 2006.

[17] K. Brinker, J. Fürnkranz, and E. Hüllermeier, "A unified model for multilabel classification and ranking," in *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI '06)*, (Rivadel Garda, Italy), pp. 489–493, 2006.

[18] K. Brinker and E. Hullermeier, "Case-based multilabel ranking," in *Proceedings of the 20th International Conference on Artificial Intelligence (IJCAI '07)*, (Hyderabad, India), pp. 702–707, Jan 2007.

[19] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: combining bayes with svm," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, (New York, NY, USA), pp. 177–184, ACM Press, 2006.

[20] C. Parker, Y. Altun, and P. Tadepalli, "Guest editorial: special issue on structured prediction," *Machine Learning*, vol. 77, pp. 161–164, 2009.

[21] A. de Carvalho and A. Freitas, "A tutorial on multi-label classification techniques," *Foundations of Computational Intelligence*, vol. Vol. 5, pp. 177–195, September 2009.

[22] R. Jin and Z. Ghahramani, "Learning with multiple labels," *Advances in Neural Information Processing Systems*, vol. 15, pp. 921 – 928, 2003.

[23] A. Zafra and S. Ventura, "G3P-MI: A genetic programming algorithm for multiple instance learning," *Information Sciences*, vol. 180, no. 23, pp. 4496 – 4513, 2010.

[24] M.-L. Zhang and Z.-H. Zhou, "Multi-label learning by instance differentiation," in *AAAI*, pp. 669–674, 2007.

[25] J. Read, *Scalable Multi-label Classification*. PhD thesis, University of Waikato, 2010.

[26] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4701 LNAI, pp. 406–417, 2007.

[27] J. Chen, X. Zhou, and Z. Wu, "A Multi-Label Chinese Text Categorization System Based on Boosting Algorithm," in *CIT '04: Proceedings of the The Fourth International Conference on Computer and Information Technology*, (Washington, DC, USA), pp. 1153–1158, IEEE Computer Society, 2004.

[28] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)*, pp. 22–30, 2004.

[29] E. Alvares Cherman, J. Metz, and M. Monard, "A Simple Approach to Incorporate Label Dependency in Multi-label Classification," in *Advances in Soft Computing* (G. Sidorov, A. Hernández Aguirre, and C. Reyes García, eds.), vol. 6438 of *Lecture Notes in Computer Science*, ch. 3, pp. 33–43, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[30] E. A. Cherman, J. Metz, and M. C. Monard, "Incorporating label dependency into the binary relevance framework for multi-label classification," *Expert Systems with Applications*, vol. 39, pp. 1647–1655, July February, 2012.

[31] J. Fürnkranz and E. Hüllermeier, "Pairwise preference learning and ranking," in *In Proc. ECML-03, Cavtat-Dubrovnik*, pp. 145–156, 2003.

[32] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, "Label ranking by learning pairwise preferences," *Artificial Intelligence*, vol. 172, pp. 1897–1916, Nov. 2008.

[33] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Mach. Learn.*, vol. 73, no. 2, pp. 133–153, 2008.

[34] J. Read, "A Pruned Problem Transformation Method for Multi-label classification," in *Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, pp. 143–150, 2008.

[35] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[36] I. Partalas, G. Tsoumakas, I. Katakis, and I. Vlahavas, "Ensemble pruning using reinforcement learning," *Advances in Artificial Intelligence*, pp. 301–310, 2006.

[37] L. Tenenboim-Chekina, L. Rokach, and B. Shapira, "Identification of Label Dependences for Multi-Label Classification," in *2nd International Workshop on Learning from Multi-Label Data (MLD'10)*, pp. 53–60, June 2010.

[38] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, p. XX, 2008.

[39] G. Tsoumakas, E. L. Mencía, I. Katakis, S. Park, and J. Fürnkrnaz, "On the combination of two decompositive multi-label classification methods," pp. 114–133, 2009.

[40] J. Read, B. Pfahringer, and G. Holmes, "Multi-label Classification Using Ensembles of Pruned Sets," in *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, vol. 0, (Washington, DC, USA), pp. 995–1000, IEEE Computer Society, 2008.

[41] J. Xu, "An extended one-versus-rest support vector machine for multi-label classification," *Neurocomputing*, vol. 74(17), pp. 3114–3124, May 2011.

[42] S. P. Wan and J. H. Xu, "A multi-label classification algorithm based on triple class support vector machine," in *Proc. 2007 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR'07)*, 2007.

[43] A. Jiang, C. Wang, and Y. Zhu, "Calibrated Rank-SVM for multi-label image categorization," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pp. 1450–1455, June 2008.

[44] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification models," *Journal of Machine Learning Research*, vol. 7, pp. 1601–1626, 2006.

[45] W. Chen, J. Yan, B. Zhang, Z. Chen, and Q. Yang, "Document Transformation for Multi-label Feature Selection in Text Categorization," *IEEE International Conference on Data Mining*, vol. 0, pp. 451–456, 2007.

[46] B. Lauser and A. Hotho, "Automatic multi-label subject indexing in a multilingual environment," in *Proceedings of the 7th European Conference in Research and Advanced Technology for Digital Libraries (ECDL 2003)*, Lecture Notes in Computer Science, (Trondheim, Norway), pp. 140–151, Springer, 2003.

[47] F. Aiolli, F. Sebastiani, and A. Sperduti, "Preference Learning for Category-Ranking based Interactive Text Categorization," in *International Joint Conference on Neural Networks (IJCNN)*, pp. 2034–2039, Aug. 2007.

[48] T. Gonçalves and P. Quaresma, *A Preliminary Approach to the Multilabel Classification Problem of Portuguese Juridical Documents*, vol. 2902/2003 of *Lecture Notes in Computer Science*. Springer, 2003.

[49] E. E. Myint and M. Pwint, "An approach for mulit-label music mood classification," in *3rd International Conference on Signal Processing Systems*, pp. V1–290–V1–294, July 2010.

[50] Z. Barutcuoglu, R. E. E. Schapire, and O. G. G. Troyanskaya, "Hierarchical multi-label prediction of gene function.," *Bioinformatics*, vol. 22, pp. 830–836, 2006.

[51] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare, "Decision trees for hierarchical multilabel classification: A case study in functional genomics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4213 LNAI, pp. 18–29, 2006.

[52] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, pp. 185–214, 2008.

[53] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Detection of Visual Concepts and Annotation of Images Using Ensembles of Trees for Hierarchical Multi-Label Classification," in *Recognizing Patterns in Signals, Speech, Images and Videos* (D. Ünay, Z. Çataltepe, and S. Aksoy, eds.), vol. 6388 of *Lecture Notes in Computer Science*, ch. 16, pp. 152–161, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[54] A. Clare and R. D. King, "Knowledge Discovery in Multi-label Phenotype Data," *Lecture Notes in Computer Science*, vol. 2168, pp. 42–53, 2001.

[55] E. Gibaja, M. Victoriano, J. L. Avila-Jimenez, and S. Ventura, "A TDIDT technique for multi-label classification," in *10th International Conference on. Intelligent Systems Design and Applications*, pp. 519–524, Nov. 2010.

[56] K. Gold and A. Petrosino, "Using information gain to build meaningful decision forests for multilabel classification," in *IEEE 9th International Conference on Development and Learning (ICDL)*, pp. 58–63, Aug. 2010.

[57] H. G. Noh, M. S. Song, and S. H. Park, "An unbiased method for constructing multilabel classification trees," *Computational Statistics & Data Analysis*, vol. 47, no. 1, pp. 149–164, 2004.

[58] F. De Comité, R. Gilleron, and M. Tommasi, "Learning Multi-label Alternating Decision Trees from Texts and Data," in *Machine Learning and Data Mining in Pattern Recognition*, pp. 251–274, 2003.

[59] K. Crammer and Y. Singer, "A family of additive online algorithms for category ranking," *The Journal of Machine Learning Research*, vol. 3, pp. 1025–1058, 2003.

[60] E. Sapozhnikova, "ART-Based Neural Networks for Multi-label Classification," in *Advances in Intelligent Data Analysis VIII* (N. Adams, C. Robardet, A. Siebes, and J.-F. Boulicaut, eds.), vol. 5772 of *Lecture Notes in Computer Science*, ch. 15, pp. 167–177, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009.

[61] M.-L. Zhang, "Ml-rbf : RBF Neural Networks for Multi-Label Learning," *Neural Processing Letters*, vol. 29, pp. 61–74, Apr. 2009.

[62] P. Marques Ciarelli and E. Oliveira, "An Enhanced Probabilistic Neural Network Approach Applied to Text Classification," in *Progress in*

Pattern Recognition, Image Analysis, Computer Vision, and Applications (E. Bayro-Corrochano and J.-O. Eklundh, eds.), vol. 5856 of *Lecture Notes in Computer Science*, ch. 78, pp. 661–668, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009.

[63] E. Oliveira, P. M. Ciarelli, A. F. Souza, and C. Badue, "Using a Probabilistic Neural Network for a Large Multi-label Problem," in *Neural Networks, 2008. SBRN '08. 10th Brazilian Symposium on*, pp. 195–200, Oct. 2008.

[64] A. F. De Souza, F. Pedroni, E. Oliveira, P. M. Ciarelli, W. F. Henrique, L. Veronese, and C. Badue, "Automated multi-label text categorization with VG-RAM weightless neural networks," *Neurocomputing*, vol. 72, pp. 2209–2217, June 2009.

[65] J.-S. Lin, P. A. Ligomenides, S. C. B. Lo, M. T. Freedman, and S. K. Mun, "A hybrid neural-digital computer-aided diagnosis system for lung nodule detection on digitized chest radiographs," in *Conference Board of the Mathematical Sciences (CBMS)*, pp. 207–212, 1994.

[66] A. Skabar, D. Wollersheim, and T. Whitfort, "Multi-label classification of gene function using MLPs," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 2234–2240, July 2006.

[67] M.-L. Zhang and Z.-H. Zhou, "A k-Nearest Neighbor Based Algorithm for Multi-label Classification," vol. 2, pp. 718–721 Vol. 2, The IEEE Computational Intelligence Society, 2005.

[68] R. Jin, S. Wang, and Z. H. Zhou, "Learning a distance metric from multi-instance multi-label data," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 896–902, 2009.

[69] E. Spyromitros, G. Tsoumakas, and I. Vlahavas, "An empirical study of lazy multilabel classification algorithms," *Artificial Intelligence: Theories, Models and Applications*, pp. 401–406, 2008.

[70] R. Rak, L. Kurgan, and M. Reformat, "A tree-projection-based algorithm for multi-label recurrent-item associative-classification rule generation," *Data & Knowledge Engineering*, vol. 64, pp. 171–197, Jan. 2008.

[71] F. A. Thabtah, P. Cowling, and Y. Peng, "MMAC: A new multi-class, multi-label associative classification approach," Proceedings - Fourth IEEE International Conference on Data Mining, ICDM 2004, (Bradford, United Kingdom), pp. 217–224, 2004.

[72] F. A. Thabtah, P. Cowling, and Y. Peng, "Multiple labels associative classification," *Knowledge and Information Systems*, vol. 9, no. 1, pp. 109–129, 2006.

[73] F. Thabtah, P. Cowling, and Y. Peng, "MCAR: multi-class classification based on association rule," in *The 3rd ACS/IEEE International Conference Computer Systems and Applications.*, 2005.

[74] F. A. Thabtah and P. I. Cowling, "A greedy classification algorithm based on association rule," *Appl. Soft Comput.*, vol. 7, no. 3, pp. 1102–1111, 2007.

[75] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, (New York, NY, USA), pp. 195–200, ACM, 2005.

[76] C. Bielza, G. Li, and P. Larrañaga, "Multi-dimensional classification with Bayesian networks," *International Journal of Approximate Reasoning*, vol. 52, pp. 705–727, Sept. 2011.

[77] J. Zhang, Z. Ghahramani, and Y. Yang, "Learning multiple related tasks using latent independent component analysis," in *Advances in Neural Information Processing Systems 18, Neural Information Processing Systems, NIPS 2005*, (Cambridge, MA), pp. 1585–1592, MIT Press, 2005.

[78] H. Wang, M. Huang, and X. Zhu, "A Generative Probabilistic Model for Multi-label Classification," in *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 628–637, IEEE Computer Society, 2008.

[79] A. S. Streich and J. Buhmann, "Classification of multi-labeled data: A generative approach," *Machine Learning and Knowledge Discovery in Databases*, pp. 390–405, 2008.

[80] A. Chan and A. A. Freitas, "A new ant colony algorithm for multi-label classification with applications in bioinfomatics," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, (New York, NY, USA), pp. 27–34, ACM Press, 2006.

[81] R. Vallim, D. Goldberg, X. Llorà, T. Duque, and A. Carvalho, "A new approach for multi-label classification based on default hierarchies and organizational learning," in *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pp. 2017–2022, 2008.

[82] Schapire, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[83] A. Esuli, T. Fagni, and F. Sebastiani, "MP-Boost: A Multiple-Pivot Boosting Algorithm and Its Application to Text Categorization," in *String Processing and Information Retrieval* (F. Crestani, P. Ferragina, and M. Sanderson, eds.), vol. 4209 of *Lecture Notes in Computer*

Science, ch. 1, pp. 1–12, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006.

[84] A. Esuli, T. Fagni, and F. Sebastiani, "Boosting multi-label hierarchical text categorization," *Information Retrieval*, vol. 11, pp. 287–313, Aug. 2008.

[85] J. Yearwood, M. Mammadov, and A. Banerjee, "Profiling Phishing Emails Based on Hyperlink Information," in *2010 International Conference on Advances in Social Networks Analysis and Mining*, pp. 120–127, IEEE, Aug. 2010.

[86] J. Li and J. Xu, "A Fast Multi-label Classification Algorithm Based on Double Label Support Vector Machine," in *Proceedings of the 2009 International Conference on Computational Intelligence and Security*, pp. 30–35, 2009.

[87] H. Blockeel, L. D. Raedt, and J. Ramong, "Top-down induction of clustering trees," in *In Proceedings of the 15th International Conference on Machine Learning*, pp. 55–63, Morgan Kaufmann, 1998.

[88] G. A. Carpenter, S. Grossberg, and S. Grossberg, "Adaptive resonance theory," in *The Handbook of Brain Theory and Neural Networks*, pp. 87–90, 2003.

[89] W. Li, J. Han, and J. Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules," in *IEEE International Conference on Data Mining series*, pp. 369–376, 2001.

[90] J. P. Lucas, *Métodos de Clasificación Basados en Asociación Aplicados a Sistemas de Recomendación.* PhD thesis, Facultad de Ciencias. Universidad de Salamanca., 2010.

[91] S. Dasgupta and Y. Freund, "Random projection trees for vector quantization," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3229–3242, 2009.

[92] V. Roth and B. Fischer, "Improved functional prediction of proteins by learning kernel combinations in multilabel settings," *BMC Bioinformatics*, vol. 8, no. Suppl 2, 2007.

[93] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," in *ICML*, pp. 170–178, 1998.

[94] M. Johnson and R. Cipolla, "Improved image annotation and labelling through multi label boosting," in *Proceedings of the British Machine Vision Conference (16th BMVC)*, (Oxford, U.K), British Machine Vision Association (BMVA), 2005.

[95] Y. Freund, "The alternating decision tree learning algorithm," in *In Machine Learning: Proceedings of the Sixteenth International Conference*, pp. 124–133, Morgan Kaufmann, 1999.

[96] P. Chan, X. Zeng, E. Tsang, D. Yeung, and J. Lee, "Neural network ensemble pruning using sensitivity measure in web applications," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pp. 3051–3056, 2007.

[97] B. Sun, X. Zhang, and R. Wang, "On constructing and pruning svm ensembles," in *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System SITIS '07.*, pp. 855–859, 2007.

[98] M. V. de Oliveira, E. D. Wandekokem, E. Mendel, F. Fabris, F. M. Varejao, T. W. Rauber, and R. Batista, "A Comparison of Two Feature-Based Ensemble Methods for Constructing Motor Pump Fault Diagnosis Classifiers," in *22th International Conference on Tools with Artificial Intelligence*, pp. 417–420, Oct. 2010.

[99] P. Nardiello, F. Sebastiani, and A. Sperduti, "Discretizing Continuous Attributes in AdaBoost for Text Categorization," in *Advances in Information Retrieval* (F. Sebastiani, ed.), vol. 2633 of *Lecture Notes in Computer Science*, ch. 23, p. 78, Berlin, Heidelberg: Springer Berlin / Heidelberg, Apr. 2003.

[100] C. D. Nguyen, T. A. Dung, and T. H. Cao, "Text Classification for DAG-Structured Categories," in *Advances in Knowledge Discovery and Data Mining* (T. B. Ho, D. Cheung, and H. Liu, eds.), vol. 3518 of *Lecture Notes in Computer Science*, ch. 36, pp. 1–18, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2005.

[101] S. N. Saleh and El, "A feature selection algorithm with redundancy reduction for text classification," in *Computer and Information Sciences, 2007. ISCIS 2007. 22nd International International Symposium on*, pp. 1–6, 2007.

[102] M. S. Ahmed, L. Khan, and M. Rajeswari, "Using Correlation Based Subspace Clustering for Multi-label Text Data Classification," in *Preceedings of the 22th International Conference on Tools with Artificial Intelligence*, pp. 296–303, Oct. 2010.

[103] Y. C. Chang, S. M. Chen, and C. J. Liau, "Multilabel text categorization based on a new linear classifier learning method and a category-sensitive refinement method," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1948–1953, 2008.

[104] Y. C. Chang, S. M. Chen, and C. J. Liau, "A new inductive learning method for multilabel text categorization," *19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006*, vol. 4031, pp. 1249–1258, 2006.

[105] T. Gonçalves and P. Quaresma, "Using IR Techniques to Improve Automated Text Classification," *Natural Language Processing and Information Systems*, pp. 374–379, 2004.

[106] D. Vilar, M. J. Castro, and E. Sanchis, "Multi-label Text Classification Using Multinomial Models," in *Advances in Natural Language Processing* (J. L. Vicedo, P. Martínez-Barco, R. Muñoz, and M. Saiz Noeda, eds.), vol. 3230 of *Lecture Notes in Computer Science*, ch. 20, pp. 220–230, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2004.

[107] Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," *Information Retrieval*, vol. 1, pp. 69–90, May 1999.

[108] K. Yu, S. Yu, and V. Tresp, "Multi-label informed latent semantic indexing," in *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, (New York, NY, USA), pp. 258–265, ACM, 2005.

[109] D. D. Lewis, Y. Yang, T. G. Rose, G. Dietterich, F. Li, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[110] T. Gonçalves and P. Quaresma, "A preliminary approach to the multilabel classification problem of portuguese juridical documents," in *Proceedings of the 11th Portuguese Conference on Artificial Intelligence (EPIA '03)*, pp. 435–444, 2003.

[111] Loza and J. Fürnkranz, "Efficient pairwise multilabel classification for large-scale problems in the legal domain," in *Machine Learning and Knowledge Discovery in Databases*, pp. 50–65, 2008.

[112] J. M. Carmona-Cejudo, M. Baena-Garcia, J. del Campo-Avila, and R. Morales-Bueno, "Feature extraction for multi-label learning in the domain of email classification," *Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on*, pp. 30–36, Apr. 2011.

[113] H. Cheng, Z. Qin, C. Fu, and Y. Wang, "A novel spam image filtering framework with Multi-Label Classification," in *International Conference on Communications, Circuits and Systems (ICCCAS)*, pp. 282–285, July 2010.

[114] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowledge and Information Systems*, vol. 22(3), pp. Pages 371–391, 2010.

[115] P. Keila and D. Skillicorn, "Structure in the enron email dataset," *Computational & Mathematical Organization Theory*, vol. 11, pp. 183–199, October 2005.

[116] P. Bhowmick, A. Basu, P. Mitra, and A. Prasad, "Multi-label Text Classification Approach for Sentence Level News Emotion Analysis," in *Pattern Recognition and Machine Intelligence* (S. Chaudhury, S. Mitra, C. Murthy, P. Sastry, and S. Pal, eds.), vol. 5909 of *Lecture Notes in Computer Science*, ch. 42, pp. 261–266, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009.

[117] T. Sobol-Shikler and P. Robinson, "Classification of Complex Information: Inference of Co-Occurring Affective States from Their Expressions in Speech," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 1284–1297, July 2010.

[118] S. Li and C. Zong, "Multi-domain sentiment classification," in *HLT '08: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, (Morristown, NJ, USA), pp. 257–260, Association for Computational Linguistics, 2008.

[119] S. Li and C. Zong, "Multi-domain adaptation for sentiment classification: Using multiple classifier combining methods," in *International Conference on Natural Language Processing and Knowledge Engineering, 2008. NLP-KE '08.*, pp. 1–8, Oct. 2008.

[120] A. Veloso, Jr, M. Gonçalves, and M. Zaki, "Multi-label lazy associative classification," in *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, vol. LNAI 4702, (Warsaw, Poland), pp. 605–612, Springer, September 2007.

[121] N. Oza, J. P. Castle, and J. Stutz, "Classification of aeronautics system health and safety documents," *Trans. Sys. Man Cyber Part C*, vol. 39, pp. 670–680, November 2009.

[122] H. Cong and L. H. Tong, "Grouping of TRIZ Inventive Principles to facilitate automatic patent classification," *Expert Systems with Applications*, vol. 34, pp. 788–795, Jan. 2008.

[123] Y. Song, L. Zhang, and C. L. Giles, "Automatic tag recommendation algorithms for social recommender systems," *ACM Trans. Web*, vol. 5, Feb. 2011.

[124] B. Chidlovskii, "Multi-label Wikipedia Classification with Textual and Link Features," in *Focused Retrieval and Evaluation* (S. Geva, J. Kamps, and A. Trotman, eds.), vol. 6203 of *Lecture Notes in Computer Science*, ch. 38, pp. 387–396, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[125] Z. Wei, D. Miao, R. Zhao, C. Xie, and Z. Zhang, "Visualizing search results based on multi-label classification," in *IEEE International Conference on Progress in Informatics and Computing.*, pp. 203–207, Dec. 2010.

[126] R. Alves, M. Delgado, and A. Freitas, "Multi-label Hierarchical Classification of Protein Functions with Artificial Immune Systems," in *Advances in Bioinformatics and Computational Biology* (A. Bazzan, M. Craven, and N. Martins, eds.), vol. 5167 of *Lecture Notes in Computer Science*, ch. 1, pp. 1–12, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2008.

[127] R. T. Alves, M. R. Delgado, and A. A. Freitas, "Knowledge discovery with Artificial Immune Systems for hierarchical multi-label classification of protein functions," in *2010 IEEE International Conference on Fuzzy Systems (FUZZ)*, pp. 1–8, July 2010.

[128] S. Ji, L. Tang, S. Yu, and J. Ye, "A shared-subspace learning framework for multi-label classification," *ACM Trans. Knowl. Discov. Data*, vol. 4, pp. 1–29, May 2010.

[129] S. Ji, L. Tang, S. Yu, and J. Ye, "Extracting shared subspace for multi-label classification," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, (New York, NY, USA), pp. 381–389, ACM, 2008.

[130] S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas, "Protein classification with multiple algorithms," *Advances in Informatics*, pp. 448–456, 2005.

[131] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, "Hierarchical multi-label prediction of gene function," *Bioinformatics*, vol. 22, pp. 830–836, Apr. 2006.

[132] J. Jung and M. R. Thon, "Gene function prediction using protein domain probability and hierarchical Gene Ontology information," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, 2008.

[133] Y. Yang and B.-L. Lu, "Prediction of Protein Subcellular Multi-locations with a Min-Max Modular Support Vector Machine," in *Advances in Neural Networks - ISNN 2006* (J. Wang, Z. Yi, J. Zurada, B.-L. Lu, and H. Yin, eds.), vol. 3973 of *Lecture Notes in Computer Science*, ch. 98, pp. 667–673, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006.

[134] L. Zhu, J. Yang, and H.-B. Shen, "Multi Label Learning for Prediction of Human Protein Subcellular Localizations," *The Protein Journal*, vol. 28, pp. 384–390, Dec. 2009.

[135] R. Duwairi and A. Kassawneh, "A framework for predicting proteins 3D structures," in *6th ACS-IEEE International Conference on Computer Systems and Applications*, pp. 37–44, Mar. 2008.

[136] S. Huang and L. Jin, "A PLSA-Based Semantic Bag Generator with Application to Natural Scene Classification under Multi-instance Multi-label Learning Framework," in *he 5th International Conference on Image and Graphics*, pp. 331–335, Sept. 2009.

[137] W. Li, M. Sun, and C. Habel, "Multi-modal Multi-label Semantic Indexing of Images Based on Hybrid Ensemble Learning," in *Advances in Multimedia Information Processing PCM 2007* (H. Ip, O. Au, H. Leung, M.-T. Sun, W.-Y. Ma, and S.-M. Hu, eds.), vol. 4810 of *Lecture Notes in Computer Science*, ch. 90, pp. 744–754, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2007.

[138] X. Li, L. Wang, and E. Sung, "Multi-label svm active learning for image classification," in *IEEE 2004 International Conference on Image Processing (ICIP '04)*, pp. 2207–2210, 2004.

[139] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang, "Two-Dimensional Multilabel Active Learning with an Efficient Online Adaptation Model for Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1880–1897, Oct. 2009.

[140] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang, "Two-Dimensional Active Learning for image classification," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.

[141] J. Shao, D. He, and Q. Yang, "Multi-semantic Scene Classification Based on Region of Interest," in *International Conference on Computational Intelligence for Modelling, Control and Automation*, pp. 732–737, 2008.

[142] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *Int. J. Comput. Vision*, vol. 81, pp. 2–23, Jan. 2009.

[143] C. Wang, S. Yan, L. Zhang, and H.-J. Zhang, "Multi-label sparse coding for automatic image annotation," in *IEEE COmputer Society Conference on Computer Vision and Pattern Recognition*, pp. 1643–1650, June 2009.

[144] H. Wang, H. Huang, and C. Ding, "Image annotation using multi-label correlated Green's function," in *International Conference on Computer Vision*, pp. 2029–2034, Sept. 2009.

[145] Z. Wang, Y. Hu, and L. T. Chia, "Multi-label learning by Image-to-Class distance for scene classification and image annotation," in *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, (New York, NY, USA), pp. 105–112, ACM, 2010.

[146] Z. Wang, W.-C. Siu, and D. Feng, "Image annotation with parametric mixture model based multi-class multi-labeling," in *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pp. 634–639, Oct. 2008.

[147] X. Zhang, J. Cheng, C. Xu, H. Lu, and S. Ma, "Multi-view multi-label active learning for image classification," in *IEEE International Conference on Multimedia and Expo*, pp. 258–261, June 2009.

[148] W. Li and M. Sun, "Multi-modal Multi-label Semantic Indexing of Images Using Unlabeled Data," in *Advanced Language Processing and Web Information Technology, 2008. ALPIT '08. International Conference on*, pp. 204–209, 2008.

[149] J. Wang, Y. Zhao, X. Wu, and X. S. Hua, "A transductive multi-label learning approach for video concept detection," *Pattern Recognition*, pp. 298–304, 2011.

[150] G. J. Qi, X. S. Hua, Y. Rui, J. Tang, T. Mei, and H. J. Zhang, "Correlative multi-label video annotation," in *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, (New York, NY, USA), pp. 17–26, ACM, 2007.

[151] J. Wang, Y. Zhao, X. Wu, and X.-S. Hua, "A transductive multi-label learning approach for video concept detection," *Pattern Recognition*, pp. 298–304, July 2010.

[152] Y. Song, L. Zhang, and C. L. Giles, "A sparse gaussian processes classification framework for fast tag suggestions," in *Proceeding of the 17th*

ACM conference on Information and knowledge management, CIKM '08, (New York, NY, USA), pp. 93–102, ACM, 2008.

[153] J. Fan, Y. Shen, C. Yang, and N. Zhou, "Structured Max-Margin Learning for Inter-Related Classifier Training and Multilabel Image Annotation," *IEEE Transactions on Image Processing*, vol. 20, pp. 837–854, Mar. 2011.

[154] W. Li and M. Sun, "Incorporating Prior Knowledge into Multi-label Boosting for Cross-Modal Image Annotation and Retrieval," in *Information Retrieval Technology* (H. Ng, M.-K. Leong, M.-Y. Kan, and D. Ji, eds.), vol. 4182 of *Lecture Notes in Computer Science*, ch. 31, pp. 404–415, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006.

[155] G. Nasierding and A. Z. Kouzani, "Empirical Study of Multi-label Classification Methods for Image Annotation and Retrieval," in *International Conference on Digital Image Computing*, pp. 617–622, Dec. 2010.

[156] G. Nasierding, G. Tsoumakas, and A. Kouzani, "Clustering Based Multi-Label Classification for Image Annotation and Retrieval," in *2009 IEEE International Conference on Systems, Man, and Cybernetics, IEEE*, 2009.

[157] S. Peters, L. Denoyer, and P. Gallinari, "Iterative Annotation of Multi-relational Social Networks," in *International Conference on Advances in Social Networks Analysis and Mining*, pp. 96–103, Aug. 2010.

[158] S. Nowak, "Overview of the photo annotation task in image-CLEF@ICPR," in *Proceedings of the 20th International conference on Recognizing patterns in signals, speech, images, and videos*, ICPR'10, (Berlin, Heidelberg), pp. 138–151, Springer-Verlag, 2010.

[159] S. Nowak, "Image CLEF@ICPR Contest: Challenges, Methodologies and Results of the Photo Annotation Task," in *20th conference of the International Association for Pattern Recognition*, pp. 489–492, Aug. 2010.

[160] K.-E. Tsay, Y.-L. Wu, M.-K. Hor, and C.-Y. Tang, "Personal Photo Organizer Based on Automated Annotation Framework," in *The Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 507–510, Sept. 2009.

[161] H. Wang, H. Huang, and C. Ding, "Image Categorization Using Directed Graphs," in *Computer Vision , ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6313 of *Lecture Notes in Computer Science*, ch. 55, pp. 762–775, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[162] M. Wang, X. Zhou, and T. S. Chua, "Automatic image annotation via local multi-label classification," in *Proceedings of the 2008 international conference on Content-based image and video retrieval*, CIVR '08, (New York, NY, USA), pp. 17–26, ACM, 2008.

[163] X. Wang, X. Liu, Z. Shi, Z. Shi, and H. Sui, "Voting conditional random fields for multi-label image classification," in *3rd International Congress on Image and Signal Processing*, pp. 1984–1988, Oct. 2010.

[164] X. Xue, H. Luo, and J. Fan, "Structured max-margin learning for multi-label image annotation," in *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, (New York, NY, USA), pp. 82–88, ACM, 2010.

[165] S. Zhang, B. Li, and X. Xue, "Semi-automatic dynamic auxiliary-tag-aided image annotation," *Pattern Recognition*, vol. 43, pp. 470–477, Feb. 2010.

[166] A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, and I. Vlahavas, "An Empirical Study of Multi-label Learning Methods for Video Annotation," *Content-Based Multimedia Indexing, International Workshop on*, vol. 0, pp. 19–24, 2009.

[167] H. Wang, H. Huang, and C. Ding, "Multi-label Feature Transform for Image Classifications," in *Computer Vision ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6314 of *Lecture Notes in Computer Science*, ch. 57, pp. 793–806, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[168] S. S. Bucak, P. Kumar Mallapragada, R. Jin, and A. K. Jain, "Efficient multi-label ranking for multi-class learning: Application to object recognition," in *IEEE International Conference on Computer Vision*, pp. 2098–2105, Sept. 2009.

[169] I. Grinias, N. Komodakis, and G. Tziritas, "Flooding and MRF-based Algorithms for Interactive Segmentation," in *International Conference in Pattern Recognition*, pp. 3943–3946, Aug. 2010.

[170] L. Yang, N. Zheng, M. Chen, Y. Yang, and J. Yang, "Categorization of Multiple Objects in a Scene without Semantic Segmentation," in *Computer Vision ACCV 2009* (H. Zha, R.-i. Taniguchi, and S. Maybank, eds.), vol. 5994 of *Lecture Notes in Computer Science*, ch. 28, pp. 303–312, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[171] S. Liapis, E. Sifakis, and G. Tziritas, "Colour and texture segmentation using wavelet frame analysis, deterministic relaxation, and fast marching algorithms," *Journal of Visual Communication and Image Representation*, vol. 15, pp. 1–26, Mar. 2004.

[172] M. A. Tahir, J. Kittler, F. Yan, and K. Mikolajczyk, "Kernel Discriminant Analysis Using Triangular Kernel for Semantic Scene Classification," *International Workshop on Content-Based Multimedia Indexing*, vol. 0, pp. 1–6, 2009.

[173] H. Wang and J. Hu, "Multi-label image annotation via Maximum Consistency," in *International Conference on Image Processing*, pp. 2337–2340, Sept. 2010.

[174] W. Zong and G.-B. Huang, "Face recognition based on extreme learning machine," *Neurocomputing*, May 2011.

[175] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and Simile Classifiers for Face Verification," in *In IEEE International Conference on Computer Vision (ICCV*, 2009.

[176] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Describable visual attributes for face verification and image search," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, October 2011.

[177] G. Nasierding and A. Kouzani, "Image to Text Translation by Multi-Label Classification," in *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence* (D.-S. Huang, X. Zhang, C. Reyes García, and L. Zhang, eds.), vol. 6216 of *Lecture Notes in Computer Science*, ch. 31, pp. 247–254, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[178] Shun, Leung, W. Lam, K. S. Leung, and J. C. Cheng, "Medical data mining using evolutionary computation," *Artificial Intelligence in Medicine*, vol. 16, no. 1, pp. 73–96, 1999.

[179] H. Wang, C. Lin, Y. Peng, and X. Hu, "Application of improved random forest variables importance measure to traditional Chinese chronic gastritis diagnosis," in *IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on*, pp. 84–89, Dec. 2008.

[180] K. C. Tan, Q. Yu, C. M. Heng, and T. H. Lee, "Evolutionary computing for knowledge discovery in medical diagnosis," *Artificial Intelligence in Medicine*, vol. 27, pp. 129–154, February 2003.

[181] J.-S. Lin, Lo, A. Hasegawa, M. T. Freedman, and S. K. Mun, "Reduction of false positives in lung nodule detection using a two-level neural classification," *IEEE Transactions on Medical Imaging*, vol. 15, pp. 206–217, Apr. 1996.

[182] L. Michielan, L. Terfloth, J. Gasteiger, and S. Moro, "Comparison of multilabel and single-label classification applied to the prediction of the isoform specificity of cytochrome p450 substrates.," *Journal of chemical information and modeling*, vol. 49, pp. 2588–2605, Nov. 2009.

[183] M. A. Mammadov, A. M. Rubinov, and J. Yearwood, "The study of drug-reaction relationships using global optimization techniques," *Optimization Methods Software*, vol. 22, pp. 99–126, Feb. 2007.

[184] L. Miclielan, F. Stephanie, L. Terfloth, D. Hristozov, B. Cacciari, K. N. Klotz, G. Spalluto, J. Gasteiger, and S. Moro, "Exploring potency and selectivity receptor antagonist profiles using a multilabel classification approach: The human adenosine receptors as a key study," *Journal of Chemical Information and Modeling*, vol. 49, no. 12, pp. 2820–2836, 2009.

[185] H. Su, M. Heinonen, and J. Rousu, "Structured Output Prediction of Anti-cancer Drug Activity," in *Pattern Recognition in Bioinformatics* (T. Dijkstra, E. Tsivtsivadze, E. Marchiori, and T. Heskes, eds.),

vol. 6282 of *Lecture Notes in Computer Science*, ch. 4, pp. 38–49, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[186] R. Rak, L. A. Kurgan, and M. Reformat, "Multilabel associative classification categorization of medline articles into mesh keywords," *IEEE Engineering in Medicine and Biology Magazine*, vol. 26, no. 2, pp. 47–55, 2007.

[187] Y. Sasaki, B. Rea, and S. Ananiadou, "Multi-topic aspects in clinical text classification," in *BIBM '07: Proceedings of the 2007 IEEE International Conference on Bioinformatics and Biomedicine*, (Washington, DC, USA), pp. 62–70, IEEE Computer Society, 2007.

[188] N. Barrett and J. H. Weber-Jahnke, "Applying natural language processing toolkits to electronic health records - an experience report.," *Studies in health technology and informatics*, vol. 143, pp. 441–446, 2009.

[189] I. Goldstein and O. Uzuner, "Role of preferred terminology in the classification of medical reports," in *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, pp. 260–264, Sept. 2009.

[190] D. Wu, L. Lu, J. Bi, Y. Shinagawa, K. Boyer, A. Krishnan, and M. Salganicoff, "Stratified learning of local anatomical context for lung nodules in CT images," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2791–2798, June 2010.

[191] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical annotation of medical images," *Pattern Recognition*, vol. 44, pp. 2436–2449, Oct. 2011.

[192] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "ImageCLEF 2009 Medical Image Annotation Task: PCTs for Hierarchical Multi-Label Classification," in *Multilingual Information Access Evaluation II. Multimedia Experiments* (C. Peters, B. Caputo, J. Gonzalo, G. Jones, J. Kalpathy-Cramer, H. Müller, and T. Tsikrika, eds.), vol. 6242 of *Lecture Notes in Computer Science*, ch. 28, pp. 231–238, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[193] F. Pachet and P. Roy, "Improving Multilabel Analysis of Music Titles: A Large-Scale Validation of the Correction Approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, pp. 335–343, Feb. 2009.

[194] A. Wieczorkowska and P. Synak, "Quality Assessment of k-NN Multi-label Classification for Music Data," in *Foundations of Intelligent Systems* (F. Esposito, Z. Ras, D. Malerba, and G. Semeraro, eds.), vol. 4203 of *Lecture Notes in Computer Science*, ch. 45, pp. 389–398, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2006.

[195] W. Jiang, A. Cohen, and Z. Raś, "Polyphonic Music Information Retrieval Based on Multi-label Cascade Classification System," in *Advances in Information and Intelligent Systems* (Z. Ras and W. Ribarsky, eds.), vol. 251 of *Studies in Computational Intelligence*, ch. 6, pp. 117–137, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009.

[196] H.-Y. Lo, J.-C. Wang, H.-M. Wang, and S.-D. Lin, "Cost-Sensitive Multi-Label Learning for Audio Tag Annotation and Retrieval," *Multimedia, IEEE Transactions on*, vol. 13, pp. 518–529, June 2011.

[197] A. Ma, I. Sethi, and N. Patel, "Multimedia Content Tagging Using Multilabel Decision Tree," in *EEE International Symposium on Multimedia*, pp. 606–611, 2009.

[198] T. Li and M. Ogihara, "Detecting emotion in music," *Proceedings of the fourth international conference on music information retrieval (ISMIR, 2003)*, October 2003.

[199] A. Wieczorkowska, P. Synak, and Z. Raś, "Multi-label classification of emotions in music," in *Proceedings of the 2006 International Conference on Intelligent Information Processing and Web Mining (IIPWM'06)*, pp. 307–315, 2006.

[200] E. Özpolat and G. B. Akar, "Automatic detection of learning styles for an e-learning system," *Comput. Educ.*, vol. 53, pp. 355–367, Sept. 2009.

[201] C. Ganapathy, J.-H. Kang, E. Shaw, and J. Kim, "Classification Techniques for Assessing Student Collaboration in Shared Wiki Spaces," in *Artificial Intelligence in Education* (G. Biswas, S. Bull, J. Kay, and A. Mitrovic, eds.), vol. 6738 of *Lecture Notes in Computer Science*, ch. 68, pp. 456–458, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2011.

[202] I. Monroy, G. Escudero, and M. Graells, *Anomaly detection in batch chemical processes*, vol. 26 of *Computer Aided Chemical Engineering*, pp. 255–260. Elsevier, 2009.

[203] S. N. Ahsan and F. Wotawa, "Impact analysis of SCRs using single and multi-label machine learning classification," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, (New York, NY, USA), ACM, 2010.

[204] S. Vogrincic and Z. Bosnic, "Multi-Label Document Classification Based on Economic Ontology," in *2nd Workshop on Learning from Multi-Label Data (MLD'10)*, pp. 69–76, June 2010.

[205] K. Ozonat and D. Young, "Towards a universal marketplace over the web: statistical multi-label classification of service provider forms with simulated annealing," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 1295–1304, ACM, 2009.

[206] D. Thorleuchter, D. V. den Poel, and A. Prinzie, "A compared R&D-based and patent-based cross impact analysis for identifying relationships between technologies," *Technological Forecasting and Social Change*, vol. 77, pp. 1037–1050, Sept. 2010.

[207] A. C. Ferguson-Smith, Y.-F. Chen, M. S. Newman, L. T. May, P. B. Sehgal, and F. H. Ruddle, "Regional localization of the interferon-2b-cell stimulatory factor 2/hepatocyte stimulating factor gene to human chromosome 7p15-p21," *Genomics*, vol. 2, no. 3, pp. 203 – 208, 1988.

[208] H. Wang, C. Ding, and H. Huang, "Multi-label Linear Discriminant Analysis," in *Computer Vision ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6316 of *Lecture Notes in Computer Science*, ch. 10, pp. 126–139, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010.

[209] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," in *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, pp. 75–83, 2008.

[210] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, pp. 50–65, 2008.

[211] A. Srivastava and B. Zane-Ulman, "Discovering recurring anomalies in text reports regarding complex space systems," in *Aerospace Conference, 2005 IEEE*, pp. 3853–3862, March 2005.

[212] M. Worring, C. G. M. Snoek, O. de Rooij, G. P. Nguyen, and A. W. M. Smeulders, "The mediamill semantic video search engine," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–1213–IV–1216, 2007.

[213] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel, and H. de Garis, "An overview of evolutionary computation," in *European Conference on Machine Learning* (P. Brazdil, ed.), vol. 667 of *Lecture Notes in Computer Science*, pp. 442–459, Springer, April 1993.

[214] L. Fogel, A. Owens, and M. Walsh, *Artificial intelligence through simulated evolution*. Chichester, WS, UK: Wiley, 1966.

[215] H. G. Beyer and H. P. Schwefel, "Evolution strategies a comprehensive introduction," *Natural Computing*, vol. 1, pp. 3–52, May 2002.

[216] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[217] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[218] N. L. Cramer, "A Representation for the Adaptive Generation of Simple Sequential Programs," in *Proceedings of an International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, July 24-26, 1985* (J. J. Grefenstette, ed.), (Hillsdale NJ), Lawrence Erlbaum Associates, 1985.

[219] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. San Francisco, CA, USA: Morgan Kaufmann, Jan. 1998.

[220] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40(2), pp. 121–144, 2010. accepted for publication.

[221] D. J. Montana, "Strongly typed genetic programming," *Evolutionary Computation*, vol. 3, no. 2, pp. 199–230, 1995.

[222] P. A. Whigham, *Grammatical Bias for Evolutionary Learning*. PhD thesis, School of Computer Science. University College, University of New South Wales, Australia, 1996.

[223] M. Wong and K. Leung, *Data Mining Using Grammar-Based Genetic Programming and Applications*. Genetic Programming Series, Kluwer Academic Publishers, 2002.

[224] M. Oltean and L. Diosan, "An autonomous GP-based system for regression and classification problems," *Applied Soft Computing*, vol. 9, pp. 49–60, January 2009.

[225] J. C. Bezdek, S. Boggavarapu, L. O. Hall, and A. Bensaid, "Genetic algorithm guided clustering," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, (Orlando, Florida, USA), pp. 34–39, IEEE, June 1994.

[226] L. Jie, G. Xinbo, and J. Li-cheng, "A GA-based clustering algorithm for large data sets with mixed numeric and categorical values," in *Proceedings of the 5th International Conference on Computational Intelligence and Multimedia Applications*, (Xián, China), pp. 102–107, IEEE, September 2003.

[227] M. Lyman and G. Lewandowski, "Genetic programming for association rules on card sorting data," in *Genetic and Evolutionary Computation*

Conference (H.-G. Beyer and U.-M. O'Reilly, eds.), (Washington DC, USA), pp. 1551–1552, ACM, June 2005.

[228] S. K. Murthy, "Automatic construction of decision trees from data: a multi-disciplinary survey," *Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 345–389, 1998.

[229] G. Folino, C. Pizzuti, and G. Spezzano, "A cellular genetic programming approach to classification," in *Proceedings of the Genetic and Evolutionary Computation Conference* (W. Banzhaf, J. M. Daida, G. E. Eiben, M. H. Garzon, V. Honavar, M. J. Jakiela, and R. E. Smith, eds.), pp. 1015–1020, Morgan Kaufmann, July 1999.

[230] G. Folino, C. Pizzuti, and G. Spezzano, "Genetic programming and simulated annealing: a hybrid method to evolve decission trees," in *Genetic Programming, EuroGP 2000* (R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, and T. C. Fogarty, eds.), vol. 1802 of *Lecture Notes in Computer Science*, pp. 294–303, Springer-Verlag, April 2000.

[231] J. R. Koza, "Concept formation and decision tree induction using the genetic programming paradigm," in *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature* (H.-P. Schwefel and R. Manner, eds.), vol. 496 of *Lecture Notes in Computer Science*, pp. 124–128, Springer-Verlag, October 1990.

[232] S. Oka and Q. Zhao, "Design of decision trees through integration of C4.5 and GP," in *Proceedings of the 4th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, pp. 128–135, 2000.

[233] J. Yu, J. Yu, A. A. Almal, S. M. Dhanasekaran, D. Ghosh, W. P. Worzel, and A. M. Chinnaiyan, "Feature selection and molecular classification of cancer using genetic programming," *Neoplasia*, vol. 9, pp. 292–303, April 2007.

[234] P. J. Bentley, ""Evolutionary, my dear Watson" - Investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims," in *Proceedings of the Genetic and Evolutionary Computation Conference* (D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer, eds.), pp. 702–709, Morgan Kaufmann, July 2000.

[235] M. L. Wong and K. S. Leung, *Data Mining using Grammar-Based Genetic Programming and Applications*. Kluwer Academic Publishers, 2000.

[236] A. Tsakonas, "A comparison of classification accuracy of four genetic programming-evolved intelligent structures," *Information Sciences*, vol. 176, pp. 691–724, March 2006.

[237] R. J. Gilbert, J. J. Rowland, and D. B. Kell, "Genomic computing: explanatory modelling for functional genomics," in *Proceedings of the Genetic and Evolutionary Computation Conference* (D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, and H.-G. Beyer, eds.), pp. 551–557, Morgan Kaufmann, July 2000.

[238] C. C. Bojarczuk, H. S. Lopes, A. A. Freitas, and E. L. Michalkiewicz, "A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets," *Artificial Intelligence in Medicine*, vol. 30, no. 1, pp. 27–48, 2004.

[239] J. Eggermont, A. E. Eiben, and J. I. van Hemert, "Adapting the fitness function in GP for data mining," in *Genetic Programming, Second European Workshop, EuroGP '99* (R. Poli, P. Nordin, W. B. Langdon, and T. C. Fogarty, eds.), vol. 1598 of *Lecture Notes in Computer Science*, pp. 193–202, Springer-Verlag, May 1999.

[240] J. Eggermont, A. E. Eiben, and J. I. van Hemert, "A comparison of genetic programming variants for data classification," in *Advances*

in Intelligent Data Analysis, Third International Symposium, IDA-99 (D. J. Hand, J. N. Kok, and M. R. Berthold, eds.), vol. 1642 of *Lecture Notes in Computer Science*, pp. 281–290, Springer-Verlag, August 1999.

[241] C. C. Bojarczuk, H. S. Lopes, and A. A. Freitas, "Genetic programming for knowledge discovery in chest pain diagnosis," *IEEE Engineering in Medicine and Biology Magazine*, vol. 19, pp. 38–44, July/August 2000.

[242] I. De Falco, A. Della Cioppa, and E. Tarantino, "Discovering interesting classification rules with genetic programming," *Applied Soft Computing Journal*, vol. 1, no. 4, pp. 257–269, 2002.

[243] R. Cattral, F. Oppacher, and D. Deugo, "Supervised and unsupervised data mining with an evolutionary algorithm," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 2, pp. 767–774, IEEE, IEEE, May 2001.

[244] L. M. Howard and D. J. DÁngelo, "The GA-P: a genetic algorithm and genetic programming hybrid," *IEEE Expert*, vol. 10, pp. 11–15, June 1995.

[245] H. F. Gray, R. J. Maxwell, I. Martínez-Pérez, C. Arús, and S. Cerdán, "Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra," in *Proceedings of the 1st Annual Conference on Genetic Programming* (J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, eds.), (Stanford, California, USA), p. 424, MIT Press, July 1996.

[246] D. Hope, E. Munday, and S. Smith, "Evolutionary algorithms in the classification of mammograms," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing*, pp. 258–265, IEEE, April 2007.

[247] G. C. Wilson and M. I. Heywood, "Introducing probabilistic adaptive mapping developmental genetic programming with redundant mappings," *Genetic Programming and Evolvable Machines*, vol. 8, pp. 187–220, June 2007.

[248] S. Sette, B. Wyns, and L. Boullart, "Comparing learning classifier systems and genetic programming: a case study," *Engineering Applications of Artificial Intelligence*, vol. 17, pp. 199–204, March 2004.

[249] P. J. Rauss, J. M. Daida, and S. A. Chaudhary, "Classification of spectral image using genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference* (L. D. Whitley, D. E. Goldberg, E. CantuPaz, L. Spector, I. C. Parmee, and H.-G. Beyer, eds.), (Las Vegas, Nevada, USA), pp. 726–733, Morgan Kaufmann, July 2000.

[250] N. Petrović and V. S. Crnojević, "Impulse noise detection based on robust statistics and genetic programming," in *Proceedings of the 7th International Conference on Advanced Concepts for Intelligent Vision Systems* (J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, eds.), vol. 3708 of *Lecture Notes in Computer Science*, (Antwerp, Belgium), pp. 643–649, Springer, September 2005.

[251] S. Silva and Y.-T. Tseng, "Classification of seafloor habitats using genetic programming," in *Applications of Evolutionary Computing. EvoWorkshops 2008* (M. Giacobini, A. Brabazon, S. Cagnoni, G. D. Caro, R. Drechsler, A. Ekárt, A. Esparcia-Alcázar, M. Farooq, A. Fink, J. McCormack, M. OÑeill, J. Romero, F. Rothlauf, G. Squillero, S. Uyar, and S. Yang, eds.), vol. 4974 of *Lecture Notes in Computer Science*, (Naples, Italy), pp. 315–324, Springer, March 2008.

[252] Z. Chen and S. Lu, "A genetic programming approach for classification of textures based on wavelet analysis," in *Proceedings of the International Symposium on Intelligent Signal Processing*, pp. 1–6, IEEE, October 2007.

[253] J.-Y. Lin, H.-R. Ke, B.-C. Chien, and W.-P. Yang, "Designing a classifier by a layered multi-population genetic programming approach," *Pattern Recognition*, vol. 40, pp. 2211–2225, August 2007.

[254] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, "Application of genetic programming for multicategory pattern classification," *IEEE Transactions on Evolutionary Computation*, vol. 4, pp. 242–258, September 2000.

[255] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 183–196, April 2004.

[256] M. Zhang, X. Gao, and W. Lou, "A new crossover operator in genetic programming for object classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 37, pp. 1332–1343, October 2007.

[257] S. Winkler, M. Affenzeller, and S. Wagner, "Advanced genetic programming based machine learning," *Journal of Mathematical Modelling and Algorithms*, vol. 6, no. 3, pp. 455–480, 2007.

[258] M. Zhang and P. Wong, "Genetic programming for medical classification: a program simplification approach," *Genetic Programming and Evolvable Machines*, vol. 9(3), pp. 229–255, 2008.

[259] J.-J. Huang, G.-H. Tzeng, and C.-S. Ong, "Two-stage genetic programming (2SGP) for the credit scoring model," *Applied Mathematics and Computation*, vol. 174, pp. 1039–1053, March 2006.

[260] M. D. Ritchie, A. A. Motsinger, W. S. Bush, C. S. Coffey, and J. H. Moore, "Genetic programming neural networks: a powerful bioinformatics tool for human genetics," *Applied Soft Computing*, vol. 7, pp. 471–479, January 2007.

[261] A. Tsakonas, G. Dounias, M. Doumpos, and C. Zopounidis, "Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming," *Expert Systems with Applications*, vol. 30, pp. 449–461, April 2006.

[262] V. N. Vapnik, *The Nature of Statistical Learning*. Springer, 2nd ed., 2000.

[263] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[264] L. Diosan, A. Rogozan, and J.-P. Pécuchet, "Optimising multiple kernels for SVM by genetic programming," in *Proceedings of the 8th European Conference on Evolutionary Computation in Combinatorial Optimization* (J. I. van Hemert and C. Cotta, eds.), vol. 4972 of *Lecture Notes in Computer Science*, (Naples, Italy), pp. 230–241, Springer, March 2008.

[265] M. Gîrdea and L. Ciortuz, "A hybrid genetic programming and boosting technique for learning kernel functions from training data," in *Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (V. Negru, T. Jebelean, D. Petcu, and D. Zaharie, eds.), (Timisoara, Romania), pp. 395–402, IEEE, September 2007.

[266] T. Phienthrakul and B. Kijsirikul, "GPES: an algorithm for evolving hybrid kernel functions of support vector machines," in *Proceedings of*

the IEEE Congress on Evolutionary Computation (D. Srinivasan and L. Wang, eds.), (Singapore), pp. 2636–2643, IEEE, IEEE, September 2007.

[267] T. Watson and T. Rakowski, "Data mining with an evolving population of database queries," in *Proceedings of the MENDEL'95* (P. Osmera, ed.), pp. 169–174, 1995.

[268] A. A. Freitas, "A genetic programming framework for two data mining tasks: classification and generalized rule induction," in *Proceedings of the 2nd Annual Conference on Genetic Programming*, pp. 96–101, Morgan Kaufmann, 1997.

[269] A. Majid, A. Khan, and A. M. Mirza, "Improving performance of nearest neighborhood classifier using genetic programming," in *Proceedings of the 2004 IEEE International Conference on Machine Learning and Applications*, (Louisville, Kentucky, USA), IEEE, December 2004.

[270] C. Ferreira, *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. Springer, 2nd ed., May 2006.

[271] C. Ferreira, "Discovery of the boolean functions to the best density-classification rules using gene expression programming," in *Proceedings of the 4th European Conference on Genetic Programming, Lecture Notes in Computer Science*, pp. 51–60, 2002.

[272] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving classification rules with gene expression programming," *IEEE Transactions on evolutionary computation.*, vol. 7, 2003.

[273] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Trans. Evolutionary Computation*, vol. 7, pp. 519–531, Jan. 2004.

[274] S. Dehuri and S.-B. Cho, "Multi-objective classification rule mining using gene expression programming," in *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology - Volume 02*, (Washington, DC, USA), pp. 754–760, IEEE Computer Society, 2008.

[275] J. Huang and C. Deng, "A novel multiclass classification method with gene expression programming," in *Proceedings of the 2009 International Conference on Web Information Systems and Mining*, WISM '09, (Washington, DC, USA), pp. 139–143, IEEE Computer Society, 2009.

[276] Z. Wu and M. Yao, "A new gep algorithm based on multi-phenotype chromosomes," in *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering - Volume 01*, IWCSE '09, (Washington, DC, USA), pp. 204–209, IEEE Computer Society, 2009.

[277] Q. Li, W. Wang, S. Han, and J. Li, "Evolving Classifier Ensemble With Gene Expression Programming," in *Third International Conference on Natural Computation, 2007. ICNC 2007.*, vol. 3, pp. 546–550, 2007.

[278] J. Zuo, C. Tang, and T. Zhang, "Mining Predicate Association Rule by Gene Expression Programming," in *Advances in Web-Age Information Management* (X. Meng, J. Su, and Y. Wang, eds.), vol. 2419 of *Lecture Notes in Computer Science*, ch. 9, pp. 281–294–294, Berlin, Heidelberg: Springer Berlin / Heidelberg, Aug. 2002.

[279] L. Duan, C. Tang, L. Tang, J. Zuo, and T. Zhang, "An effective microarray data classifier based on gene expression programming," in *Proceedings of the 2009 Fifth International Conference on Natural Computation - Volume 04*, ICNC '09, (Washington, DC, USA), pp. 523–527, IEEE Computer Society, 2009.

[280] V. Karakasis and A. Stafylopatis, "Efficient Evolution of Accurate Classification Rules Using a Combination of Gene Expression Programming and Clonal Selection," *IEEE Trans. Evolutionary Computation*, vol. 12, no. 6, pp. 662–678, 2008.

[281] X. Liu, Z. Cai, and W. Gong, "An improved gene expression programming for fuzzy classification," in *Proceedings of the 3rd International Symposium on Advances in Computation and Intelligence*, ISICA '08, (Berlin, Heidelberg), pp. 520–529, Springer-Verlag, 2008.

[282] M. H. Marghny and I. E. El-Semman, "Extracting fuzzy classification rules with gene expression programming," in *Proceedings of the International Conference on Artificial Intelligence and Machine Learning, AIML 2005*, 2005.

[283] J. Jedrzejowicz and P. Jedrzejowicz, "Gep-induced expression trees as weak classifiers," in *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, vol. 5077 of *Lecture Notes in Computer Science*, pp. 129–141, Springer Berlin / Heidelberg, 2008.

[284] J. Jedrzejowicz and P. Jedrzejowicz, "Cellular gep-induced classifiers," in *Proceedings of the Second international conference on Computational collective intelligence: technologies and applications - Volume PartI*, ICCCI 10, (Berlin, Heidelberg), pp. 343–352, Springer-Verlag, 2010.

[285] J. Jedrzejowicz and P. Jedrzejowicz, "A family of gep-induced ensemble classifiers," in *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, vol. 5796 of *Lecture Notes in Computer Science*, pp. 641–652, Springer Berlin / Heidelberg, 2009.

[286] Q. Yu, K. Tan, and T. Lee, "Knowledge discovery in data mining via an evolutionary algorithm," *Evolutionary Computation in Data Mining*, pp. 101–123, 2005.

[287] P. S. Ngan, K. S. Leung, M. L. Wong, and J. C. Y. Cheng, "Using grammar based genetic programming for data mining of medical knowledge," *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp. 254–259, 1998.

[288] W. Lu and I. Traore, "Detecting new forms of network intrusion using genetic programming," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 3, pp. 2165–2172 Vol.3, 2003.

[289] M. L. Zhang and Z. H. Zhou, "Multi-label neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[290] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Trans. Evolutionary Computation*, vol. 7, no. 6, pp. 519–531, 2003.

[291] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[292] S. Garcia and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[293] S. Ventura, C. Romero, A. Zafra, J. A. Delgado, and C. Hervás, "JCLEC: A Java framework for evolutionary computation," *Soft Computing*, vol. 12, no. 4, pp. 381–392, 2008.

[294] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[295] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.