UNIVERSIDAD DE GRANADA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS
INFORMÁTICA Y DE TELECOMUNICACIÓN

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E
INTELIGENCIA ARTIFICIAL

Algoritmos para la Clasificación Multinstancia

Memoria de tesis presentada por

Dánel Sánchez Tarragó

como requisito para optar al grado de

DOCTOR EN INFORMÁTICA

Granada

Junio de 2014

Editor: Editorial de la Universidad de Granada
Autor: Dánel Sánchez Tarragó
D.L.: GR 2128-2014
ISBN: 978-84-9083-148-9

UNIVERSIDAD DE GRANADA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E
INTELIGENCIA ARTIFICIAL

Algoritmos para la Clasificación Multinstancia

Memoria de tesis presentada por

Dánel Sánchez Tarragó

como requisito para optar al grado de

DOCTOR EN INFORMÁTICA

DIRECTORES

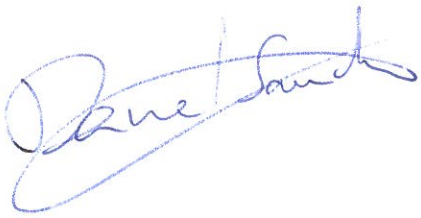
Francisco Herrera Chris Cornelis Rafael Bello

Granada

Junio de 2014

El doctorando Dánel Sánchez Tarragó y los directores de la tesis Francisco Herrera, Chris Cornelis y Rafael Bello garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Junio 2014

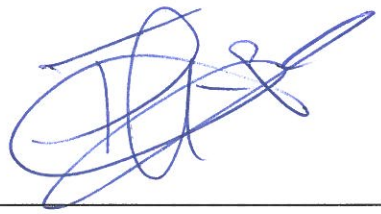


Fdo. Dánel Sánchez Tarragó

Doctorando

Fdo. Chris Cornelis

Director



Fdo. Francisco Herrera

Director

Fdo. Rafael Bello

Director

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología (CICYT) con el proyecto TIN2011-28488 y por la Junta de Andalucía con los proyectos de excelencia P10-TIC-6858 y P12-TIC-2958. También ha recibido el apoyo invaluable de la AUIP.

Resumen

Dentro del campo de estudio del aprendizaje automático, la clasificación multinstancia tiene como objetivo construir, a partir de un conjunto de ejemplos, un modelo matemático que permita clasificar objetos descritos por múltiples vectores de atributos. En la tesis se realiza una profunda revisión bibliográfica que contribuye a organizar la literatura dispersa y sistematizar el conocimiento existente sobre clasificación multinstancia, y se proponen dos soluciones para sendos problemas identificados en el campo de la clasificación multinstancia.

Uno de los problemas identificados es el problema de las clases desbalanceadas, el cual ocurre cuando hay mucha diferencia en el tamaño de las clases y provoca que el modelo de aprendizaje inducido a partir de los datos no represente adecuadamente el concepto que se pretende aprender y consecuentemente incurra en muchos errores de clasificación. En la tesis se propone un nuevo método general basado en técnicas de muestreo para enfrentar este problema en la clasificación multinstancia. El algoritmo propuesto, bautizado como MISMOTE, consiste a groso modo en modificar la distribución de los datos de entrenamiento, insertando ejemplos sintéticos de objetos multinstancia en la clase minoritaria, para equilibrar el balance de las clases. Las pruebas experimentales validadas por métodos estadísticos mostraron que la solución propuesta mejora significativamente la calidad de la clasificación y es competitiva con otras soluciones existentes en la literatura.

El otro problema identificado es que aún no han sido adaptados al enfoque del aprendizaje multinstancia algunos algoritmos de clasificación tradicionales orientados específicamente a determinados dominios de aplicación. En la tesis se propone un nuevo algoritmo de clasificación multinstancia que permite aplicar el clasificador de Rocchio en el escenario multinstancia. El clasificador de Rocchio es muy usado para la recuperación y clasificación de documentos textuales en aplicaciones *on-line*. El algoritmo propuesto, bautizado como MIRocchio, implementa una nueva hipótesis multinstancia introducida en la tesis, la cual está basada en proporción de instancias positivas. La validez del algoritmo se comprueba experimentalmente en el problema de la recomendación de páginas web índices, siendo capaz de mejorar el desempeño de los clasificadores multinstancia del estado del arte en este dominio de aplicación.

Índice general

Índice de figuras

Índice de cuadros

Introducción

Planteamiento

La construcción de las primeras computadoras digitales a mediados del siglo XX, propiciada por el desarrollo de la electrónica, dio inicio a un nuevo período en la historia de la humanidad conocido como revolución informática, caracterizado por avances tecnológicos sin precedentes que facilitan la creación, distribución y manipulación de la información. La revolución informática ha tenido un profundo impacto en todas las áreas de la actividad humana, cambiando desde la forma en que se comunican las personas hasta cómo se toman las decisiones. Teléfonos que indican al usuario el camino a seguir para llegar a un destino, edificios inteligentes que aprenden las características de sus habitantes para ahorrar energía y aumentar el confort, coches que parquean por sí mismos, son algunos ejemplos de la introducción de la informática en todos los ámbitos de la vida cotidiana.

La gestión automática de la información y la inteligencia artificial son disciplinas que están en el centro de esta revolución. Una de las ramas de la inteligencia artificial que ha alcanzado mayor auge es el aprendizaje automático, el cual se ocupa de la construcción y estudio de sistemas que pueden aprender de los datos, i.e., sin necesidad de programar explícitamente el nuevo conocimiento adquirido. El auge del aprendizaje automático se debe a su gran aplicabilidad: prácticamente todo conocimiento es susceptible de ser aprendido, si bien el uso del aprendizaje automático en determinadas aplicaciones puede estar limitado por razones técnicas (como por ejemplo, el acceso a los datos), económicas (por ejemplo, los costos de desarrollo), legales u otras. Los métodos del aprendizaje automático son usados como herramientas básicas en otras ramas de la inteligencia artificial, como la minería de datos y el reconocimiento de patrones.

Uno de los tipos de problema más importante que busca resolver el aprendizaje automático es el problema de clasificación. En este tipo de problema se parte de un conjunto de objetos de clasificación, cada uno descrito por un vector de valores (cada valor describiendo a un atributo) y una etiqueta de clase. Los objetos de clasificación se toman como ejemplo para inducir un modelo matemático capaz de predecir las etiquetas de clase de

nuevos objetos carentes de clasificación. El aprendizaje automático ha producido una variedad importante de soluciones para determinados tipos de problema de clasificación. Mayormente, estas soluciones han estado enfocadas a problemas donde los objetos de clasificación están descritos por un único vector de atributos, de tal forma que a este tipo de problemas de clasificación se le ha identificado con el epíteto «tradicional».

Los problemas de clasificación tradicional son relativamente simples de modelar y pueden encontrarse en un número importante de aplicaciones. Sin embargo, otras aplicaciones demandan formas distintas, y por lo general más complejas, de modelar el problema de clasificación. Una de las alternativas que han surgido recientemente al modelo de clasificación tradicional es la clasificación multinstancia [27], en la cual cada objeto de clasificación está descrito, ya no por uno, sino por *múltiples* vectores de atributos. Cada vector de atributos representa una *instancia* del objeto, y cada objeto de clasificación se conoce comúnmente como *bolsa*. Las etiquetas de clase se asocian con las bolsas pero no con las instancias, sin embargo es la información contenida en las instancias la que determina en último caso la etiqueta de clase de las bolsas. El objetivo de la clasificación multinstancias es aprender un modelo matemático que represente la relación que existe entre las instancias y la etiqueta de la bolsa para predecir las etiquetas de clase de nuevas bolsas carentes de clasificación.

El aprendizaje multinstancia ha despertado un amplio interés debido fundamentalmente a su capacidad para modelar problemas de clasificación con datos ambiguos y problemas con la relación *uno-a-muchos* entre sus datos, la cual es característica del aprendizaje relacional. Múltiples estudios empíricos han mostrado, para estos tipos de problema, la superioridad del enfoque multinstancia sobre los algoritmos de clasificación tradicional, los cuales tratan a todos los datos por igual sin tomar en cuenta las relaciones que puedan existir entre ellos. Se han diseñado algunos algoritmos específicos para el aprendizaje de problemas de clasificación multinstancia. Adicionalmente, muchos algoritmos de clasificación basados en métodos tradicionales como los árboles y reglas de decisión, las máquinas de soporte vectorial, la máxima verosimilitud y los métodos basados en instancias han sido adaptados al enfoque de aprendizaje multinstancia. Varios de estos algoritmos han sido usados con éxito en aplicaciones de campos tan diversos como la bioinformática, las finanzas, el diagnóstico médico o la recuperación de información. Cada día aparecen nuevos problemas de clasificación que pueden beneficiarse con los algoritmos del enfoque multinstancia.

A pesar de los avances obtenidos en el desarrollo y aplicación de métodos de clasificación multinstancia, ésta es un área de investigación muy joven dentro del aprendizaje automático, y tiene aún muchos aspectos sin resolver que requieren ser estudiados. Es

por ello que en esta tesis nos hemos dado a la tarea de identificar algunos de estos problemas y proponerles una solución.

Particularmente, uno de los problemas que afecta a la clasificación multinstancia es el problema de las clases desbalanceadas [51]. Este problema ocurre cuando hay mucha diferencia en el tamaño de las clases y provoca que el modelo de aprendizaje inducido a partir de los datos no represente adecuadamente el concepto que se pretende aprender y consecuentemente incurra en muchos errores de clasificación. El problema de clases desbalanceadas fue identificado originalmente en el escenario de la clasificación tradicional. En este escenario ha sido propuesta una gran variedad de soluciones[1] para el problema de clases desbalanceadas. Lamentablemente, los algoritmos propuestos en estas soluciones no pueden ser aplicados para enfrentar el problema de clases desbalanceadas en la clasificación multinstancia debido a las diferencias entre estos dos enfoques del problema de clasificación. El problema de clases desbalanceadas en la clasificación multinstancia ha recibido poca atención hasta el momento, y solo aparecen escasas discusiones sobre él y aisladas soluciones en la literatura. En el escenario de la clasificación tradicional, uno de los métodos más usados para tratar el problema de clases desbalanceadas consiste en modificar la distribución de los datos de entrenamiento a través técnicas de muestreo para equilibrar el balance de las clases. Hasta donde hemos podido conocer, no existe ninguna solución al problema de clases desbalanceadas para la clasificación multinstancia que esté basada en técnicas de muestreo. En esta tesis se propone un nuevo método general basado en técnicas de muestreo para enfrentar el problema de clases desbalanceadas en la clasificación multinstancia.

Otro de los puntos que necesitan más trabajo en la clasificación multinstancia es el desarrollo de algoritmos de clasificación orientados a determinados dominios de competencia. En el aprendizaje automático existe el teorema *No-Free-Lunch* (NFL) [118], que establece que no existe un algoritmo de clasificación que sea el óptimo en todos los dominios de competencia, i.e., necesariamente habrán dominios de competencia en los cuales un clasificador alcance resultados inferiores a los de otros clasificadores. La mayoría de los algoritmos desarrollados para la clasificación multinstancia han sido concebidos como soluciones de propósito general, y por tanto no obtienen generalmente ventajas de la información específica de ningún dominio de aplicación. Nosotros nos enfocamos aquí en el problema de la recuperación y clasificación de documentos textuales, el cual puede modelarse de una forma natural como un problema de clasificación multinstancia. Sin

[1] Aunque se les llama «soluciones», dando la impresión de que los métodos actuales dejan completamente resuelto el problema de clases desbalanceadas, realmente lo que hacen estos métodos es mitigar el problema hasta cierto grado (mayor o menor).

embargo, se han desarrollado muy pocos algoritmos de clasificación multinstancia para este dominio de aplicación en particular. Uno de los algoritmos de clasificación que han sido usados tradicionalmente en este campo es el clasificador de Rocchio [52]. Este algoritmo es muy valorado por su eficiencia, lo cual hace muy atractiva su aplicación en colecciones documentales muy grandes y cuando se necesitan tiempos de respuesta muy cortos, como por ejemplo en las aplicaciones *on-line*. Dado que no existen adaptaciones del algoritmo de clasificación de Rocchio al aprendizaje multinstancia, este es otro de los objetivos de la tesis.

Objetivos

A partir de lo expuesto anteriormente podemos declarar que el objetivo general de la tesis es proponer nuevos métodos de clasificación multinstancia, particularmente, abordando el problema de clases desbalanceadas y la carencia de algoritmos multinstancia específicos para el problema de la recuperación y clasificación de documentos textuales. Más concretamente, los objetivos específicos de la tesis son los siguientes:

1. Describir las características fundamentales del aprendizaje multinstancia y particularmente de los métodos de clasificación multinstancia. Este objetivo no solo persigue desarrollar el marco teórico de la investigación, sino también contribuir a organizar la literatura dispersa y sistematizar el conocimiento existente sobre clasificación multinstancia.

2. Proponer un nuevo método de solución al problema de clases desbalanceadas en la clasificación multinstancia basado en técnicas de muestreo. Con este objetivo pretendemos verificar la siguiente hipótesis de investigación: los algoritmos basados en técnicas de muestreo para mitigar el problema de clases desbalanceadas en la clasificación multinstancia *deben* mejorar de forma significativa el desempeño de los algoritmos de clasificación.

3. Diseñar un algoritmo de aprendizaje multinstancia que permita aplicar eficazmente el clasificador de Rocchio para solucionar problemas de recuperación y clasificación de documentos textuales. Con este objetivo pretendemos verificar la siguiente hipótesis de investigación: un algoritmo de clasificación multinstancia basado en el clasificador de Rocchio *debe* alcanzar resultados competitivos con respecto al de otros clasificadores multinstancia orientados a la solución de problemas de recuperación y clasificación de documentos textuales.

Estructura

La memoria está compuesta por tres capítulos y una sección de comentarios finales. En cada capítulo se desarrolla uno de los objetivos que nos hemos propuesto en la tesis. La sección de comentarios finales incluye las conclusiones generales de la investigación y los trabajos futuros. Un apéndice al final de la obra contiene información más detallada de la experimentación del Capítulo 2.

El Capítulo 1, dando respuesta al primer objetivo, comienza con una introducción al aprendizaje automático y al aprendizaje multinstancia para luego exponer los fundamentos de la clasificación multinstancia; se argumenta sobre la importancia del enfoque multinstancia y se ilustra con varios ejemplos de sus aplicaciones; se describen las hipótesis más importantes en las cuales se basan los algoritmos de clasificación multinstancia; se presenta un nuevo sistema de categorización para algoritmos de clasificación multinstancia que facilita la organización, comprensión y comparación de los algoritmos; y finalmente, se explica someramente el funcionamiento de algunos algoritmos representativos de cada categoría.

El Capítulo 2 comienza con una breve exposición de las causas y consecuencias del problema de las clases desbalanceadas, asi como de las soluciones propuestas, tanto en la clasificación tradicional como en la clasificación multinstancia. Seguidamente, se describe el método basado en técnicas de muestreo propuesto por nosotros para solucionar el problema de clases desbalanceadas en la clasificación multinstancia. El método es puesto a prueba con un experimento, y su validez es confirmada por métodos estadísticos. En un segundo experimento mostramos que el algoritmo propuesto es muy competitivo con las mejores soluciones existentes en la literatura.

El Capítulo 3 comienza describiendo el clasificador de Rocchio, que servirá de base para el algoritmo de clasificación multinstancia que proponemos, el cual es explicado seguidamente con todo detalle. En este punto se introduce una nueva hipótesis multinstancia que sirve de fundamento al algoritmo propuesto. Debido a que la velocidad de respuesta del sistema es un tema de preocupación en problemas de grandes dimensiones, como son frecuentes en los problemas de recuperación y clasificación de documentos textuales, se presenta el análisis de la complejidad temporal del algoritmo propuesto. Finalmente, para verificar el cumplimiento del objetivo trazado, el algoritmo diseñado es probado en una aplicación de minería web, la cual fue seleccionada como caso de estudio por ser una aplicación *on-line* típicamente enfocada desde el campo de la recuperación de documentos textuales.

En la sección de comentarios finales se resumen los resultados obtenidos en esta tesis,

se presentan algunos comentarios sobre los mismos y se plantean algunos temas de investigación que se desprenden de este estudio y parecen ser líneas interesantes para abordar en próximos trabajos.

1 Clasificación multinstancia

La clasificación multinstancia es una de las tareas del aprendizaje multinstancia. Específicamente, en la clasificación multinstancia el atributo predictor, i.e., la etiqueta de cada bolsa, es de tipo nominal y representa la clase o categoría de la bolsa. El objetivo es construir un modelo a partir de las bolsas de entrenamiento que permita predecir las etiquetas de clase de nuevas bolsas. Esta es una de las tareas que ha recibido mayor atención en el aprendizaje multinstancia debido a su gran aplicabilidad en los más variados campos de la actividad humana.

Este capítulo se dedica a exponer los fundamentos de la clasificación multinstancia. Primeramente, se presenta el contexto en el que se inserta la clasificación multinstancia, con explicaciones sobre el aprendizaje automático y sobre el aprendizaje multinstancia. Luego, se formaliza la noción de clasificación multinstancia y se introduce la notación matemática que será usada en este informe. Seguidamente, se aborda la importancia que tiene el aprendizaje multinstancia, y con éste la clasificación multinstancia, usando distintos argumentos teóricos y prácticos. Cada algoritmo de clasificación multinstancia basa su funcionamiento en una hipótesis sobre cómo se determina la etiqueta de clase de una bolsa. Por su importancia, se dedica una sección a explicar algunas de las hipótesis más relevantes. Por último, se presenta un sistema de categorización para algoritmos de clasificación multinstancia que facilita la organización, comprensión y comparación de los algoritmos, y se explica someramente el funcionamiento de algunos algoritmos relevantes de cada categoría.

1.1. Contexto y formalización de la clasificación multinstancia

En esta sección se introduce el marco contextual de la investigación, haciendo un recorrido desde el aprendizaje automático, pasando por el aprendizaje multinstancia, hasta la clasificación multinstancia.

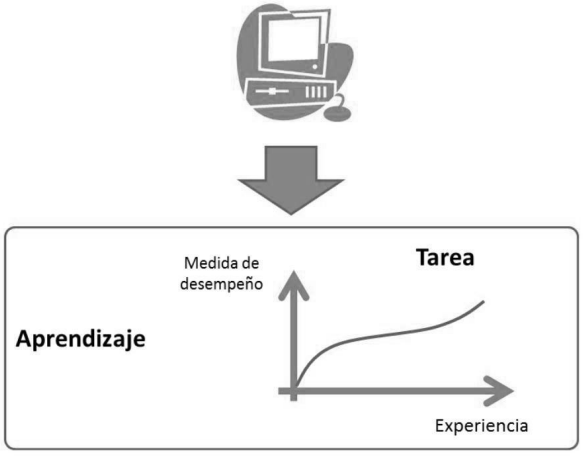


Figura 1.1: Esquema conceptual de aprendizaje automático.

1.1.1. Definición de aprendizaje automático

El tema de la investigación, la clasificación multinstancia, está contenido en el marco general del aprendizaje automático. El aprendizaje automático según Mitchell [67] puede definirse de la siguiente manera: «Un programa de computadora aprende de la experiencia E respecto a alguna clase de tarea T y una medida del desempeño D, si su desempeño en la tarea T, medido a través de D, aumenta con la experiencia E.» Es necesario identificar estos tres elementos para plantear correctamente un problema de aprendizaje: el tipo de tarea a realizar por el aprendiz (T), la medida del desempeño que se debe mejorar (D) y la fuente de experiencia (E). Un ejemplo del propio Mitchell [67] ilustra esta idea: «Un programa de computadora que aprenda a jugar damas puede mejorar su desempeño *medido por su habilidad para ganar* [D] en la clase de tarea que consiste en *jugar el juego de damas* [T], a través de la experiencia [E] obtenida por *jugar contra él mismo.*» La Figura 1.1 esquematiza el aprendizaje automático según esta definición. Los tres elementos mencionados determinan el proceso de aprendizaje en sí, y el hecho de que este proceso sea llevado a cabo por medio de un programa de computadora explica el empleo del adjetivo *automático.*

1.1.2. Tipos de aprendizaje automático

Varios tipos de aprendizaje automático han sido formalizados. El más ampliamente utilizado y al mismo tiempo el más estudiado es llamado *aprendizaje supervisado.* En este tipo de aprendizaje, los algoritmos requieren el suministro de un conjunto de ejemplos, cada ejemplo descrito por un conjunto de atributos y una etiqueta asociada que

corresponde a alguna propiedad importante o decisión relativa al ejemplo. La tarea del algoritmo de aprendizaje es construir un modelo que genere predicciones precisas de las etiquetas de futuros ejemplos. El siguiente ejemplo tomado de [33] ilustra el concepto de aprendizaje supervisado. «Supongamos que somos aprendices de botánica, y queremos aprender a distinguir entre las instancias de las distintas especies de plantas con flores del género iris. Un experto nos ha dado un lote de estas plantas indicándonos a qué especie pertenece cada ejemplo. Después de ver algunos ejemplos de cada especie, podemos intentar inferir las características distintivas de cada una. Una vez que hemos descubierto el patrón, podemos convertirnos en expertos etiquetando plantas iris.» El conjunto de datos Iris del repositorio UCI [35] es un conjunto de datos de referencia estandar para la evaluación de algoritmos de aprendizaje automático. Contiene datos de 50 ejemplos de cada una de las tres especies de plantas iris. La colección de datos que representa a un único ejemplo es referida como una instancia. Cada instancia en el conjunto de datos Iris contiene el ancho y el alto de los pétalos del ejemplo correspondiente; estos elementos de dato son llamados atributos descriptivos o simplemente atributos. Cada instancia está etiquetada con la especie a la que pertenece; esta etiqueta, que también es considerada otro atributo de la instancia, es llamada etiqueta de clase o de decisión.

El *aprendizaje no supervisado* es un tipo de aprendizaje automático donde los ejemplos no presentan etiquetas de clase. Una forma de aprendizaje no supervisado, llamado agrupamiento, intenta dividir el conjunto de datos dado en grupos de instancias relacionadas. El grado de relación se mide típicamente utilizando una medida de proximidad, tal como la distancia Euclideana entre vectores de atributos.

El *aprendizaje semisupervisado* se aplica a conjuntos de datos donde algunas instancias están etiquetadas y otras no. Por tanto, puede considerarse que es un tipo intermedio entre el aprendizaje supervisado y el aprendizaje no supervisado. El etiquetado de los datos es con frecuencia un proceso manual y costoso, mientras que los datos sin etiquetar pueden ser adquiridos automáticamente y relativamente a bajo costo en algunos dominios de problema. Como en el caso supervisado, el objetivo es inferir una regla para predecir las etiquetas de instancias individuales. En algunos problemas se puede mejorar la exactitud de la clasificación/regresión tomando provecho de grandes fuentes de datos sin etiquetar.

Los tipos de aprendizaje descritos anteriormente trabajan con datos organizados en forma de una tabla de plana (i.e. las instancias en las filas y los atributos en las columnas), pero existen muchos problemas de aprendizaje que tienen una estructura relacional, es decir, que están formados por varias tablas que se relacionan entre sí a través de atributos claves (similar a las bases de datos relacionales). Este tipo de problemas de aprendizaje se

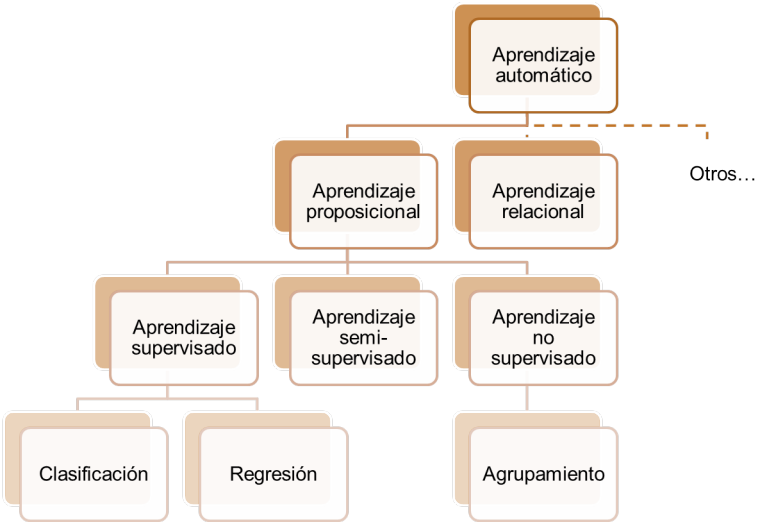


Figura 1.2: Tipos de aprendizaje automático

conoce como aprendizaje relacional, y se contrapone con los problemas que involucran una tabla plana, a los cuales se les llama problemas de aprendizaje atributo-valor o *aprendizaje proposicional*. La *programación lógica inductiva* es un tipo de aprendizaje automático que, mediante lenguajes de programación como Prolog, se basa en la lógica de primer orden para representar y resolver problemas relacionales.

La Figura 1.2 muestra la relación entre los distintos tipos de aprendizaje automático descritos. La clasificación y la regresión son tipos específicos de aprendizaje supervisado, en cuya descripción se profundiza a continuación.

1.1.3. Formalización del aprendizaje supervisado

La clasificación multinstancia es una forma de aprendizaje supervisado. En la subsección anterior introducimos la noción y la terminología del aprendizaje supervisado por medio de un ejemplo sencillo, ahora le daremos una definición formal. Una instancia es un par (x, y), donde $x = \langle x_1, x_2, ..., x_N \rangle \in \mathcal{X}$ es un vector de N atributos y $y \in \mathcal{Y}$ es la etiqueta de decisión de la instancia. Los atributos y las etiquetas de decisión son típicamente o bien elementos de los números reales (atributos numéricos) o bien conjuntos de nombres de un dominio específico (atributos nominales). El espacio N dimensional $\mathcal{X}$ de donde x toma valores se conoce como *espacio de instancias* o espacio de atributos, y $\mathcal{Y}$ es el conjunto de etiquetas de decisión. El Cuadro 1.1 muestra la forma en que se representa la información en el aprendizaje supervisado. Cada ejemplo x_i es descrito por un vector de atributos y una etiqueta de decisión y_i.

Ejemplos	Atrib.1	Atrib.2	...	Atrib.N	Decisión
x_1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,N}$	y_1
x_2	$x_{2,1}$	$x_{2,2}$	...	$x_{2,N}$	y_2
...	...	...	...	...	...
x_M	$x_{M,1}$	$x_{M,2}$	...	$x_{M,N}$	y_M

Cuadro 1.1: Representación de la información en el aprendizaje supervisado tradicional.

La tarea del aprendizaje supervisado es encontrar $f(x) = y$, basado en un conjunto de instancias de entrenamiento $D = \{(x_1, y_1), \cdots, (x_M, y_M)\}$. Generalmente, $f(x)$ es una función que asigna a cada elemento del dominio una etiqueta de decisión. Cuando la etiqueta es un atributo nominal, este proceso es llamado *clasificación*. Cuando la etiqueta es un atributo numérico, el proceso es llamado *regresión*. El proceso de clasificación subyacente $f(x)$ se conoce como un *concepto* en la terminologia del aprendizaje automático. Dado un conjunto de ejemplos de entrenamiento D a partir de los cuales aprender, un algoritmo de aprendizaje supervisado devuelve un modelo de los datos $h(x)$, el cual se pretende que sea la mejor aproximación a $f(x)$. Dicho modelo se conoce como *hipótesis* o descripción del concepto.

1.1.4. Aprendizaje multinstancia

El aprendizaje multinstancia es una variación del aprendizaje proposicional en el que cada ejemplo está descrito, ya no por un único vector, sino por muchos vectores atributo-valor. El enfoque multinstancia se ha aplicado a distintos tipos de aprendizaje proposicional, sin embargo es mucho más popular en el aprendizaje supervisado.

Como se vió en la sección anterior, en el aprendizaje supervisado estándar cada ejemplo es una instancia que consiste en un vector de atributos, aumentado con una etiqueta de decisión. La tarea es aprender una función que prediga la etiqueta de decisión de un ejemplo, dado el vector de atributos. En tanto, en el aprendizaje multinstancia cada ejemplo consiste en una bolsa de instancias. El término bolsa evoca la imagen de un saco que contiene las instancias sin orden ni concierto, pudiendo estar incluso repetida la misma instancia dentro de la bolsa. Cada bolsa tiene asociada una etiqueta de decisión, pero las instancias en sí no están etiquetadas. El problema de aprendizaje es construir un modelo, basado en las bolsas de ejemplo dadas, que pueda predecir con precisión la etiqueta de decisión de futuras bolsas. Un ejemplo ayudará nuevamente a iluminar el concepto. Este ejemplo, llamado problema del cerrajero simple, pertenece a [19]. «Imagine que hay una puerta cerrada con cerrojo, y tenemos M llaveros, cada uno

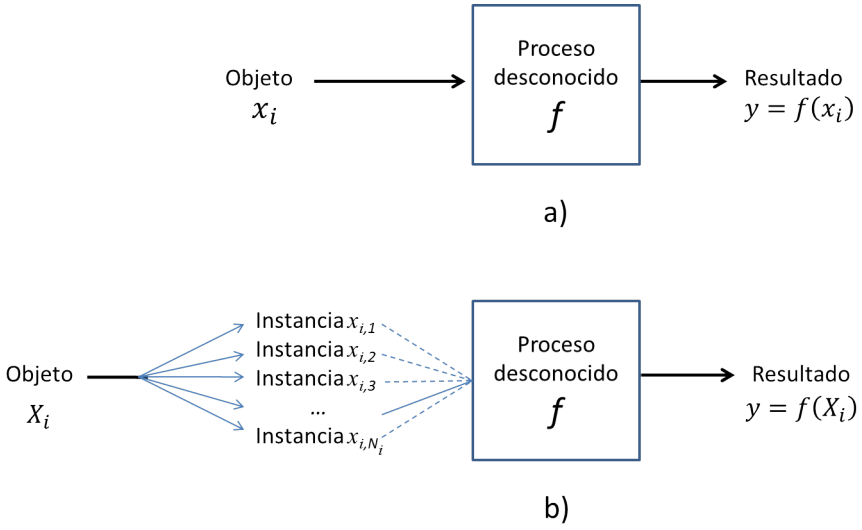


Figura 1.3: Diferencia entre el (a) aprendizaje supervisado tradicional (monoinstancia) y (b) aprendizaje supervisado multinstancia.

contiene un manojo de llaves. En esta metáfora el llavero representa la bolsa, y una llave particular, una instancia. Si un llavero contiene una llave que puede abrir la cerradura, ese llavero es considerado útil. El problema de aprendizaje es construir un modelo que pueda predecir cuándo un llavero dado es o no útil.»

El aprendizaje multinstancia es de hecho una generalización del aprendizaje proposicional, ya que cada ejemplo en cualquier problema del aprendizaje proposicional tradicional puede considerarse una bolsa que contiene una sola instancia. A partir de la introducción del aprendizaje multinstancia podemos llamar *aprendizaje monoinstancia* al aprendizaje proposicional tradicional en contraposición al aprendizaje multinstancia. La Figura 1.3[1] ilustra la diferencia entre el aprendizaje supervisado tradicional (monoinstancia) y el aprendizaje supervisado multinstancia.

1.1.5. Formalización de la clasificación multinstancia

En la clasificación multinstancia un ejemplo es un par (X, y), donde $X = \{x_1, \cdots, x_T\} \in \mathbb{N}^{\mathcal{X}}$ es un *multiconjunto* de T instancias y $y \in \mathcal{Y}$ es la etiqueta de clase del ejemplo. Multiconjunto es la palabra que en jerga matemática equivale a bolsa. Se dice que es un multiconjunto y no simplemente un conjunto porque el multiconjunto, a diferencia de éste, permite tener instancias repetidas. En notación matemática, $X \in \mathbb{N}^{\mathcal{X}}$ significa que

[1]Basada en una figura semejante de [27].

Ejemplos	Instancias	Atrib.1	Atrib.2	...	Atrib.N	Decisión
X_1	$x_{1,1}$	$x_{1,1,1}$	$x_{1,1,2}$	...	$x_{1,1,N}$	y_1
	...	...	...	...	...	
	x_{1,T_1}	$x_{1,T_1,1}$	$x_{1,T_1,2}$	...	$x_{1,T_1,N}$	
...	...	...	...	...	...	...
X_M	$x_{M,1}$	$x_{M,1,1}$	$x_{M,1,2}$	...	$x_{M,1,N}$	y_M
	...	...	...	...	...	
	x_{M,T_M}	$x_{M,T_M,1}$	$x_{M,T_M,2}$	...	$x_{M,T_M,N}$	

Cuadro 1.2: Representación de la información en el aprendizaje multinstancia.

X es un multiconjunto (una bolsa) de elementos de $\mathcal{X}$, donde $\mathbb{N}$ representa el conjunto de los números naturales. Las instancias $x_i \in \mathcal{X}$, $i = 1 \dots T$ son vectores de N atributos que, a diferencia del aprendizaje supervisado estándar, carecen de etiquetas de decisión. El conjunto $\mathcal{X}$ es un espacio N-dimensional formado por el producto vectorial de los N atributos que describen a las instancias, y $\mathcal{Y}$ es el conjunto de etiquetas de clase. La mayoría de los trabajos desarrollados en clasificación multinstancia asume que el atributo de decisión es binario, i.e., $\mathcal{Y} = \{+, -\}$, pero, en general, $\mathcal{Y}$ puede tener más de dos elementos. En este caso se dice que el problema de clasificación es *multiclase*. El Cuadro 1.2 muestra la forma en que se representa la información en el aprendizaje multinstancia. Cada ejemplo X_i es descrito por T_i vectores de atributos y una etiqueta de decisión y_i. Observe que en nuestra notación los ejemplos se representan con letra mayúscula, para indicar que es una bolsa (un conjunto de vectores), mientras que las instancias dentro de la bolsa se representan con letra minúscula, indicando que son vectores simples.

La tarea de la clasificación multinstancia es encontrar $f(X) = y$, basado en un conjunto de instancias de entrenamiento $D = \{(X_1, y_1), \cdots, (X_M, y_M)\}$, donde $f : \mathbb{N}^{\mathcal{X}} \to \mathcal{Y}$ es un concepto multinstancia. Un algoritmo de aprendizaje multinstancia construye a partir de un conjunto de ejemplos de entrenamiento D un clasificador o modelo de los datos $h(x)$, el cual es una hipótesis de $f(x)$.

1.1.6. Otros tipos de aprendizaje multinstancia

La mayoría de la investigación en aprendizaje multinstancia se ha consagrado a los problemas de clasificación, sin embargo también se han hecho algunos estudios en otros escenarios de aprendizaje. Los más notables de estos son la regresión multinstancia [84, 1], el agrupamiento multinstancia [128, 57] y el aprendizaje multinstancia multietiqueta [134]. De forma similar a la clasificación, cualquier método de aprendizaje en estos escenarios dependerá de la hipótesis implícita que considera el tipo de relación

entre instancias y bolsas.

En la Conferencia Internacional sobre Aprendizaje Automático en 2001, [84] y [1] formularon independientemente la regresión multistancia, donde las bolsas están asociadas con etiquetas de valores reales en lugar de las usuales etiquetas de clase binarias. La tarea es nuevamente predecir estas etiquetas. De forma similar a la clasificación multistancia, la regresión multinstancia está motivada por el problema de la predicción de la actividad farmacológica, ya que muchos desarrolladores de fármacos prefieren predicción de niveles de actividad de las drogas en vez de una clasificación en activo/inactivo. Algunas áreas de aplicación identificadas por otros autores incluyen predicción de la profundidad óptica de aerosoles para la investigación climática [114] y modelación del rendimiento de las cosechas [109].

En los problemas donde no se dispone de las etiquetas de las bolsas de entrenamiento, el agrupamiento multinstancia permite agrupar las bolsas en tipos de bolsas similares con alta semejanza entre las bolsas del grupo y baja semejanza entre bolsas de distintos grupos. El agrupamiento es uno de los métodos de aprendizaje más populares dentro del aprendizaje no supervisado ya que puede ayudar a encontrar la estructura inherente a los datos y tener una comprensión de su distribución, sin embargo el agrupamiento también puede ser usado como un paso de preprocesamiento para otras tareas, como la clasificación o la regresión [128].

En el aprendizaje supervisado tradicional, los problemas de aprendizaje multiclase contienen más de dos categorías de clasificación, pero cada ejemplo pertenece a solo una de esas categorías. Una extensión de esto es el *aprendizaje multietiqueta*, donde las categorías no son mutuamente exclusivas, de forma que cada ejemplo puede pertenecer a varias clases a la vez [92]. Zhou & Zhang [134] formalizaron el aprendizaje multinstancia multietiqueta (MIME), donde cada bolsa multinstancia puede estar asociada con múltiples etiquetas de clase. La tarea en MIME es aprender una función de la forma $f_{MIME} : \mathbb{N}^{\mathcal{X}} \rightarrow 2^{\mathcal{Y}}$ donde $\mathcal{X}$ es el espacio de instancia, y $\mathcal{Y}$ es el conjunto de categorías de clase. El aprendizaje multinstancia y el aprendizaje multietiqueta son ambos generalizaciones naturales del aprendizaje monoinstancia tradicional, y MIME es una generalización de ambos.

1.2. Importancia de la clasificación multinstancia

El aprendizaje multinstancia es una rama muy joven del aprendizaje automático, sin embargo ha despertado un gran interés entre los investigadores y profesionales del ramo debido a su capacidad para modelar problemas de aprendizaje caracterizados por la

presencia de ambigüedad en sus datos, y también porque ha venido a ser un eslabón que permite enlazar el aprendizaje proposicional con el relacional. El debut del aprendizaje multinstancia fue con los algoritmos de rectángulos paralelos a los ejes para la predicción de la actividad farmacológica. Hoy en día existen decenas de métodos de solución y siguen apareciendo nuevas propuestas que mejoran aspectos específicos del problema. Igualmente, ha crecido significativamente el número y la variedad de los campos de aplicación que se han beneficiado con el aprendizaje multinstancia. Algunos ejemplos de aplicaciones recientes son la anotación automática de imágenes [32, 100, 137], recuperación de imágenes/videos basados en su contenido [60, 126, 46, 62, 77], seguimiento visual [130, 127, 71, 4], bioinformática [66, 39, 108], diagnóstico médico asistido por ordenador [101, 74, 22], imaginología [111, 78, 28], reconocimiento de emociones [58, 55, 107, 59], finanzas [56], detección de anomalías [122], detección de minas terrestres [13, 12], entre otras. En esta sección se argumentará con más detalles sobre la importancia del aprendizaje multinstancia y, particularmente, de la clasificación multinstancia.

1.2.1. Origen del aprendizaje multinstancia

El aprendizaje multinstancia fue definido formalmente en 1997 por Dietterich et al. [27]. Ellos trabajaban a la sazón en la Corporación Farmacéutica Arris en el problema de predicción de la actividad de fármacos. Este problema es de gran importancia, pues permite ahorrar tiempo y dinero al concentrar los esfuerzos de biólogos y químicos en la síntesis y prueba de compuestos para los cuales se predice una alta actividad. El problema de predicción de la actividad de fármacos consiste en encontrar moléculas que se enlacen estrecha y selectivamente a una proteína dada. El factor más importante para determinar si una molécula se enlaza a una proteína es su forma. Empleando el paradigma de aprendizaje supervisado se podría aprender la forma adecuada de una molécula a partir de una serie de ejemplos positivos y negativos. Un ejemplo positivo sería una molécula que se sabe que se enlaza a la proteína dada, mientras que un ejemplo negativo sería una molécula que no se enlaza con la proteína. Sin embargo, el principal problema de este enfoque es que las moléculas son objetos flexibles. Por tanto, una misma molécula puede tomar muchas conformaciones diferentes. La Figura 1.4[2] muestra un ejemplo de varias conformaciones que puede tomar una molécula de butano al rotar uno de sus enlaces. En este caso, las distintas conformaciones ocurren porque la molécula puede rotar alrededor del enlace de los dos átomos centrales de carbono.

El problema multinstancia aparece entonces porque no existe una única descripción

[2]Tomado de [64].

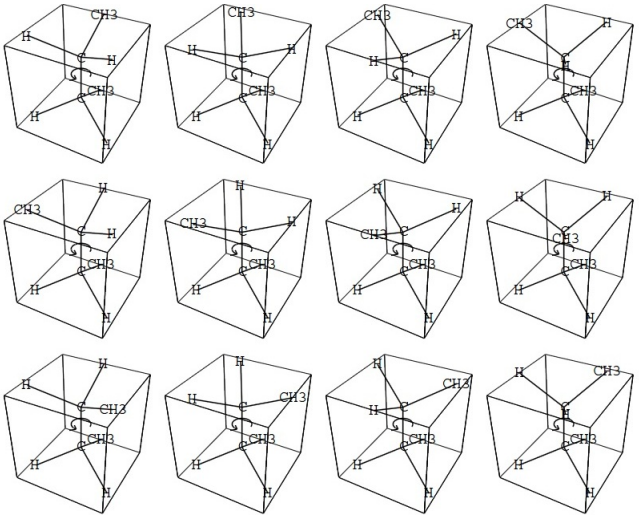


Figura 1.4: Ejemplo de diferentes conformaciones que una molécula de butano (C_4H_{10}) puede tomar.

de una molécula; es decir, una misma molécula es descrita a través de múltiples conformaciones. Por tanto, un ejemplo de entrenamiento está formado por una colección de conformaciones. Si una molécula es positiva, sabemos que al menos una de sus posibles conformaciones es la correcta, aunque no sabemos cual de ellas es la correcta. Si una molécula es negativa, sabemos que ninguna de sus conformaciones es correcta.

Específicamente, Dietterich et al. [27] presentaron el caso de los fármacos almizcleros (*musk drug activity problem*, en inglés), un caso particular del problema de predicción de la actividad de fármacos. En este problema, las moléculas activas emiten un olor a almizcle que es detectable por observadores humanos. El problema de aprendizaje consiste en predecir, basado en un conjunto de moléculas de ejemplo, si una molécula dada emitirá el olor. Dietterich et al. formularon el aprendizaje multinstancia para resolver este problema. Ellos representaron cada molécula como una bolsa que contiene las diferentes conformaciones que puede adoptar la molécula. Muchos otros enfoques habían sido empleados con anterioridad para predecir el olor de almizcle de las moléculas (e.g. [70, 31, 7, 26]), sin embargo, el enfoque multinstancia fue en efecto el que mejor funcionó.

El trabajo de Dietterich et al. no solo ofreció una mejor solución al problema de predicción de la actividad de fármacos sino que fundó un nuevo paradigma del aprendizaje automatizado que ha tenido importantes repercusiones tanto desde el punto de vista práctico, con aplicaciones a nuevos y más complejos problemas, como desde el punto de vista teórico por su conexión con el aprendizaje relacional (Sección 1.2.3.2) y su capacidad para manejar la ambigüedad en los datos (Sección 1.2.4).

Figura 1.5: Ejemplo de representación multinstancia en las variaciones en el trazado de los dígitos.

1.2.2. Ejemplos de aplicaciones de clasificación multinstancia

Con el objetivo de ilustrar cómo el enfoque multistancia puede ser usado para solucionar determinados problemas de clasificación, se esbozan a continuación algunos ejemplos de aplicaciones de clasificación multistancia, identificando la relación entre bolsas e instancias.

En el problema de reconocer dígitos manuscritos, cada persona (y aun la misma persona) cada vez que escribe un dígito lo hace de una forma ligeramente diferente a las demás. Representantes de una misma clase de dígito manuscrito pueden diferir en cuanto a traslación, rotación, grosor del trazo y otras transformaciones, como se aprecia en la Figura 1.5. En cada fila hay múltiples representantes (instancias) de una misma clase de dígito (ejemplar). Este importante problema del campo del reconocimiento de patrones ha sido enfocado de muchas maneras, por ejemplo, en [98, 97] un conjunto de N transformaciones aplicadas a un patrón se interpreta como una variedad matemática N-dimensional del patrón. Los autores presentan una red neuronal de retropropagación para aprender el conjunto de variaciones de cada patrón. Desde el enfoque multinstancia cada clase de dígito (digamos, un 3) se corresponde con un ejemplo, y las distintas transformaciones de un dígito se pueden entender como distintas variaciones del mismo dígito, por tanto, serían múltiples instancias del mismo ejemplo.

El problema de predicción de enfermedad en las frutas es un caso natural para el aprendizaje multinstancia. Cuando las frutas son recolectadas en diferentes huertos, normalmente son dispuestas en lotes dentro de almacenes, un lote por cada huerta. Después de algún tiempo (entre 3 y 5 meses, que es el tiempo que usualmente dura la transportación), aparecen algunos síntomas de enfermedad en algunos lotes. Como

la enfermedad puede ser epidémica, con frecuencia todo el lote de frutas es infectado y exhibe síntomas similares. Nos interesa predecir antes del embarque cuáles lotes son propensos a enfermar basado en algunas mediciones no destructivas hechas a cada fruta tan pronto como son recolectadas de las huertas. La tarea es predecir, dado un nuevo lote de frutas, si éste será afectado o no por cierta enfermedad (algunos meses después). Éste es obviamente un problema multinstancia. Cada lote es un ejemplo (bolsa) y cada fruta dentro de éste es una instancia. En los datos de entrenamiento, un lote es etiquetado *propenso a enfermar* (positivo) si los síntomas son observables después del embarque, en otro caso *libre de enfermedad* (negativo).

La recuperación de imágenes basada en su contenido (CBIR, siglas del término inglés *content-based image retrieval*) es una de las áreas de aplicación más frecuentes en el estudio del aprendizaje multistancia. En CBIR, los usuarios presentan ejemplos de las imágenes que desean y la tarea es determinar si una nueva imagen será de su interés. Una imagen es representada por un conjunto de segmentos (regiones de pixeles) que están caracterizadas por descriptores de color, textura y forma. Una bolsa representa una imagen y una instancia dentro de una bolsa representa un segmento de la imagen. No se conoce cuáles segmentos y atributos de una imagen están relacionados con el contenido deseado, i.e. las etiquetas de clase de las instancias individuales son desconocidas.

La clasificación multinstancia tiene también aplicaciones en la categorización de texto, donde la tarea es asignar etiquetas semánticas a los documentos de texto. Un documento puede ser representado como una bolsa y las instancias son obtenidas al dividir el documento en pasajes más pequeños. Se pueden extraer atributos tales como las frecuencias de ocurrencia de las palabras en cada pasaje para formar las instancias.

1.2.3. Relación con otros tipos de aprendizaje automático

La programación lógica inductiva y el aprendizaje proposicional o atributo-valor parecían irreconciliables, sin embargo, el aprendizaje multinstancia, surgido como una extensión del aprendizaje supervisado, ha resultado ser un puente entre ambos paradigmas. Esta sección aborda la relación que guarda el aprendizaje multinstancia con el proposicional y el relacional.

1.2.3.1. Relación con el aprendizaje proposicional

A pesar de la semejanza entre el aprendizaje multinstancia y el semisupervisado, en el sentido de que en el escenario multinstancia solo las bolsas tienen etiquetas de decisión mientras que las instancias no las tienen, existe una diferencia fundamental entre ambos

tipos de aprendizaje, y es la relación que hay entre instancias y bolsas en el aprendizaje multistancia que no existe en el aprendizaje semisupervisado. En este último, las instancias etiquetadas están al mismo nivel que las instancias no etiquetadas y no existe una relación específica entre ellas. Mientras que en el aprendizaje multinstancia existe una estructura de datos en la cual bolsas e instancias están en dos niveles distintos; todas las instancias dentro de una bolsa están relacionadas entre sí a través de la etiqueta de la bolsa. La etiqueta de la bolsa multinstancia modela la relación 1-muchos, característica de las bases de datos relacionales.

El aprendizaje multinstancia es más bien una generalización del aprendizaje supervisado estándar, o dicho de otra manera, el aprendizaje supervisado estándar se puede entender como un caso particular de aprendizaje multinstancia en el cual cada bolsa contiene una sola instancia. De igual manera que en el aprendizaje supervisado estándar, en el multinstancia se puede realizar clasificación o regresión dependiendo de si las etiquetas de clase de las bolsas son nominales o numéricas. También se puede hacer agrupamiento en problemas multinstancias en los cuales se desconozca o se prescinda de las etiquetas de clase.

1.2.3.2. Relación con el aprendizaje relacional

Para el aprendizaje proposicional se ha desarrollado un impresionante número de algoritmos, sin embargo estos, al operar solo sobre datos organizados en forma de tabla plana, no pueden aplicarse al aprendizaje relacional. Por otro lado, los datos relacionales pueden transformarse en datos atributo-valor en un proceso llamado *proposicionalización*, pero el costo computacional es muy alto debido a que implica una explosión combinatoria que hace imposible su aplicación a problemas reales [85]. El aprendizaje multinstancia ha venido a considerarse el eslabón perdido entre el aprendizaje proposicional y el relacional ya que, como se dijo anteriormente, la etiqueta de la bolsa multinstancia modela la relación 1-muchos. De Raedt [23] mostró que los problemas multinstancia también pueden ser considerados como un caso especial de programación lógica inductiva. Todos los problemas de programación lógica inductiva (en forma de bases de datos relacionales) pueden ser transformados mediante operaciones de unión de bases de datos en una sola relación 1-muchos para representarse como problemas multinstancia [85, 86]. Además, es fácil adaptar los algoritmos del aprendizaje supervisado clásico al escenario multinstancia, y como se verá en la Sección 1.4, muchos de estos algoritmos clásicos han sido ya adaptados. Esta característica del aprendizaje multinstancia hace posible que muchos problemas relacionales puedan ser resueltos mediante métodos propios del aprendizaje supervisado.

1.2.4. La ambigüedad como fuente de problemas multinstancia

El aprendizaje multinstancia modela el problema de aprendizaje a partir de entradas ambiguas debido al polimorfismo. Los casos más comunes de entradas polimórficas son:

- Múltiples instancias de la misma modalidad. Un objeto puede tener múltiples apariencias diferentes en el espacio de entrada y no se conoce cual de ellas es responsable de la etiqueta del objeto. En este tipo de problema cada bolsa describe diferentes puntos de vista del mismo objeto. Por ejemplo, en aplicaciones de diseño de fármacos a veces es conveniente utilizar una representación polimórfica para las moléculas [27]. Esto se debe a que un molécula es dinámica, asumiendo muchas conformaciones (formas) moleculares diferentes que pueden ser reactivas o no-reactivas, pero no pueden ser distinguidas durante una reacción química. La hipótesis, propuesta por la química orgánica, es que la reactividad molecular se deriva a partir de las conformaciones individuales. Una representación polimórfica de la molécula se obtiene muestreando las conformaciones. Al asociar la reactividad con cada conformación surgen interpretaciones distintas.

- Múltiples partes de un problema multipartes. Un objeto puede estar compuesto por piezas o partes, cada una de las cuales tiene una representación en el espacio de entrada, y la etiqueta se asigna al objeto como un todo y no a las partes, aunque solo una (o unas pocas) de las partes sean responsables de la etiqueta. Por ejemplo, en anotación automática de imágenes y recuperación de imágenes basadas en su contenido [2, 18, 8, 83], la representación polimórfica de imágenes se obtiene típicamente segmentando una imagen en regiones. Por otro lado, cuando un humano está etiquetando imágenes para el aprendizaje supervisado, es más típico y conveniente asignar etiquetas de clase a nivel de imagen (a toda la imagen), que hacerlo para cada segmento individualmente. La hipótesis es que tales etiquetas se derivan individualmente de los segmentos. Las imágenes etiquetadas de esta manera son ambiguas ya que se desconoce la asociación entre la etiqueta y un segmento particular. En otras palabras, hay muchas maneras de interpretar a nivel de segmento la etiqueta dada a la imagen.

Los métodos de solución del aprendizaje multinstancia, y en particular los algoritmos de clasificación multinstancia, resuelven la ambigüedad presente en los datos basándose en una hipótesis sobre cómo se relaciona la etiqueta de clase de una bolsa con las instancias de la bolsa. La proxima sección examina con detalle algunas de las hipótesis que han alcanzado más relevancia. Estas hipótesis son aplicables por igual a distintas

tareas del aprendizaje multinstancia, como pueden ser el agrupamiento, la regresión y la clasificación.

1.3. Hipótesis del aprendizaje multinstancia

En general, los problemas multinstancia son mucho más difíciles de resolver que los problemas del aprendizaje supervisado estándar ya que el espacio de hipótesis de los multinstancia es mucho mayor. Esto se deduce a partir de la definición de ambos tipos de aprendizaje: en el supervisado estándar $f_{SE} : \mathcal{X} \to \mathcal{Y}$, mientras que en el multinstancia $f_{MI} : \mathbb{N}^{\mathcal{X}} \to \mathcal{Y}$; teniendo en cuenta que $\mathcal{X} \subseteq \mathbb{N}^{\mathcal{X}}$, se tiene entonces que $f_{SE} \subseteq f_{MI}$. Para poder acotar la búsqueda de la función objetivo f_{MI} generalmente se recurre a *asumir* la existencia de cierta relación entre las instancias y la etiqueta de decisión de las bolsas. A la relación particular que se asume para resolver un problema multinstancia se le llama *hipótesis multinstancia*.

Actualmente existe una variedad de hipótesis multinstancia que han sido introducidas a medida que se han desarrollado nuevos métodos de solución para los problemas multinstancia. Algunas de estas hipótesis se plantean de forma explícita en los algoritmos que las implementan, otras aparecen de manera implícita en diversos métodos de solución multinstancia. En todos los casos, la hipótesis multinstancia subyacente es un elemento fundamental para entender el funcionamiento del algoritmo de aprendizaje.

La Figura 1.6 muestra una jerarquía con los principales tipos de hipótesis multinstancia que han sido desarrollados. Existen dos grandes grupos: el de las hipótesis basadas en instancia y el de aquellas basadas en metadatos. La característica distintiva de las hipótesis basadas en instancia es asumir que las instancias, aunque desconocidas, tienen sus propias etiquetas ocultas determinadas por algún proceso $\mathsf{g} : \mathcal{X} \to C$, donde C es el conjunto de conceptos subyacente a las instancias. Las hipótesis en esta categoría interpretan un concepto multinstancia f_{MI} como una composición de funciones $f_{MI} = \mathsf{g} \circ \mathsf{h}$ (Figura 1.7). La función g determina la etiqueta de cada instancia. La función $\mathsf{h} : \mathbb{N}^{\mathrm{C}} \to \mathcal{Y}$ determina la etiqueta de la bolsa a partir de las etiquetas de las instancias. Las distintas hipótesis basadas en instancia se diferencian en la función h que éstas asumen. Los algoritmos que implementan estas hipótesis multinstancia generalmente persiguen encontrar una aproximación de la función g que les permitan determinar las etiquetas de las instancias para luego aplicar la función h que han asumido y obtener de esta forma la etiqueta de la bolsa. Por esto se le conoce a este tipo de algoritmos como métodos en dos pasos: el primero de clasificación de instancias y el segundo de clasificación de la bolsa.

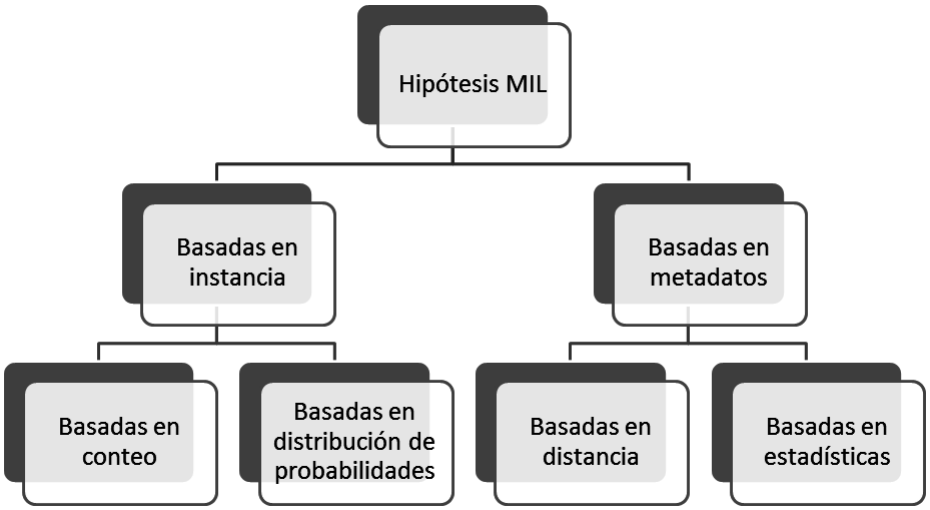


Figura 1.6: Jerarquía de hipótesis multinstancia.

Entre las hipótesis basadas en instancias están aquellas basadas en conteo, las cuales determinan la etiqueta de la bolsa según el número de instancias de cada concepto que haya en la bolsa. Otro tipo de hipótesis basada en instancias es aquella en la que las bolsas se consideran muestras aleatorias de cierta distribución de probabilidades, y la etiqueta de la bolsa está determinada por el valor de clase esperado de la población de la bolsa.

En tanto, las hipótesis que usan el enfoque de metadatos asumen que las etiquetas de las bolsas están determinadas por alguna información en un metanivel que describe a los ejemplos. Esta metainformación puede consistir en distancias, ya sea entre bolsas o entre instancias y bolsas, o en datos estadísticos que se extraen de las bolsas. La metainformación sirve para mapear las bolsas en un nuevo espacio de atributos monoinstancia en el que son aplicables los algoritmos tradicionales de aprendizaje. En las proximas secciones se describe más detalladamente algunos ejemplos importantes de los distintos tipos de hipótesis multinstancia mencionados.

1.3.1. Hipótesis estándar

La primera hipótesis que se empleó para definir el aprendizaje multinstancia[27] tuvo una gran influencia en el planteamiento de los problemas multinstancia y el desarrollo de los primeros métodos de solución. Por ello se le dió el nombre de hipótesis multinstancia estándar.

La hipótesis estándar establece que una bolsa será positiva si y solo si contiene alguna

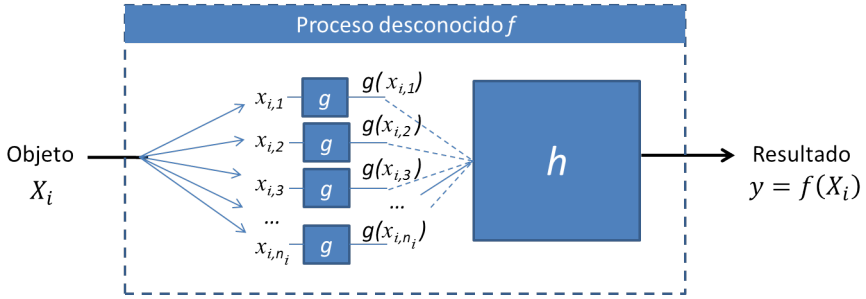


Figura 1.7: Representación de las hipótesis basadas en etiquetas de instancia.

instancia positiva. Es decir, si la bolsa es negativa todas sus instancias serán negativas, si la bolsa es positiva al menos una de sus instancias será positiva. Una forma de describir matemáticamente la hipótesis estándar es mediante la disyunción de las etiquetas de clase de las instancias:

$$\begin{aligned} f(X) &= \mathrm{g}(x_1) \vee \mathrm{g}(x_2) \vee \cdots \vee \mathrm{g}(x_n) \ , \ x_i \in X \\ &= \bigvee_{x_i \in X} \mathrm{g}(x_i) \end{aligned} \tag{1.1}$$

En esta fórmula se asume que las etiquetas positivas toman valor *verdadero* y las negativas toman *falso*. Otra forma de escribir la hipótesis estándar, en el caso en que a las etiquetas positivas se les asigne valor 1 y a las negativas el valor 0 o -1 es con la función máx:

$$f(X) = \max_{x_i \in X} \mathrm{g}(x_i) \tag{1.2}$$

Observe que la hipótesis estándar es asimétrica: si las etiquetas positiva y negativa se invierten, la hipótesis cobra un significado diferente. Por tanto, cuando apliquemos esta hipótesis, tenemos que estar bien claros de cuál debe ser la etiqueta positiva.

La hipótesis multinstancia estándar fue adoptada porque en un principio se creyó que era la hipótesis más apropiada para el problema *musk*. En el problema *musk* se considera que una molécula (representada por una bolsa de instancias) emitirá un olor a almizcle (en inglés, *musk*) si y solo si una de sus conformaciones moleculares (representada por una instancia individual) se acopla con determinado sitio de la molécula receptora del estímulo en el aparato olfativo [27]. No obstante, posteriormente se encontró que otras hipótesis multinstancia podían ofrecer soluciones al problema *musk* con calidad igual o superior a las otorgadas por la hipótesis multinstancia estándar.

La hipótesis estándar se ha usado en un número importante de algoritmos multinstancia como por ejemplo en los algoritmos APR [27], densidad diversa [65], EM-DD [129],

árboles de decisión [11, 19], reglas de decisión [19], amplificación [3] redes neuronales [82], máquinas de soporte vectorial [2], etc.

1.3.2. Generalización de la hipótesis estándar

En la hipótesis estándar se considera que el conjunto de etiquetas C que las instancias pueden tomar coincide con el conjunto de etiquetas $\mathcal{Y}$ que toman las bolsas. Estas son positivas y negativas, i.e. $C = \mathcal{Y} = \{c_+, c_-\}$. Por otro lado, en la hipótesis estandar basta que una instancia pertenezca al concepto positivo para que la bolsa sea positiva. En fórmulas, podemos escribir que un concepto multistancia estandar es una función $f_{est} : \mathbb{N}^{\mathcal{X}} \to \mathcal{Y}$ que se define como

$$f_{est}(X) \Leftrightarrow \exists x \in X : \mathsf{g}(x) = c_+$$

Sin embargo, de forma general el conjunto de conceptos o etiquetas de clase C que toman las instancias no tiene que coincidir con el conjunto de etiquetas $\mathcal{Y}$ que toman las bolsas. De esta forma las instancias pueden pertenecer a n conceptos distintos, $C = \{c_1, c_2, \ldots, c_n\}$, independientemente de que las bolsas sigan siendo solo positivas o negativas. A partir de aquí (Weidmann et al., 2003) [115] introducen tres extensiones a la hipótesis multinstancia estándar. Ellos extienden el proceso que combina las etiquetas de instancia para formar la etiqueta de la bolsa. En el caso multinstancia estándar, una instancia que es positiva con respecto al concepto proposicional subyacente hace a la bolsa positiva. En tanto Weidmann et al., en lugar de un único concepto subyacente, usan un conjunto de conceptos subyacentes y hacen que la bolsa positiva requiera cierto número de instancias de cada uno de ellos.

La primera generalización está definida en términos de la presencia de instancias de cada concepto en una bolsa. Por ejemplo, un concepto multinstancia de esta categoría es «la bolsa es positiva solo si en la bolsa hay presencia de instancias del concepto c_1 e instancias del concepto c_2». Para describir formalmente esta hipótesis definiremos primero una función de conteo Δ: $\mathbb{N}^{\mathcal{X}} \times C \to \mathbb{N}$, la cual cuenta los miembros de un concepto dado en una bolsa. Según la *hipótesis multinstancia basada en presencia* f_p una bolsa será etiquetada positiva si contiene al menos una instancia de cada uno de los conceptos subyacentes:

$$f_p(X) \Leftrightarrow \forall c \in \mathsf{C} : \Delta(X, c) \geq 1$$

donde $\mathsf{C} \subseteq C$ es el conjunto de conceptos subyacentes que deben ser satisfechos en la premisa de la hipótesis. El siguiente ejemplo de [115] ilustra un concepto multinstancia basado en presencia. «Imaginemos que nos dan varios mazos de llaves. Un mazo cor-

responde a una bolsa, las llaves son sus instancias. La tarea es predecir si un nuevo mazo puede usarse para abrir la cerradura de una puerta. Si esta puerta solo tiene una cerradura, cada mazo con al menos una llave para esa cerradura sería positiva. Esto se corresponde con el problema multinstancia estándar. Sin embargo, asumamos que existen n cerraduras distintas que deben ser descerradas antes que la puerta pueda ser abierta. Este es un ejemplo de concepto multinstancia basado en presencia, porque necesitamos al menos una instancia (una llave) de cada uno de los n conceptos (las cerraduras) para clasificar la bolsa como positiva.» Así, la hipótesis multinstancia estándar es un caso especial de concepto basado en presencia que tiene solo un concepto subyacente, o sea con $|\mathsf{C}| = 1$.

En lugar de la mera presencia de ciertos conceptos en una bolsa, uno puede requerir que simultáneamente esté presente un cierto número de instancias de cada concepto. Si para cada concepto, el correspondiente número de instancias de ese concepto excede un umbral determinado (que puede ser diferente para cada concepto), la bolsa se etiqueta positiva, en otro caso negativa. La *hipótesis multinstancia basada en umbral* f_u requiere un número mínimo de instancias (dependiente del concepto) de cada concepto. Formalmente,

$$f_u(X) \Leftrightarrow \forall c_i \in \mathsf{C} : \Delta(X, c_i) \geq t_i$$

donde $t_i \in \mathbb{N}$ es el umbral inferior del concepto i-ésimo. Una extensión del ejemplo de «las llaves y las cerraduras» de [115] mencionado anteriormente ilustra un concepto multinstancia basado en umbral. «Asumamos que la puerta tiene más de una cerradura de cada tipo, y que las llaves tienen que usarse simultáneamente para poder abrir la puerta, es decir, necesitamos una llave para cada cerradura. Aquí necesitamos al menos tantas llaves para cada tipo de cerradura como cerraduras de este tipo haya en la puerta.» En este tipo de problemas, cada bolsa positiva (un mazo de llaves) tiene que tener un número mínimo de llaves (instancias) de cada tipo de cerradura (los conceptos).

En la *hipótesis multinstancia basada en conteo* f_c el número de instancias por concepto está acotado por un límite superior, así como un límite inferior. Formalmente,

$$f_c(X) \Leftrightarrow \forall c_i \in \mathsf{C} : t_i \leq \Delta(X, c_i) \leq z_i$$

donde $t_i \in \mathbb{N}$ es el umbral inferior del concepto i-ésimo y $z_i \in \mathbb{N}$ es el umbral superior del concepto i-ésimo. En [115] se presenta el siguiente ejemplo de aprendizaje para este tipo de problema. «Nos dan estadísticas diarias de los pedidos procesados por una compañía. Usualmente, un pedido es asignado a uno de los departamentos de la compañía. Queremos predecir si la carga de trabajo de la compañía está dentro de un rango óptimo,

donde ninguno de los departamentos tenga muy pocos pedidos (porque su eficiencia sería baja), o tenga demasiados pedidos (porque estaría sobrecargado). En una representación multinstancia de este problema, las bolsas son colecciones de pedidos (las instancias) en determinado día, y la etiqueta de clase indica si la compañía tuvo una carga de trabajo eficaz en ese día. Cada uno de los conceptos subyacentes $c_i \in \mathsf{C}$ asigna un pedido a un departamento. Para que la compañía trabaje eficientemente en un día, cada departamento tiene que trabajar dentro de su rango óptimo, es decir, el número de instancias de ese concepto debe estar limitado inferior y superiormente. Estos límites pueden ser diferentes para cada concepto.»

Estos tres tipos de conceptos multinstancia forman una jerarquía, i.e.,

$$basados\,en\,presencia \subset basados\,en\,umbral \subset basados\,en\,conteo.$$

Por tanto, la hipótesis multinstancia estándar, que como ya vimos, es un caso particular de hipótesis basada en presencia, es también, en último caso, una hipótesis basada en conteo, y es el ejemplo clásico de hipótesis multinstancia basada en conteo en la jerarquía de hipótesis multinstancia de la Figura 1.6.

Entre los algoritmos que implementan la hipótesis multinstancia basada en conteo están TLC [115] y CCE [135] que serán descritos en la Sección 1.4.4.1.

1.3.3. Hipótesis colectiva

Bajo la hipótesis multinstancia estándar, solo unas pocas instancias especiales (aquellas con una instancia «positiva») pueden tener alguna influencia sobre la etiqueta de clase. Al contrario, la hipótesis colectiva es una hipótesis multinstancia donde todas las instancias en una bolsa contribuyen igualmente a la etiqueta de la bolsa [120]. Un ejemplo en [33] ilustra la motivación de esta hipótesis. Considere la tarea de encontrar todas las imágenes de paisajes playeros. No se puede garantizar el cumplimiento de la hipótesis multinstancia estándar ya que no existe un elemento de un segmento de la imagen que por sí solo pertenezca a la categoría «playa». En cambio, esperaríamos encontrar un número de elementos que cada uno contribuya a la probabilidad de que la imagen sea una escena playera, tales como arena, agua, cielo, palmeras, pelotas de playa, caracolas, castillos de arena, y así sucesivamente.

La hipótesis colectiva está motivada por una visión de la naturaleza de las bolsas basada en la teoría de las probabilidades. Bajo está visión, una bolsa no es una colección finita de elementos fijos, como generalmente se asume, sino una muestra de una población específica subyacente a esa bolsa particular. Aquí, una bolsa X se modela

como una distribución de probabilidad $Pr(\mathcal{X}|X)$ sobre el espacio de instancia $\mathcal{X}$, donde las instancias observadas fueron generadas mediante muestreo aleatorio de esta distribución. En la jerarquía de hipótesis multinstancia de la Figura 1.6, la hipótesis colectiva es el ejemplo clásico de hipótesis multinstancia basada en distribución de probabilidad.

Se asume que a las instancias le son asignadas etiquetas de clase de acuerdo a alguna (comunmente desconocida) función de probabilidad (o proceso probabilístico no determinístico) $g(x) = Pr(\mathcal{Y}, x)$. Bajo la hipótesis colectiva, la función de probabilidad de clase a nivel de bolsa está determinada por el valor de clase esperado de la población de la bolsa. Sea $c \in \mathcal{Y} = \{0, 1\}$ una etiqueta de clase, y sea X una bolsa. Entonces

$$Pr(c|X) = E_{\mathcal{X}}[Pr(c|x)|X] = \int_{\mathcal{X}} Pr(c|x) Pr(x|X) dx$$

Para calcularlo exactamente, debemos conocer $Pr(x|X)$, la distribución de probabilidad para la bolsa. Sin embargo, en la práctica, ésta, por lo general, no se conoce. Por tanto, se utiliza la muestra provista por las instancias que están en la bolsa:

$$Pr(c|X) = \frac{1}{|X|} \sum_{i=1}^{|X|} Pr(c|x_i)$$

donde $|X|$ es el número de instancias en la bolsa. En el límite, a medida que el tamaño de la muestra tiende a infinito, la versión muestral de la ecuación tiende a la ecuación poblacional.

Entre los algoritmos que implementan la hipótesis multinstancia colectiva están MI-Wrapper [36], IFLIW [33], MILR [121] y el algoritmo amplificador de Xu & Frank [121] (los dos últimos se comentan en la Sección 1.4.3.4).

1.3.4. Hipótesis basadas en distancia

Existen varias hipótesis multinstancia basadas en distancia. Algunas cuantifican la distancia entre una instancia objetivo y una bolsa, mientras que otras miden la distancia entre una bolsa objetivo y otra bolsa. Un ejemplo de hipótesis multinstancia basada en la distancia entre instancias y bolsas es la del algoritmo MILES, que se describe en la sección 1.4.4.2. En MILES se toma la distancia entre la instancia objetivo y su más cercana instancia en la bolsa, y se le aplica la función Gaussiana para obtener un valor inverso a la distancia que es usado como criterio de semejanza entre la instancia objetivo y la bolsa. La hipótesis multinstancia de MILES asume que las etiquetas de clase a nivel de bolsa están de alguna forma determinadas por estos valores de semejanza.

Por otro lado, las hipótesis multinstancia basadas en distancias entre bolsas asumen

Figura 1.8: Distancia de Hausdorff.

que son las distancias entre las bolsas las que determinan en alguna manera las etiquetas de clase a nivel de bolsa. Para medir la distancia entre bolsas es común el uso de la *distancia de Hausdorff*. La distancia tradicional de Hausdorff entre dos conjuntos de puntos (o bolsas) $X = \{x_1, \ldots, x_{n_X}\}$ y $Z = \left\{z_1, \ldots, z_{n_{z'}}\right\}$ es la mayor distancia entre un punto X y su más cercano punto en Z, o vice versa. Formalmente, la distancia (máxima) de Hausdorff se define como

$$H_{max}(X, Z) = \text{máx}\{h(X, Z), h(X, Z)\} \tag{1.3}$$

donde

$$h(X, Z) = \underset{x\in X}{\text{máx}} \underset{z\in Z}{\text{mín}} \|x - z\|$$

y $\|x - z\|$ es la distancia entre los puntos x y z bajo alguna norma, usualmente la distancia Euclideana. La Figura 1.8 muestra una representación gráfica de la distancia de Hausdorff.

La distancia mínima de Hausdorff fue propuesta por (Wang & Zucker, 2000) [110] en el contexto de una actualización multinstancia del vecino más cercano discutida en la sección 1.4.3.1. En esta variante, la función h es remplazada por h_1, donde

$$h_1(X, Z) = \underset{x\in X}{\text{mín}} \underset{z\in Z}{\text{mín}} \|x - z\|$$

Observe que la distancia mínima de Hausdorff es simplemente la distancia más corta entre un punto en X y un punto en Z. Se puede formular como

$$H_{min}(X, Z) = \underset{x\in X, z\in Z}{\text{mín}} \|x - z\| \tag{1.4}$$

Zhang & Zhou [128] proponen adicionalmente la distancia promedio de Hausdorff, la

cual define la distancia promedio entre un punto en una bolsa y su punto más cercano en la otra bolsa como

$$H_{avg}(X,Z) = \frac{\sum_{x \in X} \min_{z \in Z} \|x - z\| + \sum_{z \in Z} \min_{x \in X} \|z - x\|}{|X| + |Z|} \tag{1.5}$$

Entre los algoritmos que implementan hipótesis multinstancia basadas en distancias entre bolsas están GMIL (Sección 1.4.2.3), BARTMIP (Sección 1.4.4.3) y los algoritmos basados en el enfoque de vecinos más cercanos (Sección 1.4.3.1).

1.3.5. Hipótesis basadas en estadísticas

Las hipótesis multinstancia basadas en estadísticas asumen que las etiquetas de clase a nivel de bolsa están relacionadas con alguna información estadística que puede extraerse de las bolsas. Los algoritmos que implementan este tipo de hipótesis sustituyen cada bolsa por un vector con información estadística obtenida de la bolsa. La información estadística extraída puede ser tan simple como los valores mínimos y máximos de cada atributo, usado por Kernel Minimax [45] (Sección 1.4.3.3), o los valores promedios de todas las instancias de la bolsa, usado por Simple-MI [120].

1.4. Algoritmos de clasificación multinstancia

El número de algoritmos de clasificación multinstancia que ha sido desarrollado en el breve lapso de tiempo transcurrido desde el nacimiento del aprendizaje multinstancia en 1997 hasta la actualidad es impresionante, de forma que sería difícil y engorroso hacer una lista que los nombrara a todos y, menos sensato aún, describirlos. Sin embargo, es posible identificar elementos claves del diseño de los algoritmos de clasificación multinstancia que permiten agruparlos en distintas categorías. En esta sección se presenta un sistema de categorías para los algoritmos de clasificación multinstancia, y se describen algunos algoritmos representativos de cada categoría.

1.4.1. Categorización de los algoritmos de clasificación multinstancia

Según la taxonomía de algoritmos multinstancia de Xu [120], los algoritmos multinstancia pueden dividirse de acuerdo al tipo de hipótesis multinstancia que implementan, distinguiendo, de esta forma, dos ramas fundamentales: aquellos que implementan una hipótesis basada en instancia y los que implementan una hipótesis basada en metadatos. En el enfoque basado en instancia se asume que las instancias tienen etiquetas de clase

ocultas. El conjunto de etiquetas ocultas a nivel de instancia puede o no coincidir con el conjunto de etiquetas de clase a nivel de bolsa. Los métodos de esta categoría tratan de estimar una función que asigne etiquetas de clase a las instancias, y luego usan esa función para hacer predicciones a nivel de bolsa. Esta categoría puede subdividirse adicionalmente de acuerdo a la hipótesis multinstancia específica que implemente al algoritmo. La división más básica es entre aquellos algoritmos que implementan la hipótesis multinstancia estándar y aquellos que implementan otras hipótesis. La importancia de la hipótesis multinstancia estándar es que fue el enfoque original del aprendizaje multinstancias, y por tanto muchos de los primeros algoritmos la implementaron y actualmente otros la siguen implementando.

Los métodos que usan el enfoque basado en metadatos se apoyan en la hipótesis de que las etiquetas de clase de las bolsas están determinadas por alguna información en un metanivel que describe los ejemplos. Los algoritmos de este tipo generalmente aplican una transformación que mapea bolsas en un nuevo espacio de atributos monoinstancia, donde los atributos consisten en algún metadato extraído de la bolsa. Luego, un algoritmo de aprendizaje monoinstancia puede aplicarse a las instancias en este espacio para hacer las predicciones. Los algoritmos del enfoque basado en metadatos pueden subdividirse según sea metadatos orientados a datos o metadatos orientados a modelo. La mayoría de los algoritmos basados en metadatos son orientados a datos, lo que significa que los metadatos se extraen directamente a partir de los datos. En los algoritmos orientados a modelo se construye un modelo de clasificación o agrupamiento sobre los datos, y los metadatos son extraídos a partir del modelo y no de los datos originales.

Otra forma de caracterizar a un algoritmo multinstancia es por el tipo de relación que tiene con los métodos del aprendizaje supervisado tradicional [33]. En este sentido pueden considerarse tres categorías: (1) aquellos algoritmos que no guardan relación alguna con los métodos del aprendizaje tradicional sino que están hechos *ad hoc*, i.e., diseñados específicamente para aprender conceptos multinstancia; (2) algoritmos obtenidos a partir de adaptaciones al enfoque multinstancia de métodos monoinstancia tradicionales; (3) algoritmos envoltorios que transforman un conjunto de datos multinstancia en uno monoinstancia equivalente al cual se aplica directamente (sin modificación alguna) un algoritmo de aprendizaje tradicional.

Las dos formas de caracterizar un algoritmo multinstancia descritas en esta sección son ortogonales (i.e., independientes entre sí) y lo suficientemente generales como para dar cabida a la mayoría de los algoritmos multinstancia existentes. Conjuntamente, estas dos formas de caracterización generan un espacio bidimensional de algoritmos multinstancia, donde pueden ser mapeados los distintos algoritmos existentes para su mejor

	Enfoque basado en instancias		Enfoque basado en metadatos	
	Hipótesis estándar	Otras hipótesis basadas en instancia	Orientado a datos	Orientado a modelo
Algoritmos ad-hoc	algoritmos APR [27] Densidad Diversa [64, 65] EM-DD [129]		GMIL [94] MI-Graph, mi-Graph [133]	
Algoritmos monoinstancia actualizados	Reglas de decisión [10, 19] Árboles de decisión [11, 19, 89] mi-SVM [2] Amplificación [3] Redes neuronales [82]	Regresión logística [121] Amplificación multinstancia [121]	CitationKNN, BayesianKNN [110] MI-SVM [2] Kernel minimax [45]	
Envoltorios para algoritmos monoinstancia	Mi-NB [69]	MI-Wrapper [36]	MILES [17] BARTMIP [128]	TLC [115] CCE [135]

Cuadro 1.3: Sistema de categorías para clasificar los algoritmos de clasificación multinstancia.

organización, comprensión y comparación con otros algoritmos.

El Cuadro 1.3 es una representación del espacio bidimensional de algoritmos multinstancia descrito anteriormente. En las columnas se han colocado las categorías tomadas de la taxonomía de Xu, y en las filas las categorías determinadas por el tipo de relación con los métodos del aprendizaje tradicional. Para ejemplificación, han sido ubicados en las entradas de la tabla algunos de los métodos más conocidos y representativos de cada categoría. De estos, se describen a continuación los algoritmos más importantes.

1.4.2. Algoritmos ad-hoc

Entre los algoritmos que han sido específicamente diseñados para resolver problemas multinstancia están los algoritmos de rectángulos paralelos a los ejes (APR) [27], los algoritmos basados en el enfoque de densidad diversa [64, 65] y el algoritmo GMIL [94] basado en resultados teóricos del reconocimiento de patrones geométricos.

1.4.2.1. Algoritmos APR

Dietterich et al. [27] presentaron en 1997 los primeros algoritmos de aprendizaje multinstancia, los cuales fueron diseñados para resolver problemas multinstancia bajo la hipótesis multinstancia estándar. Específicamente, los algoritmos se concibieron

para que funcionaran bien con el problema de predicción de actividad farmacológica *Musk*. Los algoritmos APR usan el enfoque basado en instancia, modelando el problema multinstancia como un proceso de dos pasos: un proceso de clasificación que es aplicado a cada instancia y luego un proceso de selección basado en la hipótesis multinstancia estándar.

Los algoritmos intentan construir un hiperrectángulo paralelo a los ejes (en inglés *axis-parallel hyper-rectangle*, APR) que identifique la región «positiva» del espacio de instancia. Un hiperrectángulo es una generalización de un rectángulo con un número arbitrario de dimensiones. Los hiperrectángulos paralelos a los ejes pueden verse como conjunciones de reglas si-entonces sobre los valores de los atributos. En el momento de la clasificación, cualquier bolsa que contenga una instancia dentro del hiperrectángulo es etiquetada como positiva, teniendo en cuenta la hipótesis multinstancia estándar. Éste es un enfoque paramétrico, en el que los parámetros a encontrar son los atributos útiles y los límites del hiperrectángulo a lo largo de estos atributos. Como enfoque paramétrico, el objetivo de estos métodos es encontrar los parámetros que, junto a la hipótesis multinstancia estándar, pueda explicar mejor la etiqueta de clase de todos los ejemplos en los datos de entrenamiento. En otras palabras, ellos buscan los parámetros involucrados en el primer paso de acuerdo a la etiqueta de clase que se obtiene en el segundo paso y a la relación asumida entre los dos pasos. Existen algoritmos estándares que pueden construir un APR (i.e. una regla si-entonces sencilla), pero estos no tienen en cuenta ninguna hipótesis multinstancia. Por lo tanto se necesita un algoritmo de propósito específico.

Dietterich et al. formularon varios tipos de algoritmos para el aprendizaje de conceptos multinstancia tipo-APR. El método APR más efectivo en el problema *Musk* fue Discriminación Iterativa. Este algoritmo tiene tres procedimientos principales: *crecimiento*, *discriminación* y *expansión*. El algoritmo comienza agrandando un hiperrectángulo sobre todos sus atributos con bordes ajustados sobre las instancias positivas. Luego discrimina entre los atributos, seleccionando un subconjunto de atributos que distinga mejor entre instancias positivas y negativas. Se repite el procedimiento de crecer, usando solamente los atributos seleccionados. El algoritmo itera, alternando entre crecimiento y discriminación hasta que converge en un hiperrectángulo estable. Sin embargo, esto crea un rectángulo muy conservador que no generaliza bien. Por tanto se añade otro paso para expandir el hiperrectángulo con el objetivo de incrementar la probabilidad de que el rectángulo contenga nuevas instancias positivas. Para refinar los límites del hiperrectángulo utilizaron estimación de densidad de kernel (KDE, siglas del inglés *kernel density estimation*).

1.4.2.2. Densidad diversa

Densidad diversa [64] es un enfoque probabilístico al aprendizaje multinstancia bajo la hipótesis estándar. La densidad diversa es una medida de la probabilidad de que un punto en el espacio de instancia sea el punto objetivo positivo (asumiendo que solo haya un punto objetivo) dadas las bolsas en los conjuntos de entrenamiento. Esto puede ser escrito como:

$$DD\left(x\right) = Pr\left(x = t \mid X_1^+, \ldots, X_n^+, X_1^-, \ldots, X_m^-\right)$$

donde $x \in \mathcal{X}$ es un punto en el espacio de instancia, y t es el concepto objetivo. La densidad diversa de un punto se calcula sobre la base del número de bolsas positivas *diferentes* que tienen instancias cercanas a ese punto, y cuán lejos están de este punto las instancias de las bolsas negativas (las cuales se asumen que son todas negativas de acuerdo a la hipótesis multinstancia estándar). La palabra «diversa» se usa para enfatizar el hecho de que las instancias deben provenir de bolsas diferentes para que contribuyan efectivamente a la medida.

Dado que se asume un único punto objetivo, la meta de un algoritmo de aprendizaje dentro del esquema de densidad diversa es encontrar el punto con la máxima densidad diversa. Asumiendo que las bolsas son condicionalmente independientes dado el concepto objetivo, y asumiendo además probabilidades previas uniforme para el punto objetivo, puede calcularse el punto de máxima densidad diversa como

$$\arg\max_x \prod_{1\leq i\leq n} Pr\left(x = t \mid X_i^+\right) \prod_{1\leq j\leq m} Pr\left(x = t \mid X_j^-\right)$$

Lógicamente, los términos de probabilidad en esta ecuación deben ser definidos para poder realizar el cálculo. Maron [64] propone dos modelos para estimar la probabilidad de que un punto sea el concepto objetivo dada una bolsa. Uno es el modelo o-ruidoso, que usa una versión probabilística del operador lógico «o», y el otro es el modelo causa-más-probable, el cual selecciona en una bolsa positiva la instancia que tiene la mayor probabilidad de ser positiva. Maron y Lozano-Perez [65] proponen un algoritmo (que llamaremos maxDD) que intenta encontrar el concepto objetivo maximizando la densidad diversa sobre el espacio de instancia. Como no es posible explorar todo el espacio de instancia en busca del punto con la mayor densidad diversa, se usa un método de gradiente ascendente. En cada instancia de una bolsa positiva se reinicia la búsqueda, porque las instancias positivas causan máximos en la densidad diversa, y por tanto el concepto objetivo debe estar cerca de alguna de esas instancias presumiblemente.

Después de hallar el punto de máxima densidad diversa se calcula una distancia umbral que optimice el desempeño de la clasificación en una validación cruzada del conjunto de entrenamiento. Una bolsa se asigna a la clase positiva si la distancia desde el punto de máxima densidad diversa hasta alguna de sus instancias está por debajo del umbral.

Zhang & Goldman [129] formularon luego el algoritmo EM-DD, el cual es una variante de maxDD basada en un enfoque de maximización de la esperanza o expectación (en inglés, *expectation-maximization*, EM). Aunque EM-DD usa el mismo marco teórico que maxDD, tiene un método distinto de buscar el punto objetivo más probable.

El algoritmo comienza con un supuesto concepto objetivo inicial h, que se obtiene seleccionando un punto de una bolsa positiva, tal como en maxDD. Luego se ejecuta un procedimiento iterativo consistente en un paso de *expectación* seguido por un paso de *maximización*. En el paso de expectación se selecciona una instancia de cada bolsa positiva que es con mayor probabilidad la causa de la etiqueta de la bolsa dada la hipótesis actual h, usando el estimador causa-más-probable de maxDD. Después, en el paso de maximización se ejecuta una búsqueda de gradiente ascendente basada en las instancias seleccionadas (de forma similar a maxDD) para encontrar una nueva h' que maximice $DD(h')$. La actual hipótesis h es reinicializada con h'. El ciclo EM se repite hasta la convergencia.

EM-DD tiene una exactitud similar a maxDD pero mejora la eficiencia computacional. El algoritmo maxDD usa una función *softmax* en lugar de una función max porque es diferenciable, lo cual es requerido para la búsqueda del gradiente ascendente. Sin embargo, la función softmax aumenta la complejidad computacional del algoritmo. Dado que EM-DD selecciona una sola instancia de cada bolsa durante el paso de expectación, el cálculo de la función max se hace trivial y de esta forma evita el costoso cálculo de softmax. Esto también causa que el algoritmo escale bien con el incremento del tamaño de las bolsas.

1.4.2.3. GMIL

Scott et al. [94] formularon la hipótesis multinstancia generalizada GMIL, donde una bolsa es positiva si y solo si contiene instancias suficientemente cerca a al menos r de k posibles puntos objetivos, y suficientemente cerca a lo sumo a s' de k' puntos de repulsión. Dada la medida de distancia empleada, «suficientemente cerca» es equivalente a estar dentro de una caja de tamaño apropiado paralela a los ejes de los puntos objetivos (o de repulsión).

Su primer algoritmo para aprender este tipo de concepto se llamó GMIL-1. El algoritmo explícitamente enumera todas las posibles cajas paralelas a los ejes, y crea un

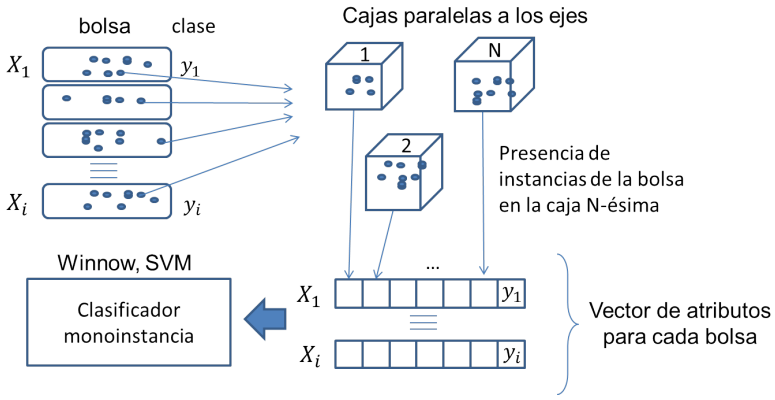


Figura 1.9: Esquema de funcionamiento general del algoritmo de clasificación multinstancia GMIL.

espacio de atributos monoinstancia. A cada bolsa asigna un vector de atributos lógicos (booleanos), un atributo por cada caja, que significan cuándo la bolsa contiene una instancia dentro de esa caja. Para reducir la dimensionalidad de este espacio, las cajas que cubren las mismas instancias son agrupadas juntas, y solo se utiliza una caja representativa para cada grupo. Todas las bolsas de entrenamiento son mapeadas a este espacio de atributos, y se entrena el algoritmo monoinstancia Winnow [61] en el conjunto de datos transformado. Winnow es similar al algoritmo Perceptron para aprender redes neuronales de capa simple, pero actualiza sus pesos de forma multiplicativa, en lugar de aditivamente. Esto provoca una mayor velocidad de convergencia, particularmente cuando muchos atributos son irrelevantes.

El funcionamiento general del algoritmo GMIL es mostrado en la Figura 1.9. Primeramente se generan todas las posibles cajas paralelas a los ejes alrededor de las instancias de todas las bolsas de entrenamiento, luego se mapean las bolsas de entrenamiento en un nuevo espacio de atributos, donde un atributo vale 1 si la caja asociada contiene una instancia de la bolsa, en otro caso vale 0. A cada vector se añade la etiqueta de clase de la bolsa correspondiente. Una vez que las bolsas han sido transformadas al nuevo espacio de atributos, se construye un modelo con un clasificador simple instancia que es usado para clasificar las nuevas bolsas una vez que son mapeadas al espacio de atributos.

La tarea de enumerar todas las cajas paralelas a los ejes es exponencial en el número de dimensiones, lo cual hace a GMIL-1 muy ineficiente. GMIL-2 [103] es un intento de mejorar la eficiencia computacional y en memoria del algoritmo. GMIL-2 es en general igual a GMIL-1, pero selecciona grupos de cajas de forma distinta. Primero, GMIL-2 reduce el número de instancias a considerar seleccionando un subconjunto de instancias

representativas, Ψ. Luego construye grupos considerando las cajas representadas por los límites de cada posible subconjunto de Ψ. Se usa un enfoque de búsqueda primero a lo ancho para intentar encontrar eficientemente los conjuntos de grupos que son geométricamente válidos, i.e. todas las instancias dentro de los límites del subconjunto de Ψ estén contenidas dentro del grupo.

Aunque GMIL-2 es mucho más eficiente que GMIL-1, todavía sufre de escalabilidad limitada [104]. En un intento adicional de mejorar la complejidad computacional del algoritmo, Tao et al. presentaron una reformulación del problema de aprendizaje GMIL basada en kernel. El kernel, $k_\wedge$, permite aplicar una máquina de soporte vectorial directamente al problema. Como el cálculo de $k_\wedge$ tiene severas dificultades con la escalabilidad, que pronto hacen el problema intratable a medida que aumenta su tamaño, los autores presentaron un esquema de aproximación aleatoria completamente-polinomial (FRAPS, acrónimo del inglés *Fully Polynomial Randomized APproximation Scheme*) para resolverlo.

Luego fue creado en [105] un nuevo kernel basado en $k_\wedge$, llamado k_{min} que extiende el modelo GMIL para manejar los conceptos multinstancia de Weidmann basados en conteo. Tao et al. encontraron que el algoritmo de kernel k_{min} ejecuta mejor que $k_\wedge$ en varios tipos de problema.

1.4.3. Algoritmos tradicionales adaptados al aprendizaje multinstancia

Un enfoque frecuente en el diseño de métodos de aprendizaje multinstancia es la modificación de un algoritmo monoinstancia para que maneje problemas de aprendizaje multinstancia. La literatura de aprendizaje supervisado tiene muchos algoritmos de aprendizaje monoinstancia que están bien sustentados tanto teórica como empíricamente, y estos algoritmos pueden proveer bases sólidas a partir de las cuales formular algoritmos multinstancia. Frecuentemente un clasificador monoinstancia puede ser transformado para que maneje problemas multinstancia con sólo cambios menores en el algoritmo, ahorrando así tiempo de diseño también. Los algoritmos de aprendizaje supervisado que han sido adaptados al escenario multinstancia incluyen el k-vecinos más cercano, los árboles de decisión, las máquinas de soporte vectorial, la regresión logística y la amplificación (*boosting*), entre otros.

1.4.3.1. Enfoque de vecinos más cercanos

En el aprendizaje supervisado monoinstancia, el algoritmo del vecino más cercano hace las predicciones de clasificación etiquetando a las instancias con la etiqueta de clase

de la instancia más cercana en los datos de entrenamiento (bajo alguna norma, como la distancia Euclideana). El algoritmo puede hacerse más robusto tomando en cuenta no solo la instancia más cercana, sino las k instancias más cercanas. Las predicciones se hacen mediante el voto mayoritario de las etiquetas de clase de los k vecinos. El método de los k vecinos más cercanos es conocido por sus siglas en inglés kNN (*k-nearest neighbour*).

La forma más directa de adaptar kNN al aprendizaje multinstancia es definiendo una norma (medida de distancia) para bolsas multinstancia. De esta forma kNN puede ser aplicado directamente. Para esta norma multinstancia, Wang & Zucker [110] usaron la distancia jerarquizada de Hausdorff (Sección 1.3.4). Ellos encontraron que el voto mayoritario usado por el kNN estándar producía frecuentemente resultados subóptimos en el escenario multinstancia. Para remediarlo, Wang y Zucker intentaron dos variaciones en el método kNN multinstancia.

El primer algoritmo propuesto fue Bayesian-kNN [110], el cual remplaza el voto mayoritario de los k vecinos con un método probabilístico que estima la etiqueta de clase más probable aplicando el teorema de Bayes.

El otro algoritmo que Wang y Zucker propusieron a partir de la modificación del kNN fue CitationKNN [110]. El método está basado en un marco teórico del campo de la bibliotecología y las ciencias de la información, el cual define relevancia entre documentos (particularmente artículos académicos) a partir de sus referencias y sus citas. Bajo este enfoque, si un artículo académico cita un trabajo anterior (conocido como *referencia*), se dice que ese artículo está relacionado con el anterior. Similarmente, un artículo que es citado por una publicación posterior (el *citador*) se considera que está relacionado con el trabajo posterior.

El concepto de R referencias más cercanas coincide con el tradicional k vecinos más cercanos, donde $k = R$. En cambio, los citadores es un nuevo concepto que hace más robusto al CitationKNN. Los citadores C-cercanos de X son las bolsas cuyos C vecinos más cercanos incluyen a X. Más precisamente, sea $Rank(X, Z)$ una función de ordenamiento tal que si Z es el n-ésimo vecino más cercano de X, entonces $Rank(X, Z)$ devuelve n. Entonces, los citadores C-cercanos de X son las bolsas que retornan valores de ordenamiento de vecinos para X inferiores a C, i.e., $citadores_C(X) = \{Z : Rank(X, Z) \leq C\}$. El concepto de los citadores más cercanos se ilustra en la Figura 1.10 a través del ejemplo de los citadores 3-cercanos de X_0. Los pequeños circulos rellenos representan los ejemplos. Los círculos grandes sin rellenar engloban a los 3 vecinos más cercanos del ejemplo que se encuentra en su centro. Por claridad, solo han sido representados los círculos que incluyen a X_0. Los citadores 3-cercanos de X_0 son (los ejemplos X_1 al X_5) aquellos cuyos 3 vecinos más cercanos incluyen a X_0.

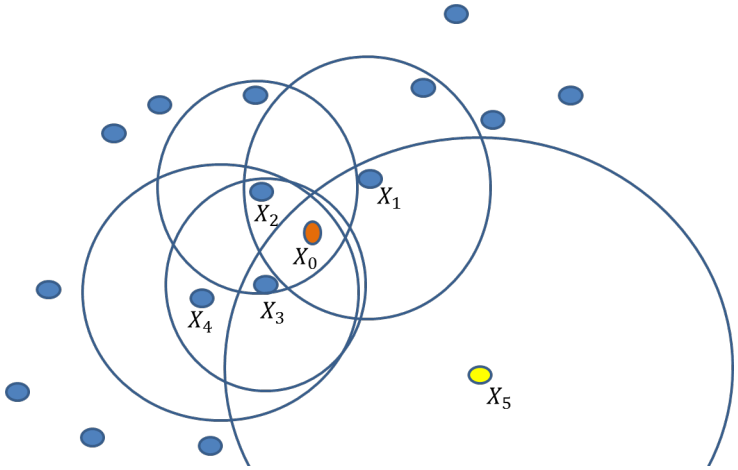


Figura 1.10: Ejemplo que ilustra el concepto de los citadores C-cercanos.

Para clasificar una instancia, CitationKNN recolecta las R referencias más cercanas y los C citadores más cercanos, y retorna una predicción positiva si y solo si hay estrictamente más instancias positivas que negativas en la colección combinada de referencias y citadores. Los empates son rotos a favor de las predicciones negativas porque las bolsas positivas pueden contener instancias negativas que pueden haber afectado erroneamente la clasificación a favor de etiquetas positivas, mientras que lo inverso no es posible bajo la hipótesis multinstancia estándar.

1.4.3.2. Árboles y reglas de decisión

Los árboles y reglas de decisión del aprendizaje supervisado estándar han sido adaptados en diversas versiones al aprendizaje multinstancia. Los métodos tradicionalmente llamados árboles de decisión estructuran el espacio de hipótesis en una jerarquía de nodos, en los cuales se compara el valor de un atributo con una constante especificada. El árbol generalmente se construye de arriba hacia abajo de manera recursiva.

Cuando se construye un árbol de decisión, el reto es determinar cuáles atributos escoger para decidir en cuál rama colocar las instancias y dónde poner los *puntos de división* para estas decisiones. Muchos algoritmos de aprendizaje modernos de árboles de decisión, tales como ID3 [79] y C4.5 [80], usan *ganancia de información* para seleccionar los atributos y los puntos de división. La cantidad de información es una medida de la teoría de la información que a grandes rasgos representa el número de *bits* requeridos para determinar si una instancia es positiva o negativa dado que la instancia alcanzó determinado nodo en el árbol. La ganancia de información entre un nodo padre y su hijo es la cantidad de

información requerida en el padre menos la información requerida en el hijo [117].

La ganancia de información se calcula usando un concepto llamado *entropía*. Chevaleyre & Zucker [19] formularon una versión de entropía para datos multinstancia, lo cual les permitió adaptar tanto ID3 como C4.5 al marco multinstancia con muy poca modificación adicional.

Por otro lado, el árbol de decisión multinstancia RELIC [89] se basa en extraer metadatos de las bolsas. La forma en que RELIC asigna ejemplos a los nodos del árbol es equivalente a extraer cierto metadato de cada ejemplo, específicamente los valores mínimos y máximos, y luego aplica el algoritmo monoinstancia C4.5 a los datos transformados. Como RELIC examina los valores de los atributos a lo largo de cada dimensión individualmente, tal metadato de un ejemplo no se corresponde con alguna instancia específica dentro de él.

MITI [11] es un algoritmo de aprendizaje de árboles de decisión de arriba hacia abajo basado en la hipótesis multinstancia estándar. Toma como entrada instancias simples, aplica pruebas estándares a los valores de los atributos, y eventualmente clasifica las instancias como positivas o negativas. En la etapa de predicción cada instancia de la nueva bolsa obtiene una clasificación particular del arbol construido, y la bolsa es clasificada como positiva si al menos una de sus instancias lo es. El árbol se hace crecer de una manera primero-el-mejor, manteniendo una cola de prioridad que contiene los nodos no expandidos en el borde del árbol. Si el nodo que encabeza la cola contiene solo instancias de bolsas positivas, se marca como nodo hoja, y se eliminan todas las bolsas positivas que contienen instancias en este nodo hoja.

La representación de reglas de decisión es una alternativa a los árboles de decisión, donde un conjunto arbitrario de reglas lógicas determina la clasificación de una instancia. Chevaleyre & Zucker [19] usaron una estrategia para actualizar algoritmos de reglas de decisión similar a la empleada en sus árboles de decisión. Muchos algoritmos de reglas de decisión aprenden conjuntos de reglas por medio de un enfoque de *cubrimiento*, donde las reglas son añadidas incrementalmente, con cada nueva regla cubriendo algunas instancias que no habían sido aún cubiertas por reglas anteriores. Definiendo una noción de cubrimiento para datos multinstancia, Chevaleyre & Zucker fueron capaces de adaptar al enfoque multinstancia los algoritmos de aprendizaje de reglas de decisión AQ, CN2 y CHARADE.

MIRI [10] es otro algoritmo de aprendizaje de reglas de decisión cuyo funcionamiento se basa en otro principio. MIRI usa árboles MITI parcialmente construidos para aprender y representar reglas. Cuando el algoritmo básico MITI encuentra una hoja positiva, se eliminan del conjunto de entrenamiento todas las bolsas positivas asociadas con esta

hoja, se genera una regla si-entonces a partir del camino que va desde el nodo raíz hasta la hoja, y se reinicia el algoritmo sobre el resto de los datos. Se desecha la estructura del árbol y se construye un nuevo árbol desde cero sobre el conjunto de datos reducido.

1.4.3.3. Máquinas de soporte vectorial

Los algoritmos de máquinas de soporte vectorial (SVM, por sus siglas en inglés) [21] son un enfoque popular al aprendizaje monoinstancia. Estos algoritmos intentan aprender un hiperplano separador que divida el espacio de instancia en regiones positiva y negativa. Un hiperplano es una generalización de la noción de línea en geometría plana Euclideana, y de un plano en geometría tridimensional. Una línea en geometría plana Euclideana divide el plano en dos regiones, una a cada lado de la línea. Un plano tiene función similar en el espacio tridimensional. Intuitivamente, un hiperplano tiene la misma función en un espacio con arbitrario número de dimensiones. El hiperplano aprendido por una máquina de soporte vectorial provee una frontera de decisión para la clasificación.

Los algoritmos SVM intentan aprender un hiperplano de *margen máximo*, donde el hiperplano separa las instancias de entrenamiento pero está tan lejos como sea posible tanto de las instancias de entrenamiento positivas como de las negativas. Cuando se aplica directamente al espacio de instancias, el hiperplano aprendido por un algoritmo SVM es una frontera de decisión lineal. Sin embargo, este modelo no es suficientemente sofisticado para algunos dominios de problema. Por ejemplo, una frontera de decisión lineal no puede representar la función o-exclusivo (XOR) de la lógica proposicional. Afortunadamente, fronteras de decisión más complejas pueden ser representadas por hiperplanos en espacios de dimensión superior, donde los atributos son derivados a partir de combinaciones de los atributos en el espacio de atributos original. Las instancias son mapeadas en espacios de dimensiones superiores, y para la clasificación se usa la frontera de decisión en el espacio dimensional superior.

Un nuevo avance fue el desarrollo de las funciones kernel para las máquinas de soporte vectorial. Las funciones kernel eliminan la necesidad de hacer realmente el mapeo en el espacio dimensional superior (y consecuentemente más gasto computacional en ese espacio) simulando el producto interno de vectores en ese espacio. Resulta que el producto interno es la única operación requerida para construir el hiperplano separador, por tanto al calcular la función kernel en el espacio de atributos original, el hiperplano para el espacio dimensional superior puede ser calculado sin ejecutar realmente el mapeo.

Para aplicar el enfoque SVM al aprendizaje multinstancia, Gärtner et al. [45] formularon dos funciones kernel para datos multinstancia: el *Kernel multinstancia*, el cual está basado en el kernel conjunto [50] y el *Kernel Minimax*, que hace un mapeo de las

bolsas en un espacio consistente en los valores mínimo y máximo de cada atributo.

A diferencia del enfoque de Gärtner et al. que solo modifican el kernel mientras dejan la formulación SVM sin cambios, Andrews et al. [2] consideran que el problema multinstancia requiere una revisión más radical de las máquinas de soporte vectorial. Para ellos, la principal cuestión a considerar cuando se adapta un SVM al enfoque multinstancia es cómo definir el margen. Andrews et al. [2] proponen dos maneras diferentes de definir el margen en un SVM multinstancia y presentan un algoritmo para cada caso. Ambos algoritmos están diseñados para aprender bajo la hipótesis multinstancia estándar. Dado que en la hipótesis multinstancia estándar todas las instancias en bolsas negativas son negativas, el margen para las bolsas negativas se define como en el SVM tradicional. En cambio, la definición de margen para las bolsas positivas debe tener en cuenta la posible presencia de instancias negativas dentro de la bolsa positiva.

Una de las alternativas propuestas por Andrews et al. [2] es el MI-SVM, que define un margen a nivel de bolsa. Para las instancias positivas se define el margen de una bolsa como la distancia máxima entre el hiperplano y cada una de las instancias de la bolsa. Como en una bolsa positiva al menos una instancia debe ser positiva, se trata de que al menos una de esas instancias tenga un margen positivo grande. Las instancias negativas en las bolsas positivas son ignoradas y solo una instancia en cada bolsa positiva contribuye a la optimización del hiperplano.

La otra alternativa desarrollada por Andrews et al. [2] fue mi-SVM. Mientras que MI-SVM trata de identificar la instancia más probablemente positiva dentro de cada bolsa positiva, mi-SVM trata de identificar la etiqueta de cada instancia en cada bolsa positiva. Este algoritmo construye un hiperplano sobre el espacio de instancias que intenta incluir al menos una instancia de cada bolsa positiva en la mitad positiva del espacio, y poner todas las instancias de las bolsas negativas en la mitad negativa del espacio, mientras mantiene todavía un margen máximo bajo estas restricciones. Las instancias negativas en las bolsas positivas, así como múltiples instancias positivas de una misma bolsa pueden ser soportes vectoriales.

1.4.3.4. Regresión logística y amplificación

La regresión logística es un método estadístico frecuentemente empleado en el aprendizaje monoinstancia, donde se construye un modelo lineal en una versión transformada de la variable objetivo. La transformación empleada es la *transformación logit*, la cual convierte una probabilidad en el logaritmo del *odds*[3] de esa probabilidad.

[3]El odds de una probabilidad p es $p/(1-p)$.

Xu & Frank [121] presentaron MILR, una adaptación al enfoque multinstancia para el algoritmo de regresión logística basada en la hipótesis colectiva (descrita en la Sección 1.3.3), donde cada instancia contribuye igual e independientemente a la etiqueta de clase de su bolsa.

En el caso monoinstancia el algoritmo calcula las probabilidades de clase a nivel de instancia. En el aprendizaje multinstancia éstas no son fáciles de determinar debido a que solo se tienen las etiquetas de clase a nivel de bolsa. Sin embargo, la probabilidad de clase a nivel de bolsa es calculada fácilmente bajo la hipótesis colectiva, y la función de probabilidad a nivel de instancia puede ser estimada a partir de ésta al maximizar el logaritmo de la verosimilitud binomial.

Xu & Frank usaron dos variantes de hipótesis colectiva: en una versión, la etiqueta de clase de una bolsa se determina por medio de la media geométrica, y en la otra versión se hace a través de la media aritmética. En el caso de la media geométrica, el algoritmo convierte el problema multinstancia en un problema de regresión logística monoinstancia, donde cada bolsa está representada por la media de las instancias en la bolsa. Desafortunadamente, esta simplificación no se aplica cuando se usa la media aritmética.

Amplificación (*boosting* en inglés) [37] es un algoritmo de metaprendizaje que fue desarrollado basado en el trabajo de la teoría del aprendizaje computacional. Sin embargo, análisis posteriores desde una perspectiva estadística [38] explicaron el algoritmo como una regresión logística aditiva. Efectivamente, el algoritmo estima la función *log-odds* (como en la regresión logística) basado en un modelo aditivo. Dada la relación entre los algoritmos, Xu & Frank [121] fueron capaces de adaptar el algoritmo de amplificación Adaboost al enfoque multinstancia usando más o menos el mismo método usado para la regresión logística.

La mayoría de los algoritmos de amplificación consisten en aprender iterativamente clasificadores débiles con respecto a una distribución y añadirlos a un clasificador final fuerte. Un clasificador débil es aquel que está ligeramente correlacionado con la verdadera clasificación, mientras que un clasificador fuerte está arbitrariamente bien relacionado con la verdadera clasificación. Cuando el clasificador débil se añade al clasificador final, comunmente se pondera de alguna manera usualmente relacionada con la exactitud del clasificador débil. El peso de los datos es recalculado después de añadir el clasificador débil: los ejemplos mal clasificados ganan peso y los ejemplos bien clasificados pierden peso. De esta forma los siguientes clasificadores débiles se enfocarán más en los ejemplos que fueron mal clasificados por los clasificadores débiles anteriores.

Otro método para aplicar algoritmos de amplificación al aprendizaje multinstancia

fue investigado por Auer & Ortner [3]. Ellos presentaron un algoritmo multinstancia de aprendizaje débil para ser usado como clasificador base para el algoritmo de amplificación Adaboost estándar. A diferencia del enfoque de Frank & Xu, el cual usa clasificadores base monoinstancia para el algoritmo de amplificación, el método de Auer y Ortner usa un clasificador base multinstancia. El algoritmo amplificador es en cambio inalterado.

1.4.4. Algoritmos envoltorio

Otro enfoque al aprendizaje multinstancia es construir algoritmos generales que sean capaces de aplicar cualquier clasificador monoinstancia arbitrario a los datos multinstancia. Esos métodos son llamados algoritmos «*envoltorios*» ya que ellos envuelven un algoritmo de aprendizaje monoinstancia dado para crear un nuevo algoritmo multinstancia. A diferencia de los métodos tratados en la sección anterior, el algoritmo monoinstancia no es modificado en lo absoluto. En lugar de ello, se aplica un proceso de proposicionalización[4] para crear una versión de los datos a la cual puedan aplicarse los algoritmos del aprendizaje supervisado. La salida del algoritmo monoinstancia se usa de alguna manera para generar las predicciones a nivel de bolsa.

Los algoritmos envoltorios multinstancia pueden aplicarse tanto a conceptos multinstancia basados en instancia como en los basados en metadatos. Para los conceptos multinstancia basados en instancias, la hipótesis es que las instancias están etiquetadas a través de algún proceso, y estas etiquetas determinan las etiquetas de clase a nivel de bolsa. Los algoritmos envoltorios que aprenden conceptos basados en instancias usualmente aplican el aprendizaje monoinstancia directamente a las instancias dentro de las bolsas de entrenamiento. En el momento de la predicción las etiquetas a nivel de bolsa son asignadas teniendo en cuenta la hipótesis multinstancia que haya asumido, por ejemplo la hipótesis multinstancia estándar.

En contraste, se asume que las etiquetas de clase a nivel de bolsa para los conceptos multinstancia basados en metadatos son determinadas por un proceso en algún espacio de atributos monoinstancia que consiste en metadatos extraídos de las bolsas. Los algoritmos envoltorios que aprenden este tipo de conceptos usualmente mapean cada bolsa en el espacio de atributos basado en metadatos a través de alguna transformación, y entonces construyen un modelo monoinstancia en el espacio de atributos transformado. Cada instancia en el espacio de atributos basado en los metadatos corresponde a una bolsa. Entonces, el clasificador base monoinstancia puede proporcionar directamente la

[4] A diferencia de los problemas relacionales donde usualmente existen múltiples relaciones que hacen intratable el método de proposicionalización, en el enfoque multinstancia solo se tiene una sola relación: bolsa-instancia, lo cual hace posible la aplicación de este método.

etiquetas de clase a nivel de bolsa.

1.4.4.1. Envoltorios orientados a modelos

El algoritmo de clasificación en dos niveles (en inglés Two-Level Classification, TLC) [115] está diseñado para aprender los tipos de conceptos multinstancia descritos en la jerarquía de conceptos de Weidmann (vea la Sección 1.3.2 para detalles).

Estos conceptos multinstancia consisten en un conjunto de conceptos a nivel de instancia que están relacionados de alguna manera a los conceptos a nivel de bolsa. Siguiendo la tradición de la literatura multinstancia, el problema de aprendizaje a nivel de bolsa es un problema de clasificación de dos clases. Sin embargo, puede haber un número arbitrario de conceptos a nivel de instancia. Se asume que las etiquetas de clase a nivel de bolsa están determinadas por la *cantidad* de cada concepto a nivel de instancia en una bolsa.

El algoritmo TLC aprende en un proceso de dos pasos. El primer paso aprende conceptos a nivel de instancia usando un árbol de decisión. El árbol se construye usando todas las instancias de todas las bolsas en los datos de entrenamiento, con las etiquetas de clase puestas según las etiquetas de las bolsas contenedoras. Las instancias son ponderadas de forma que cada bolsa tenga el mismo peso y la suma de los pesos totaliza el número de instancias. Se usa la ganancia de información para seleccionar el atributo a emplear en cada nodo de decisión. Se aplica una heurística simple de prepoda, donde los nodos dejan de dividirse cuando la suma del peso de las instancias en ese nodo es menor que dos. No se aplica ningún otro método de poda.

Se considera que cada nodo del árbol representa un concepto. Entonces, cada bolsa se convierte a una representación monoinstancia, con un atributo para cada nodo del árbol (i.e. cada concepto), el valor del cual será el número de instancias que alcanzan ese nodo en el árbol de decisión.

El segundo paso aprende conceptos a nivel de bolsa basándose en los conceptos a nivel de instancia descubiertos en el primer paso. Se aplica un algoritmo de aprendizaje monoinstancia a los datos transformados. El mismo mapeo se aplica en la etapa de clasificación a las nuevas bolsas, y el clasificador monoinstancia hace las predicciones a nivel de bolsa.

Weidmann et al. [115] encontraron que el método TLC, usando troncos de decisión amplificados (en inglés *boosted decision stumps*) como clasificadores base, fue muy competitivo en el problema Musk, y también se desempeñó extremadamente bien en conjuntos de datos artificiales donde se usó un modelo generativo a partir de la jerarquía de concepto de Weidmann.

Otro algoritmo envoltorio orientado a modelo, el método de ensamblaje de agrupamiento constructivo (en inglés *constructive clustering ensemble*, CCE) de Zhou & Zhang [135], usa también un método de proposicionalización que puede ser apropiado para algunos conceptos tipo Weidmann. El algoritmo usa un método de agrupamiento para formar d grupos a partir de todas las instancias de las bolsas de entrenamiento. Las bolsas son mapeadas en un espacio de atributos lógicos (booleanos) donde cada atributo corresponde a un grupo. El valor del atributo es verdadero si el grupo contiene alguna instancia de la bolsa, en otro caso es falso. Luego se utiliza un clasificador monoinstancia para construir un modelo sobre el conjunto de datos resultante. El algoritmo se repite k veces para distintos valores de d. Las predicciones de la clasificación se obtienen por medio del voto mayoritario de los k clasificadores monoinstancia del ensamble resultante.

Dado que el espacio de atributos construido por CCE representa la presencia o ausencia de instancias con ciertas propiedades, este algoritmo puede ser más adecuado para aprender conceptos multinstancia basados en presencia. Sin embargo, el algoritmo puede extenderse fácilmente para aprender conceptos basados en conteo, si el espacio de atributos transformado se modifica para incluir el número de ocurrencias de conceptos a nivel de instancia. Es decir, el atributo correspondiente a un grupo, en lugar de contener un valor booleano, tendría el número de instancias de la bolsa que están incluídas en el grupo.

1.4.4.2. MILES

MILES [17] es un enfoque de aprendizaje multinstancia basado en el marco de densidad diversa [65]. MILES es el acrónimo de *Multiple-Instance Learning via Embedded Instance Selection* que significa aprendizaje multinstancia mediante la selección de instancias incrustadas. MILES incrusta bolsas en un espacio de atributo monoinstancia, y aplica un algoritmo de soporte vectorial con norma 1 al conjunto de datos transformado. Sin embargo, en lugar del SVM norma 1, podrían usarse otros algoritmos de aprendizaje monoinstancia, por tanto clasificamos este algoritmo como un enfoque envoltorio.

Los primeros algoritmos basados en densidad diversa usan la hipótesis multinstancia estándar y asumen además la existencia de un único punto objetivo. MILES relaja esta hipótesis. En su lugar, usa una hipótesis simétrica, donde se permite múltiples puntos objetivos que pueden estar relacionados lo mismo a bolsas positivas como a bolsas negativas. Bajo esta hipótesis, y usando el estimador causa-más-probable del marco densidad diversa, Chen et al. definen una medida que estima la probabilidad de que un punto x sea un punto objetivo dada una bolsa X_i, sin importar la etiqueta de

clase de la bolsa:

$$Pr\left(x \mid X_i\right) \propto s\left(x \mid X_i\right) = \max_j \exp\left(-\frac{\left\|x_{ij} - x\right\|^2}{\sigma^2}\right) \tag{1.6}$$

donde x_{ij} son las instancias dentro de la bolsa X_i, y σ es un factor escala predefinido. Observe que $s\left(x \mid X_i\right)$ puede considerarse como una medida de similitud entre una bolsa y una instancia, la cual es determinada por la instancia x y la instancia más cercana dentro de la bolsa X_i.

Hasta aquí, permanece la cuestión de como encontrar los puntos objetivos. En aras de eficiencia computacional, MILES usa la hipótesis de que los puntos objetivos pueden aproximarse por las instancias dentro de las bolsas. En otras palabras, cada instancia es candidata para un punto objetivo. Los candidatos son representados como atributos en un espacio de atributos basado en instancia $\mathbb{F}_{\mathfrak{X}}$. Cada bolsa en el conjunto de entrenamiento se mapea en $\mathbb{F}_{\mathfrak{X}}$ por medio de la transformación

$$m\left(X_i\right) = \left[s\left(x^1 \mid X_i\right), s\left(x^2 \mid X_i\right), \ldots, s\left(x^n \mid X_i\right)\right]^T$$

donde $x^i \in \mathfrak{X}$ es una instancia del conjunto $\mathfrak{X}$ de todas las instancias en todas las bolsas de entrenamiento. Si se añaden las etiquetas de clase $y \in \mathcal{Y}$, el espacio $(\mathbb{F}_{\mathfrak{X}} \mid \mathcal{Y})$ es un espacio de atributos monoinstancia al cual pueden aplicarse los algoritmos del aprendizaje supervisado tradicional. La salida de este clasificador puede usarse para asignar etiquetas de clase a nivel de bolsa para datos futuros.

Chen et al. usaron el algoritmo SVM norma-1 como el clasificador base, debido a la alta dimensionalidad del espacio de atributos. SVM norma-1 es razonablemente eficiente para aprender en conjuntos de datos de alta dimensionalidad porque puede ser entrenado usando programación lineal, y se sabe que generalmente pone la mayoría de los atributos a cero, lo cual realiza efectivamente una selección de atributos. Esto ahorra el tener que realizar un proceso adicional de selección de atributos, el cual puede ser computacionalmente muy costoso.

1.4.4.3. BARTMIP

El algoritmo BARTMIP [128] está muy relacionado con MILES, e implícitamente se apoya en una hipótesis multinstancia relacionada. Mientras que MILES asume que las etiquetas de bolsa están relacionadas con distancias a nivel de instancia desde conjuntos de puntos objetivos, el método BARTMIP asume que las etiquetas de bolsa están relacionadas con distancias desde *bolsas* objetivo.

Las distancias entre las bolsas pueden calcularse a través de la distancia de Hausdorff. Zhang & Zhou [128] usan tres variantes diferentes de distancia de Hausdorff para definir la distancia entre bolsas: la distancia de Hausdorff máxima, mínima y promedio (definidas en la Sección 1.3.4). La medida de distancia que se escoja determinará los conceptos multinstancia que pueden ser representados. Podrían usarse otras medidas de distancias a nivel de bolsa si estas fueran apropiadas para un dominio de problema específico.

Inicialmente el algoritmo BARTMIP hace un agrupamiento a nivel de bolsa con las bolsas de entrenamiento usando el algoritmo k-medoides adaptado al aprendizaje multinstancia mediante el empleo de una variante de medida de Hausdorff como su medida de distancia. Además de agrupar las bolsas en k grupos, el algoritmo de agrupamiento también devuelve el medoide de cada grupo, es decir, el elemento más cercano al centro del grupo.

Luego, las bolsas de entrenamiento son mapeadas en un espacio de atributos monoinstancia k-dimensional, donde el atributo i-ésimo corresponde a la distancia de la bolsa al i-ésimo medoide, empleando la misma medida de distancia entre bolsas que la usada en la etapa de agrupamiento. Las etiquetas de clase de las bolsas originales se añaden a las instancias tranformadas, y se aplica un algoritmo base de aprendizaje monoinstancia al espacio de atributos resultante. En la etapa de clasificación se realiza el mismo mapeo a la bolsa de prueba y se obtienen las predicciones del algoritmo base de aprendizaje monoinstancia. La Figura 1.11 esquematiza el funcionamiento del algoritmo. Observe que este método es idéntico a MILES excepto porque usa una transformación a un espacio de atributos diferente.

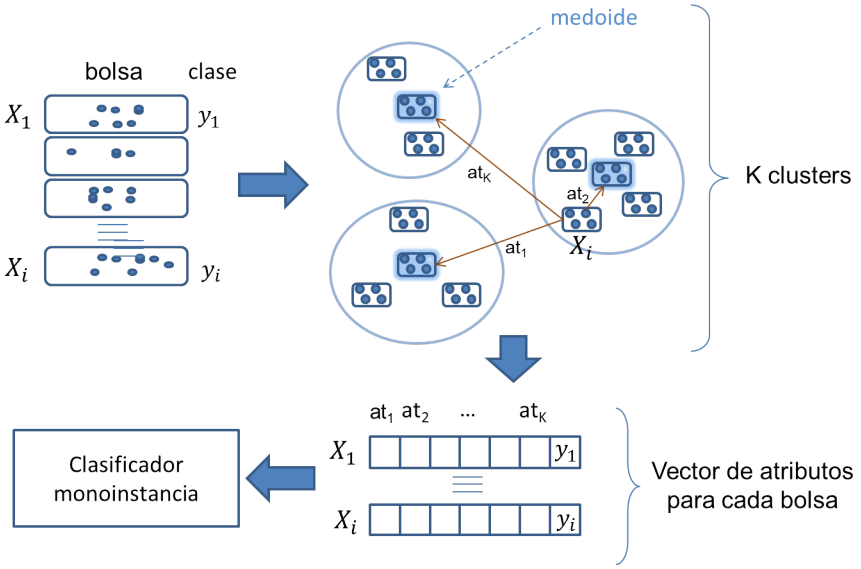


Figura 1.11: Esquema de funcionamiento del algoritmo de clasificación multinstancia BARTMIP.

2 MISMOTE: una solución al problema de clases desbalanceadas en clasificación multinstancia

Los problemas de clasificación multinstancia son susceptibles a padecer del problema de las clases desbalanceadas, lo cual afecta negativamente el desempeño de los algoritmos de clasificación multinstancia. Sin embargo, este importante problema ha recibido poca atención por parte de los investigadores y profesionales del área. Solo conocemos una propuesta de solución general para enfrentar el problema de las clases desbalanceadas en la clasificación multinstancia: los algoritmos amplificadores sensibles al costo, los cuales no están exentos de limitaciones. Una de las alternativas más importantes de solución frente a los algoritmos sensibles al costo en la clasificación monoinstancia, es la de modificar la distribución de los datos de entrenamiento a través de técnicas de muestreo para equilibrar el balance de las clases. Sin embargo, no conocemos la existencia de algún algoritmo que implemente este tipo de soluciones para la clasificación multinstancia, por lo que su realización sería muy deseable.

El objetivo propuesto en este trabajo fue diseñar un algoritmo que permita aplicar sobremuestreo aleatorio de la clase minoritaria en problemas de clasificación multinstancia, introduciendo bolsas sintéticas en la clase pequeña. El algoritmo fue bautizado MISMOTE (acrónimo del inglés *Multi-Instance Synthetic Minority Over-sampling TEchnique*). Las técnicas de muestreo que introducen ejemplos sintéticos en la clase minoritaria a partir de los datos originales han resultado muy exitosas en la clasificación monoinstancia. La hipótesis de trabajo fue que el rebalanceo de los conjuntos de entrenamiento usando el método MISMOTE en los problemas de clasificación multinstancia con clases desbalanceadas permitiría mejorar de forma apreciable el desempeño de la clasificación. Esta hipótesis fue puesta a prueba con un experimento y confirmada con métodos estadísticos. Otro experimento mostró que el algoritmo propuesto es muy competitivo con los amplificadores sensibles al costo de la literatura.

2.1. Problema de las clases desbalanceadas

El problema de las clases desbalanceadas aparece en los problemas de clasificación cuando existe una gran diferencia entre el tamaño de las clases. Bajo esta condición se observa un deterioro en el desempeño de los sistemas de clasificación tradicionales. Particularmente, altos niveles de error en la clasificación de los ejemplos de la clase minoritaria. Este problema es significativo porque generalmente la clase minoritaria es la más importante. Por ejemplo, es común encontrar enfermedades severas que afectan a una entre miles de personas. Este es un caso típico de desbalance de clases. Aunque la enfermedad sea rara, es muy importante identificarla en la persona que la padece (acertar el caso positivo) para poder tratarla. Otro ejemplo es el de la detección de operaciones financieras fraudulentas. Aunque la gran mayoría de las operaciones sean lícitas, es sumamente importante detectar cualquier operación fraudulenta, por las grandes pérdidas económicas que una sola de ellas puede ocasionar.

Aunque a este fenómeno se le ha dado en llamar «problema de las clases desbalanceadas», en realidad la diferencia entre el tamaño de las clases no es la esencia del problema, sino una condición que favorece la manifestación de otros problemas que subyacen en segundo plano. Entre las causas primarias que dan lugar a los efectos observados tras el problema de las clases desbalanceadas, algunos de los más conocidos son los siguientes:

- Uso de medidas del desempeño inadecuadas para guiar el proceso de aprendizaje: Los algoritmos de aprendizaje que usen de forma explícita o implícita heurísticas basadas en maximizar medidas del desempeño tales como la exactitud de la clasificación (*accuracy*) tenderán a ignorar los ejemplos de la clase minoritaria. Como estas medidas tratan los aciertos de ejemplos positivos de igual manera que los aciertos de ejemplos negativos, al ser estos últimos mucho más abundantes, los aciertos de ejemplos positivos pasan a ser irrelevantes. Consideremos un conjunto de datos desbalanceado, que tenga, digamos, un ejemplo positivo por cada 100 negativos. Un clasificador que trate de maximizar el *accuracy* de su regla de clasificación podría obtener una exactitud del 99 % con solo ignorar los ejemplos de la clase positiva, clasificando todos los ejemplos como negativos.

- Pequeño tamaño de la muestra: Para que el aprendizaje sea efectivo se necesitan conjuntos de ejemplos representativos de cada clase. Si el número de ejemplos de la clase minoritaria es muy bajo el desempeño de la clasificación puede afectarse por falta de representatividad [53].

- Solapamiento entre clases: Las clases pueden estar (en alguna medida) solapadas por una pobre definición intrínseca de los conceptos que ellas representan, o porque los atributos seleccionados no son suficientes para diferenciarlas adecuadamente [5]. Naturalmente, mientras mayor es el solapamiento entre las clases más difícil se hace su separación. Sin embargo, las clases desbalanceadas amplifican el efecto: menores niveles de solapamiento son suficientes para observar un marcado deterioro del desempeño de la clasificación [75, 43].

- Presencia de subconceptos en la clase minoritaria: La presencia de subconceptos es otro factor que aumenta la complejidad de un problema de clasificación debido a que implica una descripción más larga de la frontera de decisión. Las fronteras de decisión más largas requieren más ejemplos para su especificación. Por eso, aunque los subconceptos pueden aparecer en cualquier clase, cuando están en la clase minoritaria, puede ser particularmente difícil su identificación por estar más esparcida la muestra [51].

Usualmente el desbalance entre el tamaño de dos clases es caracterizado mediante la *tasa de desbalance* (IR[1]), que se define como el cociente del número de ejemplos en la clase mayoritaria entre el número de ejemplos en la clase minoritaria [44, 73]. Mientras mayor es el IR, mayor es el desbalance entre las clases. Un elevado IR está asociado con una mayor dificultad para su clasificación. Sin embargo, no puede decirse explícitamente a partir de qué valor de IR es apreciable el deterioro de la clasificación, debido al carácter heterogéneo de la complejidad de los conjuntos de datos.

La mayoría de los algoritmos de aprendizaje tradicionales no son apropiados para datos con clases desbalanceadas, generando modelos de clasificación subóptimos en estos casos, i.e., una buena cobertura de los ejemplos de la clase mayoritaria, pero frecuentes errores en la clasificación de los ejemplos de la clase minoritaria.

Actualmente es ampliamente reconocida la necesidad de emplear medidas robustas para evaluar la calidad de la clasificación cuando las clases están desbalanceadas, i.e., medidas que no sean falseadas por la prevalencia de una clase, y cuantifiquen fielmente cuán bueno es un clasificador, ya sea identificando ejemplos de la clase minoritaria, o identificando ejemplos de ambas clases, como mejor convenga a la aplicación. En la sección siguiente se presentan algunas de estas medidas, las cuales son usadas en este estudio.

[1] IR son las siglas del término inglés *imbalance ratio*.

	Clasificado como Positivo	Clasificado como Negativo
Realmente Positivo (AP)	Verdaderos Positivos (TP)	Verdaderos Negativos (FN)
Realmente Negativo (AN)	Falsos Positivos (FP)	Falsos Negativos (TN)

Cuadro 2.1: Matriz de confusión para un problema de dos clases.

2.1.1. Cómo medir el desempeño de la clasificación cuando las clases están desbalanceadas

Varias medidas del desempeño de la clasificación han sido sugeridas para el problema del desbalance de clases sin llegar a consenso sobre la conveniencia de alguna en particular. Cada medida de desempeño permite evaluar un aspecto diferente de la clasificación, es por ello que en distintos campos de aplicación se prefieren medidas de desempeño específicas.

En un problema de clasificación de dos clases hay cuatro posibles salidas, las cuales están representadas por la matriz de confusión que se muestra en el Cuadro 2.1[2]. Esta puede considerarse la información más básica sobre el desempeño de la clasificación, a partir de la cual se definen medidas más avanzadas. En este trabajo se usan las siguientes medidas de desempeño:

- *Precision* y *recall*: estas medidas provienen del campo de la recuperación de información. *Precision* (Fórmula (2.1)) es una medida de exactitud, ya que da la fracción de ejemplos clasificados como positivos que son realmente positivos. Por otro lado *recall* (Fórmula (2.2)), también llamada exactitud de la clase positiva, es una medida de completitud ya que da la fracción de ejemplos de la clase positiva que han sido clasificados correctamente. Observe que estas medidas se centran en la identificación de la clase minoritaria.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

- F_1: es también una medida usada en el campo de la recuperación de información. Se define mediante la Fórmula (2.3) como la media armónica de *precision* y *recall*. Por lo general, el objetivo principal de un algoritmo de aprendizaje es maximizar el *recall*, sin sacrificar la *precision*. Sin embargo, ambos objetivos entran en conflicto frecuentemente, y es difícil decidir el mejor entre dos algoritmos si uno aventaja al

[2]La nomenclatura entre paréntesis responde a los nombres en inglés.

otro en *precision* y el otro aventaja al primero en *recall*. F1 incorpora el equilibrio costo-beneficio de *recall* y *precision*, dando igual peso a ambas medidas, para evaluar el desempeño de un clasificador a través de un solo número. Al igual que *precision* y *recall*, F1 se enfoca solo en la clase positiva.

$$F_1 = \frac{2}{1/recall + 1/precision} = \frac{2 \times recall \times precision}{recall + precision} \tag{2.3}$$

- AUC: es el área bajo una curva ROC[3]. La curva ROC describe equilibrios entre beneficios (TP) y costos (FP) a lo largo de un rango de umbrales de un modelo de clasificación. Los gráficos ROC son consistentes para un problema dado aun si la distribución de instancias positivas y negativas está altamente desbalanceada [76, 30]. El AUC permite medir el desempeño de un clasificador para evaluar cuál modelo es en promedio mejor usando un único valor numérico obtenido a partir de la curva ROC. Su valor representa la probabilidad de que un ejemplo positivo seleccionado aleatoriamente sea evaluado (correctamente) con más seguridad que un ejemplo negativo seleccionado aleatoriamente. AUC tiene un sólido significado estadístico ya que es equivalente a la prueba de rangos de Wilcoxon. Dados n puntos (TP_i, FP_i) de una curva ROC creada a partir de AP ejemplos positivos y AN ejemplos negativos, usamos la Fórmula (2.4) para el cálculo exacto del AUC basado en la regla trapezoidal. El proceso de cálculo se ilustra en la Figura 2.1. $\Delta TP_i = TP_i - TP_{i+1}$, $\Delta FP_i = FP_i - FP_{i+1}$ y $S_i = AN - FP_i$.

$$AUC = \frac{1}{AP \cdot AN} \sum_{i=1}^{n-1} \Delta TP_i \left(S_i + \frac{1}{2} \Delta FP_i \right) \tag{2.4}$$

A diferencia de las medidas anteriores, AUC toma en cuenta la identificación correcta de ambas clases.

2.1.2. Soluciones existentes para mitigar el problema de clases desbalanceadas en la clasificación monoinstancia

De acuerdo con la literatura especializada [40, 63, 102, 51], el problema de clases desbalanceadas en la clasificación monoinstancia puede enfrentarse por medio de cuatro tipos principales de soluciones:

[3]ROC son las siglas del término inglés *Receiver Operating Characteristic*, ya que originalmente se utilizó para caracterizar el funcionamiento de los radares. Mientras que AUC son las siglas de *Area Under the (ROC) Curve*.

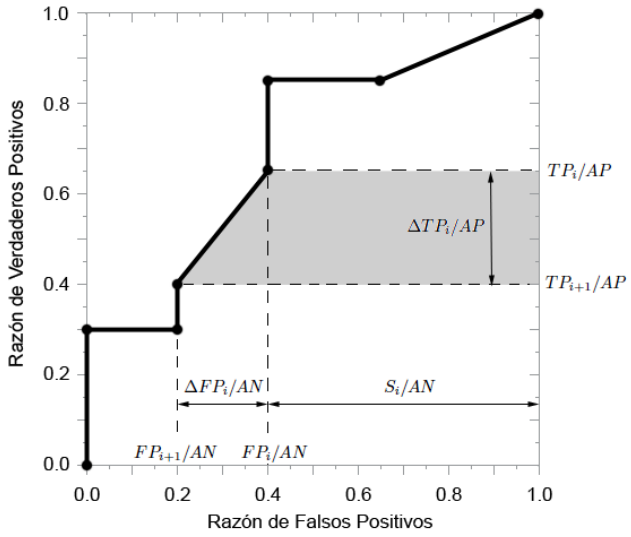


Figura 2.1: Un ejemplo de curva ROC y el cálculo del AUC.

1. Soluciones a nivel de algoritmo: Los algoritmos tradicionales de aprendizaje de clasificadores son adaptados para favorecer el aprendizaje de la clase pequeña. Se ha probado que estas soluciones a nivel de algoritmo son efectivas para ciertos algoritmos de aprendizaje de clasificadores y en ciertos dominios de aplicaciones, sin embargo son difíciles de extender a otros contextos porque requieren un profundo conocimiento del algoritmo de aprendizaje y del por qué fallan cuando la distribución de clases de los datos disponibles está desbalanceada. Por este motivo, son preferibles las soluciones en las que la estrategia de aprendizaje subyacente permanece sin cambios.

2. Soluciones sensibles al costo: Se basan en los diferentes costos asociados a los errores de clasificación en las clases o incluso en ejemplos específicos, ya que el costo del error en la clasificación de un ejemplo de la clase más pequeña es, por lo general, más elevado. La estrategia en este tipo de solución es minimizar el costo de la clasificación general. La principal desventaja del aprendizaje sensible al costo es que está basado en una matriz de costo para diferentes tipos de errores o ejemplos. Sin embargo, muchas veces no se dispone de una matriz de costo para un conjunto de datos dado.

3. Soluciones de remuestreo: Alteran la distribución de clases mediante un remuestreo del espacio de datos buscando el rebalance de las clases. Las formas más sencillas de remuestreo son el submuestreo aleatorio de la clase mayoritaria y el sobremuestreo

aleatorio de la clase minoritaria. Estas estrategias pueden conducir a la pérdida de información y el sobreajuste, respectivamente. Para tratar de evitar estos inconvenientes se han usado algunas heurísticas para seleccionar deliberadamente los ejemplos que son eliminados, en el llamado submuestreo informado de la clase mayoritaria, y aquellos que son remuestreados en el llamado sobremuestreo informado de la clase minoritaria.

4. Soluciones de ensambles: Los ensambles construyen múltiples clasificadores a partir de los datos originales y luego agregan sus predicciones cuando clasifican ejemplos desconocidos. Los ensambles pueden ser de tipo remuestreo o sensibles al costo. Los ensambles de remuestreo se basan en la aplicación de una técnica de remuestreo a una parte de los datos antes del entrenamiento de cada uno de los clasificadores que forman parte del ensamble. En los ensambles sensibles al costo el peso de los ejemplos se calcula tomando en cuenta, no solo la frecuencia de las clasificaciones erradas, sino también los costos de los errores.

La solución que proponemos en esta tesis es un algoritmo de sobremuestreo aleatorio que genera ejemplos sintéticos para la clase minoritaria. Este algoritmo está inspirado en el algoritmo SMOTE (acrónimo del inglés *Synthetic Minority Over-sampling TEchnique*) que realiza la misma tarea en el aprendizaje monoinstancia. A continuación, se describe el algoritmo SMOTE.

2.1.3. SMOTE

En SMOTE se sobremuestrea la clase minoritaria tomando cada ejemplo de la clase minoritaria e introduciendo ejemplos sintéticos a lo largo de los segmentos de línea que unen el ejemplo con cada uno de los k vecinos más cercanos de la clase minoritaria. Dependiendo de la cantidad de sobremuestreo que se vaya a realizar, se seleccionan aleatoriamente varios ejemplos de entre los k vecinos más cercanos. Sea x_0 un ejemplo de la clase minoritaria, y x_i uno de los vecinos más cercanos de x_0 seleccionado aleatoriamente, entonces se genera un nuevo ejemplo sintético s_{0i} a partir de x_0 y x_i, calculándo su valor mediante la Fórmula (2.5).

$$s_{0i} = x_0 + \lambda \left(x_i - x_0 \right) \tag{2.5}$$

En esta fórmula, λ es un número aleatorio entre 0 y 1 generado para cada componente del vector s_{0i}. Esto causa la selección de un punto aleatorio a lo largo del segmento de línea entre los valores de dos atributos específicos. El proceso se ilustra en la Figura 2.2,

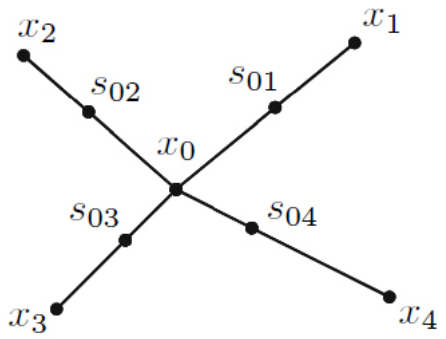


Figura 2.2: Ejemplo de interpolación aleatoria hecha por SMOTE.

donde x_0 es el punto que se toma como partida para generar las instancias sintéticas. En este ejemplo, se seleccionaron para x_0 los cuatro vecinos más cercanos $x_1, \ldots x_4$, y se crearon los puntos sintéticos $s_{01}, \ldots s_{04}$ por la interpolación aleatoria de x_0 con cada uno de los puntos $x_1, \ldots x_4$. Este enfoque fuerza a la región de decisión de la clase minoritaria a extenderse más al interior de la clase mayoritaria, haciéndose más general.

2.1.4. Soluciones al problema de las clases desbalanceadas en el aprendizaje multinstancia

Wang et al. [113] estudiaron el uso de algoritmos amplificadores sensibles al costo en conjuntos de datos multinstancia desbalanceados. Los métodos amplificadores, introducidos en la Sección 1.4.3.4, son algoritmos ensambles secuenciales que recalculan el peso de los datos después de cada iteración, de forma que el próximo clasificador débil en ser entrenado se centre más en los ejemplos que fueron mal clasificados la vez anterior. A continuación se describe más detalladamente el amplificador AdaBoost [91], el cual es por mucho el algoritmo amplificador más popular para clasificación multinstancia.

Sea $\mathcal{T} = \{(x_1, y_1), \cdots, (x_M, y_M)\}$ un conjunto de ejemplos de entrenamiento donde cada instancia x_i pertenece a un espacio de instancias $\mathcal{X}$ y, enfocándonos en problemas de clasificación de dos clases, cada etiqueta y_i pertenece al conjunto $\mathcal{Y} = \{-1, +1\}$. El amplificador usa un algoritmo de aprendizaje base (también llamado débil) que acepta como entrada un conjunto de ejemplos de entrenamiento $\mathcal{T}$ junto con una distribución de pesos D sobre $\{1, \cdots, M\}$, i.e., sobre los índices de $\mathcal{T}$. Dada esa entrada, el algoritmo de aprendizaje base calcula una hipótesis base (o débil) h. El signo de $h(x)$ es interpretado como la etiqueta que debe ser asignada a x. La idea de la amplificación es usar el algoritmo de aprendizaje base para formar una regla de predicción de gran exactitud al

invocar el algoritmo de aprendizaje base repetidamente en diferentes distribuciones sobre los ejemplos de entrenamiento. Inicialmente los pesos son uniformemente distribuidos con valor $1/M$. En cada iteración el peso de los ejemplos correcta (o incorrectamente) clasificados por h es disminuido (o aumentado). Específicamente, el peso $D^{t+1}(i)$ del ejemplo x_i en la iteración $t+1$ se calcula mediante las Fórmulas (2.6) y (2.7).

$$D^{t+1}(i) = \frac{D^t(i) K_t(x_i, y_i)}{Z_t} \tag{2.6}$$

$$K_t(x_i, y_i) = \exp(-\alpha_t y_i h_t(x_i)) \tag{2.7}$$

Z_t es un factor de normalización escogido de forma tal que D^{t+1} sea una distribución y α_t controla la influencia de cada hipótesis base h_t. La hipótesis final del amplificador, dada por la Fórmula (2.8), es el signo de la suma ponderada de las predicciones de los T clasificadores base.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{2.8}$$

En [91] se demuestra que el error de entrenamiento del clasificador final está acotado como

$$\frac{1}{M} |i : H(x_i) \neq y_i| \leq \prod_{t=1}^{T} Z_t \tag{2.9}$$

donde

$$Z_t = \sum_i D^t(i) K_t(x_i, y_i) \tag{2.10}$$

La fórmula de α_t es inducida al minimizar Z_t en cada iteración como

$$\alpha_t = \frac{1}{2} \log\left(\frac{\sum_{i, y_i = h_t(x_i)} D^t(i)}{\sum_{i, y_i \neq h_t(x_i)} D^t(i)}\right) \tag{2.11}$$

Los amplificadores pueden usarse con cualquier algoritmo de aprendizaje base siempre que este pueda manejar pesos en los ejemplos. Si se usa un algoritmo de aprendizaje multinstancia como algoritmo de aprendizaje base, entonces el amplificador será capaz de aprender a partir de conjuntos de datos multinstancia (en lugar de conjuntos de datos monoinstancia). La estrategia propuesta en [113] es usar un clasificador multinstancia como algoritmo de aprendizaje base de un ensamble amplificador *sensible al costo*. Ellos incluyen en su estudio diversas variantes de ensambles amplificadores sensibles al costo. Cada variante introduce, de una forma diferente, información de costo dentro de la

Nombre de la variante	Modificación en la fórmula de actualización de peso
Ab2	$K_t(X_i, y_i) = C_i \exp(-\alpha_t y_i h_t(X_i))$
Ab3	$K_t(X_i, y_i) = C_i \exp(-C_i \alpha_t y_i h_t(X_i))$
Ab4	$K_t(X_i, y_i) = C_i^2 \exp\left(-C_i^2 \alpha_t y_i h_t(X_i)\right)$

Cuadro 2.2: Tres variantes de AdaBoost sensible al costo usadas para clasificación multinstancia en problemas con clases desbalanceadas.

fórmula de actualización de pesos de AdaBoost. El Cuadro 2.2 presenta las tres mejores variantes de AdaBoost sensible al costo reportadas por [113]. El costo del error de la clasificación del ejemplo X_i es representado por C_i. Para cada variante, la fórmula que le corresponde en el Cuadro 2.2 sustituye a la Fórmula (2.7) en el cálculo de los pesos. Observe que las x_i de la Fórmula (2.7), que representan las instancias, han sido sustituidas en la tabla por X_i, que representan bolsas. En la primera variante, Ab2, la información de costo C_i se introduce fuera del exponente de la fórmula de actualización de peso. En la segunda variante, Ab3, la información de costo C_i se introduce dentro y fuera del exponente. Y en la tercera, Ab4, la información de costo C_i se introduce dentro y fuera del exponente, elevada al cuadrado. En [113] asumen que todos los ejemplos de una misma clase tienen igual costo. Como la matriz de costo puede ser escalada al multiplicar cada elemento de la matriz por una constante positiva, la búsqueda de la matriz óptima se reduce a encontrar la proporción adecuada entre el costo para la clase positiva y el costo para la clase negativa. La proporción se busca iterativamente tratando de maximizar una medida de desempeño.

Los clasificadores base usados en [113] para los algoritmos amplificadores sensibles al costo son MITI [11] y MIRI [10]. MITI fue descrito en la Sección 1.4.3.2. MIRI construye reglas de decisión a partir de árboles MITI parcialmente construidos.

Wang et al. [113] muestran que en los conjuntos de datos multinstancia estudiados los algoritmos amplificadores sensibles al costo mejoran el desempeño de la clasificación tanto de los clasificadores base como de los amplificadores estándares.

En otro trabajo, Zhou et al. [136] aborda los problemas multinstancia y multietiqueta (introducidos en la Sección 1.1.6 como MIME) al aplicar directamente un SVM en un marco de regularización. En el aprendizaje MIME cada ejemplo de entrenamiento, además de contener muchas instancias (sin etiquetas), también pueden ser asignados a múltiples etiquetas de clase. El aprendizaje multinstancia puede considerarse como un caso particular del aprendizaje MIME donde el conjunto de etiquetas asignadas a cada bolsa contiene solo un elemento. El algoritmo D-MimlSvm que ellos proponen incor-

pora un mecanismo para tratar con el desbalance de clases. Basados en el método de rescalamiento [131], ellos estiman la proporción de desbalance para cada etiqueta de clase y ponderan la función de pérdida de acuerdo a estas proporciones. Aunque la comparación hecha en [136] entre D-MimlSvm y otros métodos MIME (que no tienen en cuenta el problema del debalance de clases), reporta mejoras para D-MimlSvm, no queda claro hasta qué punto las mejoras observadas se deben al mecanismo incluido para tratar el desbalance de clases o a las diferencias en las estrategias de aprendizaje. Observe que esta es una solución a nivel de algoritmo, por tanto con aplicabilidad limitada, como se explicó en la Sección 2.1.2.

2.2. Algoritmo MISMOTE para el preprocesamiento de datos con clases desbalanceadas

La esencia del algoritmo MISMOTE consiste en tomar cada ejemplo de la clase minoritaria e introducir ejemplos sintéticos a lo largo del segmento de línea que une a algunos de los k vecinos más cercanos de la clase minoritaria. Más detalladamente, a partir de un ejemplo de entrenamiento P, se genera un nuevo ejemplo sintético S seleccionando aleatoriamente uno de los k vecinos más cercanos de P, que llamaremos Q, e interpolándolos a ambos P y Q. Sin embargo, en el escenario multinstancia cada ejemplo es una bolsa de instancias en lugar de un único vector, y los nuevos ejemplos sintéticos también son bolsas. Por tanto, la pregunta es *¿cómo puede definirse la interpolación de dos bolsas*?

Como las bolsas de entrenamiento están definidas por sus instancias, una forma razonable de obtener la interpolación de dos bolsas es interpolando las instancias de ambas bolsas. Sean P y Q dos bolsas de entrenamiento de la clase minoritaria que contienen N_P y N_Q instancias respectivamente. Entonces, a partir de P y Q puede generarse una nueva bolsa S que contiene $N_P \times N_Q$ instancias sintéticas por la interpolación de los pares de puntos tomados respectivamente de P y Q. Sin embargo, la introducción de bolsas sintéticas en el conjunto de entrenamiento cuyos tamaños son cuadráticos con respecto a los de las bolsas de entrenamiento original podría impactar negativamente el desempeño de muchos algoritmos de entrenamiento. Por un lado, podría introducirse un sesgo o predisposición en la clasificación al ser modificada la distribución del tamaño de las bolsas en la clase minoritaria, por otro lado, esto podría conducir a un sustancial incremento del costo computacional. Además, algunos problemas multinstancias pueden requerir que todas las bolsas tengan el mismo tamaño. Por tanto, es deseable que el tamaño de las bolsas sintéticas sea similar al de las bolsas de entrenamiento original.

Por otro lado, como en algunos problemas multinstancia la distribución del tamaño de la bolsa puede estar relacionada con la presencia de subconceptos, también es importante adaptar el tamaño de la bolsa sintética a las variaciones de la distribución del tamaño de la bolsa que puedan ocurrir *localmente* en el espacio de bolsas. Por esta razón, definimos el tamaño de una bolsa sintética S como el promedio de los tamaños de las $k\prime$ bolsas más cercanas a la bolsa de entrenamiento que da lugar a S. El número $k\prime$ de vecinos más cercanos define el alcance del concepto de localidad en el espacio de bolsa y, aunque no necesariamente tiene que ser el mismo número k de vecinos más cercanos a partir de los cuales se selecciona la bolsa involucrada en la interpolación, en nuestra implementación hacemos $k\prime = k$.

Una forma de contraer la bolsa S en una bolsa reducida s que retenga todavía las características que la hacen pertenecer a la clase positiva es tomando una muestra representativa de S. Recurrimos al muestreo aleatorio simple con remplazo para seleccionar respectivamente de P y Q pares de instancias para ser interpoladas. En otras palabras, cada instancia sintética se obtiene muestreando aleatoriamente una instancia de P y otra de Q, e interpolándolas.

La Figura 2.3 presenta el seudocódigo de MISMOTE. El algoritmo se basa solamente en la información de las T bolsas de entrenamiento $D = \{X_1, \ldots, X_T\}$ de la clase minoritaria para generar bolsas sintéticas. Para simplificar la descripción, asumimos que para cada bolsa original se generan N bolsas sintéticas, aunque realmente esto solo sucede cuando el número total de bolsas a ser generadas es un múltiplo entero de T. En otro caso, después de repartir equitativamente la responsabilidad de generar las bolsas sintéticas entre las T bolsas de entrenamiento, el resto de las bolsas sintéticas serán generadas por bolsas de entrenamiento seleccionadas aleatoriamente (sin remplazo). Por ejemplo, si se tienen 10 bolsas de entrenamiento y se desean generar 25 bolsas sintéticas, se generarán dos bolsas sintéticas a partir de cada bolsa de entrenamiento y además otras 5 bolsas sintéticas serán generadas por 5 bolsas de entrenamiento seleccionadas al azar.

El algoritmo itera sobre las T bolsas de entrenamiento de la clase minoritaria. Se determinan los k vecinos más cercanos de la clase minoritaria para cada bolsa de entrenamiento X_i, utilizando la distancia de Hausdorff definida en la Fórmula (1.3), y se calcula el tamaño $Size_i$ que será asignado a las bolsas sintéticas generadas a partir de X_i. El número k de vecinos más cercanos es un parámetro del algoritmo.

A partir del paso 6 se procede a generar N bolsas sintéticas para la bolsa de entrenamiento X_i. Para generar una bolsa sintética a partir de X_i, se selecciona a partir de los k vecinos más cercanos de X_i otra bolsa X_{ij} y se realiza la interpolación entre X_i

y X_{ij}. Para generar la bolsa sintética S_{ij}, a partir del paso 9, se genera un número $Size_i$ de instancias sintéticas. Para generar la instancia sintética r-esima de S_{ij}, s_{ijr}, se seleccionan aleatoriamente una instancia x_{ir} de X_i y una x_{ijr} de X_{ij}. Del paso 12 al 17 se recorre el espacio de atributos, y en el paso 16 se realiza, atributo por atributo, la interpolación de ambas instancias. Cada instancia generada se añade a la bolsa sintética S_{ij} en el paso 18, y tras completarse la bolsa sintética se adiciona al conjunto de bolsas sintéticas D'. Al terminar el algoritmo, retorna el conjunto de todas las bolsas sintéticas D'.

Este algoritmo permite aplicar la técnica de sobremuestreo al escenario multinstancia por medio de la generación de bolsas con características intermedias a aquellas que le rodean. Los elementos principales de su diseño son: (1) la definición del tamaño de la bolsa sintética como el promedio de los tamaños de las bolsas más cercanas, y (2) la selección de las instancias a ser interpoladas a través de muestreo simple aleatorio con remplazo. El método preserva la distribución local del tamaño de la bolsa y es independiente de cualquier hipótesis multinstancia. En la próxima sección se muestra la efectividad del método.

2.3. Experimentos

En esta sección se presentan dos experimentos que fueron realizados para verificar la eficacia del método propuesto. El objetivo del primero fue determinar si se obtiene una diferencia significativa en la calidad de la clasificación al aplicar MISMOTE, comparado con el resultado que se obtendría con el mismo clasificador entrenado sin aplicar MISMOTE. La Figura 2.4 representa esquemáticamente el diseño del experimento. El procedimiento consiste en tomar un algoritmo de clasificación estándar y generar con él dos clasificadores: uno entrenado sin MISMOTE a partir de los datos desbalanceados originales, y el otro entrenado a partir de los datos que han sido preprocesados con MISMOTE. Por *algoritmo de clasificación estándar* nos referimos aquí a un algoritmo de clasificación que no incluye ningún mecanismo específico en su proceso de entrenamiento para contrarrestar el problema de las clases desbalanceadas. Este procedimiento se repite para N conjuntos de datos y M algoritmos de clasificación estándar, según un esquema de validación que se describe en la próxima sección, y se aplican pruebas estadísticas para determinar diferencias significativas. El resultado de este experimento, que se presenta en la Sección 2.3.2, mostró que sí se obtienen mejoras significativas en varias medidas de desempeño.

El segundo experimento estuvo dirigido a comparar la calidad de la solución dada

```
Algoritmo: MISMOTE
Entrada:
  conjunto D = {X_1, ..., X_T} de todas las bolsas de entrenamiento
positivas originales
  número N de bolsas sintéticas a generar por cada bolsa positiva
original
  número k de vecinos más cercanos
Salida: conjunto de todas las bolsas sintéticas positivas D'
1. D' ← ∅
2. NumAttrs ← número de atributos
3. repetir desde i ← 1 hasta T
4.   NearestNeighbors ← calcular los k vecinos más cercanos para X_i
5.   Size_i ← calcular el tamaño promedio de los k vecinos más cercanos
6.   repetir desde j ← 1 hasta N
7.     S_ij ← ∅ inicializar una nueva bolsa sintética
8.     X_ij ← escoger aleatoriamente una bolsa de NearestNeighbors
9.     repetir desde r ← 1 hasta Size_i
10.      x_ir ← escoger aleatoriamente una instancia de X_i
11.      x_ijr ← escoger aleatoriamente una instancia de X_ij
12.      repetir desde a ← 1 hasta NumAttrs
13.        x_ira ← valor del atributo a − ésimo de x_ir
14.        x_ijra ← valor del atributo a − ésimo de x_ijr
15.        λ ← número aleatorio entre 0 y 1
16.        s_ijra ← x_ira + λ (x_ijra − x_ira)
17.      fin de repetir
18.      S_ij ← S_ij ∪ s_ijr
19.    fin de repetir
20.    D' ← D' ∪ S_ij
21.  fin de repetir
22. fin de repetir
23. retornar D'
```

Figura 2.3: Algoritmo MISMOTE.

por MISMOTE con la obtenida por otros métodos de la literatura, específicamente los algoritmos amplificadores sensibles al costo descritos en la Sección 2.1.4. La cuestión es si un clasificador estándar, entrenado tras aplicar MISMOTE a los datos de entrenamiento desbalanceados, tiene el mismo desempeño que un clasificador amplificador sensible al costo entrenado con los datos desbalanceados. La Figura 2.5 representa esquemáticamente el diseño del experimento. En este experimento se compara la calidad de la clasificación que se obtiene con dos maneras diferentes de enfrentar el problema de las clases desbalanceadas: se compara un clasificador sensible al costo entrenado a partir de los datos desbalanceados originales con un clasificador estándar entrenado con los datos rebalanceados obtenidos por MISMOTE. El resultado de este experimento, que se presenta en la Sección 2.3.3, mostró que ambas soluciones obtienen resultados comparables.

2.3.1. Elementos y métodos

Para presentar los elementos involucrados en el desarrollo de los experimentos, la Sección 2.3.1.1 hace una breve descripción de los algoritmos de clasificación usados en cada experimento, mientras que en la Sección 2.3.1.2 se exponen los conjuntos de datos seleccionados para el estudio. Además, en la Sección 2.3.1.3 se explica cómo se evaluó el desempeño de la clasificación y cómo se determinó la significación estadística de las comparaciones.

2.3.1.1. Algoritmos de clasificación

Para el primer experimento seleccionamos seis algoritmos de clasificación estándar. La selección fue hecha sobre la base de la diversidad en la forma de representación de la hipótesis de aprendizaje y el enfoque del método de aprendizaje. En el Cuadro 2.3 se nombra cada algoritmo, se especifican los parámetros con que fueron usados en el experimento y se indica brevemente la justificación de su selección. En el algoritmo MILES usamos C4.5 como clasificador base debido a que es un clasificador ampliamente usado y ha sido incluido como uno de los 10 algoritmos más importantes en minería de datos [119]. Para todos los algoritmos de el Cuadro 2.3 y el clasificador base C4.5 usamos las implementaciones disponibles en el paquete de aprendizaje automático WEKA [48]. En cada algoritmo usamos la configuración por defecto establecida en WEKA, sino se especifica lo contrario.

Para el segundo experimento, los algoritmos amplificadores sensibles al costo usados en la comparación fueron Ab2, Ab3 y Ab4, descritos en la Sección 2.1.4. Por otro lado,

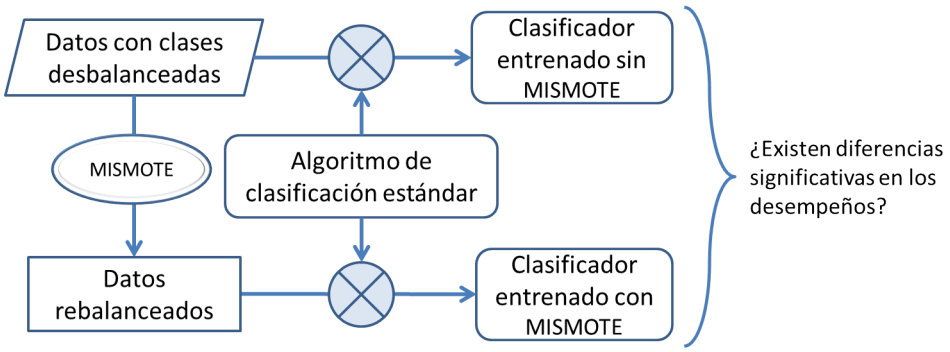


Figura 2.4: Diseño del experimento para determinar si MISMOTE introduce una mejora significativa en la calidad de la clasificación.

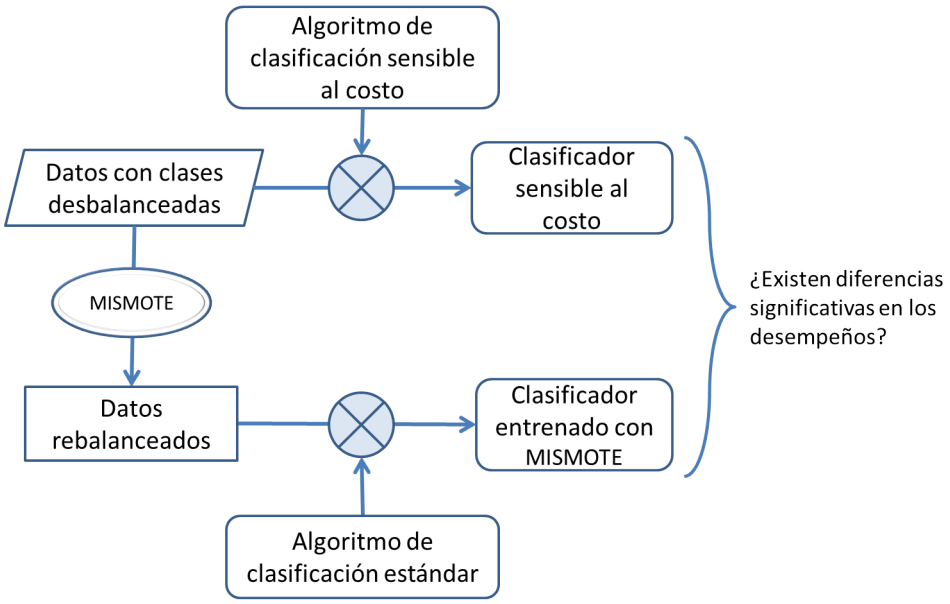


Figura 2.5: Diseño del experimento para comparar la mejora en la calidad de la clasificación introducida por MISMOTE con la de los algoritmos amplificadores sensibles al costo.

Clasificador	Parámetros	Justificación de la selección
CitationKNN[110]	Citadores: 3 Referencias: 3	Encarna el enfoque de aprendizaje basado en instancias (Sección 1.4.3.1).
MITI[11]		Encarna el enfoque de aprendizaje basado en árboles de decisión (Sección 1.4.3.2).
MILR [121]	Hipótesis multinstancia colectiva con media aritmética para el cálculo de las probabilidades posteriores.	Encarna el enfoque de aprendizaje basado en máxima verosimilitud (Sección 1.4.3.4)
MI-SVM[2]	Kernel: lineal	Encarna el enfoque de aprendizaje basado en máximo margen (Sección 1.4.3.3)
MILES[17]	Clasificador base: C4.5	Encarna el enfoque de aprendizaje multinstancia basado en envoltorio (Sección 1.4.4.2)
AdaBoost	Clasificador base: MITI	Es un algoritmo amplificador estándar, comparable con los amplificadores sensibles al costo usados en el segundo experimento.

Cuadro 2.3: Algoritmos de clasificación multinstancia seleccionados para verificar la eficacia del método MISMOTE.

para llevar a cabo la solución basada en MISMOTE seleccionamos, como clasificador estándar, el AdaBoost estándar (última fila de el Cuadro 2.3). Diversos estudios teóricos y empíricos [88, 25, 72] han demostrado que un ensamble de clasificadores hace generalmente predicciones más exactas que un solo clasificador. Fue por este motivo, para que la comparación fuera válida, que seleccionamos un algoritmo amplificador para comparar la solución basada en MISMOTE con los amplificadores sensibles al costo. Además, el AdaBoost estándar es el algoritmo amplificador que fue adaptado por los métodos amplificadores sensibles al costo en [113].

En [113] los algoritmos amplificadores sensibles al costo fueron estudiados con los métodos multinstancia MITI y MIRI como clasificadores base. Nosotros, para todos los algoritmos amplificadores involucrados en este estudio (Ab2, Ab3, Ab4 y el AdaBoost estándar), hemos seleccionado a MITI como clasificador base porque tiene similar desempeño que MIRI, pero es más eficiente.

En ambos experimentos MISMOTE fue configurado para generar tantas bolsas sintéticas como fueran necesarias para equilibrar el balance de las clases (alcanzar $IR = 1$). El parámetro k, correspondiente al número de vecinos más cercanos, fue puesto a 5.

Conjuntos de datos	Abre viatura	Nivel de instancia					Nivel de bolsa			
		# atributos	# minoritarios	# total	% minoritarios	IR	# minoritarios	# total	% minoritarios	IR
atoms	ato	10	365	1438	25,4	2,9	42	167	25,1	3,0
bonds	bon	16	603	3558	16,9	4,9	35	160	21,9	3,6
chains	cha	24	514	4630	11,1	8,0	27	152	17,8	4,6
process	pro	200	281	3154	8,9	10,2	29	142	20,4	3,9
fox	fox	230	134	781	17,2	4,8	21	121	17,4	4,8
tiger	tig	230	164	708	23,2	3,3	26	126	20,6	3,8
elephant	ele	230	150	912	16,4	5,1	25	125	20,0	4,0
wir-v1	w1	599	555	3173	17,5	4,7	21	113	18,6	4,4
wir-v2	w2	571	562	3430	16,4	5,1	21	113	18,6	4,4
wir-v3	w3	606	642	3912	16,4	5,1	21	113	18,6	4,4
wir-v4	w4	568	738	3742	19,7	4,1	24	113	21,2	3,7
wir-v5	w5	619	754	3678	20,5	3,9	24	113	21,2	3,7
wir-v6	w6	557	891	3925	22,7	3,4	24	113	21,2	3,7

Cuadro 2.4: Características de los conjuntos de datos usados en los experimentos con MISMOTE.

2.3.1.2. Conjuntos de datos

Ambos experimentos fueron ejecutados sobre 13 conjuntos de datos multinstancia desbalanceados. El Cuadro 2.4 presenta las características más importantes de estos conjuntos de datos. Los siete primeros fueron introducidos por [113] como una modificación de los empleados previamente en [34] y [10]. Los conjuntos de datos originales no eran desbalanceados, por tanto para convertirlos en conjuntos de datos desbalanceados escogieron solo una parte de las bolsas en una clase. Los últimos seis son conjuntos de datos originalmente desbalanceados tomados del problema de recomendación de páginas web índices [132] y usados en [125, 124, 106]. Este es un problema de aprendizaje complicado por tener atributos muy numerosos y dispersos. Los conjuntos de datos abarcan tres dominios de aplicación: bioinformática (atoms, bonds y chains), recuperación de imágenes por su contenido (fox, tiger y elephant) y clasificación textual (el resto).

2.3.1.3. Métodos de evaluación y comparación

La evaluación de la clasificación se llevó a cabo mediante un esquema de validación cruzada (CV, por las siglas del término inglés *cross validation*) 5×5 CV. Lo que significa que cada conjunto de datos fue dividido cinco veces (i.e., cinco repeticiones de CV) en

cinco particiones distintas en las cuales se registraron y promediaron varias medidas de desempeño. Las medidas del desempeño de la clasificación usadas fueron AUC, *precision*, *recall* y F1, las cuales fueron presentadas en la Sección 2.1.1. Para evaluar el desempeño de un clasificador en un conjunto de datos se agregaron los resultados de la clasificación (valores de la matriz de confusión) en cada una de las cinco particiones del conjunto de datos (en una sola repetición), y se calcularon las cuatro medidas del desempeño a partir de la matriz de confusión agregada. Este procedimiento se hizo para las cinco repeticiones de CV, obteniendo cinco registros de las cuatro medidas, los cuales fueron promediados para tener finalmente un estimado más preciso de las cuatro medidas del desempeño del clasificador en el conjunto de datos.

Se usaron tests estadísticos para probar la existencia de diferencias significativas entre los algoritmos comparados. Específicamente, nos apoyamos en los tests no paramétricos, de acuerdo a las recomendaciones hechas en [24, 42, 41], porque no tenemos garantías de que estén satisfechas las condiciones iniciales que aseguran la fiabilidad de los tests paramétricos.

Como en todos los casos debemos comparar pares de resultados (uno, del clasificador entrenado con MISMOTE, y el otro, del clasificador de control), usamos el test de rangos pareados con signo de Wilcoxon [116] (en lo adelante *test de Wilcoxon*) para descubrir si existen diferencias significativas en las comparaciones realizadas en ambos experimentos. El *valor p* calculado por el test de Wilcoxon representa la probabilidad de obtener un resultado al menos tan extremo como el obtenido en el experimento (valor del estadístico calculado), suponiendo que los dos clasificadores tienen desempeños similares (hipótesis nula). Un valor p muy pequeño (por ejemplo, menor que 0,05) sugiere que la hipótesis de partida es falsa, i.e., realmente existen diferencias significativas entre los métodos comparados.

2.3.2. Eficacia del MISMOTE

En esta sección se presentan y comparan los resultados de los clasificadores entrenados tras aplicar MISMOTE a los datos, y los resultados de los clasificadores entrenados con los datos sin preprocesar, i.e., con las clases desbalanceadas. Para organizar mejor la exposición de los resultados conformaremos dos grupos de clasificadores:

- Sin-M: contiene a los clasificadores entrenados directamente con los datos desbalanceados.

- Con-M: contiene a los clasificadores entrenados tras aplicar MISMOTE a los datos.

En el Apéndice de este informe se presentan los resultados detallados obtenidos en este experimento, que consisten en una tabla por cada medida del desempeño, y en cada tabla el valor de la medida de desempeño de cada clasificador, en cada conjunto de datos, en ambos grupos (Sin-M y Con-M). A continuación se analizarán los promedios por algoritmo de clasificación y por conjunto de datos.

El Cuadro 2.5 muestra el desempeño promedio de la clasificación en cada conjunto de datos, en ambos grupos (Sin-M y Con-M) y para cada medida del desempeño de la clasificación. El valor de cada entrada de la tabla es el promedio de los seis clasificadores dentro del grupo correspondiente, para el conjunto de datos y la medida de desempeño correspondientes. Al aplicar MISMOTE (grupo Con-M), aumenta el AUC en todos los conjuntos de datos menos en uno (w2), disminuye *precision* en ocho conjuntos de datos y aumenta en cinco (marcados con el símbolo ↑ en la tabla), y aumentan *recall* y F1 en todos los conjuntos de datos. Este resultado sugiere que un clasificador entrenado tras aplicar MISMOTE a un conjunto de datos con clases desbalanceadas, de forma general, va a mejorar su habilidad para identificar ambas clases, de acuerdo al incremento observado en el AUC. Por otro lado, el costo de prestar más atención a la clase minoritaria es que frecuentemente aumenta el número de falsos positivos, disminuyendo de esta manera *precision*. Sin embargo, la ventaja es que permite identificar correctamente un mayor número de ejemplos positivos (aumentando *recall*) lo cual, como los ejemplos positivos son escasos, representa una gran recompensa. El aumento de F1 en el grupo que aplica MISMOTE reafirma la aserción de que en este grupo el balance entre *precision* y *recall* es mejor.

El Cuadro 2.6 muestra para cada clasificador el desempeño promedio de la clasificación en ambos grupos (Sin-M y Con-M), para cada medida del desempeño de la clasificación. El valor de cada entrada de la tabla es el promedio de los 13 conjuntos de datos para el clasificador, el grupo y la medida de desempeño correspondientes. La clasificación del conjunto de datos promedio muestra una clara mejora respecto al AUC, el *recall* y F1 con cada clasificador. Solo CitationKNN es el único caso en el cual AUC no mejora al aplicar MISMOTE. Con respecto a *precision*, los resultados no siempre mejoran con MISMOTE. Dos clasificadores, MILR y MI-SVM, pierden *precision* en el Cuadro 2.6, lo cual es consistente con las disminuciones de *precision* observadas en el Cuadro 2.5. Como se comentó anteriormente, en un intento por reducir los falsos negativos e incrementar los verdaderos positivos, lo cual incrementaría *recall*, algunos clasificadores cometen un alto número de falsos positivos, decrementando de hecho su *precision*. Sin embargo, F1 mejora en todos los clasificadores cuando se aplica MISMOTE, lo cual indica que la ganancia en *recall* es superior a la pérdida en *precision*. Este argumento es corroborado

Conj.	AUC		*Precision*		*Recall*		F1	
datos	Sin-M	Con-M	Sin-M	Con-M	Sin-M	Con-M	Sin-M	Con-M
ato	0,665	0,689	0,466	0,421	0,222	0,459	0,338	0,466
bon	0,657	0,668	0,523	0,427	0,223	0,378	0,363	0,441
cha	0,650	0,672	0,437	0,385	0,182	0,403	0,275	0,421
pro	0,725	0,777	0,510	0,736↑	0,453	0,754	0,530	0,756
fox	0,499	0,507	0,177	0,198↑	0,080	0,374	0,114	0,274
tig	0,632	0,688	0,473	0,502↑	0,303	0,658	0,403	0,584
ele	0,583	0,636	0,503	0,447	0,187	0,570	0,331	0,518
w1	0,630	0,637	0,265	0,252	0,350	0,478	0,323	0,346
w2	0,740	0,719↓	0,392	0,363	0,543	0,595	0,478	0,479
w3	0,635	0,650	0,274	0,266	0,318	0,459	0,311	0,347
w4	0,596	0,606	0,441	0,375	0,213	0,460	0,275	0,388
w5	0,555	0,567	0,262	0,293↑	0,135	0,399	0,178	0,313
w6	0,614	0,636	0,397	0,417↑	0,275	0,535	0,339	0,462

Cuadro 2.5: Valores promedios de AUC, *precision*, *recall* y F1 obtenidos para cada conjunto de datos por un grupo de clasificadores multinstancia estándares entrenados con MISMOTE y otro grupo con los mismos tipos de clasificadores pero entrenados en los datos con clases desbalanceadas.

por los tests estadísticos que presentaremos seguidamente, indicando que mientras la disminución de *precision* no es significativa, el aumento de *recall* sí lo es.

A continuación presentaremos los resultados del test de Wilcoxon para determinar la significación estadística de las diferencias en el desempeño de los clasificadores que aplican y que no aplican MISMOTE. En esta configuración, $R+$ es la suma de los ranking para los conjuntos de datos en los cuales el método que aplica MISMOTE es mejor que el otro, y $R-$ es la suma de los ranking para los conjuntos de datos en los cuales el método que aplica MISMOTE es peor que el otro. Una diferencia grande entre $R+$ y $R-$ sugiere el rechazo de la hipótesis nula. La hipótesis nula establece que no existe diferencia en el desempeño de un clasificador que aplique MISMOTE y otro que no lo aplique. Por tanto, para que el test confirme la ventaja de aplicar MISMOTE es necesario que la hipótesis nula sea rechazada y que $R+$ sea mayor que $R-$.

Las tablas desde la 2.7 a la 2.9 presentan la salida del test de Wilcoxon para AUC, *precision* y *recall*, respectivamente. La última fila de cada tabla considera el desempeño promedio sobre los seis clasificadores, representando al clasificador promedio. Con respecto al AUC, el Cuadro 2.7 muestra que para todos los clasificadores, con excepción de CitationKNN, el valor p es muy pequeño, y $R+$ mayor que $R-$ indica que debe rechazarse la hipótesis nula a favor del grupo de clasificadores que aplican MISMOTE.

Clasificador	AUC		*Precision*		*Recall*		F1	
	Sin-M	Con-M	Sin-M	Con-M	Sin-M	Con-M	Sin-M	Con-M
CitationKNN	0,644	0,641 ↓	0,324	0,338	0,190	0,635	0,240	0,441
MITI	0,617	0,652	0,475	0,498	0,348	0,422	0,421	0,494
MILR	0,729	0,751	0,507	0,438 ↓	0,356	0,558	0,284	0,367
MI-SVM	0,570	0,597	0,387	0,363 ↓	0,225	0,371	0,251	0,394
MILES	0,592	0,616	0,278	0,316	0,229	0,522	0,402	0,457
AdaBoost	0,728	0,747	0,546	0,529	0,412	0,471	0,470	0,498

Cuadro 2.6: Valores de AUC, *precision*, *recall* y F1 obtenidos como promedios sobre 13 conjuntos de datos por cada algoritmo de clasificación multinstancia, entrenados por un lado con MISMOTE, y por otro lado, en los datos con clases desbalanceadas.

Clasificador	R+	R-	valor p
CitationKNN	35	43	0,790
MITI	78	13	0,021
MILR	74	17	0,046
MI-SVM	46	9	0,064
MILES	71	20	0,077
AdaBoost	83	8	0,006
Promedio	77,5	13,5	0,023

Table 2.7: Test de Wilcoxon para estimar la significación estadística de la diferencia en AUC que se obtiene entre un clasificador multinstancia estándar entrenado con MISMOTE y otro entrenado en los datos con clases desbalanceadas.

El Cuadro 2.8, referente a *precision*, muestra que tres clasificadores tienen un valor p considerablemente pequeño. Uno de ellos, MITI, a favor de MISMOTE, los otros dos, MILR y MI-SVM, en contra. Para otros cuatro clasificadores, incluyendo al clasificador promedio, no se rechaza la hipótesis nula, lo cual solo permite afirmar que en general no hay una diferencia significativa en *precision* al aplicar MISMOTE. Por otro lado, considerando el *recall*, el Cuadro 2.9 muestra que la hipótesis nula se rechaza para todos los clasificadores. Además, el valor $p = 0,000$ del clasificador promedio permite asegurar que aplicar MISMOTE consistentemente conduce a un aumento del *recall* en la clasificación.

Resumiendo, la mayoría de los clasificadores que aplican MISMOTE mejoran el AUC, generalmente no presentan una diferencia significativa de *precision* y con seguridad mejoran el *recall*.

Clasificador	R+	R-	valor p
CitationKNN	52	39	0,685
MITI	71	20	0,080
MILR	13	65	0,042
MI-SVM	10	45	0,084
MILES	34	21	0,557
AdaBoost	33	58	0,414
Promedio	22	69	0,110

Table 2.8: Test de Wilcoxon para estimar la significación estadística de la diferencia en *precision* que se obtiene entre un clasificador multinstancia estándar entrenado con MISMOTE y otro entrenado en los datos con clases desbalanceadas.

Clasificador	R+	R-	valor p
CitationKNN	91	0	0,000
MITI	67	11	0,025
MILR	88	3	0,001
MI-SVM	55	0	0,002
MILES	54	1	0,004
AdaBoost	76,5	1,5	0,001
Promedio	91	0	0,000

Table 2.9: Test de Wilcoxon para estimar la significación estadística de la diferencia en *recall* que se obtiene entre un clasificador multinstancia estándar entrenado con MISMOTE y otro entrenado en los datos con clases desbalanceadas.

2.3.3. Comparación con algoritmos amplificadores sensibles al costo

En esta sección se presentan y comparan los resultados del clasificador AdaBoost estándar entrenado tras aplicar MISMOTE a los datos, y los resultados de los clasificadores amplificadores sensibles al costo Ab2, Ab3 y Ab4, entrenados con los datos sin preprocesar, i.e., con las clases desbalanceadas. El método AdaBoost estándar entrenado tras aplicar MISMOTE fue usado en el experimento de la sección anterior, donde lo llamábamos simplemente «AdaBoost», especificando que pertenecía al grupo «Con-M». En esta sección, lo rebautizaremos como MISM-AB para indicar que ambos algoritmos, MISMOTE y AdaBoost estándar, se aplican dentro del método. Los resultados usados en esta sección para comparar el MISM-AB son los mismos presentados en la sección anterior para AdaBoost Con-M. Por otra parte, los resultados de los algoritmos Ab2, Ab3 y Ab4 presentados en esta sección fueron gentilmente proporcionados por los autores de [113].

El Cuadro 2.10 muestra el AUC de los métodos comparados para cada conjunto de datos, así como el AUC promedio de cada método en la última fila. Los valores remarcados en negrita indican el método vencedor para cada conjunto de datos. Decimos que un método obtiene una victoria en un conjunto de datos si éste obtiene el mejor valor entre todos los métodos comparados para ese conjunto de datos. Los números de la tabla y los valores promedios indican que los cuatro métodos tienen resultados muy parejos respecto al AUC. El Cuadro 2.11 presenta los resultados de los test de Wilcoxon correspondientes a la comparación de MISM-AB con cada uno de los amplificadores sensibles al costo. En los dos primeros casos (MISM-AB contra Ab2 y Ab3) no es posible rechazar la hipótesis nula, por lo cual se infiere que no hay diferencias significativas entre el AUC obtenido con MISM-AB y el obtenido con Ab2, ni tampoco entre el obtenido con MISM-AB y el obtenido con Ab3. Sin embargo, en el último caso (MISM-AB contra Ab4) el valor p está muy cerca de 0,1. De lo cual se puede inferir que existe una probabilidad del 90 % de obtener mejor AUC con MISM-AB que con Ab4.

El Cuadro 2.12 muestra la *precision* de los métodos comparados para cada conjunto de datos y la *precision* promedio de cada método en la última fila. El método que obtiene más victorias y mejor promedio es MISM-AB, seguido de cerca por Ab2. El Cuadro 2.13 presenta los resultados de los test de Wilcoxon correspondientes. En los tres casos que se comparan, el valor p no deja lugar a dudas de que la variante del MISM-AB alcanza significativamente más *precision* que los métodos sensibles al costo.

El Cuadro 2.14 muestra el *recall* de los métodos comparados para cada conjunto de datos y el *recall* promedio de cada método en la última fila. El *recall* promedio de MISM-AB es muy similar al de Ab2, pero está por debajo del de Ab3 y Ab4. El Cuadro 2.15

Conjunto de dato	Ab2	Ab3	Ab4	MISM-AB
ato	**0,796**	0,787	0,776	0,774
bon	0,813	0,812	0,781	**0,837**
cha	0,778	0,797	0,776	**0,858**
pro	0,942	0,944	0,938	**0,961**
fox	0,600	**0,658**	0,568	0,639
tig	0,821	0,819	0,828	**0,842**
ele	**0,820**	0,797	0,748	0,797
w1	0,702	**0,740**	0,688	0,709
w2	**0,828**	0,808	0,798	0,819
w3	0,628	0,656	**0,658**	0,643
w4	0,607	**0,635**	0,600	0,604
w5	0,603	0,612	**0,653**	0,575
w6	0,674	**0,678**	0,672	0,656
Promedio	0,739	**0,749**	0,730	0,747

Table 2.10: Valores de AUC obtenidos para cada conjunto de datos, y en promedio, por los clasificadores amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas y el AdaBoost estándar entrenado con MISMOTE (MISM-AB).

Método	R+	R-	valor p
MISM-AB vs Ab2	54	37	0,588
MISM-AB vs Ab3	32	46	0,606
MISM-AB vs Ab4	69	22	0,106

Cuadro 2.11: Test de Wilcoxon para estimar la significación estadística de la diferencia en AUC que se obtiene entre el AdaBoost estándar entrenado con MISMOTE (MISM-AB) y los amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas.

Conjunto de dato	Ab2	Ab3	Ab4	MISM-AB
ato	0,582	0,496	0,378	**0,741**
bon	0,620	0,520	0,419	**0,720**
cha	0,533	0,499	0,401	**0,668**
pro	**0,867**	0,737	0,517	0,837
fox	0,320	0,277	0,201	**0,376**
tig	**0,733**	0,577	0,400	0,697
ele	0,677	0,484	0,319	**0,720**
w1	0,282	0,269	0,234	**0,319**
w2	0,392	0,309	0,272	**0,503**
w3	0,241	0,222	0,220	**0,284**
w4	**0,338**	0,337	0,243	0,321
w5	0,221	0,277	**0,279**	0,220
w6	**0,494**	0,368	0,309	0,474
Promedio	0,485	0,413	0,323	**0,529**

Table 2.12: Valores de *precision* obtenidos para cada conjunto de datos, y en promedio, por los clasificadores amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas y el AdaBoost estándar entrenado con MISMOTE (MISM-AB).

Método	R+	R-	valor p
MISM-AB vs Ab2	76	15	0,032
MISM-AB vs Ab3	87	4	0,002
MISM-AB vs Ab4	90	1	0,000

Cuadro 2.13: Test de Wilcoxon para estimar la significación estadística de la diferencia en *precision* que se obtiene entre el AdaBoost estándar entrenado con MISMOTE (MISM-AB) y los amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas.

Conjunto de dato	Ab2	Ab3	Ab4	MISM-AB
ato	0,467	0,667	**0,881**	0,448
bon	0,571	0,766	**0,789**	0,571
cha	0,311	0,652	**0,719**	0,444
pro	0,710	0,841	**0,945**	0,779
fox	0,095	0,524	**0,600**	0,200
tig	0,469	0,677	**0,808**	0,562
ele	0,328	0,632	**0,760**	0,360
w1	0,676	**0,895**	0,876	0,638
w2	0,876	0,886	**0,933**	0,810
w3	0,638	0,838	**0,914**	0,638
w4	0,183	0,508	**0,608**	0,183
w5	0,150	0,475	**0,758**	0,133
w6	0,392	0,567	**0,667**	0,350
Promedio	0,451	0,687	**0,789**	0,471

Table 2.14: Valores de *recall* obtenidos para cada conjunto de datos, y en promedio, por los clasificadores amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas y el AdaBoost estándar entrenado con MISMOTE (MISM-AB).

Método	R+	R-	valor p
MISM-AB vs Ab2	37	18	0,375
MISM-AB vs Ab3	0	91	0,000
MISM-AB vs Ab4	0	91	0,000

Table 2.15: Test de Wilcoxon para estimar la significación estadística de la diferencia en *recall* que se obtiene entre el AdaBoost estándar entrenado con MISMOTE (MISM-AB) y los amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas.

presenta los resultados de los test de Wilcoxon correspondientes, los cuales confirman que entre MISM-AB y Ab2 no existen diferencias significativas de *recall*, y también que Ab3 y Ab4 alcanzan valores de *recall* significativamente superiores a los de MISM-AB.

El Cuadro 2.16 muestra el F1 de los métodos comparados para cada conjunto de datos y el F1 promedio de cada método en la última fila. Casi todas las victorias se reparten entre los métodos MISM-AB y Ab3. El Cuadro 2.17 presenta los resultados de los test de Wilcoxon correspondientes. Según el test, MISM-AB es significativamente mejor que Ab2. Este resultado está en correspondencia con lo observado en los dos test anteriores: ambos métodos no son significativamente diferentes en *recall*, pero MISM-AB es significativamente mejor que Ab2 en *precision*. En tanto, no se aprecian diferencias

Conjunto de dato	Ab2	Ab3	Ab4	MISM-AB
ato	0,518	**0,569**	0,529	0,558
bon	0,595	0,619	0,547	**0,637**
cha	0,393	**0,565**	0,515	0,534
pro	0,781	0,786	0,668	**0,807**
fox	0,147	**0,362**	0,301	0,261
tig	0,572	**0,623**	0,535	0,622
ele	0,442	**0,548**	0,449	0,480
w1	0,398	0,414	0,369	**0,426**
w2	0,541	0,458	0,421	**0,621**
w3	0,350	0,351	0,355	**0,393**
w4	0,238	**0,405**	0,348	0,233
w5	0,179	0,350	**0,408**	0,166
w6	0,437	**0,447**	0,422	0,403
Promedio	0,430	**0,500**	0,451	0,472

Table 2.16: Valores de F1 obtenidos para cada conjunto de datos, y en promedio, por los clasificadores amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas y el AdaBoost estándar entrenado con MISMOTE (MISM-AB).

significativas entre el valor de F1 alcanzado por MISM-AB y los alcanzados por Ab3 y Ab4. Remitiéndonos nuevamente a lo observado en los dos test anteriores, esta prueba confirma que lo que es una ventaja significativa de Ab3 y Ab4 sobre MISM-AB respecto a *recall*, representa una pérdida significativa respecto a la *precision* de Ab3 y Ab4 frente a MISM-AB, de forma tal que sus valores de F1 no se diferencia significativamente.

Considerando que en la sección previa se mostró que aplicar MISM-AB mejora significativamente el *recall* sin que ocurra una disminución significativa de *precision*, y considerando lo que se ha presentado en esta sección respecto a *recall, precision* y F1, podemos sacar la siguiente conclusión. Realmente Ab3 y Ab4 alcanzan más *recall* que MISM-AB, pero lo hacen a expensas de *precision*. Ab3 y Ab4 pierden el equilibrio entre estas dos medidas de desempeño, maximizando *recall* mientras descuidan la *precision*. Nuestro método, en cambio, obtiene un balance adecuado entre estas dos medidas de desempeño.

En resumen, el análisis hecho muestra que ambas soluciones, la basada en el preprocesamiento con MISMOTE y los algoritmos sensibles al costo estudiados, tienen desempeños comparables respecto al AUC. Mientras que la solución basada en MISMOTE tiene mejor *precision* que Ab2 y un mejor equilibrio *recall-precision* que Ab3 y Ab4.

Método	R+	R-	valor p
MISM-AB vs Ab2	83	8	0,006
MISM-AB vs Ab3	30	61	0,305
MISM-AB vs Ab4	60,5	30,5	0,313

Table 2.17: Test de Wilcoxon para estimar la significación estadística de la diferencia en F1 que se obtiene entre el AdaBoost estándar entrenado con MISMOTE (MISM-AB) y los amplificadores sensibles al costo, Ab2, Ab3 y Ab4, entrenados en los datos con clases desbalanceadas.

2.4. Conclusiones del capítulo

El aprendizaje multinstancia, como el aprendizaje monoinstancia, es afectado por el problema de las clases desbalanceadas. Para enfrentar este problema proponemos el método MISMOTE, el cual es un algoritmo de preprocesamiento de los datos de entrenamiento para sobremuestrear la clase minoritaria. La estrategia del método consiste en interpolar instancias muestreadas de dos bolsas adyacentes para generar bolsas sintéticas en la imediación de ambas bolsas.

El análisis estadístico de los resultados experimentales mostró que, en un problema de clasificación multinstancia de dos clases desbalanceadas, aplicar MISMOTE antes del entrenamiento del clasificador generalmente aumenta la exactitud de la clasificación de ambas clases, y particularmente de la clase pequeña, permitiendo identificar significativamente más ejemplos positivos sin un aumento apreciable de falsos positivos. En los experimentos, se usó el MISMOTE para obtener conjuntos de entrenamiento con clases de igual tamaño ($IR = 1$) a partir de los datos con clases desbalanceadas. Sin embargo, esta no es necesariamente la proporción óptima en el tamaño de las clases, la cual dependería en última instancia de las características del conjunto de datos. La búsqueda de la proporción óptima en el tamaño de las clases, aun cuando implica una carga computacional adicional, podría mejorar todavía más el desempeño de la clasificación. Sin embargo, aun cuando la proporción en el tamaño de las clases no sea optimizada, el estudio ha mostrado que con solo igualar el tamaño de ambas clases es suficiente para obtener mejoras significativas del desempeño de la clasificación en la mayoría de los casos.

El problema de las clases desbalanceadas en el aprendizaje multinstancia ha recibido poca atención hasta la fecha. La implementación de algoritmos amplificadores sensibles al costo con clasificadores de base multinstancia es el tipo de solución más importante conocido hasta ahora para el problema. La evidencia experimental mostró que MISMOTE es competitivo con esas soluciones. Sin embargo, el uso de métodos basados en muestreo, co-

mo MISMOTE, pueden ser preferibles por diversas razones. En las metodologías basadas en ensambles el tiempo de aprendizaje puede ser alto y el modelo de salida puede ser difícil de entender por el usuario final. Además, en los enfoques amplificadores sensibles al costo, la necesidad de definir una matriz de costo óptima impone duras restricciones a los usuarios. En cambio, como las soluciones basadas en muestreo son externas al clasificador que se utilice, el usuario tiene una gran libertad para seleccionar el clasificador que más le convenga para optimizar aspectos tales como el tiempo de aprendizaje o la interpretabilidad del modelo. Además, este enfoque permite la reutilización eficiente de la solución de remuestro que se aplique a un conjunto de entrenamiento determinado.

3 MIRocchio: Algoritmo de clasificación multinstancia basado en el clasificador de Rocchio

El clasificador de Rocchio es un algoritmo de aprendizaje tradicional del campo de la recuperación y clasificación de documentos textuales. Es apreciado por su bajo costo computacional, lo cual lo hace indicado para aplicaciones interactivas en-línea. Aunque las tareas de recuperación y clasificación de textos pueden enfocarse naturalmente como problemas multinstancia, se han desarrollado muy pocos algoritmos de aprendizaje específicos para este dominio de aplicación. Por tanto, es muy atractivo adaptar el algoritmo de Rocchio para trabajar con el enfoque multinstancia.

El objetivo propuesto en este trabajo fue diseñar un algoritmo de aprendizaje multinstancia que permita aplicar eficazmente el clasificador de Rocchio para solucionar problemas de recuperación y clasificación de documentos textuales desde el enfoque multinstancia sin perder el atractivo fundamental de este algoritmo que es su eficiencia computacional. Como resultado se concibió una nueva hipótesis multinstancia apropiada para la recuperación y clasificación de textos y se obtuvo el algoritmo de clasificación multinstancia MIRocchio que encapsula a un clasificador de Rocchio basándose en la referida hipótesis multinstancia. Para verificar el cumplimiento del objetivo trazado pusimos a prueba el algoritmo MIRocchio en una aplicación de minería web, la cual ha sido enfocada típicamente desde el campo de la recuperación de documentos textuales, obteniéndose resultados satisfactorios.

3.1. Diseño del clasificador MIRocchio

El principio de diseño adoptado para el clasificador MIRocchio es el de los métodos de envoltorio descritos en la Sección 1.4.4. En la etapa de entrenamiento el envoltorio transforma los datos multinstancia en monoinstancia, para construir un modelo de clasificación mediante el clasificador de Rocchio. En la etapa de clasificación el envoltorio aplica la transformación inversa para convertir las predicciones hechas por el clasifi-

cador de Rocchio a las instancias de la bolsa en una predicción de la etiqueta de clase de la bolsa. Las transformaciones que aplica el envoltorio en uno y otro sentido están determinadas por la hipótesis multinstancia que adopta el clasificador MIRocchio.

El estudio experimental del prototipo inicial del algoritmo MIRocchio usando la hipótesis multinstancia estándar arrojó resultados discretos, lo cual motivó que concibiéramos una nueva hipótesis multinstancia orientada al campo de la recuperación de documentos textuales que mostró resultados superiores.

A continuación se describe detalladamente el diseño del algoritmo de clasificación MIRocchio. Se comienza describiendo el clasificador monoinstancia de Rocchio que es usado por MIRocchio como clasificador base. Seguidamente, se explica la hipótesis multinstancia que es introducida como novedad en esta tesis, luego, el envoltorio que implementa a la hipótesis y el cálculo del umbral que la caracteriza. Finalmente, se sumariza el funcionamiento del algoritmo y se comenta sobre el cálculo de los parámetros.

3.1.1. Clasificador de Rocchio

El algoritmo de Rocchio [87] fue diseñado originalmente para recuperación interactiva de información basada en la refinación de términos de búsqueda usando retroalimentación de los usuarios, los cuales indican al sistema si el resultado de la recuperación fue o no relevante. Posteriormente el algoritmo fue adaptado para categorización textual [52] y, dada su elevada eficiencia tanto en entrenamiento como en su etapa operativa, ha sido ampliamente usado en este dominio de aplicación.

Dado un conjunto de entrenamiento el clasificador de Rocchio construye un prototipo o perfil para cada clase. El perfil de clase es un vector que sintetiza la descripción que se tiene de los ejemplos de entrenamiento de la clase. Cada componente del vector corresponde a un atributo en el espacio de instancia. En las aplicaciones textuales los atributos usualmente representan palabras o términos textuales de los documentos. El componente i-ésimo de un vector que representa a un documento dado d equivale al peso que el atributo i-ésimo del conjunto de entrenamiento asume en d (usualmente evaluado como $TF - IDF$ [90], siglas del inglés *Term Frequency - Inverse Document Frequency*). Dado un nuevo ejemplo x, el clasificador de Rocchio lo asigna a la clase cuyo perfil tenga más semejanza con x.

Específicamente, para un problema de clasificación de dos clases (con etiquetas $\mathcal{Y} = \{+, -\}$) se tiene un perfil de clase positivo ψ^+ y otro negativo ψ^-. Empleando la fórmula de Rocchio como en [68], el atributo i-ésimo en los perfiles de clase positivo y negativo se calcula mediante las Fórmulas (3.1) y (3.2) respectivamente.

$$\psi_i^+ = \text{máx}\left\{0, \frac{1}{|R^+|}\sum_{x\in R^+} x_i - \frac{\rho^+}{|R^-|}\sum_{x\in R^-} x_i\right\} \tag{3.1}$$

$$\psi_i^- = \text{máx}\left\{0, \frac{1}{|R^-|}\sum_{x\in R^-} x_i - \frac{\rho^-}{|R^+|}\sum_{x\in R^+} x_i\right\} \tag{3.2}$$

R^+ es el conjunto de instancias positivas, R^- el conjunto de instancias negativas y x_i el valor del atributo i-ésimo en la instancia x. El algoritmo de Rocchio, de forma implícita, aplica una selección de atributos, y los parámetros ρ^+ y ρ^- controlan la envergadura de esta selección [68]. Mientras mayor es el valor del parámetro, más intensa es la selección de atributos.

De acuerdo a la fórmula de Rocchio, el perfil de clase trata de capturar la importancia que cada término textual tiene, específicamente, para los documentos relevantes de la clase, más allá de la que pueda tener para los documentos no relevantes. Efectivamente, en las Fórmulas (3.1) y (3.2), se calcula para cada atributo el valor medio en las instancias positivas (lo cual representa la importancia del término para los documentos relevantes) y se le sustrae el valor medio del atributo en las instancias negativas (representando la importancia del término para los documentos irrelevantes). Si la importancia de un término textual para los documentos relevantes es mayor que la importancia para los documentos irrelevantes, esto se refleja en el perfil con un valor más elevado en el atributo que representa a dicho término. Si por el contrario, la importancia de un término para los documentos relevantes es menor o igual que la importancia para los documentos irrelevantes, entonces el término no es específico de la clase, y por tanto no debe ser incluido en el perfil: el valor del atributo se pone a cero, lo que en la práctica constituye una selección de atributos.

A partir de los perfiles de clase ψ^c, $c \in \mathcal{Y}$, la etiqueta de clase de una nueva instancia x se obtiene mediante la siguiente expresión:

$$\mathsf{g}(x) = \arg\underset{c\in\mathcal{Y}}{\text{máx}} S_{\cos}(x, \psi^c) \tag{3.3}$$

donde $S_{\cos}(x, z)$ representa la semejanza coseno entre los vectores x y z, definida como

$$S_{\cos}(x, z) = \frac{x \cdot z}{|x|\,|z|} \tag{3.4}$$

donde $|x|$ es la norma vectorial de x.

El algoritmo de Rocchio permite clasificar conjuntos de datos monoinstancia. Esta tesis propone la adaptación del algoritmo de Rocchio al enfoque multinstancia mediante el uso de un algoritmo envoltorio que implementa la siguiente hipótesis multinstancia.

3.1.2. Hipótesis multinstancia basada en proporción de instancias positivas

Según la hipótesis multinstancia estándar, una bolsa es positiva si contiene al menos una instancia positiva, en otro caso es negativa. Si aplicamos esta hipótesis al problema de la clasificación textual donde una bolsa representa un documento textual (e.g. artículos, libros, páginas webs) y una instancia representa un fragmento del documento (e.g. sección, párrafo, segmento), podemos interpretar que un documento será relevante si contiene al menos un fragmento relevante. Aunque esta hipótesis es en principio razonable puede no ser aplicable a un número importante de casos pues empíricamente mostró resultados limitados.

Es posible que en algunos documentos no sea suficiente la existencia de un solo fragmento relevante, y se requiera más de un fragmento relevante para que todo el documento sea relevante. Esta idea nos remite a la hipótesis basada en umbral propuesta en [115], la cual establece que una bolsa es positiva si contiene al menos cierto número de instancias positivas. Sin embargo, el número de fragmentos positivos necesarios para que un documento sea positivo puede ser muy variable de un documento a otro.

Por este motivo proponemos una nueva hipótesis que es una variación de la hipótesis basada en umbral, y que asume que el número de instancias positivas requerido para que la bolsa sea positiva es proporcional al tamaño de la bolsa. El razonamiento detrás de esta propuesta es que un documento grande puede tener un número más o menos grande de fragmentos relevantes para determinado concepto, mientras que un documento pequeño puede tener un número mucho menor de fragmentos relevantes y, sin embargo, suficiente para clasificar el documento como relevante para ese concepto. De esta forma, aunque el número de fragmentos relevantes difiera entre documentos de distintos tamaños, es más probable que la proporción de fragmentos relevantes que hacen relevante al documento se mantenga aproximadamente constante.

A esta nueva hipótesis multinstancia la llamamos *hipótesis basada en proporción*, la cual establece que una bolsa es positiva si contiene al menos cierta proporción (relativa al tamaño de la bolsa) de instancias de cada concepto relevante. En general, sea $\mathsf{C} \subset C$ el conjunto de conceptos subyacentes a la clasificación de las bolsas de un problema de aprendizaje multinstancia y Δ: $\mathbb{N}^{\mathcal{X}} \times C \to \mathbb{N}$ una función que cuenta el número de instancias de un concepto dado en una bolsa. Se define una función de mapeo f_{Pr} bajo

la hipótesis basada en proporción como

$$f_{Pr}(X) \Leftrightarrow \forall c_i \in \mathsf{C} : \frac{\Delta(X, c_i)}{|X|} \geq t_i \tag{3.5}$$

donde t_i es el umbral inferior de la proporción del concepto c_i y $|X|$ representa el número de instancias en la bolsa X.

La Fórmula (3.5) abarca el caso general en que las instancias deban pertenecer a varios conceptos relevantes (no negativos) para hacer positiva la bolsa. Por ejemplo, para que un documento sea clasificado en la categoría «termodinámica» podría requerirse que el documento incluya fragmentos de los conceptos «calor» y «mecánica». En el caso en que las instancias solo sean asignadas a las mismas clases positiva y negativa que las bolsas, i.e., coincide el concepto positivo de instancias y bolsas, entonces el conjunto de conceptos subyacentes es $\mathsf{C} = \{+\}$, y la función de mapeo puede simplificarse como

$$f_{Pr}(X) \Leftrightarrow \pi(X) \geq t \tag{3.6}$$

donde $\pi(X) = \frac{\Delta(X,+)}{|X|}$ es la proporción de instancias positivas en la bolsa X, y t es el umbral a partir del cual el concepto es positivo, y que llamaremos en lo adelante *umbral de proporción*.

A continuación se describe el algoritmo envoltorio que implementa a esta hipótesis multinstancia.

3.1.3. Algoritmo envoltorio

La función del envoltorio es servir de interfaz entre los datos multinstancia del problema de clasificación y el clasificador monoinstancia subyacente. En la etapa de entrenamiento el envoltorio transforma los datos multinstancia en monoinstancia para que puedan ser usados por el clasificador monoinstancia en la construcción del modelo de clasificación. Nuestro envoltorio es de tipo *basado en instancia*, lo que significa que el nuevo espacio monoinstancia estará formado por el conjunto de todas las instancias en todas las bolsas de entrenamiento, y que el envoltorio, aplicando un método de proposicionalización, deberá determinar las etiquetas de clase de las instancias para poder entrenar el clasificador monoinstancia con ellas.

Para determinar las etiquetas de clase de las instancias el envoltorio debería basarse en la hipótesis multinstancia de proporción, pero la única información con que cuenta es la etiqueta de clase de la bolsa, con la cual solo puede inferir, cuando la etiqueta es positiva, que la proporción de instancias positivas de la bolsa ha superado determinado

umbral y, cuando la etiqueta es negativa, que no ha superado el umbral. Umbral que, por demás, es desconocido. A diferencia de la hipótesis multinstancia estándar, donde una bolsa negativa implica que todas sus instancias son negativas, según la hipótesis multinstancia de proporción, en una bolsa negativa pueden existir instancias positivas siempre que su proporción no exceda el umbral.

Teniendo en cuenta que con la información disponible no podemos saber cuáles de las instancias que están dentro de la bolsa son positivas, acudimos a la hipótesis multinstancia estándar para obtener una imputación inicial. Siguiendo esta heurística, toda instancia que sea encontrada en alguna bolsa negativa es asignada a la clase negativa, el resto se asigna a la clase positiva. Después de este proceso de proposicionalización, cada instancia tiene una etiqueta de clase asignada, y los datos, originalmente multinstancia, han sido transformados de forma que pueden ser usados por el algoritmo de aprendizaje monoinstancia para construir el clasificador de Rocchio.

El próximo paso del algoritmo envoltorio es reconsiderar las etiquetas de clase de las instancias a partir de la nueva información obtenida por el clasificador de Rocchio, la cual ha sido condensada en los perfiles de clase (i.e. el peso de los términos relevantes para cada clase). Para cada instancia se calcula su semejanza con el perfil de cada clase y se asigna a la clase cuyo perfil sea más semejante. En concordancia con la hipótesis multinstancia de proporción, este paso permite que muchas instancias que fueron inicialmente imputadas a la clase negativa siguiendo la hipótesis multinstancia estándar, sean reasignadas a la clase positiva si en realidad se asemejan más al perfil positivo. Aunque es menos frecuente (debido a la asimetría de la hipótesis multinstancia estándar), también es posible que algunas instancias que fueron inicialmente imputadas a la clase positiva, sean reasignadas a la clase negativa.

Una vez que han sido reconsideradas las etiquetas de clase de las instancias, el envoltorio calcula la proporción de instancias positivas de cada bolsa, y con esta información determina el umbral de proporción a partir del cual se clasifica un bolsa en la clase positiva. En la próxima sección se explica detalladamente el procedimiento del cálculo del umbral.

En la etapa de clasificación se lleva a cabo el proceso contrario. El algoritmo envoltorio usa el clasificador monoinstancia de Rocchio construido durante el entrenamiento para asignar una etiqueta de clase a cada instancia dentro de la nueva bolsa. Luego, basándose en la hipótesis multinstancia de proporción, el envoltorio determina la etiqueta de clase de la bolsa a partir de la etiqueta de sus instancias.

3.1.4. Aprendizaje del umbral de proporción de instancias positivas

De forma general, el umbral t es desconocido para un conjunto de datos determinado, y resulta necesario determinar su valor durante la etapa de aprendizaje.

Sean μ_p y μ_n los valores medios de $\pi(X)$ sobre todas las bolsas de entrenamiento positivas y negativas, respectivamente. Sean V_p y V_n las varianzas de $\pi(X)$ sobre todas las bolsas de entrenamiento positivas y negativas, respectivamente. Se desea que el umbral t separe las bolsas positivas y negativas adecuadamente. Dado que las bolsas positivas y negativas tienen distintos niveles de dispersión debemos colocar el umbral en un punto medio entre μ_p y μ_n que tenga en cuenta la varianza muestral de cada clase. Una posibilidad es utilizar la Fórmula (3.7), donde el umbral t_l es asignado al promedio de las medias μ_p y μ_n ponderadas por las varianzas V_n y V_p respectivamente.

$$t_l = \frac{\mu_p V_n + \mu_n V_p}{V_p + V_n} \tag{3.7}$$

A esta forma de calcular el umbral le llamamos *modelo de ponderación lineal*. La Fórmula (3.7) calcula un valor de umbral t_l situado entre μ_p y μ_n, el cual está más cercano de la media μ que tenga la menor varianza. Sin embargo, empíricamente se ha encontrado que en algunos casos este umbral se acerca demasiado a la media muestral ya que la varianza poblacional es mayor que la varianza muestral. Específicamente, cuando μ_n y V_n se acercan a cero el umbral t se aproxima demasiado a cero y se afecta la capacidad de generalización del clasificador.

Para aliviar este problema empleamos la Fórmula (3.8) que calcula el umbral t_e como el promedio de las medias μ_p y μ_n ponderadas por el exponencial de las varianzas V_p y V_n respectivamente. A esta forma de calcular el umbral le llamamos *modelo de ponderación exponencial*.

$$t_e = \frac{\mu_p e^{-\eta V_p} + \mu_n e^{-\eta V_n}}{e^{-\eta V_p} + e^{-\eta V_n}} \tag{3.8}$$

El término η es un parámetro que controla la inclinación de la curva exponencial. La Figura 3.1 muestra cómo el umbral varía con cada modelo de ponderación, el lineal y el exponencial, a medida que la varianza V_n decrece desde 0.1 hasta cero. Los demás parámetros han sido fijados en $\mu_p = 0{,}4$, $\mu_n = 0{,}008$, $V_p = 0{,}25$ y $\eta = 10$, valores típicos del problema WIR descrito en la Sección 3.3. A diferencia del modelo lineal, el exponencial es una función convexa, la cual desacelera el decremento del umbral a medida que la varianza se aproxima a cero.

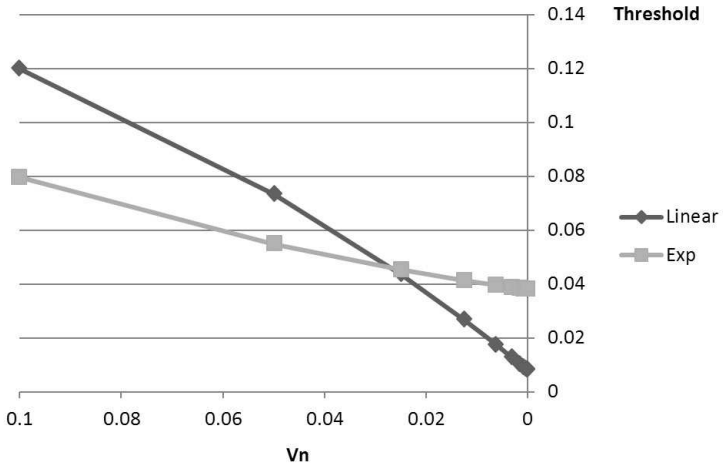


Figura 3.1: Valores del umbral de proporción en función de V_n de acuerdo a los modelos de ponderación lineal y exponencial de MIRocchio.

3.1.5. Clasificador MIRocchio

Hasta el momento hemos descrito en detalle cada uno de los elementos que forman parte del clasificador que proponemos. En esta sección se ponen todos los elementos juntos para mostrar cómo se combinan y sumarizar el funcionamiento del clasificador MIRocchio, describiendo los procesos que se llevan a cabo en las etapas de entrenamiento y clasificación. La Figura 3.2, que muestra esquemáticamente el funcionamiento del algoritmo MIRocchio, servirá para guiar la explicación. En la figura, x representa una instancia, X^- una bolsa negativa y la flecha $\leftarrow$ representa asignación de etiqueta de clase.

El primer paso en la etapa de entrenamiento es el proceso de proposicionalización llevado a cabo por el envoltorio siguiendo la heurística de la hipótesis multinstancia estándar: una instancia es asignada a la clase negativa si aparece en alguna bolsa negativa, en otro caso es asignada a la clase positiva. Las instancias con etiquetas de clase imputadas son usadas en las fórmulas de Rocchio (3.1) y (3.2) para calcular los perfiles de clase positivo ψ^+ y negativo ψ^-. Seguidamente, se reasignan etiquetas de clase a todas las instancias ya que en este momento el algoritmo dispone de más información sobre cada instancia: su cercanía con los perfiles de clase calculados. Una instancia x será asignada a la clase positiva si su semejanza coseno con el perfil de clase positivo es mayor que su semejanza coseno con el perfil de clase negativo. En caso contrario será asignada a la clase negativa. Luego, se calcula el umbral de proporción t sobre la base de los valores de $\pi(X)$ para cada bolsa de entrenamiento X. El umbral de proporción t puede calcularse mediante

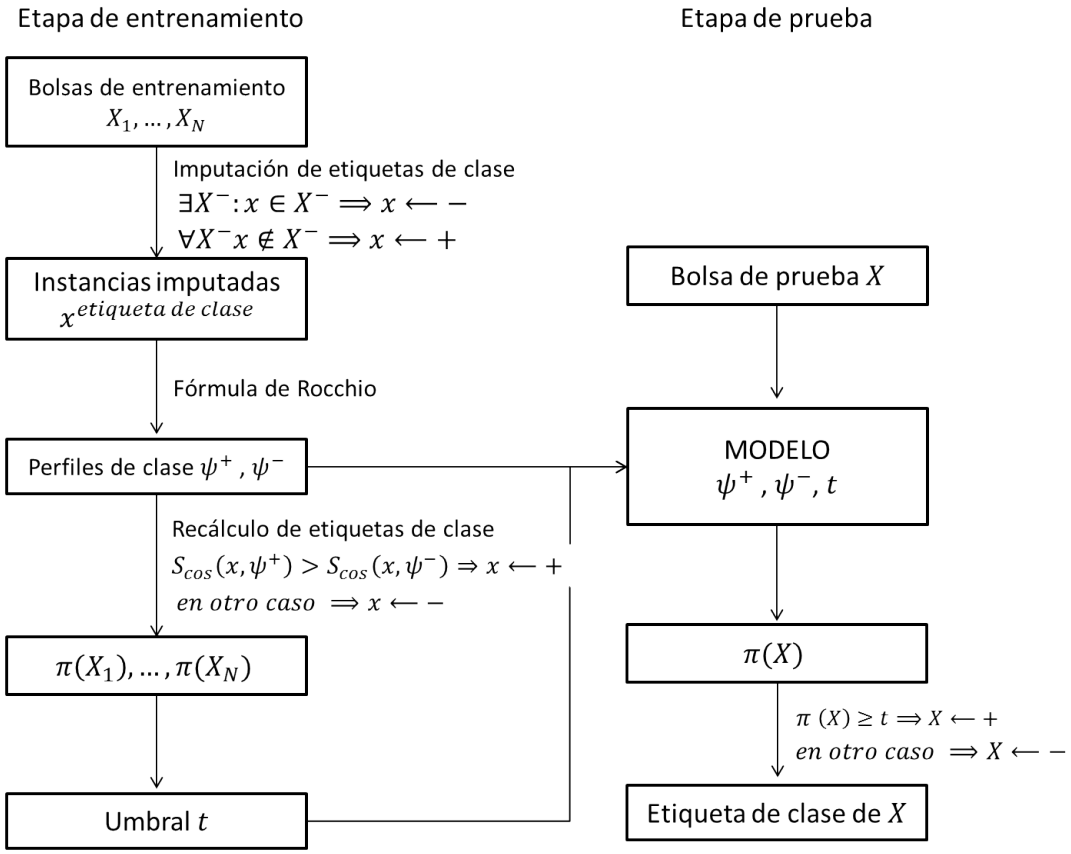


Figura 3.2: Esquema del algoritmo MIRocchio.

las Fórmulas (3.7) o (3.8), dependiendo del modelo de ponderación que se prefiera usar. Los perfiles de clase ψ^+ y ψ^-, junto al umbral de proporción t, constituyen el modelo de clasificación construido por el algoritmo MIRocchio durante su entrenamiento. A partir de este modelo se realiza la clasificación de nuevos ejemplos multinstancia.

Para clasificar una nueva bolsa X primeramente se determina la etiqueta de clase de las instancias de X. Cada instancia se asigna a la clase cuyo perfil es más semejante. Luego, se calcula $\pi(X)$ de igual manera que en el entrenamiento y se usa la Fórmula (3.6) para determinar la etiqueta de clase de la bolsa. Siguiendo la hipótesis multinstancia basada en proporción de instancias positivas, la bolsa X es asignada a la clase positiva si $\pi(X) \geq t$, en otro caso es asignada a la clase negativa.

3.1.6. Cálculo de los parámetros

El desempeño del clasificador de Rocchio depende significativamente del ajuste de los parámetros ρ^+ y ρ^- en las Fórmulas (3.1) y (3.2). Cohen & Singer [20] han sugerido que un adecuado ajuste de los parámetros del clasificador de Rocchio puede mejorar la exactitud del algoritmo en la clasificación textual, y que estos parámetros dependen en gran medida del conjunto de entrenamiento dado. Distintos autores [52, 9, 47, 93] han usado diferentes configuraciones para los parámetros del clasificador de Rocchio con resultados diversos. Moschitti [68] muestra que cuando se aplica un procedimiento sistemático de estimación de parámetros pueden obtenerse valores confiables para estos parámetros. Los experimentos en [68] muestran que la relación entre ρ y la exactitud de la clasificación se aproxima a una curva convexa con un único punto máximo. A partir de este resultado, Moschitti sugiere que el mejor valor para ρ puede obtenerse probando valores incrementales de ρ hasta que la exactitud de la clasificación en un conjunto de validación alcance un valor máximo.

Para el cálculo de los parámetros en el algoritmo MIRocchio se ha utilizado un diseño factorial que selecciona en la etapa de entrenamiento los valores óptimos de ρ^+ y ρ^- a partir de los valores de los conjuntos$\Lambda^+ = \{15, 20, 25, 30, 40\}$ y $\Lambda^- = \{2, 3, 4, 8, 15, 20\}$, respectivamente. En total son comparadas $5 \times 6 = 30$ configuraciones diferentes de ρ^+ y ρ^-, usando la exactitud (*accuracy*) en entrenamiento para guiar la búsqueda. Por regla general, el valor de ρ^+ debe ser mayor que el de ρ^-, por la siguiente razón. Según la imputación inicial de la hipótesis multinstancia basada en proporción, *todas* las instancias de una bolsa negativa son negativas, mientras que de las instancias de una bolsa positiva solo *una determinada proporción* es positiva, y el resto son negativas. Esto significa que las bolsas positivas involucran mayor incertidumbre que las negativas ya que contienen valores de atributos que no son específicos de la clase positiva, sino que por el contrario

pueden aparecer en instancias positivas y negativas por igual. Por tanto, la selección de atributos debe ser más intensa en las instancias positivas (mediante un valor mayor de ρ^+) que en las negativas para reducir el efecto de la incertidumbre y beneficiar la exactitud de la clasificación. Los valores de los conjuntosΛ^+ y Λ^- fueron determinados empíricamente. Los valores máximos deΛ^+ y Λ^- (40 y 20) están alrededor del límite superior recomendado [68, 123], ya que con estos valores es posible realizar una reducción de atributos bastante agresiva.

Este procedimiento para el cálculo de los parámetros no es inherente al algoritmo MIRocchio, por lo cual no fue reflejado en la Figura 3.2 que describe al algoritmo. Otros métodos más avanzados de optimización de parámetros pueden ser usados en su lugar.

3.2. Análisis de la eficiencia temporal de MIRocchio

Nuestra intensión no solo es obtener un algoritmo competitivo en términos de exactitud de la clasificación, sino también en términos de eficiencia temporal, como es requerido para la gestión de grandes volúmenes documentales y las aplicaciones *on-line*. Por este motivo, en esta sección se analiza la complejidad en tiempo de ejecución de nuestra propuesta.

En la etapa de entrenamiento, el método de imputación inicial tiene una complejidad temporal $\mathcal{O}\left(m \times N^2\right)$, donde m es el número de atributos y N es el número de instancias del conjunto de entrenamiento. Esto se debe a que para cada instancia positiva es necesario verificar si está incluida en una bolsa negativa. Este es el costo del peor caso, cuando el número de instancias positivas y negativas es igual a $N/2$. Sin embargo, lo característico en los problemas de categorización y recuperación de información es que el número de instancias positivas sea mucho menor que el número de instancias negativas. Por tanto, esta es una cota superior del costo computacional del método. Por otro lado, el algoritmo de Rocchio, en el cálculo de los perfiles de clase, comporta una complejidad temporal $\mathcal{O}\left(m \times N\right)$. El recálculo de las etiquetas de clase de las instancias se hace en tiempo $\mathcal{O}\left(m \times N\right)$ y el umbral de proporción se determina en tiempo $\mathcal{O}\left(N\right)$. Por tanto, la eficiencia temporal de MIRocchio en entrenamiento es $\mathcal{O}\left(m \times N^2\right)$, i.e., lineal respecto al número de atributos y cuadrática respecto al número de instancias.

Durante la etapa de clasificación, las instancias de la nueva bolsa solo tienen que ser comparadas con los perfiles de clase, por lo que la complejidad computacional de la etapa de generalización es $\mathcal{O}\left(m\right)$. Esto significa que la clasificación se realiza en un tiempo que depende solo linealmente del número de atributos, lo cual lo hace ideal para aplicaciones interactivas como, por ejemplo, las aplicaciones web. En la siguiente sección

se toma como caso de estudio una aplicación de minería web para probar las ventajas del algoritmo propuesto.

3.3. Aplicación de MIRocchio al problema de recomendación de páginas web índices

Para mostrar la eficacia del algoritmo MIRocchio hemos seleccionado como caso de uso su aplicación al problema de recomendación de páginas web índices. Este problema ha sido enfocado típicamente desde el campo de la recuperación de documentos textuales.

La red de Internet contiene un cúmulo de información tan vasto y enmarañado que se hace dificil encontrar información de calidad sobre un tema específico que interese a un usuario. Entre los esfuerzos por organizar la Internet están las páginas web índices. Estas son páginas webs que contienen gran cantidad de titulares y resúmenes breves mientras dejan la presentación detallada de los temas para páginas enlazadas. Por ejemplo, la entrada de la NBA en Yahoo! (sports.yahoo.com/nba/) es una página web índice.

Cada día un usuario de Internet puede encontrar muchas páginas web índices. Algunas pueden contener temas de interés para el usuario y otras no. Sería deseable tener un navegador web inteligente que analice automáticamente las páginas web índices visitadas por el usuario y trate de identificar si una nueva página web índice es interesanto o no para el usuario, de forma que solo le presente páginas que sean de su interés. Este problema se llama *recomendación de páginas web índices* (WIR, por sus siglas en inglés: *web index recommendation*), y es específicamente una tarea de minería de uso de la web.

Este problema ha sido abordado desde la perspectiva del aprendizaje supervisado tradicional, y los experimentos han mostrado que los desempeños de los mejores algoritmos investigados están alrededor de un 76.3 % para la *accuracy*, 63.4 % para *recall* y 66.1 % para *precision* [132]. En esta sección se muestra que el algoritmo MIRocchio obtiene desempeños mucho mejores, alcanzando alrededor de un 86.3 % para la *accuracy*, 78.4 % para el *recall* y 79.8 % para *precision*.

3.3.1. Enfoque multinstancia del problema de recomendación de páginas web índices

El problema WIR se caracteriza porque el usuario solo especifica si está interesado en una página web índice, en lugar de especificar en cuáles enlaces específicos está interesado. Por tanto, parece conveniente modelarlo con el enfoque multinstancia. En este caso la ambigüedad aparece por entradas polimórficas debido a múltiples partes.

El problema WIR fue presentado como un problema multinstancia por Zhou et al. [132]. Desde el punto de vista multinstancia, en el problema WIR una bolsa es una página web índice que llamaremos *página índice* por abreviar. A su vez, las instancias de una bolsa son las páginas webs enlazadas a la página índice, a las cuales llamaremos *páginas enlazadas*.

Zhou et al. recogieron y prepararon nueve conjuntos de datos multinstancia del problema WIR, en los cuales mostraron los beneficios del enfoque multinstancia sobre los algoritmos de clasificación tradicionales [132]. Estos conjuntos de datos fueron posteriormente usados en [125, 124] para probar y comparar nuevos métodos de solución al problema WIR desde el enfoque multinstancia.

Para construir los conjuntos de datos fueron seleccionadas 113 páginas índices, y se le pidio a nueve voluntarios que las clasificaran de acuerdo a sus intereses. Los nueve conjuntos de datos fueron el resultado de la manera particular en que cada voluntario clasificó las páginas índices. Cada página índice dió lugar a una bolsa que fue etiquetada positiva si la pagina índice fue del interés del usuario y negativa en caso contrario.

Los datos de las bolsas consisten solo en información textual; la información multimedia tal como audio, imágenes y video fue excluida para simplificar el análisis. Además, los datos fueron preprocesados usando selección de atributos, eliminando términos textuales con poco contenido semántico. Términos triviales como artículos, preposiciones, conjunciones, etc. fueron descartados.

Dentro de las bolsas, cada instancia es representada como un conjunto de términos $T = \{t_1, t_2, ..., t_n\}$, donde t_i $(i = 1, \ldots, n)$ es uno de los n términos más frecuentes que aparecen en la correspondiente página enlazada. Este tipo de representación no es usual en el aprendizaje automático e impide que los algoritmos de clasificación multinstancia desarrollados para uso general puedan ser aplicados directamente a estos conjuntos de datos.

3.3.2. Soluciones actuales del problema

Zhou et al. [132] propusieron el algoritmo Fretcit-kNN para resolver el problema WIR desde el enfoque multinstancia. Posteriormente, nuevos algoritmos multinstancia de la familia de programación genética fueron introducidos en [124] y [125] para resolver este problema. Estas propuestas son examinadas a continuación.

3.3.2.1. Fretcit-kNN

Fretcit-kNN [132] está basado en el algoritmo multinstancia CitationKNN [110], el cual fue descrito en la Sección 1.4.3.1. CitationKNN usa la representación vectorial del espacio de instancias clásica del aprendizaje automático. Fretcit-kNN es una adaptación de CitationKNN para trabajar con el tipo de representación basado en conjunto de términos frecuentes que usan los conjuntos de datos del problema WIR presentados en [132].

Para medir distancia entre dos bolsas $X = \{x_1, \ldots, x_{n_X}\}$ y $Z = \left\{z_1, \ldots, z_{n_{z'}}\right\}$ CitationKNN usa la distancia mínima de Hausdorff definida en la Fórmula (1.4). Fretcit-kNN sustituye la usual distancia Euclideana por una nueva medida de distancia que toma en cuenta el número de términos compartidos entre instancias. En Fretcit-kNN la distancia entre dos instancias x y z se define según la Fórmula (3.9).

$$fret\text{-}dist\,(x, z) = 1 - \sum_{\substack{i,j=1 \\ x_i = z_j}}^{n} \frac{1}{n} \tag{3.9}$$

Donde x_i representa el término i-ésimo de la instancia x. En otras palabras, Fretcit-kNN es equivalente al algoritmo CitationKNN donde $\|x - z\| = fret\text{-}dist\,(x, z)$.

Zhou et al. [132] reportaron resultados de Fretcit-kNN sobre los conjuntos de datos de referencia con conjuntos de términos de tamaño 5, 8, 10, 12 y 15. Fretcit-kNN fue comparado con una versión simplificada del clasificador de Rocchio que ellos llamaron TF-IDF [54], así como con una implementación clásica de kNN y una adaptación de kNN que considera tanto las referencias como los citadores de un nuevo objeto a clasificar (igual que CitationKNN pero con distancia Euclideana). Estos tres algoritmos se aplicaron directamente a las páginas índices ignorando las páginas enlazadas a éstas. Sin embargo, como el grueso de la información de las páginas índices está presente en sus páginas enlazadas, sus correspondientes desempeños fueron muy pobres [132]. Además, Zhou et al. modificaron la versión clásica y la adaptada del kNN para considerar la información de las páginas enlazadas, asignando a cada instancia la etiqueta de la bolsa que la contiene y remplazando la distancia Euclideana por $fret\text{-}dist$ para permitir su aplicación a la representación de conjuntos de términos frecuentes. Fretcit-kNN superó a los otros algoritmos en todas estas comparaciones [132].

3.3.2.2. G3P-MI y MOG3P-MI

Zafra et al. [125] introdujeron un enfoque de programación genética guiada por gramática para resolver el problema WIR. A diferencia del enfoque de Zhou et al., ellos usan una representación vectorial de las instancias del problema cuyos componentes corresponden a los términos textuales del universo documental. Ellos aplican, como preprocesamiento, una técnica de selección de atributos a los conjuntos de datos de Zhou et al., eliminando aquellos términos que aparecen en menos de 3 documentos y más de 40. Ellos entienden por documento el conjunto de páginas enlazadas (instancias) a una página índice.

Dentro de su enfoque, ellos proponen dos algoritmos, *boolean-G3P-MI* y *frequency-G3P-MI*, cuyas diferencias están en la forma usada para representar las instancias. En *boolean*-G3P-MI cada componente del vector es un valor lógico (verdadero o falso), el cual indica la presencia de un término en el documento. Mientras que en *frequency*-G3P-MI, cada componente del vector representa la frecuencia absoluta de un término en el documento.

Ambos métodos usan una gramática de libre contexto para hacer evolucionar un programa evolutivo en el cual los individuos son reglas que codifican las condiciones bajo las cuales una bolsa es positiva. Los métodos adoptan la hipótesis multinstancia estándar, según la cual la etiqueta de una bolsa se obtiene por la disyunción de las etiquetas de sus instancias, como en la Fórmula (1.1). En *boolean*-G3P-MI las reglas generadas se refieren a la presencia o ausencia de un término en una página dada, mientras que en *frequency*-G3P-MI indican si un término aparece en una página dada más veces o menos veces que un umbral predefinido. La función de aptitud usada en ambos casos trata de maximizar la exactitud de la clasificación (*accuracy*), buscando un balance entre *precision* y *recall* mediante la Fórmula (3.10).

$$aptitud = accuracy \times recall \times precision \tag{3.10}$$

Sin embargo, estos métodos no fueron capaces de obtener un buen balance entre *precision* y *recall*, por lo cual Zafra et al. [124] propusieron una nueva versión de su enfoque llamada MOG3P-MI la cual es un método multiobjetivo de programación genética guiada por gramática. MOG3P-MI está basado en el algoritmo evolutivo multiobjetivo SPEA2 [138], y es capaz de optimizar simultáneamente dos medidas de desempeño, específicamente la sensibilidad y la especificidad. La sensibilidad (resp., especificidad) mide la proporción de ejemplos realmente positivos (resp., negativos) que han sido correctamente clasificados. Los experimentos mostraron que MOG3P-MI mejora el desempeño de la clasificación de sus antecesores (*boolean*-G3P-MI y *frequency*-G3P-MI) [124].

3.3.3. Representación y procesamiento de los datos

Al igual que los algoritmos *frequency*-G3P-MI [125] y MOG3P-MI [124], nuestro algoritmo usa una representación vectorial clásica cuyos componentes son frecuencias de términos, y aplica en el preprocesamiento el mismo paso de selección de atributos. Sin embargo, mientras que estos métodos usan frecuencias absolutas en los atributos, nosotros calculamos una frecuencia normalizada y ponderada. Específicamente, usamos el esquema de ponderación de atributos $TF - IDF$ [95] mediante la Fórmula (3.11).

$$tfidf\,(t,x) = tf\,(t,x)\log\left(\frac{P}{df\,(t)}\right) \tag{3.11}$$

Donde $tf\,(t,x)$ es el número de veces que el término t aparece en una página enlazada x, P es el número total de páginas enlazadas en el conjunto de entrenamiento, y $df\,(t)$ denota el número de páginas enlazadas en el conjunto de entrenamiento en las cuales aparece el término t. Aplicamos la normalización coseno por medio de la Fórmula (3.12).

$$w_{xk} = tfidf\,(t_k,x) \Bigg/ \sqrt{\sum_{i=1}^{d}\left(tfidf\,(t_i,x)\right)^2} \tag{3.12}$$

Donde w_{xk} es el peso normalizado del término t_k en la página enlazada x, y d es la dimensión del vector atributo, i.e., el número total de términos que quedan después del proceso de selección de atributos.

3.3.4. Resultados experimentales

En esta sección se presentan los resultados experimentales que permiten validar el algoritmo MIRocchio. Primeramente, se presentan los conjuntos de datos, las medidas de desempeño y los parámetros de los algoritmos usados en el estudio experimental. Luego se comparan dos variantes de MIRocchio con aquellos algoritmos multinstancia desarrollados para dominios textuales y aplicados específicamente en el problema WIR: Fretcit-kNN y MOG3P-MI. No hacemos la comparación con G3P-MI porque en [124] queda claro que sus resultados son inferiores a los de MOG3P-MI. Las variantes de MIRocchio analizadas son (1) aquella que usa el modelo de ponderación lineal, que llamaremos MIRocchio-L, y (2) la que usa el modelo de ponderación exponencial, que llamaremos MIRocchio-E.

	Conjunto de entrenamiento		Conjunto de prueba	
Conjunto de datos	Positivas	Negativas	Positivas	Negativas
V1	17	58	4	34
V2	18	57	3	35
V3	14	61	7	31
V4	56	19	33	5
V5	62	13	27	11
V6	60	15	29	9
V7	39	36	16	22
V8	35	40	20	18
V9	37	38	18	20

Cuadro 3.1: Distribución de bolsas positivas y negativas en los conjuntos de dato del problema WIR de Zhou et al.

3.3.4.1. Conjuntos de datos

Zhou et al. [132] utilizaron un esquema de validación *hold-out* para su experimentación. Por cada conjunto de datos, usaron 75 páginas web índices como bolsas de entrenamiento y las 38 restantes como bolsas de prueba. Este esquema propuesto en [132] fue posteriomente usado en [125] y [124]. Por tanto, en esta comparación nosotros usaremos este mismo esquema de validación.

El Cuadro 3.1 muestra el número de bolsas positivas y negativas en las particiones de entrenamiento y prueba de cada conjunto de datos. Observe que el balance entre el tamaño de las clases positivas y negativas varía considerablemente en los conjuntos de datos. Ciertamente, la presencia de clases desbalanceadas es una característica muy frecuente en los problemas de recomendación. En este caso, los tres primeros voluntarios han marcado positivas solamente alrededor de un tercio de las bolsas, los tres voluntarios siguientes marcaron positivas alrededor de dos tercios de las bolsas, mientras que los últimos tres voluntarios marcaron positivas aproximadamente igual número de bolsas positivas y negativas.

3.3.4.2. Medidas de desempeño

Evaluamos la calidad de la clasificación usando cinco medidas del desempeño: *accuracy*, *precision*, *recall*/sensitividad, especificidad y AUC (*Area Under the ROC Curve*). Las tres primeras medidas fueron propuestas por [132] para analizar el comportamiento del algoritmo Fretcit-kNN. Mientras que la primera, tercera y cuarta medidas fueron usadas por [124] para medir el desempeño de MOG3P-MI. Nosotros hemos añadido aquí la

medida AUC para tener en cuenta el desbalance de clase que ocurre en seis de los nueve conjuntos de datos.

Las medidas *precision* y *recall* fueron definidas en las Fórmulas (2.1) y (2.2) respectivamente, mientras que *accuracy*, especificidad y AUC son usadas en esta sección de acuerdo a las Fórmulas (3.13) - (3.15).

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.13}$$

$$especificidad = \frac{TN}{TN + FP} \tag{3.14}$$

$$AUC = \frac{1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}}{2} \tag{3.15}$$

Observe que la Fórmula (3.15) es una aproximación de la definición de AUC dada por la Fórmula (2.4) en el Capítulo 2. El motivo de que usemos esta aproximación aquí es que para el cálculo del AUC exacto se requieren n puntos (TP_i, FP_i) de la curva ROC, sin embargo no tenemos acceso a esos puntos porque no disponemos de las implementaciones de los algoritmos del estado del arte, Fretcit-kNN y MOG3P-MI. De estos algoritmos solo disponemos de los resultados experimentales reportados por sus autores para estos conjuntos de datos, los cuales representan un solo punto de la curva ROC. La Fórmula (3.15) es usada con frecuencia en la literatura para casos como este [99].

3.3.4.3. Configuraciones algorítmicas

Por un lado, comparamos el modelo de ponderación lineal para seleccionar el umbral de proporción (MIRocchio-L) con el modelo exponencial (MIRocchio-E). En el algoritmo MIRocchio-E, el parámetro η para el cálculo de t_e, en la Fórmula (3.8), es puesto a 10, porque se comprobó experimentalmente que este valor funciona bien para la mayoría de los conjuntos de datos.

Aquí presentamos los resultados reportados por sus autores para las mejores configuraciones de cada algoritmo: 15 términos frecuentes para describir las instancias en Fretcit-kNN [132] y atributos booleanos en MOG3P-MI [124].

3.3.4.4. Comparación y análisis

El Cuadro 3.2 muestra los valores de *accuracy*, *precision*, *recall*/sensitividad, especificidad y AUC de MIRocchio-L/MIRocchio-E y sus clasificadores rivales sobre los nueve conjuntos de datos V1-V9. La última columna muestra el promedio de la medida de

Medida	Modelo	V1	V2	V3	V4	V5	V6	V7	V8	V9	Promedio
Accuracy											
	MIRocchio-L	**0.921**	0.737	**0.868**	**0.947**	0.711	0.842	0.763	0.605	0.816	0.801
	MIRocchio-E	**0.921**	**0.947**	0.842	0.895	0.816	0.868	0.816	**0.737**	**0.921**	**0.863**
	Fretcit-kNN [132]	**0.921**	0.868	**0.868**	0.895	**0.895**	**0.895**	0.816	**0.737**	0.632	0.836
	MOG3P-MI [124]	**0.921**	0.921	**0.868**	0.921	0.842	0.842	**0.842**	0.711	0.763	0.848
Precision											
	MIRocchio-L	0.571	0.231	**0.667**	0.943	0.722	0.829	0.706	0.600	0.762	0.670
	MIRocchio-E	0.600	**0.600**	**0.667**	**1.000**	0.813	0.875	**0.909**	0.778	**0.941**	**0.798**
	Fretcit-kNN [132]	0.600	0.333	**0.667**	0.968	**0.897**	**0.903**	0.800	**0.813**	0.667	0.739
	MOG3P-MI [124]	**0.667**	0.500	**0.667**	0.917	0.862	0.829	0.778	0.737	0.737	0.744
Recall / Sensitividad											
	MIRocchio-L	**1.000**	**1.000**	**0.571**	**1.000**	**0.963**	**1.000**	0.750	**0.750**	**0.889**	**0.880**
	MIRocchio-E	0.750	**1.000**	0.286	**1.000**	**0.963**	0.966	0.625	0.700	**0.889**	0.784
	Fretcit-kNN [132]	0.750	0.667	**0.571**	0.909	**0.963**	0.966	0.750	0.650	0.444	0.741
	MOG3P-MI [124]	0.500	0.667	**0.571**	**1.000**	0.926	**1.000**	**0.875**	0.700	0.778	0.780
Especificidad											
	MIRocchio-L	0.912	0.714	0.935	0.600	0.091	0.333	0.773	0.444	0.750	0.617
	MIRocchio-E	0.941	**1.000**	**0.968**	**1.000**	0.455	0.556	**0.955**	0.778	**0.950**	**0.838**
	Fretcit-kNN [132]	0.941	0.886	0.935	0.800	0.727	**0.667**	0.864	**0.833**	0.800	0.828
	MOG3P-MI [124]	**0.971**	0.943	0.936	0.480	**0.765**	0.426	0.818	0.722	0.750	0.757
AUC											
	MIRocchio-L	**0.956**	0.857	**0.753**	0.800	0.527	0.667	0.761	0.597	0.819	0.749
	MIRocchio-E	0.846	**0.971**	0.627	**0.939**	0.709	0.761	0.790	0.739	**0.919**	**0.811**
	Fretcit-kNN [132]	0.846	0.776	**0.753**	0.855	**0.845**	**0.816**	0.807	**0.742**	0.622	0.785
	MOG3P-MI [124]	0.735	0.805	**0.753**	0.700	0.827	0.722	**0.847**	0.711	0.764	0.763

Cuadro 3.2: Resultados de las variantes de MIRocchio y los clasificadores multinstancia de la literatura Fretcit-kNN y MOG3P-MI en el problema WIR según las medidas de desempeño *accuracy*, *precision*, *recall*, especificidad, y AUC.

desempeño de cada clasificador. El mejor método para cada conjunto de datos está resaltado en negritas.

Comparando primero las dos implementaciones propuestas de MIRocchio, es claro que la versión que usa un umbral de proporción ponderado exponencialmente se desempeña mucho mejor que la que asume el modelo de ponderación lineal. Como se dijo en la Sección 3.1.4, esto tiene que ver con el hecho de que cuando μ_n y V_n se acercan mucho a cero, el umbral t_l también se hace muy pequeño. Esto es más notable en los casos V4-V6, en los que V_p es muy elevado, provocando con frecuencia que el umbral de proporción caiga a cero, lo cual impide separar las bolsas positivas de las negativas.

Sin embargo, para los conjuntos de datos V1-V3, en los que la mayoría de los ejemplos

son negativos, el modelo de poderación lineal se desempeña muy bien, alcanzando incluso los más altos valores de AUC para estos conjuntos de datos. Este comportamiento puede explicarse de la siguiente manera. Los conjuntos de datos V1-V3 se corresponden con usuarios que tienen intereses muy específicos. A ellos les basta con que la página web índice tenga una pequeña proporción de enlaces interesantes (instancias positivas) ya que hay muy pocas páginas web índices que incluyen los temas de su interés. Por tanto, estos usuarios requieren que el valor de $\pi(b)$ sea más bajo. Consecuentemente, los conjuntos de datos V1-V3 se ajustan mejor con umbrales más bajos como los producidos por el modelo de ponderación lineal. Los conjuntos de datos balanceados V7-V9, y los conjuntos de datos V4-V9, donde la mayoría de los ejemplos son positivos, se corresponden con usuarios con intereses más amplios. Como para estos usuarios hay mayor abundancia de páginas web índices que incluyen temas de su interés, ellos tienen la posibilidad, si lo desean, de ser más exigentes en cuanto a la proporción de enlaces interesantes (instancias positivas) que debe incluir la página web índice. Por tanto, en estos conjuntos de datos hay mayor variabilidad de $\pi(b)$, por lo que el modelo de ponderación exponencial logra un mejor ajuste. Este resultado sugiere que podría usarse un modelo híbrido para la selección del umbral de proporción, basándose en el modelo de ponderación lineal cuando la mayoría de los ejemplos son negativos y en el modelo de ponderación exponencial en otro caso.

Cuando comparamos MIRocchio-E con Fretcit-kNN y MOG3P-MI, es claro que nuestro método muestra muy buen comportamiento, desempeñándose mejor, en promedio, para todas las medidas de desempeño seleccionadas. Como solo hay nueve conjuntos de datos, y frecuentemente ocurren empates en los valores de las medidas de desempeño, no podemos aplicar pruebas de significación estadística a estos resultados. Las pruebas no paramétricas recomendadas por estudios como [24, 41] tampoco son de ayuda, por la misma razón. Para comparar el desempeño de los clasificadores, podemos contar el número de conjuntos de datos en los cuales un algoritmo es el ganador. El Cuadro 3.3 muestra el número de victorias de cada algoritmo en la comparación entre MIRocchio-E y Fretcit-kNN para cada medida de desempeño. Decimos que el algoritmo A obtiene una victoria frente al algoritmo B en la medida de desempeño M si en el Cuadro 3.2 se observa que $M(A) > M(B)$. En general, ambos algoritmos tienen un desempeño comparable, excepto para especificidad, para la cual MIRocchio-E obtiene más victorias, indicando que MIRocchio-E es mejor que Fretcit-kNN evitando los falsos positivos (páginas índices irrelevantes recomendadas al usuario); y para AUC, donde Fretcit-kNN gana más casos, lo cual parece indicar que es mejor que MIRocchio-E manejando datos con clases desbalanceadas. Sin embargo, si consideramos solo los conjuntos de datos con

	Accuracy	*Precision*	*Recall*/Sensitividad	Especificidad	AUC
MIRocchio-E	4	5	4.5	5.5	3.5
Fretcit-kNN	5	4	4.5	3.5	5.5

Cuadro 3.3: Número de victorias en la comparación MIRocchio-E vs. Fretcit-kNN sobre los 9 conjuntos de datos del problema WIR.

	Accuracy	*Precision*	*Recall*/Sensitividad	Especificidad	AUC
MIRocchio-E	4.5	6.5	4.5	6.5	6
MOG3P-MI	4.5	2.5	4.5	2.5	3

Cuadro 3.4: Número de victorias en la comparación MIRocchio-E vs. MOG3P-MI sobre los 9 conjuntos de datos del problema WIR.

clases desbalanceadas V1-V6, esta diferencia es menos pronunciada, y si además para V1-V3 usamos el modelo de podenración lineal, como fue sugerido en el párrafo anterior, la ventaja es realmente para MIRocchio.

El Cuadro 3.4 muestra el número de victorias de cada algoritmo en la comparación entre MIRocchio-E y MOG3P-MI. En este caso, la ventaja de MIRocchio sobre MOG3P-MI considerando la *precision* es muy clara, lo cual significa que el primero hace recomendaciones mucho más precisas (más páginas índices recomendadas por MIRocchio-E son realmente relevantes). Similar observación se tiene para especificidad y para AUC. En términos de *recall*, MOG3P-MI y MIRocchio-E se desempeñan de forma comparable, pero los valores menores de precisión de MOG3P-MI indican que éste tiene cierta predisposición para asignar ejemplos a la clase positiva, lo cual resulta en menos falsos negativos pero más falsos positivos. Esto también es consistente con su modesta especificidad. Sin embargo, MIRocchio-E es más conservador para asignar un ejemplo a la clase positiva, generando, por tanto, menos falsos positivos y más falsos negativos, lo cual explica su alta precisión, su alta especificidad y su moderado *recall*. Esto puede verse como una ventaja de nuestro método: en el problema WIR, debido al gran número de páginas web índices que hay en Internet, recomendar una página irrelevante es más costoso que no recomendar una página web relevante. En otras palabras, la recomendación al usuario de páginas *potencialmente* interesantes (maximizando *recall*) no es tan importante como la recomendación de páginas *realmente* interesantes (maximizando *precision*).

3.4. Conclusiones del capítulo

En este capítulo hemos presentado un nuevo algoritmo de clasificación multinstancia, MIRocchio, orientado fundamentalmente a la recuperación/categorización de documentos textuales. El algoritmo consiste en un envoltorio que encapsula a un clasificador de Rocchio monoinstancia, al cual le sirve de interfaz con el problema multinstancia. El envoltorio fue diseñado en el marco de la investigación para implementar una nueva hipótesis multinstancia que hemos propuesto para el dominio de recuperación/categorización de documentos textuales. La hipótesis, bautizada como hipótesis multinstancia basada en proporción, asume que una bolsa es positiva si contiene una proporción de instancias positivas por encima de cierto umbral que caracteriza al problema y que es susceptible de ser aprendido.

El problema de recomendación de páginas web índices fue seleccionado para probar la eficacia del algoritmo MIRocchio porque es una aplicación en-línea de clasificación textual que se presta de forma natural para la modelación multinstancia. Los resultados experimentales mostraron que MIRocchio es capaz de mejorar el desempeño de los clasificadores multinstancia del estado del arte en este dominio de aplicación, aventajándolos sobre todo en precisión. Además, el estudio del costo computacional del algoritmo mostró que el tiempo de clasificación solo depende linealmente del número de atributos, lo cual lo hace ideal para aplicaciones interactivas como es el caso de esta aplicación de minería web.

El algoritmo propuesto implementa la hipótesis multinstancia basada en proporción mediante un envoltorio que encapsula al clasificador de Rocchio. El clasificador de Rocchio es un algoritmo clásico en el campo de la recuperación de documentos textuales, sin embargo otros algoritmos de clasificación típicos de este dominio de aplicación pueden ser usados como clasificador monoinstancia base sin necesidad de ser modificados, tan solo siendo encapsulados por el envoltorio. En otras palabras, a través del algoritmo envoltorio es posible tomar cualquier algoritmo de clasificación monoinstancia y usarlo para el aprendizaje multinstancia bajo la hipótesis multinstancia basada en proporción.

Comentarios finales

A continuación se resumen brevemente los resultados alcanzados en este trabajo y se destacan las principales conclusiones obtenidas. Se presentan las publicaciones asociadas a la investigación desarrollada y se comentan algunos temas de investigación que se desprenden del estudio y que parecen ser líneas interesantes para ser abordadas en próximos trabajos.

Conclusiones

En la investigación se han desarrollado dos algoritmos para la clasificación multinstancia. El primero, denominado MISMOTE, ofrece una solución al problema de clases desbalanceadas que afecta a los problemas de clasificación multinstancia. El segundo, denominado MIRocchio, es un algoritmo de clasificación multinstancia, basado en el clasificador de Rocchio y orientado a la solución de problemas de recuperación y clasificación de documentos textuales. Los siguientes apartados resumen brevemente los resultados obtenidos en cada uno de ellos y presentan las conclusiones más relevantes.

Algoritmo MISMOTE

Para mitigar el problema de clases desbalanceadas en la clasificación multinstancia se desarrolló el algoritmo MISMOTE, el cual es un método de preprocesamiento de conjuntos de datos multinstancia que permite sobremuestrear la clase minoritaria introduciendo bolsas sintéticas generadas a partir de la interpolación de bolsas adyacentes. Para solucionar el problema que impone la interpolación de dos bolsas se definió el tamaño de la bolsa sintética como el promedio de los tamaños de las bolsas más cercanas. Las instancias de la bolsa sintética se obtienen interpolando instancias seleccionadas aleatoriamente de las dos bolsas adyacentes que dan origen a la bolsa sintética.

El análisis estadístico de los resultados experimentales con 13 conjuntos de datos y 6 clasificadores mostró que, en un problema de clasificación multinstancia de dos clases desbalanceadas, aplicar MISMOTE antes del entrenamiento del clasificador generalmente aumenta la exactitud de la clasificación de ambas clases, y particularmente de

la clase pequeña, permitiendo identificar significativamente más ejemplos positivos sin un aumento apreciable de falsos positivos. Un segundo experimento donde se comparó el AdaBoost estándar entrenado con MISMOTE con algoritmos amplificadores sensibles al costo mostró que ambas soluciones tienen desempeños comparables respecto al AUC y F1.

Por último, aunque el método MISMOTE implica un costo adicional de preprocesamiento también supone menor costo computacional en el entrenamiento, mayor interpretabilidad del modelo y la reutilización eficiente de la solución de remuestro en un conjunto de entrenamiento determinado.

Concretamente, dando respuesta a la hipótesis que dió lugar al objetivo de esta propuesta: el método de restablecer el balance entre el tamaño de las clases introduciendo bolsas sintéticas en la clase minoritaria del conjunto de entrenamiento, como paso previo antes de construir el clasificador, permite mitigar el problema de clases desbalanceadas en la clasificación multinstancia, mejorando de forma significativa el desempeño de los algoritmos de clasificación multinstancia.

Algoritmo MIRocchio

Con el objetivo de adaptar el clasificador de Rocchio al aprendizaje multinstancia se desarrolló el algoritmo de clasificación MIRocchio que implementa la hipótesis multinstancia basada en proporción. Esta hipótesis fue concebida en la tesis como una adaptación del algoritmo a las características de los problemas de recuperación y clasificación de documentos textuales. La hipótesis asume que una bolsa es positiva si contiene una proporción de instancias positivas por encima de cierto umbral que caracteriza al problema y que es susceptible de ser aprendido. El algoritmo MIRocchio funciona como una interfaz que comunica los datos del problema multinstancia con el clasificador monoinstancia de Rocchio basada en la hipótesis multinstancia de proporción.

El clasificador MIRocchio fue comprobado experimentalmente en el problema de recomendación de páginas web índices, el cual fue seleccionado como caso de estudio por ser una aplicación *on-line* típicamente enfocada desde el campo de la recuperación de documentos textuales. MIRocchio fue comparado en este problema con dos clasificadores multinstancia del estado del arte orientados para el dominio de aplicación textual. Los resultados experimentales, que incluyeron 9 conjuntos de datos, mostraron que MIRocchio es capaz de mejorar el desempeño de los clasificadores multinstancia del estado del arte en este dominio de aplicación, aventajándolos sobre todo en precisión. Además, el estudio del costo computacional del algoritmo mostró que el tiempo de clasificación solo depende linealmente del número de atributos, lo cual lo hace ideal para aplicaciones

interactivas como es el caso de esta aplicación de minería web.

En conclusión, respondiendo a la hipótesis que nos planteamos resolver con el desarrollo de esta propuesta: es posible alcanzar resultados competitivos con respecto al de otros clasificadores multinstancia en la solución de problemas de clasificación de documentos textuales con un algoritmo de clasificación multinstancia basado en el clasificador de Rocchio.

Publicaciones asociadas a la tesis

Como resultado de la presente investigación se realizó la siguiente producción científica:

Monografía

- D.S. Tarragó, Aprendizaje Multinstancia, Editoral Feijoó, UCLV, Santa Clara, 2013.

Revistas indizadas

- D.S. Tarragó, C. Cornelis, F. Herrera, MISMOTE: synthetic minority over-sampling technique for multiple instance learning with imbalanced data, enviada a Information Sciences.
- D.S. Tarragó, C. Cornelis, R. Bello, F. Herrera, A Multi-Instance Learning Wrapper Based on the Rocchio Classifier for Web Index Recommendation, Knowledge-Based Systems. (2014). doi:http://dx.doi.org/10.1016/j.knosys.2014.01.008.

Trabajos futuros

En el desarrollo de la investigación, y a partir de sus resultados, han surgido un conjunto de ideas y motivaciones que, yendo más allá del alcance de esta tesis, decidimos plantearlas como temas para trabajos futuros.

MISMOTE para atributos mixtos

Actualmente, MISMOTE solo maneja conjuntos de datos con atributos continuos. Por tanto, se necesita claramente una extensión que permita manejar datos con atributos nominales y mixtos. Sin embargo, esta extensión aparentemente simple requiere la solución de otro problema de investigación. Recuérdese que MISMOTE necesita calcular los vecinos más cercanos de cada bolsa, y para ello usa la distancia de Hausdorff.

Tradicionalmente existieron medidas de distancia para vectores numéricos. Por ejemplo, la más usada, la distancia Euclideana, se conoce desde hace varios milenios. Sin embargo, en el aprendizaje supervisado frecuentemente los vectores que describen a los ejemplos codifican una mezcla de atributos numéricos con atributos nominales. Por tanto, se ha requerido el desarrollo de medidas de distancia más avanzadas que manejen adecuadamente distintos tipos de atributos. A estas medidas se les ha llamado *medidas de distancia heterogeneas*.

Hoy en día existen diversas medidas de distancia heterogeneas para algoritmos de aprendizaje supervisado tradicional y se ha demostrado que su uso mejora significativamente las capacidades de predicción y generalización de estos algoritmos. Sin embargo, no se conoce la existencia de medidas de distancia heterogeneas para algoritmos multi-instancia. Particularmente, en este caso sería necesario diseñar una distancia de Hausdorff heterogenea.

MISMOTE con muestreo dirigido

El objetivo implícito del algoritmo MISMOTE es generar bolsas con alta probabilidad de que sean positivas. Mientras más instancias positivas tiene una bolsa mayor es su probabilidad de ser positiva. Asumiendo que es más probable obtener una instancia positiva si al menos una de las instancias interpoladas es positiva, y aun más probable si las dos instancias interpoladas son positivas, la actual política de MISMOTE de seleccionar instancias al azar para interpolar sugiere que la probabilidad de obtener instancias positivas en la bolsa sintética es baja. En cambio, si el algoritmo seleccionara las instancias que va a interpolar basado en que tengan altas probabilidades de ser positivas, entonces cabría esperar un mayor número de instancias positivas dentro de la bolsa sintética y por tanto mayor probabilidad de que ésta represente a una bolsa positiva. Este es un tema interesante para explorar en trabajos futuros.

Ensambles basados en MISMOTE

El estudio hecho sobre MISMOTE abre un nuevo sendero en la búsqueda de soluciones al problema de clases desbalanceadas en conjuntos de datos multinstancia. Otros métodos de muestreo pueden ser adaptados también al escenario multinstancia. Por ejemplo, el submuestreo aleatorio de la clase mayoritaria tiene una implementación relativamente simple en el enfoque multinstancia, y se han reportado [15, 29] buenos resultados en la clasificación monoinstancia para la combinación de este método con técnicas de sobremuestreo tales como MISMOTE. Se han realizado muchas mejoras del algoritmo básico

SMOTE, incluyendo el uso de técnicas de edición y limpieza, y ubicación estratégica de los ejemplos sintéticos (e.g. [49, 14, 81, 6]), las cuales también pueden ser adaptadas al escenario multinstancia. Por último, el desarrollo de MISMOTE como una técnica de muestreo básica para la clasificación multinstancia, también permite adaptar al escenario multinstancia muchos algoritmos exitosos que combinan los ensambles con técnicas de muestreo tales como SMOTEBoost [16], RUSBoost [96] y SMOTEBagging [112], entre otros.

Clasificadores basados en perfiles de bolsa

En cuanto al desarrollo de algoritmos de clasificación orientados a determinados dominios de competencia, existen muchas posibilidades en dos sentidos: hay muchos dominios de aplicación que requieren algoritmos de clasificación multinstancia específicos, y hay muchos métodos de clasificación específicamente orientados a un dominio de aplicación que pueden ser adaptados al enfoque multinstancia.

En lo que respecta al algoritmo MIRocchio, una desventaja del tipo de algoritmo propuesto es que en cualquier proceso de proposicionalización necesariamente se introducen sesgos en el aprendizaje. Para superar este problema sería muy interesante como trabajo futuro modificar el clasificador de Rocchio para transformarlo en un algoritmo multinstancia *per se*, sin necesidad de un envoltorio. Los perfiles de clase generados por tal algoritmo serían bolsas en lugar de vectores de atributos, y para clasificar una nueva bolsa se utilizaría una distancia entre conjuntos, como la de Hausdorff, para determinar la similitud con los perfiles de clase.

Otra alternativa para diseñar clasificadores basados en perfiles de bolsa es definiendo los perfiles de clase en función de las aproximaciones rugosas y/o difusas de las bolsas de entrenamiento.

Apéndice

Para respaldar los resultados experimentales discutidos en la Sección 2.3.2, los Cuadros 3.5, 3.6 y 3.7 presentan los detalles de las medidas de desempeño AUC, *precision* y *recall*, respectivamente. En cada tabla se muestra el valor de la medida de desempeño de cada clasificador, en cada conjunto de datos, y en ambos grupos: en la fila encabezada por «Sí», los que entrenaron con MISMOTE; y en la fila siguiente, encabezada por «No», los que no entrenaron con MISMOTE. Las dos últimas filas de cada tabla representa el desempeño del clasificador promedio. La última columna de cada tabla representa el desempeño del conjunto de datos promedio.

MISMOTE	Clasificador	ato	bon	cha	pro	fox	tig	ele	w1	w2	w3	w4	w5	w6	Prom
Sí	CitationKNN	0,719	0,741	0,678	0,481	0,481	0,485	0,500	0,689	0,788	0,757	0,681	0,638	0,735	0,644
No		0,718	0,744	0,689	0,500	0,500	0,500	0,500	0,671	0,769	0,717	0,686	0,634	0,704	0,641
Sí	MILR	0,701	0,761	0,789	0,972	0,494	0,722	0,758	0,722	0,836	0,696	0,744	0,620	0,683	0,731
No		0,777	0,723	0,823	0,962	0,483	0,757	0,792	0,725	0,849	0,745	0,751	0,635	0,764	0,753
Sí	MI-SVM	0,500	0,500	0,500	0,507	0,501	0,668	0,547	0,633	0,789	0,631	0,552	0,519	0,566	0,570
No		0,500	0,500	0,500	0,642	0,523	0,743	0,636	0,638	0,698	0,655	0,563	0,532	0,630	0,597
Sí	MILES	0,775	0,646	0,704	0,943	0,515	0,643	0,614	0,475	0,475	0,475	0,479	0,479	0,479	0,592
No		0,760	0,676	0,679	0,957	0,485	0,705	0,739	0,500	0,500	0,500	0,500	0,500	0,500	0,616
Sí	MITI	0,630	0,636	0,577	0,724	0,502	0,644	0,602	0,629	0,810	0,614	0,523	0,521	0,606	0,617
No		0,692	0,698	0,670	0,826	0,543	0,733	0,608	0,648	0,777	0,635	0,532	0,532	0,581	0,652
Sí	AdaBoost	0,761	0,806	0,828	0,949	0,593	0,815	0,815	0,711	0,803	0,616	0,569	0,560	0,639	0,728
No		0,774	0,837	0,858	0,961	0,639	0,842	0,797	0,709	0,819	0,643	0,604	0,575	0,656	0,747
Sí	Promedio	0,681	0,682	0,679	0,763	0,514	0,663	0,639	0,643	0,750	0,632	0,591	0,556	0,618	0,647
No		0,704	0,696	0,703	0,808	0,529	0,713	0,679	0,649	0,735	0,649	0,606	0,568	0,639	0,668

Table 3.5: Valores de AUC obtenidos para cada conjunto de datos y algoritmo de clasificación multinstancia, entrenado por un lado con MISMOTE, y por otro lado, en los datos con clases desbalanceadas.

MISMOTE	Clasificador	ato	bon	cha	pro	fox	tig	ele	w1	w2	w3	w4	w5	w6	Prom
Sí	CitationKNN	0,556	0,568	0,367	0,000	0,000	0,000	0,000	0,360	0,587	0,342	0,557	0,385	0,487	0,324
No		0,392	0,393	0,291	0,204	0,174	0,206	0,200	0,377	0,559	0,457	0,408	0,336	0,399	0,338
Sí	MITI	0,697	0,676	0,636	0,771	0,203	0,580	0,557	0,304	0,487	0,276	0,279	0,260	0,445	0,475
No		0,715	0,665	0,728	0,828	0,306	0,641	0,498	0,318	0,494	0,303	0,295	0,282	0,405	0,498
Sí	MILR	0,483	0,830	0,691	0,790	0,211	0,510	0,600	0,376	0,496	0,458	0,435	0,298	0,465	0,511
No		0,451	0,409	0,440	0,950	0,170	0,485	0,510	0,318	0,481	0,306	0,435	0,293	0,488	0,441
Sí	MI-SVM	0,000	0,000	0,000	0,200	0,200	0,835	0,933	0,286	0,389	0,296	0,933	0,367	0,586	0,387
No		0,000	0,000	0,000	0,939	0,197	0,733	0,624	0,248	0,281	0,262	0,522	0,342	0,578	0,363
Sí	MILES	0,596	0,542	0,489	0,787	0,271	0,440	0,484	0,000	0,000	0,000	0,000	0,000	0,000	0,278
No		0,549	0,667	0,465	0,762	0,143	0,447	0,443	0,000	0,000	0,000	0,212	0,212	0,212	0,316
Sí	AdaBoost	0,743	0,765	0,649	0,886	0,337	0,796	0,844	0,307	0,470	0,260	0,314	0,235	0,499	0,546
No		0,741	0,720	0,668	0,837	0,376	0,697	0,720	0,319	0,503	0,284	0,321	0,220	0,474	0,529
Sí	Promedio	0,513	0,563	0,472	0,572	0,204	0,527	0,570	0,272	0,405	0,272	0,420	0,257	0,414	0,420
No		0,475	0,476	0,432	0,753	0,228	0,535	0,499	0,263	0,386	0,269	0,366	0,281	0,426	0,414

Table 3.6: Valores de *precision* obtenidos para cada conjunto de datos y algoritmo de clasificación multinstancia, entrenado por un lado con MISMOTE, y por otro lado, en los datos con clases desbalanceadas.

MISMOTE	Clasificador	ato	bon	cha	pro	fox	tig	ele	w1	w2	w3	w4	w5	w6	Prom
Sí	CitationKNN	0,243	0,360	0,200	0,000	0,000	0,000	0,000	0,200	0,476	0,095	0,317	0,175	0,408	0,190
No		0,529	0,554	0,533	1,000	1,000	1,000	1,000	0,400	0,619	0,371	0,425	0,325	0,500	0,635
Sí	MITI	0,305	0,314	0,178	0,490	0,057	0,354	0,256	0,543	0,819	0,571	0,150	0,175	0,317	0,348
No		0,443	0,463	0,370	0,690	0,171	0,546	0,288	0,581	0,724	0,571	0,167	0,192	0,275	0,422
Sí	MILR	0,052	0,131	0,133	0,870	0,162	0,423	0,400	0,371	0,514	0,352	0,492	0,267	0,483	0,358
No		0,781	0,509	0,630	0,807	0,224	0,600	0,605	0,505	0,677	0,486	0,541	0,343	0,575	0,560
Sí	MI-SVM	0,000	0,000	0,000	0,014	0,010	0,354	0,096	0,638	0,905	0,571	0,108	0,058	0,167	0,225
No		0,000	0,000	0,000	0,290	0,324	0,538	0,320	0,905	0,952	0,867	0,167	0,133	0,325	0,371
Sí	MILES	0,510	0,309	0,400	0,890	0,171	0,385	0,312	0,000	0,000	0,000	0,000	0,000	0,000	0,229
No		0,543	0,366	0,481	0,986	0,152	0,608	0,656	0,000	0,000	0,000	1,000	1,000	1,000	0,522
Sí	AdaBoost	0,400	0,491	0,252	0,697	0,076	0,446	0,336	0,629	0,790	0,590	0,158	0,142	0,350	0,412
No		0,448	0,571	0,444	0,779	0,200	0,562	0,360	0,638	0,810	0,638	0,183	0,133	0,350	0,471
Sí	Promedio	0,252	0,268	0,194	0,493	0,079	0,327	0,233	0,397	0,584	0,363	0,204	0,136	0,287	0,294
No		0,457	0,410	0,410	0,759	0,345	0,642	0,538	0,505	0,630	0,489	0,414	0,354	0,504	0,497

Table 3.7: Valores de *recall* obtenidos para cada conjunto de datos y algoritmo de clasificación multinstancia, entrenado por un lado con MISMOTE, y por otro lado, en los datos con clases desbalanceadas.

Bibliografía

[1] R.A. Amar, D.R. Dooly, S.A. Goldman, and Q. Zhang. Multiple-instance learning of real-valued data. In *Machine Learning-International Workshop*, pages 3–10, 2001.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 561–568, 2002.

[3] P. Auer and R. Ortner. A boosting approach to multiple instance learning. In J.F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201 of *Lecture Notes in Computer Science*, pages 63–74. Springer Berlin / Heidelberg, 2004.

[4] B. Babenko, M. H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.

[5] M. Basu and T.K. Ho, editors. *Data Complexity in Pattern Recognition*. Advanced Information and Knowledge Processing. Springer, 2006.

[6] G. E. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

[7] I.B. Bersuker, A.S. Dimoglo, M. Yu. Gorbachov, P.F. Vlad, and M. Pesaro. *New Journal of Chemistry*, 15:307, 1991.

[8] J. Bi, Y. Chen, and J. Z. Wang. A sparse support vector machine approach to region-based image categorization. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 1121–1128, 2005.

[9] Y. Bi, D. Bell, H. Wang, G. Guo, and J. Guan. Combining multiple classifiers using dempster's rule for text categorization. *Applied Artificial Intelligence*, 21(3):211–239, 2007.

[10] L. Bjerring and E. Frank. Beyond trees: Adopting MITI to learn rules and ensemble classifiers for multi-instance data. In *Proceedings of the 24th international conference on Advances in Artificial Intelligence*, pages 41–50. Springer, 2011.

[11] H. Blockeel, D. Page, and A. Srinivasan. Multi-instance tree learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 57–64, 2005.

[12] J. Bolton and P. Gader. Application of multiple-instance learning for hyperspectral image analysis. *IEEE Geoscience and Remote Sensing Letters*, 8(5):889–893, 2011.

[13] J. Bolton, P. Gader, H. Frigui, and P. Torrione. Random set framework for multiple instance learning. *Information Sciences*, 181(11):2061–2070, 2011.

[14] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Advances in Knowledge Discovery and Data Mining*, pages 475–482. Springer, 2009.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.

[16] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer. SMOTEBoost: improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003.

[17] Y. Chen, J. Bi, and J.Z. Wang. MILES: multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1931–1947, 2006.

[18] Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research*, 5:913–939, 2004.

[19] Y. Chevaleyre and J.D. Zucker. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. In *Canadian Conference on AI*, pages 204–214, 2001.

[20] W.W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141–173, 1999.

[21] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[22] S. Das, B. Amoedo, F. De la Torre, and J. Hodgins. Detecting parkinsons' symptoms in uncontrolled home environments: a multiple instance learning approach. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, volume 2012, pages 3688–3691. IEEE, 2012.

[23] L. De Raedt. Attribute-value learning versus inductive logic programming: The missing links. In D. Page, editor, *Inductive Logic Programming*, volume 1446 of *Lecture Notes in Computer Science*, pages 1–8. Springer Berlin / Heidelberg, 1998.

[24] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[25] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[26] T. G. Dietterich, A. N. Jain, R. H. Lathrop, and T. Lozano-perez. A comparison of dynamic reposing and tangent distance for drug activity prediction. In *Advances in Neural Information Processing Systems 6 (NIPS Conference)*, pages 216–223. Morgan Kaufmann, 1994.

[27] T. G. Dietterich, R. H. Lathrop, and T. LozanoPerez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[28] J. Ding, H. Cheng, J. Huang, J. Liu, and Y. Zhang. Breast ultrasound image classification based on multiple-instance learning. *Journal of Digital Imaging*, 25(5):620–627, 2012.

[29] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[30] T. Fawcett. ROC graphs with instance-varying costs. *Pattern Recognition Letters*, 27(8):882–891, 2006.

[31] C. Fehr, J. Galindo, R. Haubrichs, and R. Perret. *Helv. Chim. Acta*, 72:1537, 1989.

[32] S. Feng, W. Xiong, B. Li, C. Lang, and X. Huang. Hierarchical sparse representation based multi-instance semi-supervised learning with application to image categorization. *Signal Processing*, 94:595–607, 2014.

[33] J. Foulds. *Learning Instance Weights in Multi-Instance Learning.* Master thesis, University of Waikato, Hamilton, New Zealand, February 2008.

[34] J. Foulds and E. Frank. Revisiting multiple-instance learning via embedded instance selection. In *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, pages 300–310. Springer-Verlag, 2008.

[35] A. Frank and A. Asuncion. *UCI Machine Learning Repository.* University of California, Irvine, School of Information and Computer Sciences, 2010.

[36] E. Frank and X. Xu. Applying propositional learning algorithms to multi-instance data. Technical report 06/03, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2003.

[37] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. pages 148–156, 1996.

[38] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998.

[39] G. Fu, X. Nan, H. Liu, R. Y. Patel, P. R. Daga, Y. Chen, D. E. Wilkins, and R. J. Doerksen. Implementation of multiple-instance learning in drug activity prediction. *BMC Bioinformatics*, 13(15):1–12, 2012.

[40] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):463–484, 2012.

[41] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010.

[42] S. García and F. Herrera. An extension on "Statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008.

[43] V. García, R. A. Mollineda, and J. S. Sánchez. On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Analysis and Applications*, 11(3-4):269–280, 2008.

[44] V. García, J. S. Sánchez, and R. A. Mollineda. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25(1):13–21, 2012.

[45] T. Gärtner, P. A. Flach, A. Kowalczyk, and A.J. Smola. Multi-instance kernels. In *In Proc. 19th International Conf. on Machine Learning*, pages 179–186. Morgan Kaufmann, 2002.

[46] I. Gondra and T. Xu. A multiple instance learning based framework for semantic image segmentation. *Multimedia Tools and Applications*, 48(2):339–365, 2010.

[47] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer. Using knn model-based approach for automatic text. In *Proc. ODBASE'03, the 2nd Internat. Conf. on Ontologies, Database and Applications of Semantics*, LNCS, pages 986–996, 2003.

[48] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[49] H. Han, W.-Y. Wang, and B.-H. Mao. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *Advances in intelligent computing*, pages 878–887. Springer, 2005.

[50] D. Haussler. Convolution kernels on discrete structures. Technical report, Technical report, UC Santa Cruz, 1999.

[51] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[52] D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 301–315, 1995.

[53] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

[54] T. Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proc. ICML1997: the Fourteenth Internat. Conf. on Machine Learning*, page 143–151, 1997.

[55] A. Katsamanis and J. Gibson. Multiple instance learning for classification of human behavior observations. In *Proceedings of the 4th international conference on Affective computing and intelligent interaction-Volume Part I*, pages 145–154. Springer-Verlag, 2011.

[56] S. Kotsiantis, D. Kanellopoulos, and V. Tampakas. Financial application of multi-instance learning: two greek case studies. *Journal of Convergence Information Technology*, 5(8):42–53, 2010.

[57] H. Kriegel, A. Pryakhin, and M. Schubert. An EM-approach for clustering multi-instance objects. In *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 139–148. Springer, 2006.

[58] C. Lee, A. Katsamanis, M. P. Black, B. R. Baucom, P. G. Georgiou, and S. S. Narayanan. Affective state recognition in married couples ' interactions using PCA-based vocal entrainment measures with multiple instance learning. In *Proceedings of the 4th international conference on Affective computing and intelligent interaction-Volume Part II*, pages 31–41. Springer-Verlag, 2011.

[59] B. Li, W. Xiong, and W. Hu. Web horror image recognition based on context-aware multi-instance learning. In *IEEE 11th International Conference on Data Mining*, pages 1158–1163. IEEE, 2011.

[60] D. X. Li, J. L. Fan, D. W. Wang, and Y. Liu. Latent topic based multi-instance learning method for localized content-based image retrieval. *Computers and Mathematics with Applications*, 64(4):500–510, 2012.

[61] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1987.

[62] H. Liu, C. Zhou, J. Shen, P. Li, and S. Zhang. Video caption detection algorithm based on multiple instance learning. In *2010 Fifth International Conference on Internet Computing for Science and Engineering*, pages 20–24. IEEE, 2010.

[63] V. López, A. Fernández, S. García, V. Palade, and F. Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[64] O. Maron. *Learning from Ambiguity*. PhD thesis, Massachusetts Institute of Technology, United States, 1998.

[65] O. Maron and T. Lozano-Pérez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS Conference)*, pages 570–576. MIT Press, 1998.

[66] F. U. Minhas and A. Ben-Hur. Multiple instance learning of calmodulin binding sites. *Bioinformatics*, 28(18):i416–i422, 2012.

[67] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, New York, NY, USA., 1997.

[68] A. Moschitti. A study on optimal parameter tuning for rocchio text classifier. In *ECIR*, LNCS2633, pages 420–435, 2003.

[69] J.F. Murray, G.F. Hughes, and K. Kreutz. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, 6:783–816, 2005.

[70] J.N. Narvaez, B.K. Lavine, and P.C. Jurs. *Chemical Senses*, 11:145–156, 1986.

[71] J. Ning, W. Shi, S. Yang, and P. Yanne. Visual tracking based on distribution fields and online weighted multiple instance learning. *Image and Vision Computing*, 31(11):853–863, 2013.

[72] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[73] A. Orriols-Puig and E. Bernadó-Mansilla. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*, 13(3):213–225, 2009.

[74] M. Popescu and A. Mahnot. Early illness recognition using in-home monitoring sensors and multiple instance learning. *Methods of information in medicine*, 51(4):359–367, 2012.

[75] R. C. Prati, G. E. Batista, and M. C. Monard. Class imbalances versus class overlapping: an analysis of a learning system behavior. In *MICAI 2004: Advances in Artificial Intelligence*, pages 312–321. Springer, 2004.

[76] F. Provost and T. Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distribution. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48, 1997.

[77] Q. Qiao and P. Beling. Classroom video assessment and retrieval via multiple instance learning. In *Proceedings of the 15th international conference on Artificial Intelligence in Education*, pages 272–279. Springer, 2011.

[78] G. Quellec, M. Lamard, M. D. Abràmoff, E. Decencière, B. Lay, A. Erginay, B. Cochener, and G. Cazuguel. A multiple-instance learning framework for diabetic retinopathy screening. *Medical Image Analysis*, 16(6):1228–1240, 2012.

[79] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[80] J.R. Quinlan. *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[81] E. Ramentol, Y. Caballero, R. Bello, and F. Herrera. SMOTE-RSB*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. *Knowledge and Information Systems*, 33(2):245–265, 2012.

[82] J. Ramon and L. De Raedt. Multi instance neural networks. In *Attribute-Value and Relational Learning: Crossing the Boundaries.*, pages 53–60, 2000.

[83] S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, pages 697–704. ACM Press, 2005.

[84] S. Ray and D. Page. Multiple instance regression. In *Machine Learning-International Workshop*, pages 425–432, 2001.

[85] P. Reutemann. *Development of a Propositionalization Toolbox*. Master's thesis, Albert Ludwigs University of Freiburg, Germany, 2004.

[86] P. Reutemann, B. Pfahringer, and E. Frank. A toolbox for learning from relational data with propositional and multi-instance learners. In G. Webb and X. Yu, editors, *AI 2004: Advances in Artificial Intelligence*, volume 3339 of *Lecture Notes in Computer Science*, pages 421–434. Springer Berlin / Heidelberg, 2005.

[87] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart Retrieval System-Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[88] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[89] G. Ruffo. *Learning Single and Multiple Instance Decision Trees for Computer Security Applications*. Ph.D. dissertation, University of Turin, 2000.

[90] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[91] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[92] R.E. Schapire and Y. Singer. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[93] R.E. Schapire, Y. Singer, and A. Singhal. Boosting and rocchio applied to text filtering. In *Proc. SIGIR 1998: the 21st Annual Internat. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 215–223, 1998.

[94] S. Scott, J. Zhang, and J. Brown. On generalized multiple-instance learning. *International Journal of Computational Intelligence and Applications*, 5(1):21–35, 2005.

[95] F. Sebastiani. Machine learning in automated text categorization. *ACM Computation Survey*, 34(1):1–47, 2002.

[96] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. RUSBoost: a hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(1):185–197, 2010.

[97] P. Simard, Y. LeCun, and J. S. Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems 5 (NIPS Conference)*, pages 50–58, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[98] P. Simard, Y. Lecun, B. Victorri, and J. Denker. Tangent prop - a formalism for specifying selected invariances in an adaptive network. In *Advances in Neural Information Processing Systems 4 (NIPS Conference)*, volume 4, pages 895–903, 1992. Bw71y Times Cited:29 Cited References Count:0 Advances in Neural Information Processing Systems.

[99] M. Sokolova, N. Japkowicz, and S. Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. *AI 2006: Advances in Artificial Intelligence*, page 1015–1021, 2006.

[100] X. Song, L. C. Jiao, S. Yang, X. Zhang, and F. Shang. Sparse coding and classifier ensemble based multi-instance learning for image categorization. *Signal Processing*, 93(1):1–11, 2013.

[101] L. Sun, Y. Lu, K. Yang, and S. Li. ECG analysis using multiple instance learning for myocardial infarction detection. *IEEE transactions on biomedical engineering*, 59(12):3348–3356, 2012.

[102] Y. Sun, A. K. C. Wong, and M. S. Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[103] Q. Tao and S. Scott. A faster algorithm for generalized multiple-instance learning. In *Proceedings of the Seventeenth Annual FLAIRS Conference*, 2004.

[104] Q. Tao, S. Scott, N. V. Vinodchandran, and T. T. Osugi. SVM-based generalized multiple-instance learning via approximate box counting. In *In Proceedings of the Twenty-First International Conference on Machine Learning*, pages 779–806. Morgan Kaufmann, 2004.

[105] Q. Tao, S. Scott, N. V. Vinodchandran, T. T. Osugi, and B. Mueller. An extended kernel for generalized multiple-instance learning. In *16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004.*, pages 272–277, 2004.

[106] D. S. Tarragó, C. Cornelis, R. Bello, and F. Herrera. A multi-instance learning wrapper based on the rocchio classifier for web index recommendation. *Knowledge-Based Systems*, 2014.

[107] D.M.J. Tax, E. Hendriks, M.F. Valstar, and M. Pantic. The detection of concept frames using clustering multi-instance learning. In *20th International Conference on Pattern Recognition (ICPR)*, pages 2917–2920, 2010.

[108] R. Teramoto and H. Kashima. Prediction of protein-ligand binding affinities using multiple instance learning. *Journal of Molecular Graphics and Modelling*, 29(3):492–497, 2010.

[109] K. Wagstaff and T. Lane. Salience assignment for multiple-instance regression. In *Proceedings of the International Conference on Machine Learning 2007 Workshop on Constrained Optimization and Structured Output Spaces.*, 2007.

[110] J. Wang and J-D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 1119–1126, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[111] S. Wang, M. T. McKenna, T. B. Nguyen, J. E. Burns, N. Petrick, B. Sahiner, and R. M. Summers. Seeing is believing: Video classification for computed tomographic colonography using multiple-instance learning. *IEEE transactions on medical imaging*, 31(5):1141–1153, 2012.

[112] S. Wang and X. Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 324–331. IEEE, 2009.

[113] X. Wang, S. Matwin, N. Japkowicz, and X. Liu. Cost-sensitive boosting algorithms for imbalanced multi-instance datasets. In *Advances in Artificial Intelligence*, pages 174–186. Springer, 2013.

[114] Z. Wang, V. Radosavljevic, B. Han, and Z. Obradovic. Aerosol optical depth prediction from satellite observations by multiple instance regression. In *Proceedings of the SIAM International Conference on Data Mining*, pages 165–176. SIAM, 2008.

[115] N. Weidmann, E. Frank, and B. Pfahringer. A two-level learning method for generalized multi-instance problems. In *Proceedings 14th European Conference on Machine Learning*, volume 2837, pages 468–479, Cavtat-Dubrovnik, Croatia, 2003. Springer.

[116] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83, 1945.

[117] I. H Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[118] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. *Tech. Rep. SFI-TR-95-02-010, Santa Fe Institute*, 1995.

[119] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[120] X. Xu. *Statistical Learning in Multiple Instance Problems*. Master thesis, University of Waikato, Hamilton, New Zealand, June 2003.

[121] X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. In *Proc. of the PacificAsia Conf. on Knowledge Discovery and Data Mining*, pages 272–281. Springer-Verlag, 2004.

[122] W. Yang, Y. Gao, and L. Cao. TRASMIL: a local anomaly detection framework based on trajectory segmentation and multi-instance learning. *Computer Vision and Image Understanding*, 117(10):1273–1286, 2013.

[123] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML-97*, pages 412–420, Nashville, US, 1997.

[124] A. Zafra, E.L. Gibaja, and S. Ventura. Multiple instance learning with multiple objective genetic programming for web mining. *Applied Soft Computing*, 11(1):93–102, 2011.

[125] A. Zafra, C. Romero, S. Ventura, and E. Herrera-Viedma. Multi-instance genetic programming for web index recommendation. *Expert Systems with Applications*, 36(9):11470–11479, 2009.

[126] D. Zhang, F. Wang, Z. Shi, and C. Zhang. Interactive localized content based image retrieval with multiple-instance active learning. *Pattern Recognition*, 43(2):478–484, 2010.

[127] K. Zhang and H. Song. Real-time visual tracking via online weighted multiple instance learning. *Pattern Recognition*, 46(1):397–411, 2013.

[128] M.L. Zhang and Z.H. Zhou. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, 31(1):47–68, 2009.

[129] Q. Zhang and S. A. Goldman. EM-DD: an improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems 14 (NIPS Conference)*, volume 1-2, pages 1073–1080, 2001.

[130] T. Zhang, S. Liu, C. Xu, and H. Lu. M4L: maximum margin multi-instance multi-cluster learning for scene modeling. *Pattern Recognition*, 46(10):2711–2723, 2013.

[131] Z.-H. Zhou and X.-Y. Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.

[132] Z.H. Zhou, K. Jiang, and M. Li. Multi-instance learning based web mining. *Applied Intelligence*, 22:135–147, 2005.

[133] Z.H. Zhou, Y.-Y. Sun, and Y.-F. Li. Multi-instance learning by treating instances as non-I.I.D. samples. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1249–1256. ACM, 2009.

[134] Z.H. Zhou and M.L. Zhang. Multi-instance multi-label learning with application to scene classification. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pages 1609–1616. MIT Press, 2006.

[135] Z.H. Zhou and M.L. Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowledge and Information Systems*, 11(2):155–170, 2007.

[136] Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320, 2012. 00044.

[137] S. Zhu and X. Tan. A novel automatic image annotation method based on multi-instance learning. *Procedia Engineering*, 15:3439–3444, 2011.

[138] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the strength pareto evolutionary algorithm. Technical report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.