Universidad Central “Marta Abreu” de Las Villas

Facultad de Matemática, Física y Computación

Licenciatura en Ciencia de la Computación

TRABAJO DE DIPLOMA

Desarrollo de clasificadores multinstancia para aplicaciones textuales basados en la fórmula de Rocchio

Autor: Carlos Arturo Díaz Figueredo

Tutor: Dr. Dánel Sánchez Tarragó

Santa Clara

2015

Agradecimientos

Detrás de cada trabajo hay un autor y muchas manos, por lo que agradecer a todos los que hacen posible la realización de los mismos sin dejar de mencionar a alguien es una tarea tan complicada como realizarlos. Por este motivo quiero agradecer particularmente a:

A mi madre, por ser padre y madre, por estar en las buenas y en las malas, y por enseñarme todo lo que sé. Por enseñarme respeto a los demás a pesar de las discrepancias, y sobre todo, por estar ahí para cada duda.

A abuelo y abuela, que a pesar de no saber muchas veces siquiera de que iba este trabajo, se mantenían interesados por el tema y siempre al tanto de su desarrollo; a ti "viejo", ¡muchas gracias!

Un agradecimiento especial para mi hermana Claudia que aunque a veces no lo crea, es alguien muy importante en mi vida. Gracias además por revisar y corregirme la tesis.

A mi novia Liana, por estar siempre disponible para darme ese apoyo necesario en los momentos difíciles y estresantes.

A mi tutor por estar disponible ante cada duda o cada preocupación. Gracias además por guiarme por el camino de la ciencia y enseñarme respeto y honestidad hacia ella.

A todos aquellos buenos profesores y amigos que con su ejemplo formaron de mí el hombre que soy.

Finalmente, no puedo dejar de agradecer a todas aquellas personas que de una forma u otra tuvieron que ver con este trabajo, y a los que no tuvieron nada que ver, !gracias también!

Resumen

La clasificación multinstancia, como parte del aprendizaje automático, tiene como objetivo construir a partir de un conjunto de ejemplos, un modelo matemático que permita clasificar objetos descritos por múltiples vectores de atributos. Específicamente, la clasificación textual es una tarea de la clasificación multinstancia, donde la idea es asignar etiquetas semánticas a los documentos. La rama de la clasificación textual es ampliamente utilizada en un sinnúmero de campos de aplicación.

Dentro de los algoritmos destacados en el área de la clasificación textual se encuentra el algoritmo clasificador simpleinstancia de Rocchio. Recientemente fue publicado el algoritmo clasificador MIRocchio que adecua la fórmula de Rocchio al enfoque multinstancia, con buenos resultados en el área de clasificación textual. Sin embargo este algoritmo presenta ciertas limitaciones durante el proceso de aprendizaje. Estas limitantes se analizan en profundidad junto con el funcionamiento de este algoritmo.

Con el objetivo de mitigar estas limitaciones el presente trabajo propone tres variantes del algoritmo MIRocchio que intentan mejorar tanto la eficiencia como la eficacia del mismo. Se propone además en esta tesis una nueva hipótesis basada en la frontera de decisión entre clases, la cual se incorporara en el diseño de dos de las propuestas. Todos los algoritmos obtenidos en este trabajo están enfocados específicamente al área de la clasificación textual.

La validez de las hipótesis propuestas durante esta investigación se comprueba experimentalmente para los problemas de la recomendación de paginas web indices y TREC9. Los experimentos arrojaron resultados favorables para dos de los tres algoritmos propuestos, siendo capaces de mejorar el desempeño del clasificador MIRocchio y siendo competitivos con algoritmos del estado del arte.

Abstract

The multinstance classification, like part of the automatic learning, has as objective to build a mathematical model that allows to classify objects described by multiple vectors of attributes starting from a group of examples. Specifically, the textual classification is a task of the multinstance classification, where the idea is assign semantic labels to the documents. The branch of the textual classification is broadly used in many application fields.

One of the most recognized algorithms in the area of the textual classification is the singleinstance classifier of Rocchio. Recently was published a classifier algorithm baptized as MIRocchio that adapts the formula of Rocchio to the multinstance learning with good results in the area of textual classification. This algorithm presents certain limitations during the learning process. These restrictive ones are analyzed in depth together with the operation of this algorithm.

With the objective of mitigating these limitations the present work proposes three variants of the algorithm MIRocchio that promise to improve as much the efficiency as the effectiveness of the same one. Is also intends in this thesis a new hypothesis based on the frontier of decision among classes, which incorporated in the design of two of the proposals. All the algorithms obtained in this work are focused specifically to the area of the textual classification.

The validity of the hypotheses proposed during this investigation is proven experimentally for the problems TREC9 and WIR. The experiments threw favorable results for two of the three proposed algorithms, being able to improve the acting of the classifier MIRocchio and being competent also with algorithms of the state of the art.

Índice general

Índice de figuras

Índice de cuadros

List of Algorithms

Introducción

Planteamiento

El aprendizaje multinstancia es una rama muy joven del aprendizaje automático que ha ganado su interés por su capacidad para modelar problemas de aprendizaje caracterizados por la presencia de ambigüedad en sus datos. Las técnicas relacionadas con este tipo de aprendizaje han crecido significativamente dado el número y la variedad de campos de aplicación que se han beneficiado con su uso. Algunos ejemplos de la aplicación reciente de estas técnicas son la anotación automática de imágenes, recuperación de imágenes y/o videos basados en su contenido, seguimiento visual, bioinformática, diagnóstico médico asistido por ordenador, reconocimiento de emociones, finanzas, detección de anomalías, detección de minas terrestres, entre otras.

La clasificación multinstancia es un tipo de aprendizaje automático en el que cada ejemplo (bolsa) está asociado a múltiples descripciones (instancias) y una etiqueta de clase. Esta se utiliza en problemas donde los ejemplos pueden ser descritos desde múltiples puntos de vista. Este tipo de aprendizaje se utiliza en problemas donde predomina la ambigüedad en los datos. La clasificación multinstancia tiene también fuertes aplicaciones en la categorización de texto, donde la tarea es asignar etiquetas semánticas a los documentos. En este tipo de problemas los documentos se presentan como un conjunto de fragmentos, de los cuales se puede obtener la información necesaria para asignar a este una clasificación. La rama de la clasificación textual es ampliamente utilizada en un sinnúmero de campos de aplicación, que van desde la recuperación de información hasta la extracción de conocimiento de la Internet, etc.

Dentro de los algoritmos destacados en el área de la clasificación textual se encuentra el algoritmo clasificador simpleinstancia de Rocchio, propuesto por (Rocchio 1971). Este algoritmo básicamente construye durante el proceso de aprendizaje un perfil para cada clase tomando como base los ejemplos (instancias) de cada una de estas. El perfil de clase es una instancia prototípica que trata de capturar la importancia que cada término textual tiene

para los ejemplos relevantes de la clase, más allá de la que pueda tener para los ejemplos no relevantes. El clasificador de ROCCHIO en la etapa de clasificación asigna a un nuevo ejemplo la clase del perfil que más se le asemeje.

Recientemente fue publicado por (Tarragó 2014) un algoritmo clasificador bautizado como MIROCCHIO. Este algoritmo adecua la fórmula de ROCCHIO al escenario de la clasificación multinstancia con buenos resultados en el área de clasificación textual. MIROCCHIO, al igual que el clasificador de ROCCHIO basa su funcionamiento en la creación de perfiles de clase a nivel de instancias, por lo que necesita obtener conjuntos de instancias etiquetadas, para lo cual realiza un proceso de proposicionalización[1]. En todo proceso de imputación de etiquetas de clase a las instancias es introducido un sesgo no conveniente en el aprendizaje debido a que las etiquetas son asignadas a las instancias asumiendo determinada hipótesis multinstancia que no siempre se cumple. Este sesgo introducido afecta por supuesto el resultado de la clasificación, sin embargo si pudiese evitarse este proceso de asignación de etiquetas de clases a las instancias, se pudiesen mitigar de cierto modo los errores en el aprendizaje.

Por este motivo el presente estudio propone una nueva hipótesis de trabajo para el desarrollo de algoritmos de clasificación multinstancia que elimina esta limitante. Para evitar el proceso de proposicionalización se plantea la creación de perfiles de clase a nivel de bolsa, obteniendo como perfil una bolsa prototípica en lugar de una instancia. Esta nueva hipótesis permite a aquellos clasificadores que la implementen utilizar directamente las bolsas y evitar el proceso de etiquetar las instancias. Las bolsas que conformarían los perfiles de clases estarían constituidas por n instancias generadas a partir de submuestrear aleatoriamente las bolsas de ejemplo. A modo de resumen esta hipótesis plantea que la utilización de la fórmula de ROCCHIO para la creación de los perfiles de clase a nivel de bolsa resultará en un aumento de la calidad de la clasificación sobre aquellos clasificadores que mantienen los perfiles de clase a nivel de instancia.

Otro punto importante es que el algoritmo MIROCCHIO concibe el espacio generado por las bolsas como un espacio linealmente separable. Esto significa que para problemas binarios (solo dos clases), trazando el hiperplano adecuado, el espacio conformado por todas las bolsas sería dividido en dos subespacios, cada uno correspondiente a cada clase. Este algoritmo utiliza un hiperplano como frontera de decisión entre la clase positiva y la negativa. En la Figura (0.1) se aprecia el resultado de aplicar un hiperplano como frontera

[1] El proceso de proposicionalización es aquel en el que las instancias de una bolsa son etiquetadas siguiendo determinada hipótesis multinstancia

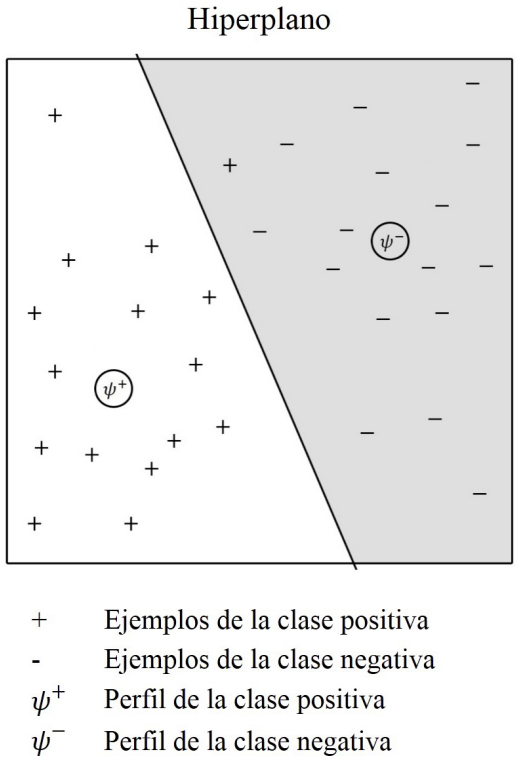

Figura 0.1: Frontera de decisión Hiperplano sobre un problema de clasificación binaria

de decisión entre clases sobre un problema de dos clases, tal como lo hace el MIROCCHIO. En este caso las bolsas positivas y negativas son representadas como puntos en un plano para su mejor comprensión y el hiperplano como una línea que las separa.

Sin embargo existen problemas donde aparecen más de dos clases, los cuales son transformados en problemas binarios manteniendo la clase de interés como positiva y concibiendo las demás como negativa. Estos problemas originalmente multiclase (más de dos clases) son por su naturaleza no linealmente separables, por lo que el uso de un hiperplano para separar las dos clases del nuevo problema binario no resultaría conveniente. En estos casos en los que los ejemplos de la clase positiva se encuentran rodeados por aquellos de las clases negativas es considerable representar la frontera de decisión como una hiperesfera que englobe las bolsas positivas excluyendo las negativas.

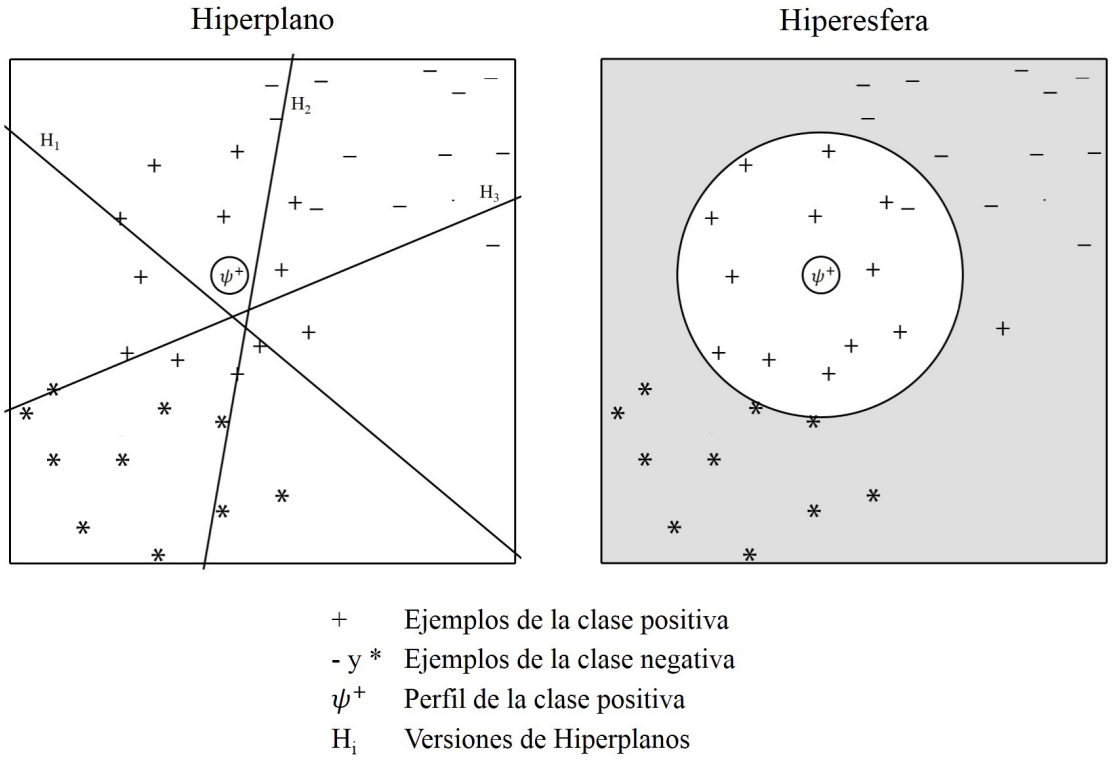

Figura 0.2: Fronteras de decisión Hiperplano e Hiperesfera sobre un problema de tres clases transformado en un problema de clasificación binaria

Siguiendo esta idea se presenta una nueva hipótesis de investigación que plantea que los algoritmos clasificadores multinstancia basados en ROCCHIO, que utilicen una hiperesfera como frontera de decisión entre clases, obtendrán mejores resultados sobre aquellos que implementen el tradicional hiperplano. En la figura (0.2) se observa un problema de tres clases transformado en un problema de clasificación binaria. Para este problema, utilizando un hiperplano como frontera de decisión entre clases, cada H_i propuesto introduce grandes errores en la clasificación. En cambio, se observa en la imagen que la utilización de una hiperesfera como frontera de decisión entre clases conlleva a una disminución considerable del error.

Finalmente, la obtención de nuevos clasificadores que superen el desempeño del MIROC-

CHIO y sean competentes además con los algoritmos del estado del arte constituiría un aporte significativo a la rama de la clasificación multinstancia, específicamente al área de la clasificación textual.

Objetivo general

Desarrollar clasificadores multinstancia utilizando como base la fórmula de ROCCHIO destinados principalmente al área de la clasificación textual.

Objetivos específicos

1. Desarrollar un algoritmo con perfiles de clase a nivel de instancia e hiperesfera como frontera de decisión entre clases.
2. Desarrollar un algoritmo con perfiles de clase a nivel de bolsa e hiperplano como frontera de decisión entre clases.
3. Desarrollar un algoritmo con perfiles de clase a nivel de bolsa e hiperesfera como frontera de decisión entre clases.
4. A partir de estos algoritmos desarrollados verificar la validez de las hipótesis planteadas.

Estructura

El presente trabajo está dividido en tres capítulos. El Capítulo 1 está orientado a introducir los conceptos y teorías necesarios para el desarrollo de esta tesis. Este comienza con una introducción al aprendizaje automático y luego al aprendizaje multinstancia, explicando los fundamentos de la clasificación multinstancia y su importancia, así como las aplicaciones reales de la misma; se describen además algunas de las hipótesis multinstancia más reconocidas que sirven de base para diversos algoritmos clasificadores del estado del arte; se expone además un sistema de categorización para los algoritmos de clasificación multinstancia. Este capítulo dedica también una sección a introducir el tema de la clasificación textual desde sus enfoques mono y multinstancia, haciendo énfasis en la representación

de los datos y los principales algoritmos destinados a la solución de estos problemas respectivamente, siendo descritos los clasificadores Rocchio y MIRocchio como principales exponentes para esta investigación.

El Capítulo 2, destinado a satisfacer los objetivos propuestos, presenta las propuestas diseñadas para implementar las hipótesis planteadas. Seguidamente se introduce el proceso de diseño del primer algoritmo propuesto, se expone la hipótesis multinstancia que implementa el clasificador y se explica minuciosamente como se realizan los ajustes de los diversos parámetros que intervienen en su funcionamiento. Para finalizar el capítulo de introducen las propuestas de clasificadores basados en perfiles de clase a nivel de bolsa y se describe el funcionamiento de estos algoritmos de manera general. Luego se presentan los clasificadores BPMR y BEMR como los segundos algoritmos propuestos y se explican al igual que en el caso anterior las hipótesis que intervienen en sus diseños, así como el proceso de cálculo de los parámetros de los mismos, etc.

Finalmente en el Capítulo 3, dedicado a la validación de los resultados obtenidos, se describe el marco experimental; dígase explicación de los experimentos, descripción de los conjuntos de datos, así como los algoritmos a comparar y las métricas de desempeño utilizadas. Finalmente se presentan y discuten los resultados de las experimentaciones obtenidas, comparándolos con los resultados obtenidos primeramente con el algoritmo MIRocchio y luego con una selección de varios algoritmos del estado del arte.

1 La clasificación multinstancia para aplicaciones textuales

La clasificación multinstancia es una de las tareas del aprendizaje multinstancia, cuyo objetivo es construir un modelo a partir de las bolsas de entrenamiento con el objetivo de predecir las etiquetas de clase de nuevas bolsas a clasificar. Esta es una tarea que ha recibido gran atención debido a sus variadas aplicaciones en importantes campos de la actividad humana.

Primeramente, este capítulo tiene como principal intención abordar los temas referentes al aprendizaje automático y sus variantes.

En una segunda sección de este capítulo se exponen las bases y fundamentos del aprendizaje multinstancia y se formaliza. Para cerrar esta primera parte del capítulo, se explica la importancia del aprendizaje multinstancia y se introducen las hipótesis que serán utilizadas en el presente trabajo, así como un sistema de categorización para algoritmos de clasificación multinstancia.

Finalmente se introduce el tema de la clasificación textual en sus variantes simpleinstancia y multinstancia, haciendo hincapié en la representación de los datos y los principales algoritmos destinados a la solución de estos problemas respectivamente. En esta sección se hace también un análisis de los principios del funcionamiento del algoritmo Rocchio por su importancia para este trabajo y se expone una breve crítica al algoritmo clasificador multinstancia MIRocchio.

1.1. Aprendizaje automático

Según (Mitchell 1997), el aprendizaje automático 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 para plantear correctamente un problema

de aprendizaje, identificar estos tres elementos: 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 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 el mismo.

Los tres elementos mencionados anteriormente determinan que el proceso en sí sea un proceso de aprendizaje, y al ser este llevado a cabo por medio de una computadora explica el carácter automático del aprendizaje.

Seguidamente se exponen los tipos de aprendizaje automático y se formaliza su definición.

1.1.1. Tipos de aprendizaje automático

Varios tipos de aprendizaje automático han sido formalizados. El más utilizado y al mismo tiempo el más estudiado es el llamado aprendizaje supervisado. En este tipo de aprendizaje, los algoritmos requieren el suministro de un conjunto de ejemplos, cada uno 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 para las etiquetas de futuros ejemplos. El siguiente ejemplo ilustra el concepto de aprendizaje supervisado. "Supongamos que somos estudiantes 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."

Los datos Iris es un conjunto de referencia estándar 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 contiene el ancho y el alto de los pétalos del ejemplo correspondiente; estos elementos de los datos 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 aplicación. 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 tomando provecho de grandes fuentes de datos sin etiquetar.

Los tipos de aprendizaje descritos anteriormente trabajan con datos organizados en forma de una tabla plana con las instancias en las filas y los atributos como columnas. Los problemas cuyos datos están organizados de esta forma reciben el nombre de problemas de aprendizaje atributo-valor o aprendizaje proposicional.

1.1.2. Formalización del aprendizaje supervisado

En esta sección se formaliza el concepto del aprendizaje supervisado descrito anteriormente por el ejemplo Iris. Una instancia es un par (x, y), donde $x = \langle x_1, x_2, \ldots, x_N \rangle \in X$ es un vector de N atributos y $y \in Y$ es la etiqueta de decisión de la instancia. Los atributos y las etiquetas de decisión son típicamente números reales (atributos numéricos) o bien conjuntos de nombres de un dominio específicos (atributos nominales). El espacio N dimensional X de donde x toma valores se conoce como espacio de instancias o espacio de atributos, y Y es el conjunto de etiquetas de decisión.

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, mas 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 terminología 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.

A continuación se introducen los conceptos de aprendizaje y clasificación multinstancia debido a su importancia para esta tesis.

1.2. Aprendizaje multinstancia

El enfoque multinstancia ha adquirido mayor popularidad en el aprendizaje supervisado, sin embargo ha sido aplicado también a distintos tipos de aprendizaje proposicional. El aprendizaje multinstancia es una variante del aprendizaje proposicional en el que cada ejemplo no está descrito ya por un único vector, sino que está descrito por muchos vectores atributo-valor.

Como se describió en la sección anterior, en el aprendizaje supervisado estándar cada ejemplo es una instancia que consiste en un vector de atributos, y una etiqueta de decisión o clase. La tarea es aprender una regla que prediga la clase de un ejemplo dado el vector de atributos que conforma la instancia referente a este. Mientras que, en el aprendizaje multinstancia cada ejemplo esta descrito como una bolsa de instancias. Cada bolsa tiene asociada una etiqueta de decisión o clase, pero las instancias contenidas en esta no presentan etiquetas. El objetivo del aprendizaje multinstancia es construir un modelo que basado en las bolsas dadas como ejemplo de entrenamiento, pueda predecir con precisión la etiqueta o clase de decisión de las futuras bolsas.

El siguiente ejemplo llamado el problema del cerrajero simple ayudará a entender mejor este concepto: "Imagine que hay una puerta cerrada con cerrojo, y tenemos M llaveros, cada uno 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 considerado una generalización del aprendizaje proposicional, ya que en cualquier problema del aprendizaje proposicional tradicional puede considerarse cada ejemplo como una bolsa que contiene una sola instancia. A partir de la introducción del aprendizaje multinstancia podemos llamar aprendizaje monoinstancia o simpleinstancia al aprendizaje proposicional tradicional.

1.2.1. Clasificación multinstancia

La clasificación multinstancia es una de las tareas del aprendizaje multinstancia. Específicamente, en la clasificación multinstancia cada ejemplo (bolsa) está compuesto por múltiples instancias y una etiqueta de tipo nominal que representa la clase o categoría de la bolsa, en contraposición con la clasificación simpleinstancia, donde cada ejemplo está descrito por un vector atributo-valor. El objetivo de la calificación multinstancia 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.

1.2.2. Formalización de la clasificación multinstancia

En la clasificación multinstancia cada ejemplo esta descrito como un par (x, y), donde $X = \{x_1, x_2, \ldots, x_T\} \in \mathbb{N}^{\mathcal{X}}$ es un multiconjunto (bolsa) de T instancias y $y \in Y$ es la etiqueta de clase de la bolsa. Se dice que es un multiconjunto y no un conjunto porque el multiconjunto permite tener instancias repetidas a diferencia de este último. En notación matemática, $X \in N^{\mathcal{X}}$ significa que X es un multiconjunto de elementos de X, donde $\mathbb{N}$ representa el conjunto de los números naturales. Las instancias $x_i \in X, i = 1 \ldots T$ son vectores de N atributos que, a diferencia del aprendizaje supervisado estándar, carecen de etiquetas de decisión. El conjunto X es un espacio N-dimensional formado por el producto vectorial de los N atributos que describen a las instancias, y 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., $Y = \{+, -\}$, pero, en general, Y puede tener más de dos elementos. En este caso se dice que el problema de clasificación es multiclase.

La tarea de la clasificación multinstancia es encontrar $f(X) = y$, basado en un conjunto de instancias de entrenamiento $D = \{f(X_1, y_1), \cdots, (X_M, y_M)\}$, donde $f : \mathbb{N}^{\mathcal{X}} \to Y$ 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 $f(x)$.

1.2.3. Importancia de la clasificación multinstancia

El aprendizaje multinstancia, a pesar de ser una rama muy joven del aprendizaje automático, debido a su capacidad para modelar problemas de aprendizaje caracterizados por la presencia de ambigüedad en sus datos, ha despertado un gran interés en los investigadores y profesionales de la rama. 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 variedad 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, recuperación de imágenes y/o videos basados en su contenido, seguimiento visual, bioinformática, diagnóstico médico asistido por ordenador, reconocimiento de emociones, finanzas, detección de anomalías, detección de minas terrestres, entre otras.

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 fragmentos más pequeños. Se pueden extraer atributos tales como las frecuencias de ocurrencia de las palabras en cada pasaje para formar las instancias. Posteriormente en la Sección (1.3) se aborda en detalle el tema referente a la calificación textual desde sus diferentes enfoques.

En la siguiente sección se introduce el concepto de hipótesis multinstancia y se describen varias de estas hipótesis del estado del arte. Se describe también, por su importancia para el presente trabajo, la hipótesis multinstancia basada en la proporción de instancias positivas presentada por (Tarragó 2014).

1.2.4. Hipótesis del aprendizaje multinstancia

De manera general los problemas enfocados desde la perspectiva de la clasificación multinstancia presentan una mayor complejidad que los problemas del aprendizaje supervisado estándar dado que el espacio de hipótesis de los multinstancia es mucho mayor.

Esta afirmación se puede inferir de la definición de ambos tipos de aprendizaje. En el aprendizaje supervisado estándar $f_{SE} : X \to Y$, mientras que en el multinstancia $f_{MI} : \mathbb{N}^{\mathcal{X}} \to Y$, teniendo en cuenta que $X \subseteq \mathbb{N}^{\mathcal{X}}$, se tiene entonces que $f_{SE} \subseteq f_{MI}$. Con el

objetivo de acotar la búsqueda de la función objetivo f_{MI} se recurre generalmente a asumir la existencia de relaciones entre la constitución de las instancias y la etiqueta de las bolsas. A esta relación particular que se asume para dar solución al problema multinstancia se denomina hipótesis multinstancia.

Existen una gran variedad de hipótesis multinstancia que han sido propuestas, algunas de estas se plantean de forma explícita en los algoritmos que las implementan mientras que otras permanecen implícitas en diferentes métodos de solución multinstancia. En ambos casos la hipótesis multinstancia es un pilar fundamental para la comprensión del algoritmo de aprendizaje. Entre las diferentes hipótesis existentes en la literatura se pueden discernir dos grandes grupos: el de las hipótesis basadas en instancias y el de aquellas basadas en metadatos.

La principal características de las hipótesis basadas en instancias es que parten de la idea de que las instancias poseen de manera implícita sus propias etiquetas de clase determinadas por algún proceso $g : X \to C$, donde C es el conjunto de conceptos subyacentes a las instancias. Las hipótesis en esta categoría interpretan la hipótesis multinstancia como una composición de funciones $f_{MI} = g \circ h$, donde g determina la etiqueta de cada instancia y la función h determina la etiqueta de la bolsa en función de las etiquetas de las instancias. La principal diferencia entre las hipótesis de este grupo radica en la función h que utilicen para predecir la etiqueta de la bolsa. Los algoritmos que implementan estas hipótesis son conocidos generalmente como algoritmos de dos pasos: primeramente se clasifican las instancias y luego se clasifica la bolsa.

Por otro lado, 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.

A continuación se describen con más detalle algunos ejemplos importantes de los distintos tipos de hipótesis multinstancia mencionados.

1.2.4.1. Hipótesis estándar

Esta hipótesis recibe el nombre de hipótesis multinstancia estándar debido a que fue la primera hipótesis que se empleó para definir el aprendizaje multinstancia. Esta tuvo una

gran influencia en el planteamiento de los problemas multinstancia y el desarrollo de los primeros métodos de solución.

La hipótesis estándar establece que una bolsa será positiva si y solo si contiene alguna 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. Observe que la hipótesis estándar es asimétrica: si las etiquetas positiva y negativa se invierten, la hipótesis cobra un significado diferente. Por tanto, cuando se aplica esta hipótesis, es importante tener bien claro cuál debe ser la etiqueta positiva.

La hipótesis estándar ha sido usada en un número importante de algoritmos multinstancia como por ejemplo en los algoritmos APR (Dietterich et al. 1997), DENSIDAD DIVERSA (Maron and Lozano-Pérez 1998), EM-DD (Zhang and Goldman 2001), ÁRBOLES DE DECISIÓN (Chevaleyre and Zucker 2001, Blockeel et al. 2005), REGLAS DE DECISIÓN (Chevaleyre and Zucker 2001), AMPLIFICACIÓN (Auer and Ortner 2004) REDES NEURONALES (Ramon and De Raedt 2000), MÁQUINAS DE SOPORTE VECTORIAL (Andrews et al. 2002), etc.

1.2.4.2. Hipótesis colectiva

La hipótesis colectiva es una hipótesis multinstancia donde, al contrario de la hipótesis estándar, todas las instancias en una bolsa contribuyen por igual a la etiqueta de la bolsa (Xu 2003). Un ejemplo en (Foulds 2008) ilustra la motivación de esta hipótesis. Teniendo en cuenta que en este caso el objetivo es 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 una composición de elementos, donde que cada uno contribuya a la probabilidad de que la imagen sea una escena playera, dígase arena, agua, cielo, palmeras, pelotas de playa, castillos de arena, entre otros.

Esta hipótesis está motivada por una visión de la naturaleza de las bolsas basada en la teoría de las probabilidades. Bajo esta 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.

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.

Algunos de los algoritmos que implementan la hipótesis multinstancia colectiva son MIWRAPPER (Frank and Xu 2003), IFLIW (Foulds 2008), MILR (Xu and Frank 2004) y el

algoritmo amplificador de Xu & Frank (Xu and Frank 2004).

1.2.4.3. Hipótesis basadas en distancia

Existen varias hipótesis multinstancia basadas en distancia entre objetivos. Algunas miden la distancia entre una instancia objetivo y una bolsa, mientras que otras miden la distancia entre una bolsa objetivo y otra. Un ejemplo de hipótesis multinstancia basada en distancia entre instancias y bolsas es la utilizada por el algoritmo MILES, donde se toma la distancia entre la instancia objetivo y su instancia más cercana en la bolsa para luego ser utilizada como criterio de semejanza entre estas.

Sin embargo, las hipótesis multinstancia basadas en distancia entre bolsas asumen que las etiquetas de las bolsas están determinadas de alguna manera por las distancias entre estas. Para cuantificar la distancia entre dos bolsas es comúnmente utilizada la distancia de Hausdorff. La distancia tradicional de Hausdorff entre dos conjuntos de puntos (bolsas) $X = \{x_1, \ldots, x_{n_X}\}$y $Z = \left\{z_1, \ldots, z_{n_{z'}}\right\}$ se puede definir como la mayor distancia entre un punto X y su más cercano punto en Z o viceversa. Formalmente la distancia (máxima) de Hausdorff se define como

$$H_{max}\left(X, Z\right) = \text{máx}\left\{h\left(X, Z\right), h(X, Z)\right\} \tag{1.1}$$

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 Euclidiana.

La distancia mínima de Hausdorff fue propuesta por (Wang and Zucker 2000) en el contexto de una actualización multinstancia del vecino más cercano. En esta variante se remplaza la función h por h_1, donde

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

La distancia mínima de Hausdorff es simplemente la distancia más corta entre un punto en X y un punto en Z, esta se puede fórmular como

$$H_{\text{mín}}(X, Z) = \min_{x \in X, z \in Z} \|x - z\| \tag{1.2}$$

Algunos de los algoritmos que implementan hipótesis basadas en distancias son GMIL, BARTMIP y los algoritmos basados en vecinos más cercanos.

1.2.4.4. Hipótesis basada en umbral

Según la hipótesis multinstancia estándar, una bolsa es positiva si contiene al menos una instancia positiva, en otro caso es negativa. De manera concisa, una hipótesis multinstancia basada en umbral establece que una bolsa es positiva si contiene al menos cierto número de instancias positivas, o es negativa en caso contrario.

1.2.4.5. 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 y una instancia representa un fragmento del documento, 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.

Es muy probable que para algunos documentos no baste con la existencia de un solo fragmento relevante, y se necesite más de un fragmento relevante para hacer que el documento sea relevante. Este razonamiento nos remite a la hipótesis basada en umbral anteriormente mencionada, 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 variar de un documento a otro.

Teniendo en cuenta estos inconvenientes aparece una nueva hipótesis (Tarragó 2014) que se obtiene como variación de la hipótesis basada en umbral, llamada hipótesis basada en proporción de instancias positivas. Esta nueva hipótesis 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

manera, aunque difiera el número de fragmentos en cada documento, es más probable que la proporción de fragmentos necesarios para hacer relevantes a los documentos se mantenga aproximadamente constante.

De manera formal la hipótesis multinstancia basada en proporción de instancias positivas 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 caso contrario es negativa.

1.2.5. Algoritmos de clasificación multinstancia

El número de algoritmos de clasificación multinstancia que ha sido desarrollado desde el nacimiento del aprendizaje multinstancia en 1997 hasta la actualidad es sumamente grande, por lo que se torna difícil y engorroso mencionarlos a todos. Sin embargo, es posible agruparlos en distintas categorías según elementos claves en sus respectivos diseños. A continuación se presentan tres grandes grupos para categorízar los algoritmos de clasificación multinstancia propuestos por (Tarragó 2014), y se nombran algunos de los exponentes más representativos de cada categoría.

1.2.5.1. 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), los algoritmos basados en el enfoque de densidad diversa y el algoritmo GMIL basado en resultados teóricos del reconocimiento de patrones geométricos.

1.2.5.2. Algoritmos tradicionales adaptados al aprendizaje multinstancia

Es frecuente que algunos métodos de aprendizaje multinstancia sean diseñados como resultado de la modificación de un algoritmo monoinstancia. La literatura provee muchos algoritmos de aprendizaje monoinstancia que están bien sustentados tanto teórica como empíricamente, los cuales en varias ocasiones proveen bases sólidas a partir de las cuales fórmular algoritmos multinstancia. Frecuentemente transformaciones menores en el algoritmo permiten a un clasificador monoinstancia manejar problemas multinstancia, ahorrando también así tiempo de diseño. Entre los algoritmos de aprendizaje supervisado más famosos que han sido adaptados al escenario multinstancia se encuentran el $k - vecinos$ más cercano, los árboles de decisión, las máquinas de soporte vectorial, la regresión logística y la amplificación, entre otros.

1.2.5.3. Algoritmos envoltorios

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 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.

Para los conceptos multinstancia basados en instancias, la hipótesis es que las instancias son 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.

1.3. Clasificación textual

La clasificación textual es aquella donde la tarea es asignar etiquetas semánticas a los documentos según su contenido. La rama de la clasificación textual es ampliamente utilizada en un sinnúmero de campos de aplicación.

Esta sección está destinada a introducir los conceptos referentes a la clasificación textual desde los enfoques simpleinstancia y multinstancia. Se describe para cada enfoque la forma de representación de los ejemplos, los algoritmos más relevantes de la literatura y se describen los clasificadores Rocchio y MIRocchio por sus respectivas influencias sobre la presente investigación.

1.3.1. Enfoque tradicional

En el enfoque tradicional o simpleinstancia cada ejemplo aparece descrito únicamente por un vector de atributos y una etiqueta de clase asociada a éste. El objetivo es aprender de un conjunto de ejemplos y crear un modelo de clasificación que luego, para un documento

X sea capaz de asignarle adecuadamente una etiqueta de clase Y. En este enfoque los documentos son considerados como bolsas de palabras llamadas instancias.

1.3.1.1. Representación bolsa de palabras (ponderación TF-IDF)

El enfoque tradicional de la clasificación textual como se mencionaba anteriormente, utiliza como ejemplos de aprendizaje un conjunto de documentos denominados bolsas. Según este enfoque cada bolsa está compuesta por un vector de atributos y una etiqueta de clase, cada elemento del vector de atributos hace referencia al peso de cada palabra (término) del documento. Estos pesos pueden ser calculados de disimiles formas, pero las más comunes son aquellas en las que intervienen las frecuencias de aparición de los términos, tanto en el documento como en la colección, denominadas $TF - IDF$ por sus siglas en inglés *Term Frequency* y *Inverse Document Frequency* respectivamente.

Seguidamente se presenta uno de los algoritmos más relevantes de la literatura destinados al área a la clasificación textual monoinstancia. Por la importancia que presenta para esta investigación el algoritmo clasificador simpleinstancia de Rocchio será analizado en profundidad en la siguiente sección.

1.3.1.2. Clasificador de Rocchio

El algoritmo Rocchio introducido por (Rocchio 1971) fue diseñado originalmente con el objetivo de recuperar información basado en la refinación interactiva de términos de búsqueda usando retroalimentación de los usuarios, para indicar al sistema si los resultado de la búsqueda eran relevante o no. Posteriormente el algoritmo fue adaptado para categorización textual (Ittner et al. 1995) y, dada su elevada eficiencia tanto en entrenamiento como en su etapa operativa, ha sido ampliamente usado en este dominio de aplicación.

Partiendo de un conjunto de datos entrenamiento el algoritmo clasificador de Rocchio construye un prototipo o perfil para cada clase. Un perfil de clase es un vector que sintetiza la descripción que se tiene de los ejemplos de entrenamiento de esta. En las aplicaciones textuales los atributos usualmente son representaciones de 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 en d, por lo general este peso es evaluado como $TF - IDF$.

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 $Y = \{+, -\}$) se tiene un perfil de clase positivo ψ^+ y otro negativo ψ^-. Empleando la fórmula de Rocchio, el atributo $i - esimo$ en los perfiles de clase positivo y negativo se calcula mediante las fórmulas siguientes respectivamente.

$$\psi_i^+ = \text{máx}\{0, \frac{1}{|R^+|} \sum_{x \in R^+} x_i - \frac{\rho^+}{|R^-|} \sum_{x \in R^-} x_i\} \tag{1.3}$$

$$\psi_i^- = \text{máx}\{0, \frac{1}{|R^-|} \sum_{x \in R^-} x_i - \frac{\rho^-}{|R^+|} \sum_{x \in R^+} x_i\} \tag{1.4}$$

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 intensidad de esta selección. Mientras mayor es el valor del parámetro, más intensa es la selección de atributos.

El perfil de clase es un vector que 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. De acuerdo a la fórmula de Rocchio, se calcula para cada atributo la importancia del término para los documentos relevantes, calculada como el valor medio en las instancias positivas, y se le sustrae la importancia del término para los documentos irrelevantes, o sea el valor medio del atributo en las instancias negativas. Si la importancia de un término textual para los documentos relevantes para la clase 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 especifico 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). De manera gráfica, el perfil de cada clase es un vector abstracto que se encuentra ubicado en el centroide del espacio generado por los ejemplos de la clase.

Una vez obtenidos los perfiles de clase ψ^c, $c \in Y$, la etiqueta de clase de una nueva instancia x se obtiene mediante la siguiente expresión

$$g(x) = \arg_{c \in Y} max S_{cos}(x, \psi^c) \tag{1.5}$$

donde

$$S_{cos}(x, z) = \frac{x * z}{|x||z|} \tag{1.6}$$

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

1.3.2. Enfoque multinstancia

A diferencia del enfoque tradicional o simpleinstancia, el enfoque multinstancia es aplicado para aquellos problemas que presentan ambigüedad en sus datos. Para estos problemas los ejemplos pueden ser descritos desde múltiples puntos de vistas, o sea que el enfoque multinstancia es una variante de la clasificación textual en el que cada ejemplo (bolsa) no está descrito ya por un único vector, sino por varios vectores atributo-valor (instancias). Según este enfoque cada documento es considerado como una bolsa asociada a una etiqueta de clase donde cada fragmento del texto constituye una instancia de la bolsa. Al igual que en el enfoque tradicional, los pesos de los atributos pueden ser calculados de diferentes maneras, pero las más comunes son aquellas en las que intervienen las frecuencias de aparición de los términos, en el documento y en la colección, $TF - IDF$ respectivamente. Una vez más el objetivo es aprender de un conjunto de ejemplos y crear un modelo de clasificación que luego, dado documento X sea capaz de asignarle adecuadamente una etiqueta de clase Y.

Seguidamente se exponen dos clasificadores multinstancia del estado del arte enfocados precisamente al área de la clasificación textual multinstancia.

1.3.2.1. Algoritmos de la literatura Fretcit-kNN y MOG3P

Fretcit-kNN propuesto por (Zhou et al. 2005) basa su funcionamiento en el algoritmo clasificador multinstancia CitationKNN (Wang and Zucker 2000) el cual se describe más adelante en la Sección (3.1.2). 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 (Zhou et al. 2005).

Para medir distancia entre dos bolsas $X = \{x_1, \ldots, x_{n_x}\}$ y $Z = \{z_1, \ldots, z_{n_z}\}$ CitationKNN usa la distancia mínima de Hausdorff definida en la fórmula (1.2). 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 como

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

donde x_i representa el termino $i-$ésimo de la instancia x. En resumen, FRETCIT-KNN es parecido al algoritmo CITATIONKNN, con la salvedad de que $\| x - z \| = fret - dist(x, z)$.

Por otro lado, MOG3P-MI es un método multiobjetivo de programación genética guiada por gramática propuesto por (Zafra et al. 2011) y diseñado específicamente para resolver el problema de recomendación de paginas web indices WIR por sus siglas en ingles (*Web index recomendation*). MOG3P-MI esta basado en el algoritmo evolutivo multiobjetivo SPEA2 (Zitzler et al. 2001), y es capaz de optimizar simultáneamente dos medidas de desempeño, específicamente la sensibilidad y la especificidad. La sensibilidad mide la proporción de ejemplos realmente positivos 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) (Zafra et al. 2011). Este algoritmo utiliza una representación vectorial clásica cuyos componentes son frecuencias de términos, y aplica en el preprocesamiento el mismo paso de selección de atributos utilizando las frecuencias absolutas.

1.3.2.2. Clasificador MIRocchio

El algoritmo clasificador MIROCCHIO (*Multi-instance* ROCCHIO por sus siglas en inglés) es una adaptación del algoritmo de ROCCHIO al enfoque multinstancia. Este algoritmo fue propuesto por (Tarragó 2014) mostrando excelentes resultados al ser comparado con otros algoritmos del estado del arte específicamente en el área de la clasificación textual. El clasificador MIROCCHIO asume como hipótesis multinstancia la hipótesis basada en la proporción de instancias positivas, que asume que el número de instancias positivas requerido para que la bolsa sea positiva es proporcional al tamaño de la bolsa. A continuación se presenta el funcionamiento del algoritmo MIROCCHIO mediante el uso de un algoritmo envoltorio que implementa la hipótesis multinstancia basada en la proporción de instancias positivas.

El algoritmo de envoltorio funciona como una interfaz entre los datos multinstancia del clasificador y el clasificador simpleinstancia subyacente (en este caso el clasificador de ROCCHIO). 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. MIROCCHIO utiliza un envoltorio de tipo basado en instancias, lo que significa que el espacio monoinstancia generado 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 se basa en la hipótesis multinstancia de proporción, pero la única información con que cuenta en el momento del aprendizaje es la etiqueta de clase de la bolsa, con la cual solo puede inferir, que cuando la bolsa es positiva, la proporción de instancias positivas de la bolsa ha superado determinado umbral y, cuando la etiqueta es negativa, que no ha superado dicho umbral. A diferencia de la hipótesis multinstancia estándar, donde una bolsa negativa implica que todas sus instancias son negativas, en una bolsa negativa pueden existir instancias positivas según la hipótesis multinstancia de proporción, siempre que su proporción no exceda el umbral. Teniendo en cuenta esta limitante, acudimos a la hipótesis multinstancia estándar para obtener una imputación inicial de las etiquetas de instancias. Siguiendo esta heurística, toda instancia que pertenezca a una bolsa positiva pero no a una negativa, es asignada a la clase positiva, de lo contrario se asigna a la clase negativa.

Una vez terminado este proceso de proposicionalización, cada instancia posee una etiqueta de clase asignada, y los ejemplos, originalmente multinstancia, han sido transformados de forma que pueden ser usados por el algoritmo de aprendizaje monoinstancia para construir el clasificador de Rocchio.

Al obtener los datos monoinstancia, se procede a entrenar el clasificador de Rocchio con el fin de obtener los perfiles relativos a cada clase. A partir de la nueva información obtenida por el clasificador de Rocchio, se recalculan las etiquetas de clase de las instancias para disminuir de cierto modo los errores de imputación iniciales. Para cada instancia se calcula su semejanza con el perfil de cada clase y se asigna a la clase cuyo perfil sea más semejante.

Teniendo reconsideradas las etiquetas de clase de las instancias, el envoltorio determina la proporción de instancias positivas para cada bolsa, y luego calcula el umbral de proporción a partir del cual se clasifica una bolsa en la clase positiva.

En la etapa de clasificación, el envoltorio usa el clasificador monoinstancia de Rocchio construido durante el entrenamiento para clasificar las instancias dentro de la nueva bolsa. Luego, a partir de la etiqueta de sus instancias, el envoltorio determina la etiqueta de clase de la bolsa basándose en la hipótesis multinstancia de proporción.

De manera formal, el algoritmo clasificador MIRocchio realiza en la etapa de entrenamiento el proceso de proposicionalización llevado a cabo por el envoltorio siguiendo la heurística de la hipótesis multinstancia estándar, donde una instancia es asignada a la clase negativa si aparece en alguna bolsa negativa, en otro caso es asignada a la clase positiva.

Seguidamente se construyen los perfiles de clase positivo ψ^+ y negativo ψ^- respectivamente utilizando las fórmulas de Rocchio (1.3) y (1.4). Una vez obtenidos los perfiles de cada clase se reasignan etiquetas de clase a todas las instancias basándose en su cercanía con los perfiles de clase calculados. Luego, se calcula el umbral de proporción t para cada bolsa de entrenamiento X. 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.

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 determina 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.

Los resultados experimentales obtenidos en (Tarragó 2014) mostraron que MIRocchio es capaz de mejorar el desempeño de varios clasificadores multinstancia del estado del arte en el dominio de aplicaciones textuales. El estudio del costo computacional del algoritmo mostró además que el tiempo de clasificación solo depende linealmente del número de atributos, lo que hace de este algoritmo un método ideal para aplicaciones en-línea. Sin embargo como se mencionaba al inicio de este trabajo, este algoritmo presenta una limitante y es que durante el proceso de aprendizaje se introduce un sesgo indeseado durante la imputación de las etiquetas de clases de las instancias. Este sesgo introducido se debe a que como se explicó con anterioridad, las instancias son etiquetadas siguiendo determinada hipótesis multinstancia que no siempre se cumple, en este caso la hipotiposis multinstancia estándar.

Con el objetivo de mitigar las limitaciones de MIRocchio, en el siguiente capítulo se proponen tres nuevos clasificadores multinstancia que prometen aumentar la eficiencia y la eficacia de los resultados de la clasificación.

2 Nuevos clasificadores multinstancia basados en Rocchio

La fórmula de Rocchio ha sido ampliamente utilizada con importantes resultados en el área de la clasificación textual, esto se debe a su alta eficiencia y a su aceptable eficacia. El algoritmo MIRocchio adecua la fórmula de Rocchio al escenario multinstancia con buenos resultados en el área de la clasificación textual. Partiendo del uso de la fórmula de Rocchio, en el presente capítulo se describen los algoritmos propuestos para validar las hipótesis planteadas. Primeramente se presentan los algoritmos diseñados para la implementación de las mismas y para cada algoritmo obtenido se explica su funcionamiento, haciendo hincapié en aquellos detalles que podrían resultar de mayor interés.

2.1. Propuestas

Como se ha planteado con anterioridad, el algoritmo MIRocchio presenta la limitación de que durante la etapa de entrenamiento introduce un sesgo no deseado en el proceso de imputación de las etiquetas de clases de las instancias, por lo que en este trabajo se exponen variantes del mismo con hipótesis de trabajo bien marcadas con el fin de mitigar estas limitaciones.

En este capítulo se proponen tres clasificadores multinstancia que implementan las dos hipótesis planteadas en esta tesis. En la Tabla (2.1) se observan los tres algoritmos propuestos acompañados del MIRocchio, todos agrupados por las hipótesis que implementan. Las filas representan las hipótesis multinstancia subyacente tras los clasificadores, en este caso dos hipótesis, la primera basada en la proporción de instancias positivas y la segunda basada en distancia entre bolsas, mientras que por las columnas representan las fronteras de decisión entre clases utilizada por los algoritmos.

Como primera propuesta de este trabajo, aparece el algoritmo clasificador multinstancia IEMR (*Instance-profile Multi-instance* Rocchio por sus siglas en inglés), que aunque no

		Frontera de decisión entre clases	
		Hiperplano	Hiperesfera
Hipótesis	Hipótesis basada en la creación de perfiles de clase a nivel de instancia	MIRocchio	IEMR
	Hipótesis basada en la creación de perfiles de clase a nivel de bolsa	BPMR	BEMR

Tabla 2.1: Clasificadores multinstancia basados en la fórmula de Rocchio agrupados por las hipótesis que implementan

elimina el sesgo insertado por MIRocchio durante el entrenamiento, si promete mejorar la calidad de la clasificación. La hipótesis subyacente tras este clasificador es al igual que en su antecesor, la hipótesis multinstancia basada en la proporción de instancias positivas. La principal diferencia entre este clasificador y el MIRocchio es que se incorpora la idea de la utilización de un solo perfil de clase (el perfil de la clase positiva) frente al usual enfoque de la utilización de un perfil para cada clase.

Esta primera hipótesis plantea que el uso del perfil de clase positiva solamente, aumentará la eficacia y la eficiencia de la clasificación sobre los algoritmos que implementen la hipótesis basada en proporción de instancias positivas frente a aquellos que implementen la misma hipótesis utilizando ambos perfiles de clase. Esta nueva hipótesis interpreta la frontera de decisión que separas ambas clases como una hiperesfera en el espacio de ejemplos. Esta hiperesfera engloba los ejemplos de la clase positiva, excluyendo a los de la negativa, lo que permite su aplicabilidad a aquellos problemas que por su naturaleza no puedan ser linealmente separables para los cuales la clase de interés es considerada como positiva y las demás como negativas. La implementación de la hiperesfera como frontera de decisión conlleva a la creación solamente del perfil de la clase positiva, lo cual disminuye el costo computacional de este algoritmo. Este algoritmo fue diseñado para demostrar la veracidad de esta primera hipótesis.

Para comprobar la validez de la segunda hipótesis presentada en la investigación que plantea la creación de perfiles de clase a niveles de bolsa se han propuesto dos clasificadores bautizados como BPMR y BEMR. Estos algoritmos multinstancia con perfiles de clases a nivel de bolsa evitan la introducción de sesgos no deseados en el aprendizaje dado que no realizan el proceso de proposicionalización. Para evitar la asignación de etiquetas de clases a las instancias estos algoritmos realizan un submuestreo aleatorio de las bolsas de entrenamiento para obtener los conjuntos R^+ y R^- utilizados en las fórmulas (1.3) y (1.4)

para confeccionar los perfiles de clase, este proceso de submuestreo aleatorio de las instancias será descrito más adelante.

En las secciones siguientes se describen en detalle cada uno de los nuevos clasificadores propuestos que implementan las hipótesis presentadas en este trabajo.

2.2. IEMR

El primer algoritmo propuesto en esta investigación basa su diseño en los métodos de envoltorio, y ha sido bautizado como IEMR por sus siglas en inglés (*Instance-profile hiper-Esfera Multi-instance* ROCCHIO). Durante el proceso de entrenamiento el envoltorio transforma los datos multinstancia en monoinstancia, para luego construir un modelo de clasificación mediante el clasificador subyacente basado en la fórmula de ROCCHIO. En la etapa de clasificación, el envoltorio aplica la transformación inversa para convertir las predicciones hechas a las instancias de la bolsa en una predicción de la etiqueta de clase de la bolsa. Estas transformaciones están determinadas por la hipótesis multinstancia de proporción descrita en el capítulo anterior e implementada por MIROCCHIO.

Este clasificador, a diferencia del MIROCCHIO, que crea perfiles para cada clase, utiliza la fórmula de ROCCHIO para la creación solamente del perfil de la clase positiva. La interpretación geométrica de la frontera de decisión que separas ambas clases en el espacio es una hiperesfera que engloba los ejemplos de la clase positiva, excluyendo a los de la negativa. A pesar de las ventajas computacionales que ofrece este algoritmo dado que solamente crea un perfil de clase, mantiene la introducción de sesgos en el aprendizaje durante el proceso de imputación de las etiquetas de clases de las instancias.

En las secciones siguientes se describe detalladamente el diseño del clasificador IEMR, comenzando por el funcionamiento del algoritmo de envoltorio. Seguidamente se explican en detalle los procesos de entrenamiento y clasificación, así como los métodos utilizados para la estimación de los parámetros que intervienen en su diseño.

2.2.1. Algoritmo envoltorio

El algoritmo de envoltorio como se ha discutido anteriormente, tiene como función principal servir de interfaz entre los datos multinstancia del clasificador y el clasificador simpleinstancia subyacente. En la etapa de entrenamiento el envoltorio transforma los datos multinstancia en monoinstancia para que puedan ser usados por el clasificador monoinstan-

cia en la construcción del modelo de clasificación. La figura (2.1) presenta esquemáticamente el funcionamiento del envoltorio, a continuación se describe cada paso de este.

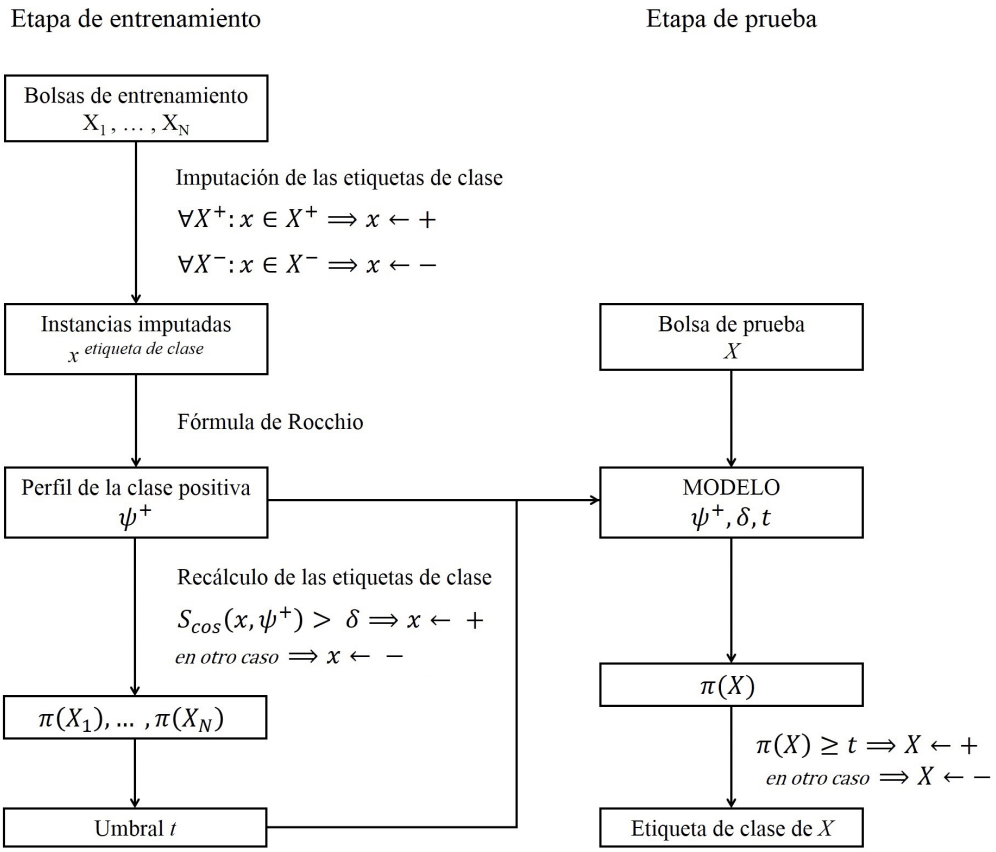

Figura 2.1: Esquema del algoritmo clasificador IEMR.

El envoltorio utilizado en este algoritmo es de tipo basado en instancias, o sea, que el espacio monoinstancia obtenido está formado por el conjunto de todas las instancias de todas las bolsas de entrenamiento. Para determinar las etiquetas de clase de las instancias el envoltorio deberá aplicar la hipótesis multinstancia de proporción de instancias positivas, pero con la información que brindan los datos, no se conoce la proporción necesaria de instancias positivas que hacen que una bolsa sea positiva, solamente se puede inferir que si una bolsa es positiva es porque la proporción de instancias positivas en esta ha superado

el umbral de proporción, y lo contrario para las bolsas negativas. Teniendo en cuenta que con la información disponible (solamente se conocen las etiquetas de clase de las bolsas) no podemos saber cuáles de las instancias que están dentro de la bolsa son positivas, aplicamos la hipótesis multinstancia estándar para obtener una imputación inicial. Siguiendo esta idea, cada instancia recibirá como etiqueta de clase la etiqueta de la bolsa donde esté contenida, o sea que las instancias pertenecientes a las bolsas positivas serán asignadas a la clase positivas y lo contrario para las instancias que pertenezcan a las bolsas negativas. Una vez determinada las etiquetas de clase de las instancias, los datos multinstancia, han sido transformados a datos simpleinstancia, y se procede a generar el perfil de la clase positiva aplicando la fórmula de Rocchio (1.3). Este perfil obtenido es un vector que sintetiza la descripción de los ejemplos de la clase positiva.

El próximo paso del algoritmo es recalcular las etiquetas de clase de las instancias a partir de la información que brinda el perfil de la clase positiva obtenido. Sin embargo, en ausencia de un perfil para la clase negativa no hay manera de calcular, dada una instancia x, a cuál perfil se asemeja más, por lo tanto aparece el concepto de umbral de semejanza. El umbral de semejanza define la frontera entre los los ejemplos de ambas clases. Al calcular la semejanza entre una instancia cualquiera y el perfil de la clase positiva, obtenemos un valor (acotado entre 0 y 1 resultante de aplicar la distancia coseno definida por la fórmula (2.3)) al cual llamaremos valor de semejanza, este indica como es lógico, cuánto se parecen estas instancias, o lo que es lo mismo, cuan semejantes son. Partiendo del hecho de que si la instancia es lo suficiente semejante al perfil positivo, esta será clasificada como positiva, aparece la siguiente incógnita: ¿Cómo saber cuándo es suficientemente semejante una instancia a otra? En resumen, el umbral de semejanza no es más que un valor que indica el mínimo de semejanza que tiene que existir entre una instancia x y el perfil de la clase positiva, para ser clasificada x como positiva.

Para cada instancia se calcula su semejanza con el perfil de la clase positiva y si esta supera el umbral de semejanza, esta se asigna a la clase positiva, de lo contrario se asigna a la clase negativa. Este paso de recalcular las etiquetas de clase permite que muchas instancias que fueron asignadas inicialmente a la clase positiva o negativa siguiendo la hipótesis multinstancia estándar, sean reasignadas a la clase a la cual en realidad se asemejan más.

Una vez reasignadas las etiquetas de clase de cada instancia, se calcula la proporción de instancias positivas en cada bolsa, para luego con esta información, determinar el umbral de proporción necesario para clasificar una bolsa como positiva. El perfil de la clase positiva

y los umbrales de semejanza y proporción constituyen el modelo de clasificación obtenido durante el proceso de entrenamiento.

En la etapa de clasificación se lleva a cabo el proceso contrario. El algoritmo usa la información brindada por el clasificador de Rocchio, o sea el perfil de la clase positiva, para determinar las etiquetas de clase de las instancias pertenecientes a una nueva bolsa. Luego siguiendo la heurística planteada por la hipótesis multinstancia de proporción, se determina la etiqueta de clase de la bolsa a partir de la etiqueta de sus instancias.

2.2.2. Clasificador IEMR

Primeramente durante la etapa de entrenamiento el algoritmo de envoltorio, partiendo del conjunto de datos de ejemplos, obtiene dos conjuntos de instancias etiquetadas, uno positivo y otro negativo siguiendo la heurística de la hipótesis multinstancia estándar. De manera que el envoltorio asigna las etiquetas de clase a las instancias de las bolsas según la etiqueta de clase de esta, este proceso es el llamado proceso de proposicionalización. O sea las instancias contenidas en bolsas positivas serán etiquetadas como positivas y las pertenecientes a bolsas negativas como negativas.

Las instancias obtenidas en este proceso de proposicionalización son utilizadas en la fórmula de Rocchio (1.3) para calcular el perfil de la clase positiva. Una vez obtenido el perfil de la clase positiva, el algoritmo recalcula las etiquetas de clase de las instancias partiendo de la semejanza entre estas y el perfil de la clase positiva obtenido mediante el paso anterior. Una instancia x será asignada a la clase positiva si su distancia coseno respecto al perfil positivo es menor que el umbral de semejanza, 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 se calcula mediante la fórmula (2.1), utilizando el modelo de ponderación exponencial. El modelo de clasificación es construido a partir del perfil de la clase positivas, junto con los umbrales de semejanza y proporción obtenidos en el proceso de entrenamiento. Este modelo servirá de base para la clasificación de los nuevos ejemplos. El algoritmo (1) explica detalladamente el proceso de entrenamiento del IEMR.

Input:

- $D = \{X_1, \ldots, X_M\}$ conjunto de ejemplos de entrenamiento
- ρ^+ intensidad de la selección de atributos para los ejemplos positivos
- δ umbral de semejanza

Output:

- Ψ^+ perfil de la clase positiva
- t umbral de proporción

$D^P \leftarrow \emptyset$
$D^N \leftarrow \emptyset$
foreach $X \in D$ **do**
  if X *es positiva* **then**
    foreach $x \in X$ **do**
      $D^P \leftarrow x$
  else
    foreach $x \in X$ **do**
      $D^N \leftarrow x$
$\Psi^+ \leftarrow \text{máx}\left\{0, \frac{1}{|D^P|}\sum_{x \in D^P} x_i - \frac{\rho^+}{|D^N|}\sum_{x \in D^N} x_i\right\}$
$D^P \leftarrow \emptyset$
$D^N \leftarrow \emptyset$
foreach $X \in D$ **do**
  foreach $x \in X$ **do**
    $d_C \leftarrow$ distancia coseno entre x y Ψ^+
    if $d_C < \delta$ **then**
      $D^P \leftarrow x$
    else
      $D^N \leftarrow x$
$t \leftarrow$ proporción de instancias positivas en D^P y D^N

Algorithm 1: IEMR (entrenamiento)

Durante el proceso de clasificación de una nueva bolsa X, se clasifican las instancias contenidas en esta. La etiqueta de clase de cada instancia de X es calculada según su distancia al perfil de la clase positiva. Luego se calcula la proporción de instancias positivas en X, la bolsa es asignada a la clase positiva si la proporción de instancias positivas contenidas en esta es mayor que el umbral de proporción t calculado durante el proceso de entrenamiento, de lo contrario es asignada a la clase negativa. En el algoritmo (2) se describe el procedimiento realizado por IEMR para clasificar un nuevo ejemplo.

Input:

- X bolsa a clasificar
- Ψ^+ perfil de la clase positiva
- δ umbral de semejanza
- t umbral de proporción

Output:

- y etiqueta de clase

```
1  I+ ← 0
2  foreach x ∈ X do
3  |   d_C ← distancia coseno entre x y Ψ+
4  |   if d_C < δ then
5  |   |   I+ ← I+ + 1
6  P ← I+ / |X|
7  if P > t then
8  |   y ← clase positiva
9  else
10 |   y ← clase negativa
```

Algorithm 2: IEMR (explotación)

A continuación se explica en detalle cómo se calcula el valor de ρ^+ para la fórmula (1.3) y los umbrales de semejanza y proporción respectivamente.

2.2.3. Estimación de los parámetros

Estudios anteriores han demostrado que el clasificador de ROCCHIO presenta significativa dependencia a los parámetros ρ^+ y ρ^- utilizados en las fórmulas (1.3) y (1.4). Los autores (Cohen and Singer 1999) han sugerido que el ajuste adecuado de los parámetros del clasificador de ROCCHIO puede aumentar considerablemente la calidad de la clasificación en la rama de la clasificación textual, y que estos parámetros dependen directamente y en gran medida del conjunto de datos utilizado para el aprendizaje. Varios estudios (Ittner et al. 1995, Bi et al. 2007, Guo et al. 2003, Schapire et al. 1998) han presentado diferentes configuraciones para la selección de los parámetros del clasificador de ROCCHIO con diversos resultados. Experimentos realizados en (Moschitti 2003) demuestran que la relación entre ρ y la exactitud de la clasificación se aproxima a una curva convexa con un único punto máximo. Partiendo de este resultado Moschitti sugiere que los mejores valores para ρ pueden obtenerse probando valores incrementales de ρ hasta que la exactitud de la clasificación alcance un valor máximo.

Además del valor de ρ^+ necesario para la fórmula de ROCCHIO (1.3) es necesario calcular el valor del umbral de semejanza δ.

La optimización de los parámetros se realiza mediante una 10×10 CV. Para cada partición de entrenamiento se ha utilizado en el algoritmo IEMR un método que selecciona los valores óptimos de ρ^+ y δ a partir de los conjuntos $\Lambda^+ = \{10, 11, 12, ..., 40\}$ y $\Delta = \{0{,}1, 0{,}2, 0{,}3, \ldots, 0{,}9\}$ respectivamente, entrenando y evaluando sobre las particiones correspondientes en cada iteración. De manera general son comparadas $31 \times 9 = 279$ configuraciones diferentes de ρ^+ y δ usando como métrica de la calidad de la clasificación $F1$ para guiar la búsqueda. El conjunto Λ^+ incluye además valores que fueron utilizados en experimentos realizados por (Tarragó 2014), validados por buenos resultados para el algoritmo clasificador multinstancia MIROCCHIO sobre el área de la clasificación textual. Los valores máximos de Λ^+ se encuentran sobre el límite superior recomendado por (Moschitti 2003, Yang and Pedersen 1997), debido a que con estos valores se logra una reducción de atributos bastante agresiva. Los valores del conjunto Δ son los términos de una serie que va desde 0,1 hasta 0,9 con un incremento de 0,1. Estos valores de Δ están acotados debido a la relación de este parámetro con el resultado de las distancias calculadas entre las instancias y el perfil de la clase positiva, distancia acotada también entre 0 y 1.

De todos los pares (ρ^+, δ) se selecciona aquel que maximice el resultado de la clasificación según la métrica F1. Al finalizar este proceso de optimización de los parámetros se procede

a la creación del perfil de la clase positiva utilizando el valor de ρ^+ encontrado, luego se recalculan las etiquetas de clase utilizando además el valor de δ y finalmente se calcula el umbral de proporción de instancias positivas en las bolsas de ejemplo.

2.2.4. Aprendizaje del umbral de proporción de instancias positivas

El valor del umbral de proporción t por lo general es desconocido para los conjuntos de datos, y es de suma importancia calcularlo correctamente durante el proceso de aprendizaje.

Para calcular el valor del umbral de semejanza fueron propuestos por (Tarragó 2014) dos modelos de ponderación, el modelo lineal y el modelo exponencial. Sin embargo, fue demostrado por este estudio que para el modelo de ponderación lineal en algunos casos el umbral t se aproxima demasiado a cero y se afecta la capacidad de generalización del clasificador.

Para aliviar este problema aplicamos el modelo de ponderación exponencial que calcula el umbral t_e como el promedio de las mediasμ_py μ_n ponderadas por el exponencial de las varianzas V_p y V_n respectivamente y se define como

$$t_e = \frac{\mu_p e^{-\eta V_p} + \mu_n e^{-\eta V_n}}{e^{-\eta V_p} + e^{-\eta V_n}} \tag{2.1}$$

donde μ_p y μ_n son los valores medios de $\pi(X)$ sobre todas las bolsas de entrenamiento positivas y negativas, respectivamente y V_p y V_n las varianzas de $\pi(X)$ sobre todas las bolsas de entrenamiento positivas y negativas, respectivamente. El término η es un parámetro que controla la inclinación de la curva exponencial. 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.

Por estos motivos ha sido seleccionado el modelo de ponderación exponencial para ser integrado al clasificador IEMR para calcular el umbral de proporción de instancia positivas.

2.2.5. Análisis de la eficiencia temporal de IEMR

En la etapa de entrenamiento el método de imputación de las etiquetas de clase de las instancias presenta una complejidad temporal de $\mathcal{O}(N)$, donde N es el número de instancias del conjunto de entrenamiento.

Por otro lado, el algoritmo de Rocchio, en el cálculo de los perfiles de clase, comporta una complejidad temporal $\mathcal{O}(m \times N)$, donde m es número de atributos. El recálculo de las

etiquetas de clase de las instancias se hace en tiempo $\mathcal{O}(m \times N)$ y el umbral de proporción se determina en tiempo $\mathcal{O}(N)$. Por tanto, la eficiencia temporal de IEMR durante la etapa de entrenamiento esta es $\mathcal{O}(m \times N)$, que es una cota superior del costo computacional del método.

Durante el proceso de clasificación, las instancias de la nueva bolsa solo tienen que ser comparadas con el perfil de la clase positiva, por lo que la complejidad computacional en esta etapa es $\mathcal{O}(m)$. Esto significa que la clasificación se realiza en un tiempo que depende solo linealmente del número de atributos.

2.3. Algoritmos multinstancia con perfiles de clases a nivel de bolsa

Los algoritmos basados en perfiles de clases a nivel de bolsa son los algoritmos obtenidos como resultado de desarrollar la segunda hipótesis propuesta en esta investigación. Estos algoritmos, al igual que el IEMR, basa su diseño en los métodos de envoltorio. La diferencia más notoria respecto a los clasificadores MIRocchio e IEMR es que estos algoritmos con perfiles de clases a nivel de bolsa utilizan la fórmula de Rocchio para la creación de los perfiles de clase, pero a nivel de bolsa, o sea el perfil de clase será una bolsa en lugar de solo una instancia. Los parámetros necesarios en la etapa de entrenamiento de estos algoritmos se calculan automáticamente mediante validación cruzada, tratando de maximizar el resultado de la clasificación en cada iteración. Estos algoritmos no introducen sesgos durante el proceso de aprendizaje porque al trabajar a nivel de bolsa, no realizan la imputación de las etiquetas de clases de las instancias.

A continuación se describe de manera general el funcionamiento de los clasificadores BMRocchio, haciendo hincapié en la hipótesis multinstancia utilizada para ambos y se describen los elementos comunes para los algoritmos BPMR y BEMR durante los procesos de entrenamiento y clasificación.

2.3.1. Hipótesis basada en distancia entre bolsas

La hipótesis multinstancia adoptada en este trabajo para los clasificadores con perfiles de clases a nivel de bolsa utiliza la distancia entre estas para predecir la etiqueta de clases de una nueva bolsa durante el proceso de clasificación, esta hipótesis plantea que una bolsa será etiquetada en función de la distancia existente entre esta y las bolsas que representan

los perfiles de clase.

Para medir distancia entre bolsas han sido propuestas por la literatura una serie de métricas, siendo la distancia de Hausdorff propuesta por (Wang and Zucker 2000) y sus variantes las más reconocidas. La distancia tradicional de Hausdorff entre dos conjuntos de puntos (bolsas) se define como

$$Hausdorff(X, \Psi) = máx\left(máx_{x \in X} mín_{\psi \in \Psi} \parallel x - \psi \parallel, máx_{\psi \in \Psi} mín_{x \in X} \parallel x - \psi \parallel\right) \quad (2.2)$$

donde $\parallel x - \psi \parallel$ es una métrica de distancia entre instancias. Tomando como basamento la semejanza coseno, se utiliza la siguiente métrica de distancia, a la cual se denomina distancia coseno

$$\parallel x - \psi \parallel = 1 - \frac{\sum_{i=1}^{A} x_i \psi_i}{\sqrt{\sum_{i=1}^{A} x_i^2} \sqrt{\sum_{i=1}^{A} \psi_i^2}} \quad (2.3)$$

donde A es el tamaño de los vectores, es decir, el número de atributos. Como la semejanza coseno arroja un valor entre 0 y 1, también la distancia coseno estará normalizada entre 0 y 1, y así será también para la distancia de Hausdorff.

Partiendo de la distancia clásica de Hausdorff, la distancia AverageHausdorff es una variación que intenta mitigar la sensibilidad que presenta la distancia original a los valores extremos, utilizando para esto la media entre los valores mínimos obtenidos. La distancia AverageHausdorff entre dos bolsas X y Ψ se define como

$$AverageHausdorff(X, \Psi) = 1 - \frac{\sum_{x \in X} mín_{\psi \in \Psi} \parallel x - \psi \parallel + \sum_{\psi \in \Psi} mín_{x \in X} \parallel x - \psi \parallel}{\mid X \mid + \mid \Psi \mid} \quad (2.4)$$

2.3.1.1. Operadores OWA aplicados a métricas de distancia entre bolsas

Tanto la distancia de Hausdorff como la de AverageHausdorff mencionadas anteriormente son sensibles en mayor o menor medida a extremos debido a la utilización de funciones mínimos y máximos, en cambio los operadores OWA por sus siglas en ingles (*ordered*

weighted aggregation) propuestos por (Yager 1988) basan su funcionamiento en un vector de peso que determina cuáles elementos de la colección son de relevancia y en que medida.

Los operadores OWA modelan un proceso de agregación en el cual una secuencia A de n valores escalares, los cuales son ordenados decrecientemente y pesados según el valor correspondiente a sus posiciones en el vector de peso $W = \{w_i\}$, donde $w_i \in [0, 1]$ y $\sum_i^n w_i = 1$. De manera general el resultado de aplicar los operadores OWA sobre una colección A utilizando un vector de pesos W se puede definir como

$$OWA_W(A) = \sum_{i=1}^{n} w_i a_i$$

donde w_i y a_i son los elementos $i-ésimo$ del vector de peso y la colección respectivamente.

Una de las ventajas del uso de los operadores OWA es su flexibilidad dado que permite modelar disimiles estrategias de agregaciones. Por ejemplo los operadores máximo, mínimo y promedio pueden ser modelados a través de los operadores OWA utilizando los vectores de pesos W_{max}, W_{min} y W_{avg} respectivamente. Estos vectores de peso se definen de la siguiente manera

$$W_{max} = \{w_i\},\ donde\, w_1 = 1, w_i = 0, i \neq 1$$

$$W_{min} = \{w_i\},\ donde\, w_n = 1, w_i = 0, i \neq n$$

$$W_{avg} = \{w_i\},\ donde\, w_i = \frac{1}{n}, i = 1, \ldots, n$$

Tratando de minimizar la sensibilidad a valores extremos de las distancias basadas en Hausdorff, estas han sido adaptadas incluyendo operadores OWA en sus respectivos diseños. Estas nuevas distancias presentadas se conocen como OWAHaus (2.5) y AveOWAHaus (2.6) y se obtienen tras aplicar los operadores OWA a las distancias Hausdorff (2.2) y AverageHausdorff (2.4) respectivamente. Estas distancias se definen de la siguiente manera

$$OWAHaus(A,B) = 1 - max\left(OWA_{W_{max}} \underbrace{OWA_{W_{min}} \underbrace{\delta(a,b)}_{b\in B}}_{a\in A}, OWA_{W_{max}} \underbrace{OWA_{W_{min}} \underbrace{\delta(a,b)}_{a\in A}}_{b\in B} \right) \tag{2.5}$$

$$AveOWAHaus(A,B) = 1 - \frac{\sum_{a\in A} OWA_{W_{min}} \underbrace{\delta(a,b)}_{b\in B} + \sum_{b\in B} OWA_{W_{min}} \underbrace{\delta(a,b)}_{a\in A}}{|A| + |B|} \tag{2.6}$$

Los vectores de peso W_{max} y W_{min} utilizados en la implementación de las distancias OWAHaus y AveOWAHaus están conformados de la siguiente manera respectivamente

$$W_{max} = \left(\frac{1}{1\sum_{i=1}^{p}\frac{1}{i}}, \frac{1}{2\sum_{i=1}^{p}\frac{1}{i}}, \ldots, \frac{1}{p\sum_{i=1}^{p}\frac{1}{i}} \right)$$

$$W_{min} = \left(\frac{1}{p\sum_{i=1}^{p}\frac{1}{i}}, \frac{1}{(p-1)\sum_{i=1}^{p}\frac{1}{i}}, \ldots, \frac{1}{1\sum_{i=1}^{p}\frac{1}{i}} \right)$$

donde p es la longitud del vector de pesos para ambos casos.

Para estas cuatro medidas de distancia entre bolsas presentadas en esta sección se realizaron varios experimentos, los cuales resaltaron el superior desempeño de la distancia AverageHausdorff (2.4) sobre las otras tres distancias presentadas. Por este motivo se decidió incorporar esta métrica de distancia entre bolsas al diseño de los dos clasificadores propuestos basados en perfiles a nivel de bolsas.

2.3.2. Algoritmos multinstancia con perfiles de clases a nivel de bolsa (Entrenamiento)

De manera general durante la etapa de entrenamiento los algoritmos con perfiles de clases a nivel de bolsa mantienen el mismo comportamiento. A partir del esquema mostrado en la Figura (2.2) se describe a continuación de manera general el funcionamiento de estos algoritmos basados en perfiles de clases a nivel de bolsa.

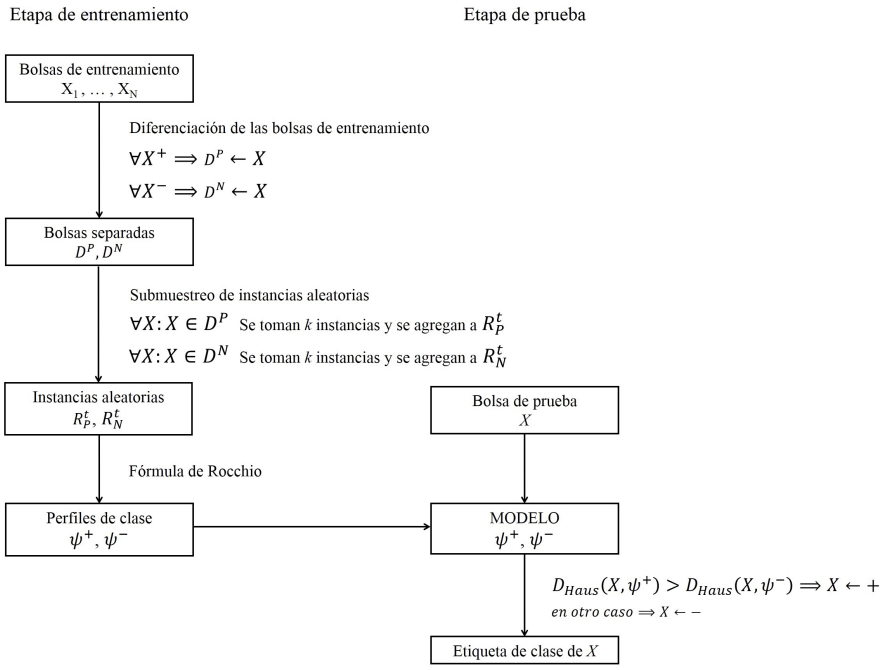

Figura 2.2: Esquema del funcionamiento general de los algoritmos propuestos basados en perfiles de clase a niveles de bolsa.

Partiendo de un conjunto de ejemplos de entrenamiento, estos algoritmos separan las bolsas positivas de las negativas creando dos nuevos conjuntos, D^P y D^N respectivamente. Una vez separadas las bolsas en conjuntos individuales, se procede a calcular el tamaño de la bolsa que conformara el perfil de la clase positiva, el cual denominaremos T. El tamaño de la bolsa perfil de la clase positiva T se define como el promedio de la cantidad de instancias en las bolsas de ejemplo, y definirá cuantas instancias prototípicas tendrán las bolsas de los perfiles de clases.

$$T \leftarrow \frac{1}{|D^p|} \sum_{X \in D^P} |X| \tag{2.7}$$

Tras este paso, se procede a calcular las T instancias para los perfiles de clases. Para esta operación es necesario crear dos conjuntos de instancias etiquetadas, a los cuales llamaremos R_P^t y R_N^t que se utilizaran en las fórmulas de Rocchio como se verá más adelante.

En cada iteración los conjuntos R_P^t y R_N^t almacenan instancias seleccionadas aleatoriamente atendiendo a varios criterios que varían según el enfoque. Entiéndase de la siguiente manera, para cada bolsa de D^P se toman aleatoriamente k instancias y luego se agregan a R_P^t. De igual manera funciona para las bolsas almacenadas en D^N, se toman aleatoriamente k instancias y luego se agregan a R_N^t. De esta manera quedan almacenados en R_P^t y R_N^t un subconjunto aleatorio de instancias positivas y negativas respectivamente que son utilizadas en las fórmulas de Rocchio para de calcular las instancias $t - ésima$ que conformaran las bolsas de los perfiles de clase.

$$\Psi_a^t \leftarrow \text{máx} \left\{ 0, \frac{1}{|R_P^t|} \sum_{x \in R_P^t} x_a - \frac{\rho}{|R_N^t|} \sum_{x \in R_N^t} x_a \right\} \tag{2.8}$$

Una vez obtenidas las T instancias anteriores se conformaran las bolsas de los perfiles de clase agrupando las instancias prototípicas positivas obtenidas en una bolsa perfil positivo e igualmente con las negativas para formar una bolsa perfil negativo. Estas bolsas serán utilizadas como base para estimar las etiquetas de clase de los nuevos ejemplos. En la próxima sección se explica detalladamente como se realizo la selección del valor de k para el diseño de estos algoritmos.

2.3.3. Estimación de k

El valor de k indica la cantidad de instancias que serán maestreadas de las bolsas de ejemplos para construir los conjuntos de instancias etiquetadas para la fórmula de Rocchio. De manera general, el valor de k determina la cantidad de instancias que representaran a la bolsa contenedora.

Una hipótesis lógica sería que al seleccionar mayor cantidad de instancias de una bolsa, la muestra seleccionada obtendría más representatividad para la bolsa en cuestión. Luego, por este motivo se realizaron varios experimentos para determinar un valor de k adecuado

para estos algoritmos. Estos algoritmos fueron probados con valores para k desde 1 hasta 10, aumentando gradualmente la cantidad de instancias seleccionadas en cada experimento. Los resultados obtenidos no mostraron diferencias significativas entre el desempeño de los algoritmos para los diferentes valores de k, por los que se seleccionó el valor 1 como un buen acercamiento para la cantidad de instancias a seleccionar.

2.3.4. BPMR

Este algoritmo llamado BPMR (por sus siglas en ingles *Bag-profile hiper-Plano Multi-instance* Rocchio) utiliza un perfil positivo y uno negativo para predecir las etiquetas de las nuevas bolsas mediante una función de distancia entre conjuntos, donde las bolsas son clasificadas en función a la cercanía que guarde con ambos perfiles. Explicado de otra manera, el espacio formado por todas las bolsas es dividido por un hiperplano que separa las bolsas positivas de las negativas, y en caso de aparecer una nueva bolsa, esta recibirá la clase cuyo perfil este a menor distancia.

BPMR pretende aplicar las fórmulas de Rocchio (1.3) y (1.4) sobre los conjuntos de instancias obtenidos muestreando aleatoriamente las bolsas de ejemplos con el fin de crear los perfiles de clases positivo y negativo. En esta sección se explica el funcionamiento de este algoritmo, haciendo mayor énfasis en el proceso de muestreo y creación de los perfiles de clases realizado durante el entrenamiento.

2.3.4.1. BPMR (Entrenamiento)

Siguiendo la filosofía de los algoritmos basados en perfiles de clases a nivel de bolsa, el algoritmo clasificador multinstancia BPMR mantiene el mismo comportamiento durante la etapa de entrenamiento. Primeramente se parte de un conjunto de ejemplos y se separan las bolsas positivas de las negativas para crear dos nuevos conjuntos, D^P y D^N respectivamente. Tras ser obtenidos los conjuntos que contienen los ejemplos negativos y positivos se calcula el tamaño de las bolsas que conformaran los perfiles de la clase positiva y negativa respectivamente, o lo que es lo mismo, cuantas instancias contendrá cada bolsa perfil, este valor será denominado T. El valor de T se define como el promedio de la cantidad de instancias en las bolsas de ejemplo.

Una vez determinados los conjuntos D^P y D^N y la cantidad T de instancias para las bolsas de los perfiles positivo y negativo, se procede a calcular las T instancias para los perfiles de clase. Cada perfil es una bolsa que contendrá T instancias resultantes de aplicar las

fórmulas de Rocchio (1.3) y (1.4) para los perfiles positivo y negativo sobre los conjuntos R_P^t y R_N^t que se obtienen submuestreando aleatoriamente las bolsas de los conjuntos D^P y D^N respectivamente. Más explícitamente los conjuntos D^P y D^N son recorridos y para cada bolsa contenida en estos son seleccionadas k instancias de manera aleatoria y son agregadas a los conjuntos R_P^t y R_N^t respectivamente. En cada iteración los conjuntos R_P^t y R_N^t contienen instancias obtenidas mediante un submuestreo aleatorio de las instancias pertenecientes a las bolsas de ejemplo, siendo estas una muestra representativa de toda la población dado el factor de aleatoriedad.

Una vez obtenidas las T instancias anteriores se conformaran las bolsas de los perfiles de clase positivo y negativo. Los perfiles de clase ψ^+ y ψ^- constituyen el modelo de clasificación construido por el algoritmo durante su etapa de entrenamiento. Todo este proceso de describe con más detalles en el algoritmo (3).

Input:

- $D = \{X_1, \ldots, X_M\}$ conjunto de ejemplos de entrenamiento
- ρ^+ intensidad de la selección de atributos para los ejemplos positivos
- ρ^+ intensidad de la selección de atributos para los ejemplos negativos

Output:

- Ψ^+ perfil de la clase positiva
- Ψ^- perfil de la clase negativa

$D^P \leftarrow$ conjunto de instancias de ejemplos positivos
$D^N \leftarrow$ conjunto de instancias de ejemplos negativos
$A \leftarrow$ número de atributos de D
$T \leftarrow \frac{1}{|D^P|} \sum_{X \in D^P} |X|$ tamaño del perfil de clase
for $t \leftarrow 1$ **to** T **do**
    $R_P^t \leftarrow \emptyset$
    $R_N^t \leftarrow \emptyset$
    foreach $X \in D^P$ **do**
        seleccionar aleatoriamente una instancia $x \in X$ y agregarla a R_P^t
    foreach $X \in D^N$ **do**
        seleccionar aleatoriamente una instancia $x \in X$ y agregarla a R_N^t
    for $a \leftarrow 1$ **to** A **do**
        $\Psi_a^{t^+} \leftarrow \text{máx}\left\{0, \frac{1}{|R_P^t|} \sum_{x \in R_P^t} x_a - \frac{\rho^+}{|R_N^t|} \sum_{x \in R_N^t} x_a\right\}$
        $\Psi_a^{t^-} \leftarrow \text{máx}\left\{0, \frac{1}{|R_N^t|} \sum_{x \in R_N^t} x_a - \frac{\rho^-}{|R_P^t|} \sum_{x \in R_P^t} x_a\right\}$

Algorithm 3: BMRocchio Hiperplano (entrenamiento)

2.3.4.2. BPMR (Prueba)

Durante el proceso de clasificación una nueva bolsa X es asigna a la clase cuyo perfil es más semejante. Para calcular la semejanza entre la bolsa y los perfiles de clase, se utiliza la función de distancia entre bolsas AverageHausdorff definida por la fórmula (2.4). En otras palabras, se miden las distancias entre la bolsa X y ambos perfiles ψ^+ y ψ^-, a las cuales denominaremos d_p y d_n respectivamente. Una vez obtenidas ambas distancias la bolsa X será clasificada como positiva si $d_p > d_n$, en otro caso será clasificada como negativa. El algoritmo (4) explica el proceso llevado a cabo por BPMR durante la etapa de clasificación.

Input:

- X bolsa a clasificar
- Ψ^+ perfil de la clase positiva
- Ψ^- perfil de la clase negativa

Output:

- y etiqueta de clase

```
1 d_H^+ ← distancia AverageHausdorff entre X y Ψ^+
2 d_H^- ← distancia AverageHausdorff entre X y Ψ^-
3 if d_H^+ < d_H^- then
4 |   y ← clase positiva
5 else
6 |   y ← clase negativa
```

Algorithm 4: BMRocchio Hiperplano (explotación)

2.3.4.3. Cálculo de los parámetros

Como se explicaba anteriormente, estudios realizados en (Moschitti 2003) demuestran la relación entre ρ y la exactitud de la clasificación. Partiendo de este resultado Moschitti sugiere que los mejores valores para ρ pueden obtenerse probando valores incrementales de ρ hasta que la exactitud de la clasificación alcance un valor máximo.

Para el algoritmo BPMR es necesario optimizar los valor de ρ^+ y ρ^- aplicados en las fórmulas de Rocchio. Al necesitar dos perfiles de clase a parece la necesidad de optimizar

dos valores de ρ, lo que aumenta el costo computacional del algoritmo. Según se conoce los valores de ρ^+ tienden a ser mucho mayores que los valores de ρ^-, esto se cree que esta asociado a la mayor ambigüedad presente en los ejemplos positivos. A partir de la hipótesis: $\rho^+ > \rho^-$ se propone fijar unos valores iniciales para ambos parámetros, a partir de los cuales podríamos explorar ρ^- variándolo por validación cruzada y manteniendo ρ^+ fijo. Una vez obtenido el mejor valor de ρ^- (con ρ^+ fijo) podemos fijar ρ^- y variar ρ^+ por validación cruzada para obtener su mejor valor.

Para este algoritmo se decidió variar los valores de ρ^- entre 0 y 9, con incremento 1 (o sea, 0, 1, 2, ..., 9), y los valores de ρ^+ entre 30 y 75 con incremento 5 (o sea, 30, 35, ..., 75). Entonces, en vez de hacer $10 \times 10 = 100$ pruebas para encontrar la mejor combinación de valores, fijamos ρ^+ en 55 y probamos los 10 valores de ρ^-, si el mejor valor fue, por ejemplo, para $\rho^- = 3$ entonces lo dejamos con ese valor probamos con los otros 9 valores de ρ^+ para encontrar el mejor valor. Esto no asegura obtener los valores óptimos, pero podemos obtener un sub-óptimo haciendo tan solo 20 pruebas y ahorrando el 80 % del costo computacional. La optimización de los parámetros se realiza mediante una 10×10 CV y para cada caso se entrena el clasificador y se evalúa el desempeño de la clasificación para los diferentes valores de ρ^+ y ρ^-.

De todos los pares (ρ^+, ρ^-) se selecciona aquel que maximice el resultado de la clasificación según la métrica $F1$. Al finalizar este proceso de optimización de los parámetros se procede a la creación de los perfiles de cada clase utilizando los valores de ρ^+ y ρ^- encontrados.

2.3.4.4. Análisis de la eficiencia temporal de BPMR

En la etapa de entrenamiento el método para conocer la cantidad de instancias T que contendrán los perfiles de cada clase presenta una complejidad temporal de $\mathcal{O}(M)$, donde M es la cantidad de bolsas del conjunto de entrenamiento. Luego para crear los perfiles de clase es necesario crear T instancias para ambos, lo que presenta una complejidad temporal de $\mathcal{O}(T \times m \times N)$, donde m es el número de atributos y N es el total de instancias del conjunto de entrenamiento. Esto se debe a que el algoritmo de Rocchio, en el cálculo de los perfiles de clase, comporta una complejidad temporal $\mathcal{O}(m \times N)$. Esto significa que BPMR presenta una complejidad temporal de $\mathcal{O}(T \times m \times N)$ durante la etapa de entrenamiento.

Durante el proceso de clasificación, la nueva bolsa se compara con los perfiles de clase, utilizando la distancia entre conjuntos AverageHausdorff, lo que implica una complejidad

temporal de $\mathcal{O}(T \times B \times m)$, donde B es la cantidad de instancias de la nueva bolsa.

A continuación se describe la variante de algoritmo basado en la creación de perfiles de clase a nivel de bolsa y que implementa la hiperesfera como frontera de decisión entre clases.

2.3.5. BEMR

Una segunda propuesta más atrevida plantea la utilización de un solo perfil de clase, en este caso el perfil de la clase positiva, teniendo como premisa que aquellas bolsas que no pertenezcan a la clase positiva serán clasificadas como negativas. Esta propuesta bautizada como BEMR (por sus siglas en ingles *Bag-profile hiper-Esfera Multi-instance* Rocchio) utiliza un umbral de distancia para definir cuándo una bolsa es lo suficientemente semejante al perfil positivo como para ser incluida en esta clase. Expresado de otro modo, este algoritmo crea una hiperesfera en cuyo centroide yace el perfil de la clase positiva y su radio es igual al umbral de distancia, por lo que todas aquellas bolsas que se encuentren dentro de la hiperesfera son consideradas como positivas a diferencia de aquellas que se encuentren en su exterior, que son clasificadas como negativas.

En contraposición con el algoritmo BPMR, esta variante pretende aplicar solamente la fórmula de Rocchio (1.3) correspondiente a la construcción del perfil positivo. Y al igual que en el caso anterior los conjuntos de instancias positivas y negativas se obtienen muestreando aleatoriamente las bolsas de ejemplos.

2.3.5.1. BEMR (Entrenamiento)

El objetivo del proceso de entrenamiento es crear un modelo de clasificación a partir de los datos de los ejemplos suministrados. El perfil de la clase positiva y el umbral de distancia obtenido conforman dicho modelo de clasificación para el algoritmo BEMR.

Al igual que el algoritmo anterior, el BEMR parte de los subconjuntos de bolsas D^P y D^N positivas negativas respectivamente, obtenidos a partir de los datos de ejemplo D. El submuestreo de las instancias se realiza al igual que en el BPMR, o sea, tomando k instancias aleatorias de los conjuntos D^P y D^N en cada iteración, formado los subconjuntos R_P^t y R_N^t. En cada iteración los conjuntos R_P^t y R_N^t contienen instancias obtenidas mediante un submuestreo aleatorio de las instancias pertenecientes a los conjuntos D^P y D^N, siendo estas una muestra representativa de toda la población por el carácter aleatorio de la selección de las muestras. Sobre los conjuntos R_P^t y R_N^t se aplica la fórmula de Rocchio (1.3) correspondiente a la creación del perfil de la clase positiva con el objetivo de obtener en

cada iteración la instancia $t - ésima$ de la bolsa.

Las T instancias obtenidas anteriormente se unifican en una bolsa, formando el perfil de la clase positiva. Esta bolsa perfil de la clase positiva ψ^+ junto al umbral de distancia δ conforman el modelo de clasificación de este algoritmo. Este proceso de entrenamiento se describe detalladamente en el algoritmo (5).

Input:

- $D = \{X_1, \ldots, X_M\}$ conjunto de ejemplos de entrenamiento
- ρ^+ intensidad de la selección de atributos para los ejemplos positivos

Output:

- Ψ^+ perfil de la clase positiva

$D^P \leftarrow$ conjunto de instancias de ejemplos positivos
$D^N \leftarrow$ conjunto de instancias de ejemplos negativos
$A \leftarrow$ número de atributos de D
$T \leftarrow \frac{1}{|D^P|} \sum_{X \in D^P} |X|$ tamaño del perfil de clase
for $t \leftarrow 1$ **to** T **do**
    $R_P^t \leftarrow \emptyset$
    $R_N^t \leftarrow \emptyset$
    foreach $X \in D^P$ **do**
        seleccionar aleatoriamente una instancia $x \in X$ y agregarla a R_P^t
    foreach $X \in D^N$ **do**
        seleccionar aleatoriamente una instancia $x \in X$ y agregarla a R_N^t
    for $a \leftarrow 1$ **to** A **do**
        $\Psi_a^t \leftarrow \text{máx}\left\{0, \frac{1}{|R_P^t|} \sum_{x \in R_P^t} x_a - \frac{\rho}{|R_N^t|} \sum_{x \in R_N^t} x_a\right\}$

Algorithm 5: BMRocchio Hiperesfera (entrenamiento)

Seguidamente se explica en detalle como se obtiene el valor del parámetro para la selección de atributos de las bolsas positivas ρ^+ utilizado en la fórmula (1.3) para calcular el perfil de la clase positiva y el valor del umbral de distancia, ambos en función de los datos de ejemplo.

2.3.5.2. BEMR (Prueba)

Durante la etapa de clasificación, dada una nueva bolsa, se calcula la distancia entre ésta y el perfil de la clase positiva calculado anteriormente en el proceso de aprendizaje. Para calcular la distancia entre la bolsa y el perfil, se utiliza la función de distancia entre bolsas AverageHausdorff definida anteriormente por la fórmula (2.4). Si la distancia obtenida entre ambas bolsas es menor que el umbral de distancia, la bolsa es clasificada como positiva, en otro caso será clasificada como negativa. El algoritmo (6) explica este proceso llevado a cabo por BMRocchio_ Hiperesfera durante la etapa de clasificación.

Input:

- X bolsa a clasificar
- Ψ^+ perfil de la clase positiva
- δ umbral de distancia

Output:

- y etiqueta de clase

```
1 d_H ← distancia AverageHausdorff entre X y Ψ
2 if d_H < δ then
3 |  y ← clase positiva
4 else
5 |  y ← clase negativa
```

Algorithm 6: BMRocchio Hiperesfera (explotación)

A continuación se explica detalladamente como se realiza la estimación de los parámetros que intervienen en el funcionamiento de este algoritmo.

2.3.5.3. Cálculo de los parámetros

Debido a la sensibilidad que presenta el clasificador de Rocchio a la variación de sus parámetros, es necesario encontrar los parámetros que maximicen en mayor grado la exactitud de la clasificación. Para el caso del algoritmo BEMR solamente es necesario optimizar el valor de ρ^+ aplicado en la fórmula (1.3), y adicionalmente encontrar el valor óptimo de δ.

La optimización de ambos parámetros está dada en dos momentos, por lo que estos parámetros se optimizan por separado. El parámetro ρ^+ interviene directamente el la creación del perfil de la clase positiva, por lo que es necesario ajustarlo durante el proceso de entrenamiento, mientras que el parámetro δ solamente interviene en la etapa de clasificación.

La optimización de los parámetros se realiza mediante una 10×10 CV. Para cada partición de entrenamiento se calculan los valores óptimos de ρ^+ y δ. Para la selección del valor óptimo de ρ^+, se entrena el clasificador (se construye el perfil de la clase positiva) utilizando cada valor ρ_i^+ del conjunto $\Lambda^+ = \{0, 1, 2, \ldots, 30\}$. Una vez entrenado el clasificador con el valor ρ_i^+ se evalúa el desempeño de la clasificación para los diferentes valores de δ contenidos en el conjunto $\Delta = \{0{,}1, 0{,}2, 0{,}3, \ldots, 0{,}9\}$. Los valores del conjunto Δ son los términos de una serie que va desde 0,1 hasta 0,9 con un incremento de 0,1. Estos valores están acotados debido a la relación de este parámetro con el resultado de las distancias calculadas entre las bolsas y el perfil de la clase positiva, acotado también entre 0 y 1.

De todos los pares (ρ^+, δ) se selecciona aquel que maximice el resultado de la clasificación según la métrica $F1$. Al finalizar este proceso de optimización de los parámetros se procede a la creación del perfil de la clase positiva utilizando el valor de ρ^+ encontrado. Luego el valor de δ servirá para asignar las etiquetas de clase durante el proceso de clasificación.

2.3.5.4. Análisis de la eficiencia temporal de BEMR

En la etapa de entrenamiento el método para conocer la cantidad de instancias T que contendrá el perfil de la clase positiva presenta una complejidad temporal de $\mathcal{O}(M)$, donde M es la cantidad de bolsas del conjunto de entrenamiento. Luego para crear el perfil positivo es necesario crear T instancias sintéticas utilizando la fórmula de Rocchio, lo que presenta una complejidad temporal de $\mathcal{O}(T \times m \times N)$, donde m es el número de atributos y N es el total de instancias del conjunto de entrenamiento. En resumen, BEMR presenta una complejidad temporal de $\mathcal{O}(T \times m \times N)$ durante la etapa de entrenamiento.

Durante el proceso de clasificación, la nueva bolsa se compara con el perfil de la clase positiva, utilizando AverageHausdorff como distancia entre conjuntos, lo que implica una complejidad temporal de $\mathcal{O}(T \times B \times m)$, donde B es la cantidad de instancias de la nueva bolsa.

3 Estudio experimental

En este capítulo se presentan y discuten los resultados de los experimentos realizados para verificar la eficacia de los algoritmos presentados y por tanto la validez de las hipótesis propuestas. En la Sección (3.1) se explica detalladamente en qué consisten los experimentos, sobre que conjuntos de datos se realizan y cuáles son los clasificadores a comparar, así como las medidas de desempeño utilizadas. Seguidamente, en la Sección (3.2) se presentan y discuten los resultados obtenidos de la comparaciones realizadas entre los algoritmos propuestos. También se realiza en esta sección una comparación entre el desempeño de los clasificadores obtenidos y algoritmos del estado del arte.

3.1. Descripción del marco experimental

En esta sección se presentan los dos experimentos que fueron realizados para verificar la eficacia de los algoritmos propuestos. El objetivo de estos experimentos es determinar si se obtiene una mejoría significativa al aplicar los algoritmos desarrollados en el presente trabajo, comparado con el resultado que se obtendría con el clasificador MIRocchio sobre los mismos conjuntos de datos. El procedimiento consiste en evaluar el desempeño de los clasificadores propuestos, en un primer experimento, para los datos TREC9 y un segundo experimento sobre los datos WIR.

Se separan ambos tipos de conjuntos de debido a que los datos desbalanceados presentes en el problema WIR, imponen dificultades adicionales a la clasificación que provocan cambios en el comportamiento de los clasificadores. Por esta misma razón se usan además medidas de desempeño distintas para ambos casos. Los algoritmos desarrollados serán comparados entre sí y con MIRocchio, y luego se seleccionaran los mejores de cada hipótesis para ser comparados con algoritmos del estado del arte.

Estos experimentos se realizan mediante una validación cruzada validación cruzada de cinco particiones. A continuación en la Sección 3.1.1 se presentan los conjuntos de datos seleccionados para el estudio. Además se exponen en las Secciones (3.1.2) y (3.1.3) los

algoritmos a comparar y los métodos utilizados para medir la eficacia de los clasificadores respectivamente.

3.1.1. Conjuntos de datos

Los experimentos fueron ejecutados sobre 16 conjuntos de datos multinstancia, de ellos 10 balanceados y 6 desbalanceados. Estos conjuntos de datos son del dominio de aplicaciones textuales y pertenecen a dos problemas de clasificación distintos: TREC9 y WIR. En ambos problemas la clase negativa esta compuesta por varios subconceptos, lo cual los hace ideales para probar la hipótesis que propone el usos de la hiperesfera como frontera de decisión entre clases.

El problema de clasificación TREC9 (también conocidos como OHSUMED) consisten en una colección de aproximadamente 54000 artículos médicos donde cada artículo está separado en fragmentos con un máximo de 50 palabras cada uno para poder lograr su representación multinstancia. Estos artículos son representados como bolsas en los que cada fragmento es una instancia del documento (bolsa). Las instancias están compuestas por las frecuencias de aparición de los términos en el documento utilizando la ponderación $TF - IDF$. El problema TREC9 original tiene 4903 clases, pero este problema originalmente de clasificación multiclase, se convirtió en 4903 problemas de clasificación binaria, donde cada problema es referido a uno de los conceptos o categorías originales. De forma que en un problema de clasificación binaria un documento es positivo si pertenece a la categoría relacionada con el problema y es negativo si pertenece a cualquiera de las 4902 categorías restantes. Por eso, en TREC9 la clase negativa está compuesta por 4902 conceptos diferentes. Los datos usados también por (Andrews et al. 2002) son un subconjunto de TREC9 conformados por siete conjuntos de datos. Cada uno de estos conjuntos presenta un balance perfecto entre la clase positiva y la negativa (cada clase contiene 200 bolsas de ejemplo).

La Tabla (3.1) muestra un resumen de las características de los conjuntos de datos del problema TREC9. Por las columnas se muestra en orden el nombre del conjunto, la cantidad de atributos, la cantidad de instancias negativas, positivas y el total de estas. Se muestra además la cantidad de bolsas negativas y positivas, asi como el total de bolsas, y por último se muestra la cantidad de subconceptos que forman las clases.

Por su parte los conjuntos WIR contienen información acerca del problema de recomendación de páginas web índices (Zhou et al. 2005) y han sido usados en (Zafra et al. 2009,

Nombre	Atributos	I -	I +	Instancias	B -	B +	Bolsas	subconceptos
TREC9-1	320	1644	1580	3224	200	200	400	4903
TREC9-2	303	1629	1715	3344	200	200	400	4903
TREC9-3	324	1620	1626	3246	200	200	400	4903
TREC9-4	306	1637	1754	3391	200	200	400	4903
TREC9-5	300	1621	1746	3367	200	200	400	4903
TREC9-6	299	1616	1684	3300	200	200	400	4903
TREC9-7	303	1635	1818	3453	200	200	400	4903

Tabla 3.1: Características de los conjuntos de datos TREC9

2011, Tarragó et al. 2014). Los datos WIR fueron obtenidos a través de varias encuestas realizadas a nueve voluntarios sobre sus preferencias en la web, donde una bolsa es positiva si la página web índice es relevante para el usuario (en este caso el voluntario) y en caso contrario es negativa. Al igual que para los datos TREC9, los conjuntos WIR corresponden a un problema inicialmente con nueve clases (una según la opinión de cada voluntario) llevado a un problema de clasificación binaria. La clase negativa en el caso de los datos WIR solo tiene 8 subconceptos porque el noveno es el que conforma la clase positiva.

Nombre	Atributos	I -	I +	Instancias	B -	B +	Bolsas	subconceptos	IR
WIR-1	304	2867	556	3423	92	21	113	9	4,400
WIR-2	298	2867	556	3423	92	21	113	9	4,400
WIR-3	303	2867	556	3423	92	21	113	9	4,400
WIR-4	303	757	2666	3423	24	89	113	9	3,700
WIR-5	302	757	2666	3423	24	89	113	9	3,700
WIR-6	304	757	2666	3423	24	89	113	9	3,700
WIR-7	303	1713	1710	3423	58	55	113	9	1,054
WIR-8	303	1713	1710	3423	58	55	113	9	1,054
WIR-9	301	1713	1710	3423	58	55	113	9	1,054

Tabla 3.2: Características de los conjuntos de datos WIR

La Tabla (3.2) muestra de manera resumida las características de los conjuntos de datos WIR. En la tabla se muestra tal y como se observa en la Tabla (3.1) los valores referentes a la cantidad de atributos, instancias y bolsas, así como la cantidad de conceptos que conforman las clases del problema. Además se expone en esta tabla la tasa de desbalance IR entre los ejemplos de cada conjunto de datos. Según la Tabla (3.2) el problema WIR esta conformado por nueve conjuntos de datos que van desde WIR-1 hasta WIR-9. Los conjuntos WIR-1,

WIR-2 y WIR-3 presentan un desbalance de clases hacia la clase negativa (mayor cantidad de bolsas negativas que positivas) con un IR (*Inbalance ratio* por sus siglas en inglés) igual a 4,380. Los conjuntos WIR-4, WIR-5 y WIR-6 también se encuentran desbalanceados, pero en estos casos hacia la clase positiva con un IR de 3,708. Finalmente los conjuntos WIR-7, WIR-8 y WIR-9 presentan desbalance imperceptible con un IR de 1,054, por lo que se consideran como balanceados. Los desbalances aparecen porque hay tres usuarios que tienen intereses muy estrechos o exigentes, resultando en muchos negativos, mientras que otros tres tienen intereses muy amplios o imprecisos, resultando en muchos positivos. Este es un problema de aprendizaje complicado debido a sus desbalances y además por tener atributos muy numerosos y dispersos.

Dada la no existencia de suficientes conjuntos de datos multinstancia en los repositorios de datos de aprendizaje automático, no fue posible extender la experimentación. Estos problemas (TREC9 y WIR) son los dos grupos de datos sobre los que serán evaluados los algoritmos clasificadores a comparar. Seguidamente se mencionan y explica brevemente estos algoritmos.

3.1.2. Algoritmos a comparar

Los tres algoritmos obtenidos en este trabajo (IEMR, BPMR y BEMR) son comparados en cuanto a su desempeño con el algoritmo MIRocchio descrito en la Sección (1.3.2.2) y considerado como el antecesor de esta investigación. A demás son comparados estos clasificadores con algoritmos del estado del arte. A continuación se describen brevemente cuáles son los clasificadores del estado del arte que se incluyen en las comparaciones.

El algoritmo clasificador multinstancia SimpleMI realiza un mapeo de las bolsas de ejemplo convirtiéndolas en instancias cuyos atributos son los promedios de los valores de los atributos de cada instancia contenida en la bolsa, convirtiendo el problema multinstancia original a un nuevo problema simpleinstancia. Luego de obtenido el nuevo problema, se le aplica un clasificador simpleinstancia, en este caso el clasificador simpleinstancia C4.5. A pesar de que este algoritmo se basa en un comportamiento relativamente simple, es un algoritmo que presenta un desempeño competitivo con otros exponentes del estado del arte.

MITI (Blockeel et al. 2005) 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 clasificación cada instancia de la

nueva bolsa obtiene una clasificación particular del árbol construido, y la bolsa es clasificada como positiva si contiene al menos una instancia positiva. 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.

El clasificador CitationKNN propuesto por Wang y Zucker en (Wang and Zucker 2000) es una variante de los $k - vecinos$ más cercanos. Este método está basado en un marco teórico que define la relevancia de los 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 esta relacionado con el anterior. Similarmente, un artículo que es citado por una publicación posterior (el citador) se considera que esta relacionado con el trabajo posterior. Los citadores $C - cercanos$ de X son las bolsas cuyos C vecinos más cercanos incluyen a X. El concepto de R referencias más cercanas coincide con al algoritmo original, donde $k = R$. En cambio, los citadores es un nuevo concepto que hace más robusto al CitationKNN. 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 erróneamente la clasificación a favor de etiquetas positivas, mientras que lo inverso no es posible bajo la hipótesis multinstancia estándar.

En la siguiente sección se explica cómo será realizada la validación de los resultados y bajo que métricas se evalúa el desempeño de los clasificadores en cuestión.

3.1.3. Validación

La evaluación de la clasificación fue llevada a cabo mediante un esquema de validación cruzada (CV, por las siglas del termino ingles *cross validation*) 5 CV. En una validación cruzada el conjunto de datos en cuestión es dividido en k particiones aleatorias con la misma cantidad de ejemplos aproximadamente. Una vez dividido el conjunto se procede a entrenar el clasificador con las primeras $k - 1$ particiones y luego se evalúa el desempeño con la partición sobrante. Este proceso se repite k veces, rotando los conjuntos de entrenamiento y prueba de manera que se pruebe el desempeño del clasificador sobre cada partición tras

ser entrenado con las otras $k-1$ particiones restantes. Los valores obtenidos sobre cada partición son registrados y promediados para obtener varias medidas de desempeño.

Con el fin de obtener resultados más fiables los conjuntos de datos fueron particionados con anterioridad, obteniendo 5 subconjuntos para cada conjunto de datos. Todos los clasificadores fueron validados utilizando las mismas particiones creadas para cada conjunto en lugar de obtenerlas al azar, reduciendo la intervención del factor de aleatoriedad en los resultados obtenidos.

Los clasificadores basados en perfiles de clase a nivel de bolsas presentan una fuerte dependencia del azar en el proceso de submuestreo de las instancias de las bolsas durante la etapa de entrenamiento por lo que son considerados como clasificadores no deterministas, lo cual hace que los valores de los resultados varíen aún cuando se realizan los experimentos sobre los mismos conjuntos de datos. Para mitigar la obtención de resultados alterados debido al factor de aleatoriedad se decidió repetir los experimentos 10 veces para cada clasificador basado en perfiles de clase a nivel de bolsas. Por esta razón los resultados obtenidos para cada uno de estos algoritmo fueron consolidados promediando los valores de desempeño, apuntando a disminuir las diferencias entre los resultados de las corridas. Los valores que se exponen en este capítulo corresponden solamente a los resultados consolidados, los resultados individuales de los 10 experimentos realizados para estos algoritmos no serán mostrados en este trabajo.

Para medir la calidad de los clasificadores existen varias métricas propuestas en la literatura, entre las que vale destacar *accuracy, Kappa, precision, recall, F1, AUC, GMean, etc.* como las reconocidas. Por excelencia las medidas *accuracy* y *Kappa* son utilizadas para medir el desempeño de los clasificadores sobre los conjuntos de datos balanceados, sin embargo *accuracy* a pesar de ser una medida reconocida no corrige aquellas concordancia que ocurren al azar. Por otra parte las métricas AUC y *GMean* han sido utilizadas por la literatura para medir el desempeño sobre los conjuntos desbalanceados. Sin embargo AUC ha sido recientemente cuestionada en nuevas investigaciones (Hanczar et al. 2010). Estas afirman que el AUC es muy ruidoso como medida de clasificación y que presenta algunos otros problemas respecto a la comparación de modelos. Un estudio crítico propuesto por (Lobo et al. 2008, Hand 2009), sugiere fallas frecuentes en la obtención de estimados de AUC validos y confiables. De esta manera, el valor práctico de la métrica AUC ha sido cuestionado (Hand 2009), surgiendo la posibilidad de que esta medida introduzca más desaciertos que aciertos en cuanto a la calidad de la clasificación.

Por estos motivos las medidas de desempeño usadas en estos experimentos para evaluar la

calidad de la clasificación fueron *Kappa* para los conjuntos de datos balanceados y *GMean* para los desbalanceados. Seguidamente se explica el significado de cada una de estas medidas y cómo se obtienen, pero primero se hace necesario explicar qué es la matriz de confusión y cómo interviene en la obtención de estas métricas.

En un problema de clasificación binaria (dos clases) hay cuatro posibles salidas, las cuales están representadas por la matriz de confusión que se muestra en la Tabla (3.3). Esta puede considerarse la información más básica sobre el desempeño de la clasificación, a partir de la cual se definen medidas más avanzadas como las que se describen a continuación.

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

Tabla 3.3: Matriz de confusión para un problema de dos clases.

- *Kappa*: mide la proporción de aciertos que pueden atribuirse propiamente al clasificador y no a la casualidad, es decir que intenta corregir el grado de concordancia (o exactitud) substrayendo la proporción de éxitos atribuibles al azar. Una forma de definir esta medida es la siguiente

$$kappa = \frac{accuracy - P_c}{1 - P_c} \tag{3.1}$$

donde P_c es la probabilidad de éxito debido a la casualidad. Para problemas de clasificación binarios P_c puede definirse como

$$P_c = \frac{(TP + FN)(TP + FP) + (FP + TN)(FN + TN)}{TP + FP + TN + FN}$$

El rango de *Kappa* es desde -1 (total desacuerdo), hasta 1 (acuerdo total). Un valor de kappa igual a cero significa una clasificación aleatoria. Por lo tanto un clasificador que obtenga un valor de *Kappa* mayor que cero indica que funciona razonablemente bien. En resumen, el estadístico *Kappa* es usado para medir concordancia entre las clasificaciones realizadas sobre un conjunto de datos pero corrigiendo aquellas concordancia que ocurren al azar. Aunque esta medida es estadísticamente robusta no tiene en cuenta el costo de los distintos tipos de errores, lo que hace que esta métrica no sea fiable para conjuntos de datos desbalanceados y sí apropiada para conjuntos

balanceados.

- *GMean*: obtiene el desempeño general del clasificador pero teniendo en cuenta los aciertos de ambas clases por separado. Esta métrica calcula la media geométrica entre los desempeños de la clasificación de ambas clases. La medida de desempeño *GMean* se puede definir como

$$gmean = \sqrt{(\frac{TN}{TN + FP} \times \frac{TP}{TP + FN})} \tag{3.2}$$

  donde $TN/(TN + FP)$ y $TP/(TP + FN)$ representan la proporción de aciertos positivos y negativos respectivamente. La propiedad de evaluar el desempeño de ambas clases por separado, independientemente del número de ejemplos de cada una, hace de *GMean* una métrica robusta y fiable para medir el desempeño de los clasificadores sobre los conjuntos de datos desbalanceados.

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, y se calcularon las medidas del desempeño a partir de la matriz de confusión agregada. Dado el carácter de estas medidas serán utilizadas como se indicaba anteriormente, *Kappa* para medir el desempeño de los clasificadores sobre los conjuntos de datos balanceados mientras que para los datos desbalanceados será utilizada la métrica *GMean*.

3.2. Presentación y discusión de los resultados

La Tabla (3.4) muestra los valores de *Kappa* obtenidos para los clasificadores propuestos sobre los conjuntos de datos TREC9. Esta muestra los clasificadores comparados por columnas y en las filas los conjuntos de datos TREC9-1 hasta TREC9-10. La última fila muestra el desempeño del clasificador de manera general sobre todos los conjuntos de datos calculado como el promedio de la columna. El mejor valor de desempeño para cada conjunto de datos se encuentra resaltado en negritas.

En la Tabla (3.4) se observa que los clasificadores que crean perfiles de clase a nivel de instancia (es decir, MIRocchio e IEMR) obtienen mejores resultados de manera general sobre los datos TREC9 que aquellos clasificadores que crean perfiles de clase a nivel de bolsa (BEMR y BPMR).

Algoritmos	MIRocchio	IEMR	BPMR	BEMR
TREC9-1	0,895	**0,905**	0,875	0,825
TREC9-2	0,665	**0,725**	0,312	0,517
TREC9-3	0,715	**0,780**	0,608	0,694
TREC9-4	**0,710**	0,685	0,574	0,630
TREC9-7	**0,635**	0,600	0,496	0,520
TREC9-9	0,130	**0,380**	0,152	0,282
TREC9-10	**0,605**	0,580	0,557	0,601
Promedio	0,622	**0,665**	0,510	0,581

Tabla 3.4: Resultados de los clasificadores propuestos y el clasificador MIROCCHIO según *Kappa* sobre los conjuntos de datos TREC9

Se puede ver además que aquellos clasificadores que usan la hiperesfera como frontera de decisión entre clases muestran mejores resultados comparados con las versiones que usan el hiperplano. Siendo más específicos, el clasificador IEMR mejora el resultado del MIROCCHIO, obteniendo cuatro victorias y mayor promedio general. BEMR a su vez, mejora el desempeño de BPMR para los datos TREC9 según el estadístico *Kappa* con seis victorias frente a una derrota y mejor promedio de forma general. Se dice que el algoritmo A obtiene una victoria frente al algoritmo B sobre la medida de desempeño M si $M(A) > M(B)$. Por lo que podemos afirmar que tanto la utilización de los perfiles de clase a nivel de instancia, como la hiperesfera como frontera de decisión entre clases, mejoran la calidad de la clasificación sobre los datos TREC9.

Luego, la unión de estas dos variantes mencionadas (la creación de los perfiles de clase a nivel de instancia y la utilización de hiperesfera como frontera de decisión entre clases, ambas reflejadas en el algoritmo IEMR) muestran un mejor desempeño de manera general, superando al clasificador MIROCCHIO en un 4,3 % según el estadístico *Kappa*. Si comparamos el desempeño de la clasificación (*Kappa*) entre este algoritmo y MIROCCHIO (por ser los dos con mejores resultados obtenidos sobre los datos TREC9), de manera general ambos mantienen un desempeño comparable. Este desempeño es ligeramente superado por IEMR con cuatro victorias sobre la tres de MIROCCHIO. Podemos resaltar que los valores obtenidos por IEMR, a pesar de no ser los mejores en cada caso, son buenos valores para la medida de desempeño Kappa.

Como el número de conjuntos de datos para TREC9 es muy reducido y además el número de victorias que presentan los clasificadores comparados tiene poca diferencia, las pruebas

estadísticas no cuentan con la suficiente potencia para obtener diferencias significativas entre estos.

Por otro lado, la Tabla (3.5) muestra los resultados obtenidos de la experimentación sobre los conjuntos de datos WIR. La tabla muestra los clasificadores comparados por columnas y en las filas los conjuntos de datos WIR-1 -WIR-9. La última fila muestra el desempeño del clasificador de manera general sobre todos los conjuntos de datos calculado como el promedio de los valores de la columna. El mejor valor de desempeño para cada conjunto de datos se encuentra resaltado en negritas.

Algoritmos	MIRocchio	IEMR	BPMR	BEMR
WIR-1	0,480	0,701	**0,738**	0,732
WIR-2	0,523	0,720	**0,746**	0,743
WIR-3	0,555	0,706	**0,761**	0,736
WIR-4	**0,864**	0,696	0,717	0,665
WIR-5	**0,899**	0,729	0,747	0,650
WIR-6	**0,784**	0,618	0,758	0,614
WIR-7	**0,744**	0,563	0,736	0,604
WIR-8	**0,718**	0,519	0,708	0,582
WIR-9	**0,651**	0,564	0,636	0,620
Promedio	0,691	0,646	**0,727**	0,661

Tabla 3.5: Resultados de los clasificadores propuestos y el clasificador MIROCCHIO según *GMean* sobre los conjuntos de datos WIR

Los resultados del experimento realizado sobre los datos WIR arrojaron que los clasificadores que crean los perfiles de clase a nivel de bolsa (BPMR y BEMR) ofrecen mejores resultados de manera general sobre estos conjuntos de datos que aquellos que crean perfiles de clase a nivel de instancia, integrando en su diseño la hipótesis de proporción de instancias positivas (MIROCCHIO e IEMR), según la Tabla (3.5). A pesar de que BPMR obtiene solamente tres victorias sobre MIROCCHIO, obtiene mejor promedio general, y supera en todos los casos y en promedio general al IEMR.

Además, aquellos clasificadores que usan el hiperplano como frontera de decisión entre clases mejoran los resultados sobre los que utilizan la hiperesfera. El clasificador MIRocchio supera en seis victorias frente a tres derrotas y de manera general al algoritmo IEMR, mientras que el BPMR mejora en todos los casos al BEMR. En otras palabras, los clasificadores MIROCCHIO y BPMR mejoran los desempeños de los algoritmos IEMR y BEMR respectivamente para los datos WIR según la métrica *GMean*. Estos resultados sugieren

que tanto la utilización de los perfiles de clase a nivel de bolsa, como el hiperplano como frontera de decisión entre clases, mejoran (aunque no de manera significativa) la calidad de la clasificación sobre los datos WIR.

Es apreciable también el desempeño del clasificador BPMR por encima de los otros algoritmos comparados, de manera general en promedio este algoritmo supera al MIROCCHIO en un 3,6 % unidades en *GMean*. Comparando este algoritmo con el MIROCCHIO podemos observar que el clasificador MIROCCHIO a pesar de obtener mayor cantidad de victorias sobre su oponente, presenta una menor calidad de la clasificación de manera general como promedio sobre los conjuntos WIR que el BPMR según los valores de *GMean* mostrados en la Tabla (3.5).

Aunque no se puede concluir que el clasificador BPMR supera significativamente los resultados obtenidos por MIROCCHIO si se aprecia que este BPMR muestra un desempeño sostenido con poca variabilidad sobre todos los conjuntos de datos WIR. Sin embargo el MIROCCHIO presenta problemas para enfrentar el desbalance hacia la clase negativa, como es el caso de los conjuntos de datos WIR-1, WIR-2 y WIR-3 como se observa en la Tabla (3.5).

De manera general se presenta el clasificador BPMR como el algoritmo con mejores resultados obtenidos en los experimentos realizados sobre los conjuntos de datos WIR. Para los datos WIR al igual que en el caso anterior como no existen suficientes conjuntos de datos, no es posible aplicar pruebas de significación estadística a los resultados.

3.2.1. Comparación con los algoritmos del estado del arte

Para comparar el desempeño de los algoritmos obtenidos han sido tomados además varios algoritmos de propósito general representativos del estado del arte. En esta sección se comparan los algoritmos IEMR y BPMR (por ser los de mejores resultados sobre los datos TREC9 y WIR respectivamente) con los clasificadores multinstancia SIMPLEMI, MITI y el CITATIONKNN presentados en la Sección (3.1.2).

Las Tablas (3.6) y (3.7) muestran los resultados de las comparaciones entre estos algoritmos y los clasificadores IEMR sobre los conjuntos de datos TREC9 y BPMR sobre los datos WIR respectivamente.

En la Tabla (3.6) se observa un desempeño superior del clasificador IEMR sobre los demás algoritmos seleccionados del estado del arte. Este algoritmo muestra el mejor desempeño para cinco conjuntos de los siete que forman los datos TREC9 sobre los demás clasificadores.

Algoritmos	IEMR	SimpleMI	MITI	CitationKNN
TREC9-1	**0,905**	0,880	0,905	0,210
TREC9-2	**0,725**	0,660	0,400	0,040
TREC9-3	0,780	**0,800**	0,715	0,035
TREC9-4	0,685	**0,790**	0,615	-0,030
TREC9-7	**0,600**	0,570	0,445	-0,070
TREC9-9	**0,380**	0,230	0,090	-0,050
TREC9-10	**0,580**	0,510	0,500	-0,070
Promedio	**0,665**	0,634	0,524	0,009

Tabla 3.6: Comparación entre el desempeño de los clasificadores IEMR y clasificadores del estado del arte según *Kappa* sobre los conjuntos de datos TREC9

IEMR presenta además el mejor resultado de manera general descrito como el promedio sobre los conjuntos de datos balanceados TREC9 según *Kappa*. Estos resultados han sido obtenidos de la experimentación sobre un número reducido de conjuntos de datos, por lo que no ha sido posible realizar pruebas estadísticas sobre los resultados. Por lo tanto, no podemos hablar de significación estadística.

Algoritmos	BPMR	SimpleMI	MITI	CitationKNN
WIR-1	**0,738**	0,489	0,733	0,629
WIR-2	0,746	0,525	**0,772**	0,466
WIR-3	0,761	0,572	**0,795**	0,469
WIR-4	0,717	0,716	0,571	**0,751**
WIR-5	**0,747**	0,710	0,492	0,691
WIR-6	**0,758**	0,710	0,404	0,719
WIR-7	0,736	**0,744**	0,465	0,593
WIR-8	**0,708**	0,673	0,361	0,637
WIR-9	0,636	**0,750**	0,428	0,664
Promedio	**0,727**	0,654	0,558	0,624

Tabla 3.7: Comparación entre el desempeño de los clasificadores BPMR y clasificadores del estado del arte según *GMean* sobre los conjuntos de datos WIR

Se observa en la Tabla (3.7) el desempeño superior del clasificador BPMR sobre los algoritmos del estado del arte presentados. BPMR presenta de manera general mejor rendimiento en cuanto a *GMean* para los datos WIR, obteniendo mayor cantidad de victorias (cuatro victorias) frente a la competencia. Este algoritmo obtiene también el mejor promedio de

desempeño sobre los datos WIR. Al igual que en los casos anteriores, los valores han sido obtenidos como resultado de la experimentación sobre un número reducido de conjuntos de datos, por lo que no ha sido posible realizar pruebas estadísticas sobre estos.

Aunque no podemos hablar de significación estadística en los resultados obtenidos, pero a pesar de esto se observa que los clasificadores IEMR y BPMR son competitivos, y superiores en algunos casos, con respecto a los algoritmos seleccionados del estado del arte.

Conclusiones

Como resultado de esta investigación han sido desarrollado tres algoritmos para la clasificación textual multinstancia. El primero, denominado IEMR, ofrece un enfoque diferente a la solución planteada por MIRocchio, utilizando una hiperesfera como frontera de decisión entre clases. Los otros dos algoritmos, denominados BPMR y BEMR, basan su funcionamiento en el clasificador de Rocchio e introducen el concepto de perfiles de clase a nivel de bolsa.

La experimentación sobre estos algoritmos arrojo varios resultados importantes. Primeramente, el rendimiento de los algoritmos IEMR y BEMR demostró que el uso de la Hiperesfera como frontera de decisión entre clases aumenta el desempeño de los clasificadores basados en Rocchio para algunos problemas pero no para todos. Los resultados presentados muestran que para los datos TREC9 aquellos algoritmos que implementan la hipótesis que sugiere la hiperesfera como frontera de decisión entre clases, presentan mejor desempeño que los que utilizan el hiperplano. En cambio, para los datos WIR ocurre lo contrario, la utilización del hiperplano muestra mejores resultados que la hiperesfera. Este resultado se cree está atribuido a que para el problema TREC9 la clase negativa está conformada por un número extremadamente superior de subconceptos (4903 conceptos) que para los datos WIR (9 conceptos). Debido al tamaño de la experimentación no se puede afirmar categóricamente la validez de esta hipótesis, pero los resultados sugieren que la utilización de la hiperesfera pudiese ser una buena solución para problemas donde la clase negativa esté compuesta por un número elevado de subconceptos como el caso TREC9. Finalmente se encontró el IEMR como el clasificador propuesto con mejor desempeño para los problemas TREC9.

Por otra parte, los resultados obtenidos por el clasificador BPMR superaron de manera general a los demás algoritmos para el problema WIR. Estos resultados sugieren que para problemas con las mismas características que WIR es conveniente la creación de perfiles de clase a nivel de bolsa. Además los algoritmos que implementan las hipótesis presentadas en esta investigación, a pesar de no mejorar los resultados para todos los conjuntos de datos del

problema WIR, presentan un comportamiento más uniforme frente al clasificador MIRocchio sobre los datos WIR. La no obtención de los resultados esperados para el desempeño de estos algoritmos basados en perfiles de clase a nivel de bolsa puede estar asociado quizás a una selección inadecuada del parámetro k, influyendo directamente en la conformación de los perfiles de clase. Otro punto interesante es que el algoritmo MIROCCHIO presenta dificultades para enfrentar los problemas WIR-1, WIR-2 y WIR-3 mientras que los demás clasificadores muestran resultados mantenidos sobre todos los conjuntos de datos, lo que siguiere que estos algoritmos pudiesen ser utilizados en aquellos problemas donde se presente un desbalance de clases hacia la clase negativa, sin verse afectados por esta característica.

Los algoritmos desarrollados en esta investigación, a pesar de que debido al tamaño de la experimentación no se puede afirmar estadísticamente la validez de las hipótesis subyacentes tras cada uno, son competitivos y mejoran en varios casos a los algoritmos del estado del arte. Tanto el algoritmo IEMR como el BPMR obtienen mayor cantidad de victorias y mejor promedio general sobre los algoritmos seleccionados del estado del arte para los problemas TREC9 y WIR respectivamente.

Recomendaciones

A partir de los resultados alcanzados en el desarrollo del presente trabajo, han aparecido una serie de recomendaciones que pudiesen constituir trabajos futuros.

El reducido número de conjuntos de datos que intervinieron en los experimentos, no hizo posible la aplicación de pruebas estadísticas para validar las hipótesis plantadas al inicio de esta investigación. Por este motivo se recomienda extender este estudio a más conjuntos de datos con las características presentadas por los problemas TREC9 y WIR. Además sería conveniente observar el comportamiento de estos algoritmos sobre otros problemas del área de la clasificación textual u otros dominios de aplicación.

A pesar de que se muestra en esta investigación al algoritmo BPMR como un clasificador competente con algoritmos del estado del arte, no se obtuvieron los resultados esperados para la hipótesis que implementa. La calidad de la clasificación obtenida (inferior a la esperada) para los algoritmos basados en perfiles de clase a niveles de bolsa, se supone puede estar dada por una selección inadecuada de la cantidad de instancias seleccionadas de cada bolsa de ejemplo durante el proceso de submuestreo. Se propone por esta razón introducir variantes en nuevas investigaciones que realicen la selección del valor de k de una mejor manera.

Bibliografía

Andrews, S., Tsochantaridis, I. and Hofmann, T.: 2002, Support Vector Machines for Multiple-Instance Learning, *Advances in Neural Information Processing Systems 15 (NIPS)*, pp. 561–568. 14, 53

Auer, P. and Ortner, R.: 2004, A Boosting Approach to Multiple Instance Learning, *in* J. Boulicaut, F. Esposito, F. Giannotti and D. Pedreschi (eds), *Machine Learning: ECML 2004*, Vol. 3201 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 63–74. 14

Bi, Y., Bell, D., Wang, H., Guo, G. and Guan, J.: 2007, Combining multiple classifiers using dempster's rule for text categorization, *Applied Artificial Intelligence* **21**(3), 211–239. 35

Blockeel, H., Page, D. and Srinivasan, A.: 2005, Multi-instance tree learning, *Proceedings of the 22nd international conference on Machine learning*, pp. 57–64. 14, 55

Chevaleyre, Y. and Zucker, J.: 2001, Solving Multiple-Instance and Multiple-Part Learning Problems with Decision Trees and Rule Sets. Application to the Mutagenesis Problem, *Canadian Conference on AI*, pp. 204–214. 14

Cohen, W. and Singer, Y.: 1999, Context-sensitive learning methods for text categorization, *ACM Transactions on Information Systems* **17**(2), 141–173. 35

Dietterich, T. G., Lathrop, R. H. and LozanoPerez, T.: 1997, Solving the multiple instance problem with axis-parallel rectangles, *Artificial Intelligence* **89**(1-2), 31–71. 14

Foulds, J.: 2008, *Learning Instance Weights in Multi-Instance Learning*, Master Thesis, University of Waikato, Hamilton, New Zealand. 14

Frank, E. and Xu, X.: 2003, Applying propositional learning algorithms to multi-instance data, *Technical report 06/03*, Department of Computer Science, University of Waikato, Hamilton, New Zealand. 14

Guo, G., Wang, H., Bell, D., Bi, Y. and Greer, K.: 2003, Using knn model-based approach for automatic text, *Proc. ODBASE'03, the 2nd Internat. Conf. on Ontologies, Database and Applications of Semantics*, LNCS, pp. 986–996. 35

Hanczar, B., Hua, J., Sima, C., Weinstein, J., Bittner, M. and Dougherty, E.: 2010, Small-sample precision of ROC-related estimates, *Bioinformatics* **26**(6), 822–830. 57

Hand, D.: 2009, Measuring classifier performance: A coherent alternative to the area under the ROC curve, *Machine Learning* **77**, 103–123. 57

Ittner, D. J., Lewis, D. D. and Ahn, D. D.: 1995, Text categorization of low quality images, *In Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 301–315. 19, 35

Lobo, J., Jiménez-Valverde, A. and Real, R.: 2008, AUC: a misleading measure of the performance of predictive distribution models, *Global Ecology and Biogeography* **17**, 145–151. 57

Maron, O. and Lozano-Pérez, T.: 1998, A Framework for Multiple-Instance Learning, *Advances in Neural Information Processing Systems (NIPS Conference)*, MIT Press, pp. 570–576. 14

Mitchell, T. M.: 1997, *Machine learning*, McGraw Hill series in computer science, McGraw-Hill, New York, NY, USA. 7

Moschitti, A.: 2003, A Study on Optimal Parameter Tuning for Rocchio Text Classifier, *ECIR*, LNCS2633, pp. 420–435. 35, 46

Ramon, J. and De Raedt, L.: 2000, Multi Instance Neural Networks, *Attribute-Value and Relational Learning: Crossing the Boundaries.*, pp. 53–60. 14

Rocchio, J.: 1971, Relevance feedback in information retrieval, *in* G. Salton (ed.), *The Smart Retrieval System-Experiments in Automatic Document Processing*, Prentice Hall, pp. 313–323. 1, 19

Schapire, R., Singer, Y. and Singhal, A.: 1998, Boosting and Rocchio applied to text filtering, *Proc. SIGIR 1998: the 21st Annual Internat. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 215–223. 35

Tarragó, D. S.: 2014, *Algoritmos para la clasificación multinstancia*, Tesis doctoral, Universidad de Granada, Granada. 2, 12, 16, 17, 23, 25, 35, 36

Tarragó, D. S., Cornelis, C., Bello, R. and Herrera, F.: 2014, A Multi-Instance Learning Wrapper Based on the Rocchio Classifier for Web Index Recommendation, *Knowledge-Based Systems* **59**, 173–181. 54

Wang, J. and Zucker, J.-D.: 2000, Solving the Multiple-Instance Problem: A Lazy Learning Approach, *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1119–1126. **URL:** *http://portal.acm.org/citation.cfm?id=645529.757771* 15, 22, 38, 56

Xu, X.: 2003, *Statistical Learning in Multiple Instance Problems*, Master Thesis, University of Waikato, Hamilton, New Zealand. 14

Xu, X. and Frank, E.: 2004, Logistic Regression and Boosting for Labeled Bags of Instances, *Proc. of the PacificAsia Conf. on Knowledge Discovery and Data Mining*, Springer-Verlag, pp. 272–281. 14, 15

Yager, R. R.: 1988, On ordered weighted averaging aggregation operators in multicriteria decisionmaking, *Systems, Man and Cybernetics, IEEE Transactions on* **18**(1), 183–190. 04749. 39

Yang, Y. and Pedersen, J.: 1997, A comparative study on feature selection in text categorization, *Proceedings of ICML-97*, Nashville, US, pp. 412–420. 35

Zafra, A., Gibaja, E. and Ventura, S.: 2011, Multiple Instance Learning with Multiple Objective Genetic Programming for Web Mining, *Applied Soft Computing* **11**(1), 93–102. 23, 54

Zafra, A., Romero, C., Ventura, S. and Herrera-Viedma, E.: 2009, Multi-instance genetic programming for web index recommendation, *Expert Systems with Applications* **36**(9), 11470–11479. 53

Zhang, Q. and Goldman, S. A.: 2001, EM-DD: An improved multiple-instance learning technique, *Advances in Neural Information Processing Systems 14 (NIPS Conference)*, Vol. 1-2, pp. 1073–1080. 14

Zhou, Z., Jiang, K. and Li, M.: 2005, Multi-Instance Learning Based Web Mining, *Applied Intelligence* **22**, 135–147. 22, 53

Zitzler, E., Laumanns, M. and Thiele, L.: 2001, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, *Technical report 103*, Gloriastrasse 35, CH-8092 Zurich, Switzerland. 23