UCLV
Universidad Central
"Marta Abreu" de Las Villas

MFC
Facultad de Matemática
Física y Computación

Departamento

Ciencia de la Computación

TRABAJO DE DIPLOMA

Título del trabajo: Agrupamiento en problemas de tipo multi-instancias

Autor del trabajo: Adrian Tomás de la Viña Espinosa

Tutora: Dra. Leticia Arco García

Santa Clara, junio de 2018

A la memoria de mi abuela María Amaranta.

Agradecimientos

Agradezco en primer lugar a mis padres Maité y Tomás por haberme apoyado siempre y ayudado a convertirme en la persona que soy hoy.

A mi hermano Yefgueny por estar siempre a mi lado.

A mi tutora Dra. Leticia Arco García por todo su esfuerzo, dedicación y paciencia que tuvo hacia mí.

A mis amigos, por haberme ayudado durante todos estos años, por estar siempre ahí.

Resumen

El aprendizaje multi-instancias es un nuevo tipo de aprendizaje automatizado que ha recibido gran interés recientemente por su capacidad para manejar cierto tipo de ambigüedad en los problemas de aprendizaje. Sin embargo, implementaciones de algoritmos de aprendizaje multi-instancias no supervisados que se han reportado en la literatura no se encuentran disponibles, y aquellas que aparecen se han implementado de manera aislada, lo que limita la comparación de nuevos métodos que se creen con los ya establecidos. De ahí que constituye un problema la no disponibilidad en una única plataforma de algoritmos de aprendizaje multi-instancias no supervisados que permita su estudio y desarrollo. Por ello, el objetivo de esta investigación consiste en incorporar al Weka los principales algoritmos, así como aquellas transformaciones tanto a los datos como a los métodos simple-instancia existentes, que permitan explorar variantes del agrupamiento multi-instancias, mediante la implementación de paquetes. Los principales resultados obtenidos son: (1) la incorporación a Weka de los algoritmos multi-instancias BAMIC y COSMIC, tres filtros que permiten transformar problemas multi-instancias en problemas simple-instancia y la adaptación de los métodos Canopy y HierarchicalClusterer para el manejo de datos multi-instancias, permitiendo realizar un estudio inicial de las variantes del agrupamiento multi-instancias, y concluyendo que BAMIC utilizando la distancia promedio de Hausdorff para comparar bolsas fue el algoritmo ganador.

Abstract

Multi-instance learning is a new type of machine learning that has received great interest recently because of its ability to handle a certain type of ambiguity in learning problems. However, implementations of unsupervised multi-instance learning algorithms that have been reported in the literature are not available, and those that appear have been implemented in isolation, which limits the comparison of new methods that are created with the already established. Hence, the unavailability in a single platform of unsupervised multi-instance learning algorithms that allows for their study and development is a research problem. Therefore, the objective of this research is to incorporate into Weka the main algorithms, as well as those transformations both to the data and to the existing simple-instance methods, which allow exploring variants of the multi-instance clustering, through the implementation of packages. The main results obtained are: (1) the incorporation into Weka of the BAMIC and COSMIC multi-instance algorithms, three filters that allow the transformation of multi-instance problems into simple-instance problems and the adaptation of the Canopy and HierarchicalClusterer methods for the management of multi-instance data, allowing an initial study of the variants of the multi-instance clustering, and concluding that BAMIC using Hausdorff's average distance to compare bags was the winning algorithm.

Tabla de contenidos

Introducción

El aprendizaje multi-instancias es un nuevo tipo de aprendizaje automatizado que ha recibido gran interés recientemente por su capacidad para manejar cierto tipo de ambigüedad en los problemas de aprendizaje, siendo una especie de vínculo entre el aprendizaje supervisado tradicional y el aprendizaje relacional (Dietterich, Lathrop and Lozano-Pérez, 1997). En el aprendizaje automático, la clasificación multi-instancias tiene como objetivo construir un modelo matemático, a partir de un conjunto de ejemplos, que permita clasificar objetos descritos por múltiples vectores de atributos. En el aprendizaje supervisado estándar cada ejemplo es una instancia que consiste en un vector de atributos, aumentado con una etiqueta de decisión. La tarea es aprender una función que prediga la etiqueta de decisión de un ejemplo, dado el vector de atributos (Mitchel, 1997). En tanto, en el aprendizaje multi-instancias cada ejemplo consiste en una bolsa de instancias. El término bolsa evoca la imagen de un saco que contiene las instancias sin orden ni concierto, pudiendo estar incluso repetida la misma instancia dentro de la bolsa. Cada bolsa tiene asociada una etiqueta de decisión, pero las instancias en sí no están etiquetadas.

Existen varias propuestas que abordan el aprendizaje multi-instancias supervisado (Wu *et al.*, 2015; Carbonneau, 2016; Hajimirsadeghi and Mori, 2017; Roy, Banerjee and Murino, 2017); sin embargo, existen muchos problemas que requieren un tratamiento multi-instancias sin supervisión, de ahí que esta investigación se focalice en este tipo de aprendizaje. Este último, comprende dos tareas principales, las cuales son: análisis de agrupamiento y minería de regla de asociación. En la primera el proceso de aprendizaje consiste en la construcción de grupos, en la cual los ejemplos similares son puestos juntos y los menos similares son separados. En la segunda, el proceso de aprendizaje consiste en la obtención de reglas que muestran relaciones desconocidas en los datos (Herrera *et al.*, 2016). Por un lado, las implementaciones de algoritmos de aprendizaje multi-instancias no supervisados que se han reportado en la literatura no se encuentran disponibles, y aquellas que aparecen se han implementado de manera aislada, lo que limita la comparación de nuevos métodos que se creen con los ya establecidos. Por otro lado, es política del laboratorio de Inteligencia Artificial del Centro de Investigaciones en Informática (CII) de la Universidad Central “Marta Abreu” de Las Villas

(UCLV) incorporar al Weka todos aquellos métodos de aprendizaje automatizado que permitan el estudio, análisis y comparación de los algoritmos existentes. Todo ello constituye una problemática a la cual aún no se le ha dado respuestas definitivas, lo cual justifica el **planteamiento del problema** siguiente: La no disponibilidad en una única plataforma de algoritmos de aprendizaje multi-instancias no supervisados que permita su estudio y desarrollo.

El **objetivo general** del trabajo de diploma consiste en incorporar al Weka los principales algoritmos, así como aquellas transformaciones tanto a los datos como a los métodos simple-instancia existentes, que permitan explorar variantes del agrupamiento multi-instancias, mediante la implementación de paquetes. Este se desglosa en los siguientes **objetivos específicos**:

1. Identificar las características y relaciones de las instancias dentro de las bolsas, los elementos esenciales del aprendizaje multi-instancias supervisado y su adaptación al no supervisado, los principales algoritmos no supervisados y las medidas asociadas a éstos.
2. Implementar paquetes para Weka que permitan realizar el agrupamiento multi-instancias ya sea a partir de algoritmos ad-hoc, algoritmos simple-instancia actualizados o envoltorios para algoritmos simple-instancia, así como las medidas que permiten comparar bolsas.
3. Verificar y validar los algoritmos incorporados al Weka a partir de su ejecución sobre colecciones de problemas multi-instancias reportados en la literatura.

Las preguntas de investigación planteadas son:

- ¿Cuáles son las características esenciales de los algoritmos de agrupamiento para problemas multi-instancias?
- ¿Cómo diseñar un paquete para Weka que encapsule un algoritmo de agrupamiento multi-instancias?
- ¿Cómo incorporar en Weka distancias entre bolsas?
- ¿Cómo crear filtros en Weka que permitan transformar datos multi-instancias en simple-instancia?

- ¿Qué medidas permiten validar algoritmos de agrupamiento multi-instancias y cuáles conjuntos de datos son los más utilizados a tales efectos?

Después de haber realizado el marco teórico se formuló la siguiente hipótesis de investigación como presunta respuesta a las preguntas de investigación: La creación de paquetes que encapsulen las transformaciones y los algoritmos de agrupamiento multi-instancias permite comparar los nuevos métodos con los clásicos ya existentes en una misma plataforma.

Para lograr los objetivos trazados y demostrar la hipótesis establecida se acometieron las tareas de investigación siguientes:

1. Estudio de los principales conceptos del aprendizaje multi-instancias.
2. Análisis de los elementos esenciales del aprendizaje multi-instancias supervisado y su adaptación al no supervisado.
3. Identificación de las principales medidas para comparar bolsas.
4. Estudio de los principales algoritmos de agrupamiento para problemas multi-instancias.
5. Estudio del diseño de la herramienta Weka y las formas de crear paquetes de algoritmos de agrupamiento.
6. Obtención de las colecciones de problemas multi-instancias.
7. Diseño de paquetes para Weka que encapsulen los principales algoritmos de agrupamiento multi-instancias.
8. Creación de filtros que permitan transformar problemas multi-instancias en simple-instancia.
9. Transformación de algoritmos simple-instancia en algoritmos multi-instancias.
10. Implementación de los paquetes para Weka que encapsulen los principales algoritmos de agrupamiento multi-instancias.
11. Incorporación al Weka de las medidas que permiten comparar bolsas.
12. Verificación de los algoritmos y medidas incorporadas al Weka.
13. Validación de los algoritmos incorporados al Weka a partir de su ejecución sobre colecciones de problemas multi-instancias reportados en la literatura.

El presente trabajo de diploma se encuentra estructurado en varios capítulos. El Capítulo 1 recoge las ideas fundamentales relacionadas con el aprendizaje multi-instancias, haciendo

énfasis en el aprendizaje multi-instancias no supervisado, específicamente el agrupamiento multi-instancias. Además, se describirán los métodos de solución de problemas multi-instancias y algunas consideraciones acerca de su aplicación en el agrupamiento multi-instancias. En el Capítulo 2 se describirán los principales algoritmos de agrupamiento multi-instancias y su implementación en Weka, la incorporación de filtros que permitan transformar problemas multi-instancias en simple-instancia, así como, la adaptación realizada a algoritmos simple-instancia para que logren realizar el agrupamiento de datos multi-instancias. Además, se detallarán las medidas que permiten la comparación entre bolsas e instancias. El Capítulo 3 estará encargado de describir las colecciones existentes para la validación, así como el diseño experimental que permitirá comparar los algoritmos implementados. Este documento culmina con las conclusiones, las recomendaciones y las referencias bibliográficas.

Capítulo 1 Acerca del aprendizaje multi-instancias no supervisado

En este capítulo se presenta una descripción y formalización de los conceptos esenciales del aprendizaje multi-instancias. Se describirán los principales escenarios de este tipo de aprendizaje, particularizando en los algoritmos de agrupamiento multi-instancias. Finalmente, se describirán los métodos de solución de problemas multi-instancias y algunas consideraciones acerca de su aplicación en el agrupamiento multi-instancias.

1.1 Aprendizaje multi-instancias

El paradigma de aprendizaje multi-instancias fue definido formalmente en el año 1997 (Dietterich, Lathrop and Lozano-Pérez, 1997). Dietterich y un colectivo de autores trabajaban en una aplicación en bioquímica: el problema de predicción de la actividad de fármacos. Dicho problema consiste en encontrar aquellas moléculas que sean capaces de enlazarse a una proteína dada. El factor principal para determinar si una molécula se enlaza a una proteína determinada es su forma. Empleando el paradigma de aprendizaje supervisado se podría aprender la forma adecuada de una molécula a partir de un conjunto de ejemplos positivos y negativos. Un ejemplo positivo sería una molécula que se sabe que se enlaza a la proteína dada, mientras que un ejemplo negativo sería una molécula que no se enlaza con la proteína. Sin embargo, el problema principal de este enfoque es que las moléculas son objetos flexibles. Por tanto, una misma molécula puede tomar variadas conformaciones. El problema multi-instancias surge entonces porque no existe una única descripción de una molécula; es decir, una misma molécula es descrita a través de múltiples conformaciones. Por tanto, un ejemplo de entrenamiento está formado por una colección de conformaciones. Si una molécula es positiva, sabemos que al menos una de sus posibles conformaciones es la correcta, aunque no sabemos cuál de ellas es la correcta. Si una molécula es negativa, sabemos que ninguna de sus conformaciones es correcta. El problema de aprendizaje consiste en predecir, basado en un conjunto de moléculas de ejemplo, si una molécula pertenece a una u otra clase. Se formula entonces el aprendizaje multi-instancias para resolver este problema. Cada molécula se representó como una bolsa que contiene las diferentes conformaciones que puede adoptar la molécula.

El ejemplo descrito previamente, el cual dio origen al surgimiento del aprendizaje multi-instancias, evidencia que la forma de describir los datos en el llamado aprendizaje simple-instancia, no permite su modelación ni su consecuente solución. En el aprendizaje simple-instancia, la descripción de los datos o información tradicional se realiza considerando que cada observación u objeto de aprendizaje es descrito por un vector de características y, posiblemente, un resultado asociado. En el aprendizaje multi-instancias la estructura de los datos es más compleja, donde un ejemplo u objeto de aprendizaje es denominado bolsa. El rasgo principal del aprendizaje multi-instancias es que una bolsa está asociada con múltiples instancias o descripciones. Cada instancia es descrita por un vector de características, pero nunca se reporta un resultado asociado a la instancia. La única información disponible acerca de una instancia, además de sus valores de características es su pertenencia a una bolsa (Dietterich, Lathrop and Lozano-Pérez, 1997; Mitchel, 1997; Herrera *et al.*, 2016).

Explícitamente, una instancia x se corresponde con un punto en el espacio de instancias $\mathbb{X}$, asumido comúnmente como $\mathbb{X} \in \mathbb{R}^d$, así, cada instancia es descrita por d valores, generalmente reales, aunque en algunas aplicaciones pueden combinarse valores de diversos dominios. Para modelar esta situación, $\mathbb{X}$ puede ser generalizado a $\mathbb{X} \subseteq A^d = A_1 \times \dots A_d$, tal que cada instancia es descrita por un vector *d*-dimensional, donde cada atributo $A_i (i = 1, \dots, d)$ toma un valor de un conjunto finito o infinito V_i, así, algunos rasgos pueden ser nominales y otros numéricos.

Una bolsa X es una colección de n instancias, donde cada instancia x_i proviene del espacio de instancias $\mathbb{X}$. Cada bolsa puede tener un tamaño diferente; por tanto, el valor n puede variar entre las bolsas en el conjunto de datos. Se dice que una bolsa es un multi-conjunto y no simplemente un conjunto ya que el primero, a diferencia del segundo, permite tener instancias repetidas. Por esta razón, muchos autores definen una bolsa como $X \in \mathbb{N}^{\mathbb{X}}$, para representar que pueden existir duplicados de las instancias en las bolsas. Esta estructura de los datos hace que el aprendizaje multi-instancias sea complejo e impone retos al desarrollar nuevos métodos.

En la Figura 1 se muestra la estructura general de un conjunto de datos multi-instancias. La primera columna representa las bolsas, que indican los ejemplos. Cada bolsa contiene un

número de instancias, representadas en la segunda columna. A cada instancia se le asocia un vector *d*-dimensional, representado por los valores asociados a los atributos representados de la columna A_1 a la columna A_d. La primera instancia $x_{1,1}$ en la primera bolsa X_1 es representada por el vector de rasgos $(x_{1,1,1}, x_{1,1,2}, \dots, x_{1,1,d})$. La última columna representa la salida asociada a la bolsa (atributo de decisión). La salida se corresponde a toda la bolsa y no a una instancia en particular. Dependiendo de la tarea de aprendizaje, la salida puede ser la etiqueta de la clase (para los problemas de clasificación) o un valor real (para los problemas de regresión). En los problemas de agrupamiento, tal es el caso de esta investigación, no están disponibles los valores de salida, ya que los métodos de agrupamiento son no supervisados, y por tanto, se encargan de descubrir las asociaciones entre las bolsas.

Bags	Instances	$\mathscr{A}_1$	$\mathscr{A}_2$	...	$\mathscr{A}_d$	Outcome
X_1	$x_{1,1}$ ... x_{1,n_1}	$x_{1,1,1}$ ... $x_{1,n_1,1}$	$x_{1,1,2}$ ... $x_{1,n_1,2}$	... ...	$x_{1,1,d}$ ... $x_{1,n_1,d}$	y_1
...	...	...	...	...	...	...
X_M	$x_{M,1}$ ... x_{M,n_M}	$x_{M,1,1}$ ... $x_{M,n_M,1}$	$x_{M,1,2}$ ... $x_{M,n_M,2}$	... ...	$x_{M,1,d}$ ... $x_{M,n_M,d}$	y_M

Figura 1. Estructura de un conjunto de datos multi-instancias con *M* bolsas. Fuente: (Herrera *et al.*, 2016).

1.2 Escenarios del aprendizaje multi-instancias

Al igual que como sucede en el aprendizaje tradicional simple-instancia, el aprendizaje multi-instancias puede enfrentar diversas tareas de aprendizaje, tanto supervisado, semi-supervisado como no supervisado (Herrera *et al.*, 2016).

El objetivo de un problema de clasificación multi-instancias consiste en determinar la clase a la que pertenecen las nuevas bolsas, basándose en el aprendizaje alcanzado a partir de las clases asignadas a las bolsas en el conjunto de entrenamiento (Li *et al.*, 2015; Hajimirsadeghi and Mori, 2017; Tran *et al.*, 2017; Melki, Cano and Ventura, 2018). Tradicionalmente, el aprendizaje multi-instancias ha estado focalizado a problemas de clasificación de dos clases

(una clase positiva y otra negativa). Sin embargo, los métodos MIL existentes pueden manejar cualquier número de clases.

Aunque este trabajo de diploma está dirigido al agrupamiento multi-instancias, resulta de interés estudiar y analizar las formas en que se ha enfrentado la clasificación multi-instancias, de forma tal que se puedan establecer analogías y probablemente extrapolar las experiencias del MIL supervisado al MIL no supervisado.

Una de las primeras clasificaciones de los métodos de clasificación multi-instancias fue propuesta por Xu en (Xu, 2003), la cual realiza la división en métodos basados en instancias y métodos basados en metadatos. Los primeros, inicialmente tratan de determinar las clases a las que pertenecen las instancias, las cuales son usadas para derivar la clase a la que pertenece la bolsa, utilizando una regla explícita. Los segundos, determinan la clase de la bolsa desde la información extraída directamente desde las bolsas. Esta clasificación sugiere dos formas de enfrentar el agrupamiento multi-instancias, pudiera pensarse en primero agrupar las instancias y a partir de esos grupos agrupar las bolsas, o directamente agrupar las bolsas considerando la información que ofrecen. Como describiremos en el epígrafe 1.3, existen métodos de agrupamiento que siguen uno u otro enfoque.

Otra categorización de la clasificación multi-instancias fue propuesta por Foulds en (Foulds, 2008) ésta divide los métodos en tres categorías:

- **Diseñados para MIL (*Purpose-built*)**: Algoritmos específicamente diseñados para aprender conceptos multi-instancias.
- **Adaptados a MIL (*Upgraded*)**: Adaptación de algoritmos de clasificación simple-instancia a la clasificación multi-instancias.
- **Envoltorios (*Wrappers*)**: Métodos que transforman la estructura de los conjuntos de datos multi-instancias a una representación simple-instancia con el objetivo de aplicar los algoritmos de clasificación simple-instancia existentes sin modificación alguna.

Las clasificaciones propuestas en (Xu, 2003; Foulds, 2008) siguen distintos criterios para conformar los tipos de métodos y no son excluyentes; por ejemplo, un métodos específicamente diseñado para aprender conceptos multi-instancias pudiera ser basado en instancias o basado en metadatos. La clasificación propuesta por Xu en (Xu, 2003) tiene en

cuenta cómo proceder para obtener las clases de la bolsa, si clasificando previamente las instancias o no; mientras que la clasificación propuesta por Foulds en (Foulds, 2008) se centra en cómo diseñar los métodos respecto a si se va a reutilizar o no, y cómo reutilizar la experiencia ya existente en la clasificación simple-instancia. Con el objetivo de combinar ambos enfoques en solo uno, Amores presentó otra taxonomía para organizar los métodos de clasificación multi-instancias en tres categorías (Amores, 2013):

- **Paradigma en el espacio de instancias (*Instance Space Paradigm*)**: Algoritmos que encuentran funciones discriminantes en el espacio de instancias. La etiqueta de la bolsa se deriva a partir de un supuesto enlace multi-instancias de las etiquetas de las instancias pertenecientes a la bolsa.
- **Paradigma en el espacio de bolsas (*Bag Space Paradigm*)**: Métodos que trabajan en el espacio de la bolsa y definen medidas de distancias o similitudes entre bolsas, permitiendo así determinar las relaciones espaciales entre las bolsas y las clases.
- **Paradigma en el espacio embebido (*Embedded Space Paradigm*)**: Algoritmos que trabajan en un espacio embebido. Transforman el espacio de entrada original en un nuevo espacio (embebido), donde las bolsas se describen por vectores simples de atributos. Así, los algoritmos simple-instancia pueden ser aplicados en un espacio inducido.

La categorización presentada por Foulds y Frank (Foulds and Frank, 2010) consiste en una organización jerárquica de los métodos de clasificación multi-instancias. Esta categorización no ha sido muy utilizada por la comunidad científica de MIL por diversas razones, entre ellas, que divida los métodos en un gran número de categorías y muchas de ellas están desconectadas del resto.

En (Herrera *et al.*, 2016) se propone otra categorización de los métodos de clasificación multi-instancias que intenta reutilizar las ventajas de las clasificaciones anteriores y mitigar sus desventajas. Esta propuesta organiza los métodos en dos niveles. La primera clasificación distingue entre métodos basados en instancias y métodos basados en bolsas. El segundo nivel subdivide los métodos basados en bolsas en: métodos basados en un espacio de bolsas original y métodos basados en un espacio de bolsas mapeado, como se describe a continuación:

- **Métodos basados en instancias (*Instance-based Methods*)**: Incluye aquellos algoritmos cuyo proceso de aprendizaje ocurre al nivel de instancias. Estos métodos asumen la existencia de diferentes clases de instancias; i.e., las instancias tienen clases de etiquetas ocultas. Además, asumen que existe una relación explícita, llamada supuesto multi-instancias (*multi-instance assumption*), entre las etiquetas de las clases de las instancias en la bolsa y la etiqueta de la clase de la bolsa en sí. Estos métodos construyen un modelo de aprendizaje para determinar las clases de las instancias y así, basados en el supuesto multi-instancias, determinan la clase de la bolsa.
- **Métodos basados en bolsas (*Bag-based Methods*):** Incluye aquellos algoritmos para los cuales el proceso de aprendizaje ocurre al nivel de bolsas. Ellos no intentan determinar sub-conceptos o etiquetas de las clases para instancias individuales, en vez de eso, consideran a la bolsa como toda una entidad. Estos métodos se dividen en dos subcategorías, métodos que trabajan con el espacio de bolsas original y métodos que trabajan con el espacio de bolsas mapeado:
    - **Métodos basados en el espacio original de bolsas (*Original Bag Space based Methods*):** Incluye clasificadores que definen una distancia o medida de similitud entre bolsas para trabajar directamente en el espacio de bolsas original. Estos métodos requieren una función de comparación de bolsas.
    - **Métodos basados en el espacio mapeado de bolsas (*Mapped Bag Space based Methods*):** Incluye clasificadores que transforman cada bolsa en una representación simple-instancia, tal que sea posible entrenar cualquier clasificador simple-instancia para etiquetar nuevas bolsas. Estos métodos difieren unos de otros dependiendo de la forma específica en que realicen el proceso de mapeo.

Los problemas de regresión multi-instancias, al igual que en el aprendizaje tradicional, se resuelven siguiendo un aprendizaje supervisado, donde los atributos de decisión tienen un dominio numérico (Wagstaff, Lane and Roper, 2008). Por tanto, la principal diferencia entre los problemas de regresión y los de clasificación es que en la regresión se reemplazan las clases, utilizadas en los problemas de clasificación, por valores numéricos.

Dentro de los escenarios del aprendizaje multi-instancias se han desarrollado variantes no supervisadas como es el caso de la minería de reglas de asociación multi-instancias (Haripriya *et al.*, 2016) y el agrupamiento multi-instancias (Kriegel *et al.*, 2006; Kriegel, Pryakhin and Schubert, 2006; Zhang and Zhou, 2009). Tal como fue descrito en el aprendizaje supervisado, el MIL no supervisado es más complejo que el aprendizaje simple-instancia tradicional no supervisado debido a la ambigüedad inherente en el aprendizaje multi-instancias. Sin embargo, como sucede en otros escenarios, MIL tiene gran flexibilidad y permite la solución de varios problemas más eficientemente que el paradigma simple-instancia (Herrera *et al.*, 2016).

El objetivo del agrupamiento multi-instancias es agrupar bolsas (no etiquetadas previamente) basándose en una medida de similitud entre ellas. Los métodos de agrupamiento multi-instancias determinan un conjunto de grupos y una función la cual asigna bolsas a los grupos de forma tal que se minimice la similitud entre bolsas que pertenecen a grupos diferentes y se maximice la similitud entre las bolsas de un mismo grupo. La selección de una medida de similitud apropiada es crucial en el agrupamiento multi-instancias. También es importante tener en cuenta que no todas las instancias en una bolsa contribuyen igualmente a la predicción de la bolsa, lo que implica que las bolsas no deberían ser consideradas en todos los casos como una colección de instancias independientes en la definición de la medida de similitud. Este hecho tiene una estrecha relación con la posible adaptación del supuesto multi-instancias al caso del agrupamiento. Debido a la importancia que reviste el agrupamiento multi-instancias para esta investigación, el próximo epígrafe está dedicado a describir los principales algoritmos de agrupamiento multi-instancias reportados en la literatura.

1.3 Algoritmos de agrupamiento multi-instancias

El análisis de grupos o agrupamiento intenta agrupar una colección dada de patrones no etiquetados en diferentes subconjuntos que contienen patrones similares. Esos subconjuntos son llamados grupos. Los métodos de agrupamiento deben producir grupos con alta similitud interna y baja similitud entre grupos (Wong, 2015; Xu and Tian, 2015; Sajana, Rani and Narayana, 2016; Carbonneau *et al.*, 2017).

El agrupamiento se ha convertido en una herramienta muy interesante para guiar el descubrimiento de grupos previamente desconocidos en muchas áreas incluyendo la minería de datos, la estadística, el aprendizaje automático, la tecnología de bases de datos espaciales, la recuperación de información, la búsqueda Web, la biología y el marketing, entre otros. Además, los algoritmos de agrupamiento se están utilizando frecuentemente en la fase de pre-procesamiento para facilitar la solución de otras tareas, debido a la alta dimensionalidad de los datos presente en los problemas a resolver. Todo ello evidencia que el análisis de grupos es un tópico muy activo en las investigaciones actuales de minería de datos (Herrera *et al.*, 2016).

La ambigüedad inherente del aprendizaje multi-instancias, donde cada observación (bolsa) contiene varias instancias no etiquetadas con una relación particular ente ellas, hace que la tarea de distribuir objetos en grupos sea aún más difícil. El agrupamiento multi-instancias tiene sus propias características y las medidas de similitud en el agrupamiento simple-instancia no son apropiadas, ya que instancias en una bolsa usualmente exhiben diferentes funcionalidades. Uno de los problemas a resolver consiste en identificar cuáles instancias de la bolsa contienen el concepto del agrupamiento y cuáles son irrelevantes. Es por ello que en (Herrera *et al.*, 2016), se insiste en el hecho que en el agrupamiento multi-instancias, las bolsas no pueden ser analizadas como una simple colección de instancias independientes. Las características y relaciones de las instancias dentro de las bolsas deben ser cuidadosamente investigadas. Como se expresó en (Herrera *et al.*, 2016), es importante identificar los conceptos/objetos determinantes en cada bolsa, de forma tal que se definan distancias que logren determinar la verdadera relación entre las bolsas. Desafortunadamente, no se han reportado trabajos que hayan estudiado esta relación para el agrupamiento multi-instancias. Por tanto, esta relación no se tendrá en cuenta en este trabajo de diploma cuando se calculen las distancias entre bolsas en el proceso de agrupamiento.

El agrupamiento multi-instancias no ha sido tan abordado por la comunidad científica como la clasificación multi-instancias. A continuación, describiremos los principales algoritmos multi-instancias reportados en la literatura para el descubrimiento de grupos de bolsas.

El algoritmo BAMIC (*Bag-level Multi-Instance Clustering*), propuesto por Zhang y Zhou (Zhang and Zhou, 2009), es el primer algoritmo de agrupamiento multi-instancias reportado en

la literatura. BAMIC adapta el algoritmo clásico *k*-medoides, para particionar un conjunto de bolsas en *k* grupos disjuntos. Para ello, Zhang y Zhou consideran las bolsas elementos atómicos y emplean el algoritmo *k*-medoides para construir los grupos basándose en las distancias entre las bolsas.

BAMIC, en su primer paso, selecciona *k* patrones (bolsas) arbitrariamente para inicializar los medoides, es decir, una bolsa representativa por cada uno de los *k* grupos a descubrir. En cada iteración, cada bolsa se asigna al grupo representado por el medoide más cercano a la bolsa. Los medoides de los grupos son recalculados mediante la selección de los objetos con el más bajo promedio de disimilitud al resto de las bolsas en el grupo.

Las medidas de Hausdorff (distancia de Hausdorff mínima, máxima y promedio) (Devaney, 2007; Deza and Deza, 2012) han sido las más utilizadas por este método para comparar bolsas. Resultados experimentales han mostrado similar desempeño del método cuando se varían estas métricas de distancias.

Si quisiéramos utilizar para clasificar los métodos de agrupamiento multi-instancias, las taxonomías establecidas para los métodos de clasificación multi-instancias, entonces BAMIC es un algoritmo adaptado a MIL según la clasificación de Foulds (Foulds, 2008), es un algoritmo basado en metadatos según la clasificación de Xu (Xu, 2003), sigue el paradigma de espacio entre las bolsas según la clasificación de Amores (Amores, 2013), y finalmente, según (Herrera *et al.*, 2016), es un método basado en el espacio original de bolsas.

El algoritmo UC-kNN (*Unsupervised Citation-kNN*), propuesto por Henegar y un colectivo de autores (Henegar, Clément and Zucker, 2006), es un método jerárquico aglomerativo para el agrupamiento multi-instancias, creado a partir de la adaptación a agrupamiento multi-instancias del método Citation-kNN creado por Wang y Zucker (Zhang and Chen, 2005). Este método sigue el enfoque del vecino más cercano. Una bolsa X_i es una buena referencia para otra bolsa X_j, si la bolsa X_i es una de las *k* bolsas más cercanas a la bolsa X_j. Los autores de UC-kNN definieron una nueva métrica de similitud basada en la información mutua para comparar las bolsas.

A pesar de que este método fue creado a partir del método Citation-kNN, consideramos que según la taxonomía de Foulds (Foulds, 2008) debe ser clasificado como un método diseñado

para MIL, ya que solo toma ideas básicas y dista de sus antecesores. Según el resto de las taxonomías, pertenece a las mismas categorías que BAMIC.

El algoritmo COSMIC (*Conceptual Specified Multi-Instance Clusters*), propuesto por Kriegel y un colectivo de autores (Kriegel *et al.*, 2006), es un algoritmo de agrupamiento multi-instancias basado en densidad que permite encontrar un número arbitrario de grupos e identificar datos ruidosos. Este algoritmo está basado en OPTICS (Ankerst *et al.*, 1999) y consiste de dos pasos. El primer paso sigue la estrategia de un algoritmo basado en densidad y calcula para cada bolsa la distancia núcleo (*core distance*) y la distancia de asequibilidad (*reachability distance*), y establece un ordenamiento de las bolsas. Las salidas del primer paso constituyen las entradas al segundo paso que se encarga de extraer los grupos de la lista de bolsas ordenada.

Consideramos que este método, al igual que como sucede con UC-kNN, debe ser clasificado como un método diseñado para MIL según la taxonomía de Foulds (Foulds, 2008), porque, aunque es basado en OPTICS, establece particularidades para el agrupamiento multi-instancias. Según el resto de las taxonomías, pertenece a las mismas categorías que BAMIC y UC-kNN.

El algoritmo M3IC-MBM (*Maximum Margin Multiple Instance Clustering with Modified Bag Margin*), propuesto por Zhang y un colectivo de autores (Zhang *et al.*, 2011), se basa en hiperplanos de margen máximo que separan los datos en diferentes clases de manera no supervisada. Por tanto, M3IC-MBM se convierte en un problema de optimización no convexo. Este algoritmo requiere que se le especifiquen como entrada los valores asociados a cinco parámetros.

Aunque este método se puede clasificar como un método basado o que sigue el paradigma en el espacio de bolsas según las categorizaciones de Xu (Xu, 2003), de Herrera y colectivo de autores (Herrera *et al.*, 2016) y de Amores (Amores, 2013), es importante destacar que en el proceso de descubrimiento de los grupos realiza un análisis interno de las instancias por cada bolsa. Además, según la clasificación de Foulds (Foulds, 2008), es un algoritmo diseñado para MIL.

El algoritmo de agrupamiento MIEM (*Multi-instance EM Clustering*), propuesto por Kriegel y un colectivo de autores (Zhang and Chen, 2005), sigue un enfoque estadístico y está basado en el algoritmo de agrupamiento EM (*Expectation Maximization*) (Kriegel, Pryakhin and Schubert, 2006). En MIEM se modelan las instancias como miembros de conceptos en algún espacio de rasgos. Cada concepto es modelado por un proceso estadístico en ese espacio de rasgos. Un objeto multi-instancias puede entonces describirse como el resultado de seleccionar varias veces un concepto y generar así una instancia. Los grupos de objetos multi-instancias pueden describirse como distribuciones multinomiales sobre los conceptos. Así, grupos diferentes tienen diferentes probabilidades sobre los conceptos descubiertos. MIEM sigue tres pasos fundamentales. En el primer paso se deriva un modelo que describe los conceptos en el espacio de instancias. En el segundo paso se encuentra una buena inicialización de la distribución de cada objeto multi-instancias a los conceptos descubiertos y se agrupan utilizando el algoritmo *k-means* (Macqueen, 1967). En el tercer paso se emplea el agrupamiento EM para optimizar la distribución por cada grupo de los objetos multi-instancias.

MIEM fue diseñado para MIL y su primer paso indica que primeramente determina los conceptos resultantes de las instancias, a partir de un proceso de agrupamiento de éstas. Por tanto, adaptando las taxonomías establecidas para los métodos de clasificación multi-instancias a la clasificación de algoritmos de agrupamiento multi-instancias, pudiéramos clasificar MIEM según las categorizaciones de Xu (Xu, 2003), de Herrera y colectivo de autores (Herrera *et al.*, 2016) y de Amores (Amores, 2013), como un método basado en instancias o que sigue el paradigma en el espacio de instancias.

Es muy difícil establecer una comparación en cuanto a la calidad de los agrupamientos obtenidos por los métodos expuestos en este epígrafe, ya que en la mayoría de las publicaciones donde ellos han sido presentados a la comunidad científica, solamente se comparan con algunos métodos de agrupamiento simple-instancia adaptados al MIL, pero no entre ellos. Además, generalmente utilizan diversas colecciones de datos y medidas de validación. Otro elemento que dificulta establecer comparaciones es que ellos no se encuentran incorporados en una plataforma común que permita su estudio y análisis bajo las mismas condiciones.

1.4 Enfoques para la solución de problemas multi-instancias

Como se pudo apreciar en el epígrafe 1.3, existen muy pocos algoritmos de agrupamiento multi-instancias. Sin embargo, se pudieran generar muchísimas más soluciones para el agrupamiento multi-instancias si se aplicaran los métodos establecidos para la solución de problemas de clasificación multi-instancias. En (Herrera *et al.*, 2016) se sintetizan tres posibles variantes para enfrentar la clasificación multi-instancias: los algoritmos ad-hoc, los algoritmos simple-instancia actualizados y los envoltorios para algoritmos simple-instancia.

- **Algoritmos ad-hoc**: Algoritmos específicamente diseñados para resolver problemas multi-instancias apartándose de los métodos tradicionales de aprendizaje supervisado.
- **Algoritmos simple-instancia actualizados**: Algoritmos tradicionales del aprendizaje supervisado transformados para poder ser aplicados a datos multi-instancias.
- **Envoltorios para algoritmos simple-instancia**: Algoritmos que aplican una transformación a los datos del problema multi-instancias de forma que puedan aplicarse los algoritmos del aprendizaje supervisado tradicional para obtener la solución.

Algunos de los métodos de agrupamiento multi-instancias descritos en el epígrafe 1.3 surgieron de la actualización de algoritmos de agrupamiento simple-instancia ya existentes o fueron creados propiamente para el problema MIL, considerándose algoritmos ad-hoc, aunque se basan en su mayoría en ideas de propuestas precedentes. Aquellos surgidos a partir de la actualización de algoritmos simple-instancia han utilizado principalmente las distancias de Hausdorff para comparar bolsas. Por tanto, quedan muchas distancias con las cuáles experimentar (Jin, Wang and Zhou, 2009; Sørensen *et al.*, 2010; Campbell, Ping and Xu, 2011; Tax *et al.*, 2011; Chen and Wu, 2012; Deza and Deza, 2012; Wang, Nie and Huang, 2013), así como métodos de agrupamiento simple-instancia por transformar (Wong, 2015; Xu and Tian, 2015; Sajana, Rani and Narayana, 2016; Carbonneau *et al.*, 2017). Por otro lado, no se ha investigado en profundidad en el desarrollo de envoltorios para algoritmos de agrupamiento simple-instancia. Todo ello nos hace pensar que aún quedan muchas variantes por explotar en el desarrollo de métodos de agrupamiento multi-instancias.

El desarrollo de métodos envoltorios permite aplicar la gran cantidad de métodos y algoritmos que han sido desarrollados para el aprendizaje no supervisado simple-instancia, en la solución de problemas multi-instancias. Por tanto, en vez de usar un método de agrupamiento simple-instancia modificado, sería viable transformar los datos multi-instancias en una representación simple-instancia de las bolsas. Así, es posible aplicar cualquier algoritmo de agrupamiento simple-instancia existente para resolver el problema multi-instancias; y no solo esto, también será posible aplicar técnicas de preprocesamiento, tales como edición, limpieza y reducción de dimensionalidad, las cuales han sido muy bien estudiadas en el aprendizaje simple-instancia (Herrera *et al.*, 2016).

Los métodos basados en mapeo se basan en una función que transforma la representación multi-instancias de una bolsa en un vector simple-instancia, pero difieren unos de otros dependiendo del proceso de mapeo específico que realicen. Se han desarrollado cuatro estrategias para realizar el mapeo de datos multi-instancias a simple-instancia con el objetivo de resolver problemas de clasificación, considerando el significado de los atributos en el nuevo espacio de representación (Herrera *et al.*, 2016). Pensamos que algunas de estas estrategias pudieran ser útiles para realizar el proceso de transformación cuando el objetivo es aplicar algoritmos de agrupamiento.

- **Métodos de mapeo basados en la estadística de la bolsa (*Mapping Methods based on Bag Statistics*):** Cada atributo del nuevo espacio mapeado es un valor estadístico obtenido a partir del conjunto de valores del atributo correspondiente en el espacio de representación original.
- **Métodos de mapeo basados en la concatenación de instancias representativas (*Mapping Methods based on Representative Instance Concatenation*):** Cada vector del nuevo espacio mapeado es la concatenación de *N* instancias de la bolsa, donde cada instancia es una representación de un patrón en el espacio de instancias.
- **Métodos de mapeo basados en conteo (*Mapping Methods based on Counting*):** Cada atributo del nuevo espacio mapeado indica la presencia, cantidad o frecuencia, de las instancias de la bolsa en una región específica del espacio de instancias.

- **Métodos de mapeo basados en distancias (*Mapping Methods based on Distance*):** Cada atributo del nuevo espacio mapeado representa la distancia (o similitud) de la bolsa a una región específica del espacio de instancias.

Los métodos de mapeo basados en la concatenación de prototipos y basados en conteo han sido diseñados pensando principalmente en realizar la trasformación de un conjunto de datos donde existe una clasificación de referencia. En el caso de los métodos basados en la concatenación de prototipos es necesario identificar patrones en los datos y calcular un nuevo vector por cada bolsa que representará el promedio de las instancias pesadas por su grado de macheo con el patrón (Tarel, Boughorbel and Boujema, 2005). Los métodos de mapeo basados en conteo representan cada bolsa como un vector simple, donde cada atributo representa la cantidad de instancias de la bolsa que se encuentran en una región específica del espacio de instancias. En otras palabras, estos métodos describen la relación entre la etiqueta de la bolsa y las clases de instancias cubiertas por diferentes regiones en el espacio de instancias. Por otra parte, los métodos de mapeo basados en distancias requieren identificar regiones específicas del espacio de instancias, de ahí que frecuentemente se utilicen algoritmos de agrupamiento a tales efectos.

Es por ello que en esta investigación solo exploraremos el uso de los métodos de mapeo basados en la estadística de la bolsa para transformar los datos multi-instancias en simple-instancia, y poder aplicarles los algoritmos de agrupamiento simple-instancia. Las transformaciones a aplicar son (Gärtner *et al.*, 2002; Dong, 2006): mapeo promedio (*average mapping*), mapeo Min-Max (*min-max mapping*) y mapeo de momentos estadísticos (*moment mapping*).

1.5 Consideraciones finales del capítulo

Al igual que como sucede en el aprendizaje tradicional simple-instancia, el aprendizaje multi-instancias puede enfrentar diversas tareas de aprendizaje, tanto supervisado, semi-supervisado como no supervisado. Dentro de ellas, la clasificación multi-instancias ha sido muy abordada, no tanto así, el agrupamiento multi-instancias.

Las diversas taxonomías que existen para categorizar los métodos de clasificación multi-instancias se pueden utilizar para establecer categorías en los algoritmos de agrupamiento

multi-instancias. Estas taxonomías no son excluyentes, ya que siguen criterios distintos para establecer las categorías. La mayoría de ellas intenta distinguir entre aquellos métodos basados en instancias y los que son basados en bolsas, siendo la presentada en (Herrera *et al.*, 2016) la más completa, aunque no permite, a diferencia de la taxonomía propuesta por Foulds (Foulds, 2008), categorizar los métodos en aquellos adaptados o diseñados propiamente para MIL.

Los principales algoritmos de agrupamiento multi-instancias reportados en la literatura son BAMIC, UC-kNN, COSMIC, M3IC-MBM y MIEM. Es muy difícil establecer una comparación en cuanto a la calidad de los agrupamientos obtenidos por estos métodos, ya que en la mayoría de las publicaciones donde ellos han sido presentados a la comunidad científica, solamente se comparan con algunos métodos de agrupamiento simple-instancia adaptados al MIL, pero no entre ellos. Además, generalmente utilizan diversas colecciones de datos y medidas de validación. Otro elemento que dificulta establecer comparaciones es que ellos no se encuentran incorporados en una plataforma común que permita su estudio y análisis bajo las mismas condiciones.

El desarrollo de métodos envoltorios permite aplicar la gran cantidad de métodos y algoritmos que han sido desarrollados para el aprendizaje no supervisado simple-instancia, en la solución de problemas multi-instancias. Sin embargo, los algoritmos envoltorios no han sido muy abordados en el agrupamiento multi-instancias. Es por ello que en esta investigación se sugiere la aplicación de métodos que permitan mapear el espacio multi-instancias en un espacio simple-instancia. Sugerimos el uso de los métodos de mapeo basados en la estadística de la bolsa para transformar los datos multi-instancias en simple-instancia, y poder aplicarles los algoritmos de agrupamiento simple-instancia. Otros métodos de mapeo están más dirigidos a la clasificación y, por tanto, son menos aplicables al aprendizaje no supervisado.

Capítulo 2 El aprendizaje multi-instancias no supervisado en Weka

En este capítulo se describe cómo se incorporaron algoritmos a Weka para aumentar sus potencialidades en el agrupamiento multi-instancias. Primeramente, se describen las principales características de Weka acerca de la forma de entrada de los datos, sus interfaces principales, estructura, diseño y formas de incorporación de distancias, algoritmos y filtros. Se presenta una descripción detallada de cómo se incorporaron medidas para comparar bolsas, cómo se crearon filtros que constituyen envoltorios de algoritmos simple-instancia, cómo se incorporaron algoritmos de agrupamiento multi-instancias, así como, la adaptación realizada a algoritmos simple-instancia para que logren realizar el agrupamiento de datos multi-instancias.

2.1 Generalidades de Weka

Weka[1] es una plataforma de software para el aprendizaje automático y la minería de datos escrito en Java y desarrollado en la Universidad de Waikato en Nueva Zelanda. Dicho desarrollo comenzó en el año 1993 con los lenguajes TCL/TK y C, aunque años más tarde se decidió reescribir el código en Java. Weka es software libre distribuido bajo la licencia GNU-GPL y es multiplataforma. Weka contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, asociados a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades.

Weka provee implementaciones de algoritmos de aprendizaje automático que pueden ser aplicados de manera sencilla a diversos conjuntos de datos. Además, incluye una variedad de herramientas para transformar conjuntos de datos, tales como los algoritmos para discretización y muestreo (Frank, Hall and Witten, 2016). Weka incluye métodos para los principales problemas de minería de datos: regresión, clasificación, agrupamiento, minería de reglas de asociación y selección de atributos (Ian H. *et al.*, 2011).

Para trabajar con Weka es posible descargar un instalador específico para determinada plataforma o un archivo ejecutable .jar, el cual se puede correr de la forma usual si Java está instalado.

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

2.1.1 Entrada de datos

La entrada de datos a Weka se realiza mediante archivos con extensión arff (*attribute-relation file format*). El formato de un fichero arff difiere en cuanto al tipo de información que se va a representar, ya que pueden ser datos simple-instancia o multi-instancias.

En la estructura para representar información tradicional o simple-instancia, se puede comenzar con signos de porciento (%), indicando que las líneas que se hallan después de estos son comentarios. A continuación de los comentarios, al principio del archivo, se ubica el nombre de la relación y el bloque que define los atributos, los cuales se agrupan en cinco tipos: atributos de tipo cadena, atributos de tipo fecha, atributos de tipo valor-relacional, de tipo nominal y de tipo numérico (Ian H. *et al.*, 2011). En los atributos de tipo cadena la palabra clave *string* se ubica después del nombre del atributo, de igual forma sucede con los atributos de tipo fecha, pero en este caso con la palabra clave *date*. Los atributos de tipo nominal están seguidos por el conjunto de posibles valores que pueden tomar encerrados entre llaves y separados por coma, mientras que los de tipo numérico tienen que estar seguidos por la palabra clave *numeric* (Ian H. *et al.*, 2011).

Después del bloque de los atributos debe escribirse *@data* en una nueva línea, que marca el inicio de las instancias en el conjunto de datos (Ian H. *et al.*, 2011). Las instancias son escritas una por línea, con los valores correspondientes para cada atributo según el orden en que fueron especificados y separadas por comas. En caso de existir algún atributo que no tenga valor, éste se representa con un signo de interrogación, indicando así que existe un valor perdido (Ian H. *et al.*, 2011).

Las especificaciones de los atributos en los ficheros arff permite que el conjunto de datos sea verificado para asegurarnos que la descripción de los objetos se corresponde con los atributos, y los programas que leen ficheros arff realizan esta comprobación automáticamente (Ian H. *et al.*, 2011). En la Figura 2 se muestra la estructura un archivo arff para representar los datos en un problema simple-instancia (Ian H. *et al.*, 2011).

```
% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no
```

Figura 2. Archivo arff simple-instancia. Fuente: (Ian H. *et al.*, 2011).

En cuanto a la conformación de un fichero para representar datos de tipo multi-instancias, también se puede comenzar con signos de porciento (%), indicando comentarios. Luego se define el nombre de la relación y posteriormente el bloque de atributos. Los atributos de tipo valor-relacional difieren de los otros tipos, ya que permiten representar problemas multi-instancias en el formato arff. El valor de un atributo relacional es un conjunto de instancias separadas. El atributo es definido con un nombre y la palabra clave *relational*, seguido por un bloque de atributos anidados que proporcionan la estructura de las instancias referenciadas. Se coloca una línea *@end* seguida del nombre del atributo relacional para cerrar el bloque de atributos anidados (Ian H. *et al.*, 2011). Luego de la declaración de los atributos, debe

escribirse *@data* en una nueva línea que marca el comienzo de las bolsas, las cuales no son más que instancias con múltiples vectores de atributos. Las bolsas pueden contener diferentes cantidades de instancias y se escribe una bolsa por cada línea. Para ello, se hace necesario poner explícitamente un cambio de línea que indique el fin de cada instancia en la bolsa. En la Figura 3 se muestra cómo se representa la información de tipo multi-instancias en un fichero arff (Ian H. *et al.*, 2011).

```
% Multiple instance ARFF file for the weather data
%
@relation weather

@attribute bag_ID { 1, 2, 3, 4, 5, 6, 7 }
@attribute bag relational
     @attribute outlook { sunny, overcast, rainy }
     @attribute temperature numeric
     @attribute humidity numeric
     @attribute windy { true, false }
@end bag
@attribute play? { yes, no }

@data
%
% seven "multiple instance" instances
%
1, "sunny, 85, 85, false\nsunny, 80, 90, true", no
2, "overcast, 83, 86, false\nrainy, 70, 96, false", yes
3, "rainy, 68, 80, false\nrainy, 65, 70, true", yes
4, "overcast, 64, 65, true\nsunny, 72, 95, false", yes
5, "sunny, 69, 70, false\nrainy, 75, 80, false", yes
6, "sunny, 75, 70, true\novercast, 72, 90, true", yes
7, "overcast, 81, 75, false\nrainy, 71, 91, true", yes
```

Figura 3. Archivo arff multi-instancias. Fuente: (Ian H. *et al.*, 2011).

2.1.2 Interfaz visual

En la interfaz de inicio de Weka se encuentran cinco posibles aplicaciones para el trabajo con conjuntos de datos: *Explorer*, *Experimenter*, *KnowledgeFlow*, *Workbench* y *Simple CLI*. En la Figura 4 se muestra esta interfaz.

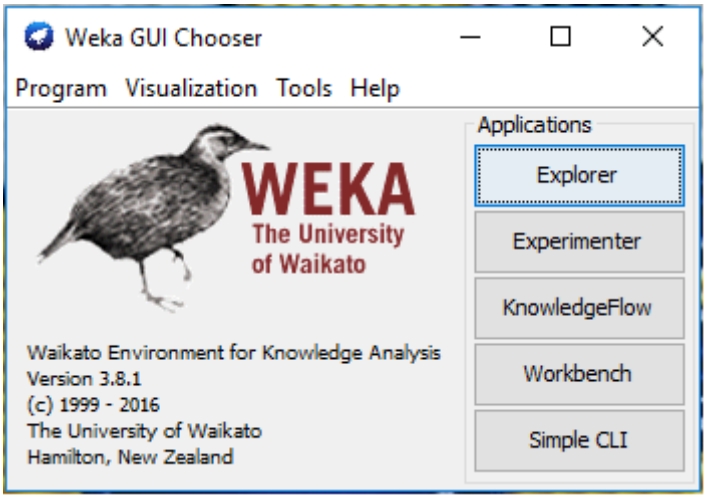

Figura 4. Interfaz principal de Weka.

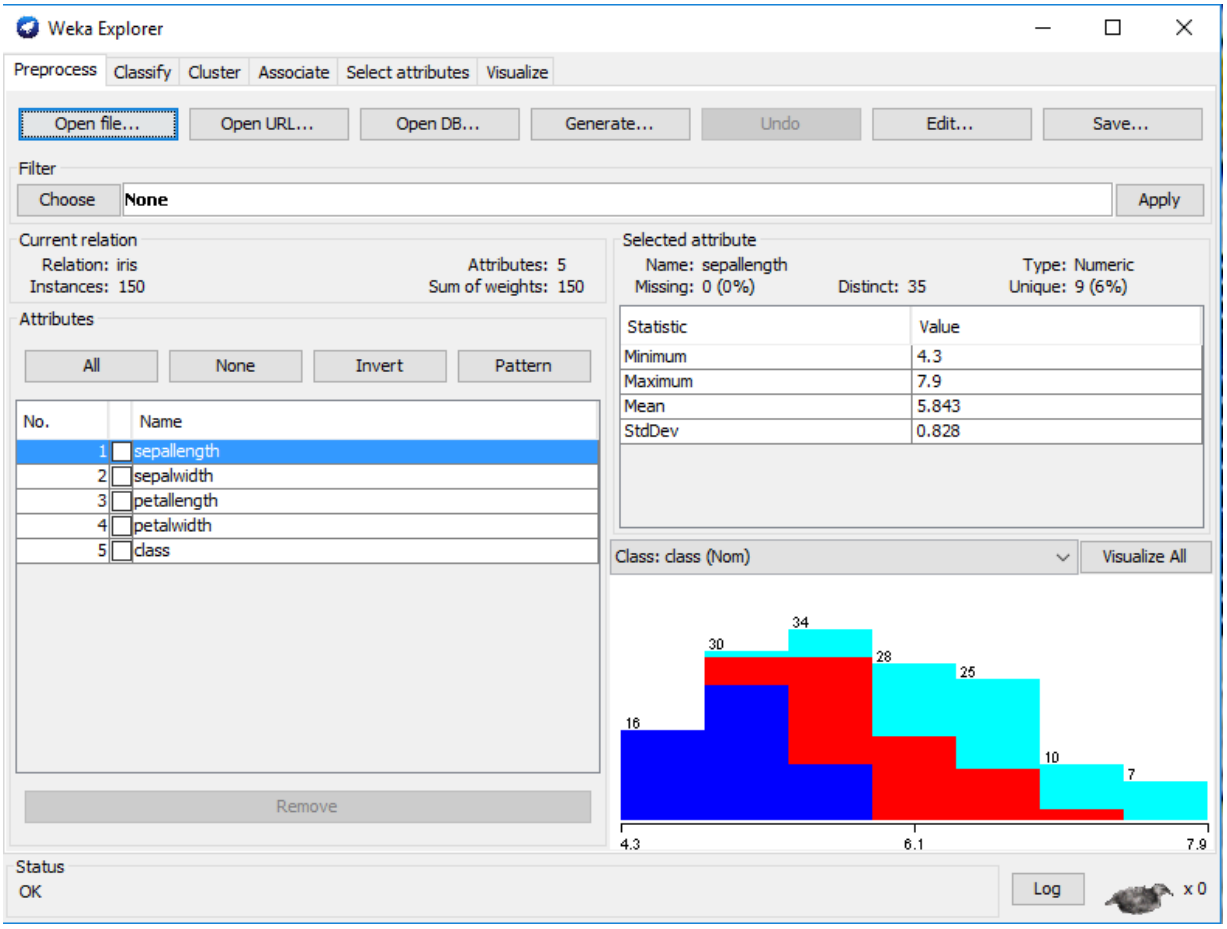

Figura 5. Interfaz del *Explorer*.

El Explorador (*Explorer*) es la interfaz gráfica de usuario principal de Weka, permite el acceso a todas sus facilidades usando un menú de selección. Contiene seis paneles diferentes en la parte superior que se corresponden con las distintas tareas de minería de datos que soporta Weka. Otros paneles pueden estar disponibles instalando los paquetes apropiados (Frank, Hall and Witten, 2016). En la Figura 5 se muestra la interfaz del *Explorer*.

El Flujo de Conocimiento (*KnowledgeFlow*) es una interfaz que provee a los usuarios la selección de componentes de Weka desde una barra de herramientas, de forma tal que se van conectando en un grafo dirigido que establece el flujo y el orden de los algoritmos a ejecutar sobre los datos a procesar. Proporciona una alternativa a la interfaz Explorer, para aquellos usuarios que les gusta pensar en términos de cómo fluyen los datos a través del sistema. Esta interfaz también permite el diseño y ejecución de configuraciones para el procesamiento de flujos de datos, opción que no está disponible en el Explorador (Frank, Hall and Witten, 2016).

El Experimentador (*Experimenter*) brinda la posibilidad de realizar largos procesos de experimentación y revisarlos luego de que hayan concluido para analizar las estadísticas de ejecución que han sido acumuladas. Las estadísticas pueden ser almacenadas en un fichero con formato arff y posteriormente procesadas. Contiene facilidades para que los usuarios avanzados distribuyan la carga de cómputo a través de múltiples máquinas que utilizan la invocación de métodos remotos de Java (*Java Remote Method Invocation*; JRMI). Puede configurar grandes experimentos y simplemente dejarlos correr (Frank, Hall and Witten, 2016).

La Interfaz de Línea de Comando (*Simple CLI*) brinda la posibilidad de realizar las mismas funciones que las interfaces anteriores, pero directamente mediante líneas de comandos. Es posible realizar invocaciones de algoritmos como si fuera hecho mediante el *Explorer* (Frank, Hall and Witten, 2016).

2.1.3 Estructura y diseño

Weka cuenta con un gran número de clases agrupadas en paquetes internos que permiten a sus usuarios realizar las operaciones más conocidas en minería de datos. Su principal paquete se denomina *core,* y las clases que contiene son empleadas por casi todas las demás clases de los

restantes paquetes (Ian H. *et al.*, 2011). Las clases principales que almacena el paquete *core* son: la clase *Attribute*, la clase *Instance* y la clase *Instances*. Un objeto de la clase *Attribute* representa un atributo. El mismo contiene el nombre del atributo, su tipo y los posibles valores que puede tomar. Un objeto de la clase Instancia contiene los valores de los atributos para una instancia en particular y un objeto de tipo Instancias sostiene un conjunto ordenado de instancias, o sea, el conjunto de datos (Ian H. *et al.*, 2011). También existen otras clases para realizar el cálculo de distancias y otras que simplemente contienen la versión de Weka que se está empleando. Otro paquete de suma importancia se nombra *classifiers*, el cual contiene implementaciones de los algoritmos más usados para la clasificación y la predicción numérica. La clase más importante en este paquete se denomina *Classifier*, la cual define el orden general de cualquier esquema para clasificación o predicción numérica y contiene tres métodos principales: *buildClassifier(), classifyInstance(),* y *distributionForInstance()* (Ian H. *et al.*, 2011). También, otro paquete de gran utilidad en Weka es *clusterers*, el cual contiene las implementaciones de los algoritmos clásicos de agrupamiento. Aquí también se halla una clase fundamental que se llama *AbstractClusterer* que se encarga de definir la estructura de cualquier esquema de agrupamiento. Igualmente, almacena métodos esenciales como: *buildClusterer(), clusterInstance()* y *distributionForInstance().* Además, Weka cuenta con un sistema manejador de paquetes el cual fue añadido con el objetivo de brindarle al usuario la posibilidad de seleccionar e instalar aquellos paquetes que sean de su interés. Otra motivación para incluir este sistema manejador de paquetes fue para hacer muy sencillo el proceso de contribución a Weka por parte de su comunidad de usuarios (Frank, Hall and Witten, 2016).

2.1.3.1 Distancias

Las distancias que se encuentran en la plataforma Weka están agrupadas dentro del paquete *core* mencionado anteriormente y son empleadas tanto por algoritmos de clasificación como de agrupamiento. Dicho paquete recoge las siguientes: la distancia Euclidiana, la distancia de Manhattan, la distancia de Chebyshev y la distancia de Minkowski. Estas distancias son empleadas por los algoritmos clásicos que manejan problemas de tipo simple-instancia. Las mismas se emplean para realizar cálculos entre pares de instancias, lo que nos permite realizar tareas de agrupamiento sobre conjuntos de datos. Deben ser escogidas cuidadosamente ya que la calidad de los resultados depende de ellas.

Por cada una de las distancias en Weka se implementa una clase en Java la cual contiene determinados métodos para realizar el cálculo de la distancia. Generalmente, la distancia más utilizada es la Euclidiana, pero casi siempre tanto los algoritmos de clasificación como los de agrupamiento presentan opciones para escoger la métrica de distancia que desee el usuario. En el caso de que sea necesario incorporar una nueva distancia se debería ubicar aquí mismo, porque como se explicó previamente este es el paquete central de Weka.

Cada clase en Java que representa una distancia, hereda de una clase común denominada *NormalizableDistance*, y esta a su vez implementa la interfaz *DistanceFunction*, la cual define el esquema general de cualquier métrica de distancia. Además, las clases que representan las distancias implementan determinadas interfaces que permiten la modificación de sus parámetros.

2.1.3.2 Filtros

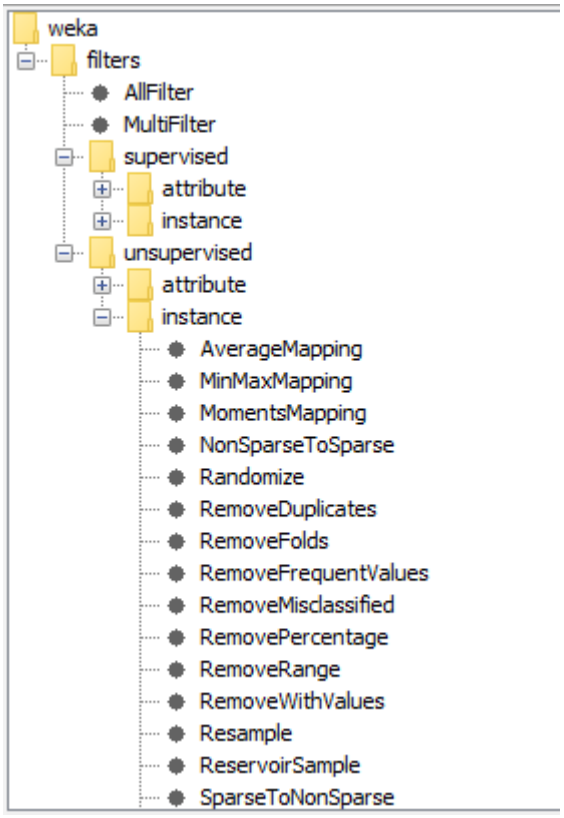

Figura 6. Estructura del paquete *filters* en Weka.

Los filtros que almacena la plataforma Weka se encuentran en el paquete *filters* y están destinados a realizar tareas de preprocesamiento o transformación al conjunto de datos de entrada. Los filtros están agrupados en dos categorías generales: supervisados y no supervisados. Éstos a su vez se clasifican en aquellos que actúan sobre las instancias o sobre los atributos. Esto significa que existen filtros específicos para realizar trabajos con los atributos de las instancias y otros para ocuparse de las instancias en sí (Frank, Hall and Witten, 2016). Los filtros se encuentran disponibles para su selección a través de la interfaz *Explorer*, haciendo clic en el botón *Choose* se despliegan todas las posibles variantes. Los filtros no supervisados no requieren de un atributo de clase para realizar transformaciones. Pueden efectuar diversas operaciones tanto a nivel de atributos como de instancias. La Figura 6 muestra la estructura del paquete *filters* en la plataforma Weka.

2.1.3.3 Agrupamiento

Los algoritmos de agrupamiento que posee Weka están localizados en el paquete *clusterers* y solamente pueden manejar problemas de tipo simple-instancia.

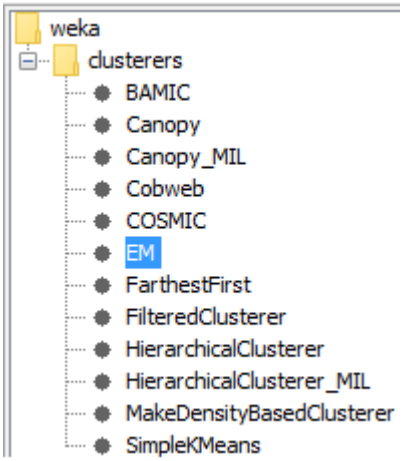

Figura 7. Estructura del paquete *clusterers* en Weka.

Los algoritmos de agrupamiento clásico disponibles en Weka son: *Cobweb*, *SimpleKMeans*, *Canopy*, *HierarchicalClusterer*, *EM*, *FarthestFirst* y *MakeDensityBasedClusterer*. Cada uno de ellos representa una clase en Java que contiene la definición de varios métodos para realizar tareas de agrupamiento. Cada una de estas clases tiene como ancestro común la clase *AbstractClusterer* que define la estructura general de cualquier esquema de agrupamiento.

También implementan determinadas interfaces que permiten la modificación de los parámetros que son utilizados para la ejecución de estos algoritmos. Dichos algoritmos están accesibles desde el panel de la interfaz *Explorer*. Para adicionar nuevos métodos de agrupamiento es necesario situarlos en dicho paquete. En la Figura 7 se muestra la estructura del paquete *clusterers*.

2.2 Incorporación a Weka de distancias para comparar bolsas

Como se describió en el subepígrafe 2.1.3.1, el Weka incluye varias distancias en el paquete *core* que permiten comparar objetos representados por vectores de rasgos. Estas distancias no permiten comparar bolsas, ya que una bolsa está formada por un conjunto de instancias. Se han desarrollado múltiples distancias que permiten comparar bolsas que resultan de gran utilidad para el desarrollo de técnicas de aprendizaje multi-instancias, entre ellas: la distancia promedio de Hausdorff, la distancia mínima de Hausdorff, la distancia máxima de Hausdorff (Jin, Wang and Zhou, 2009; Sørensen *et al.*, 2010; Campbell, Ping and Xu, 2011; Chen and Wu, 2012; Deza and Deza, 2012; Wang, Nie and Huang, 2013), la distancia de Hausdorff adaptada, la divergencia de Cauchy-Schwarz y la distancia de Mahalanobis (Eiter and Mannila, 1997; Herrera *et al.*, 2016).

Estas distancias se encuentran clasificadas en dos grandes grupos: las distancias que manejan las bolsas como un conjunto de puntos y las distancias que manejan las bolsas como distribuciones de probabilidad. Las del primer grupo consideran cada bolsa como un conjunto de puntos o un subconjunto de un espacio multidimensional. Típicamente, una distancia entre dos bolsas es definida como una agregación de distancias entre instancias particulares de cada bolsa. Generalmente, la distancia Euclidiana se utiliza para medir la distancia entre instancias; sin embargo, otras pueden aplicarse como la distancia de Manhattan o la de Chebyshev. Las distancias del segundo grupo consideran cada bolsa como una distribución de probabilidad en el espacio de instancias. Las diferencias entre las bolsas son caracterizadas en términos de las diferencias entre las distribuciones de sus instancias. Para cada bolsa, se estima una densidad probabilística y se calculan las distancias entre las distribuciones (Herrera *et al.*, 2016).

En este trabajo de diploma se decidió incorporar a Weka las distancias de promedio, mínima y máxima de Hausdorff, ya que son las más utilizadas para comparar bolsas en el aprendizaje multi-instancias.

La distancia promedio de Hausdorff define la distancia promedio entre las instancias de una bolsa y su instancia más cercana en otra bolsa, como se muestra en la ecuación (1), donde *A* y *B* representan bolsas, y *a* y *b* representan instancias de las bolsas *A* y *B*, respectivamente. Para calcular la distancia *d*(*a*,*b*) entre las instancias *a* y *b* se puede utilizar cualquier métrica que permita comparar vectores, generalmente se utiliza la distancia Euclidiana.

$$D_{avg}(A,B) = \frac{\sum_{a\in A} min_{b\in B} d(a,b) + \sum_{b\in B} min_{a\in A} d(a,b)}{|A| + |B|} \quad (1)$$

La distancia mínima de Hausdorff define la distancia más corta entre una instancia en una bolsa *A* y una instancia en una bolsa *B*, como se muestra en la ecuación (2).

$$D_{min}(A,B) = min_{a\in A} min_{b\in B} d(a,b) \quad (2)$$

La distancia máxima de Hausdorff especifica la mayor distancia entre una instancia en una bolsa *A* y su más cercana instancia de la bolsa *B*, o viceversa, como se muestra en la ecuación (3).

$$D_{max}(A,B) = max\{max_{a\in \boldsymbol{A}} min_{b\in \boldsymbol{B}} d(a,b), max_{a\in \boldsymbol{A}} min_{b\in \boldsymbol{B}} d(a,b)\} \quad (3)$$

Para la incorporación al Weka de estas tres distancias fue necesario, primeramente, crear una estructura de directorios determinada que permitirá ubicar las distancias dentro del paquete *core*. La Figura 8 muestra cómo se conforma este directorio. Para llevar a cabo la implementación de dichas distancias, es necesario crear una clase en Java por cada una de ellas con los métodos necesarios para realizar el cálculo de distancia. La Figura 9 muestra el diagrama de clases para la distancia de Hausdorff, la cual incluye sus tres variantes. Esta clase, denominada *HausdorffDistance*, implementa tres interfaces, una de ellas es *DistanceFunction*,

la cual es la encargada de proveer el esquema general de las distancias. Además, se definieron algunos métodos que le permiten al usuario seleccionar aquella que sea de su interés.

```
+-miDistances.jar
+-Description.props
+-build_package.xml
+-src
|     +-main
|     |      +-java
|     |            +-weka
|     |                  +-core
|     |                        +-HausdorffDistance.java
|     +-test
|           +-java
|                 +-weka
|                       +-core
|                             +-HausdorffDistanceTest.java
+-lib
+-doc
```

Figura 8. Estructura de directorios para la incorporación de distancias a Weka.

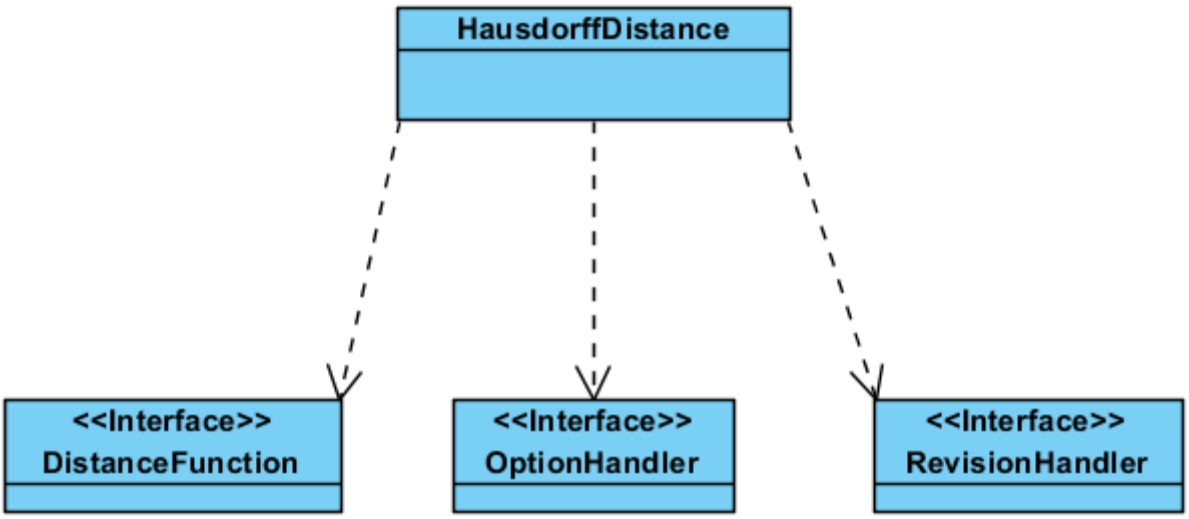

Figura 9. Diagrama de clases para la distancia de Hausdorff.

2.3 Envoltorios para algoritmos simple-instancia

Como fue descrito en el Capítulo 1, existen tres tipos de métodos de solución de problemas multi-instancias: algoritmos ad-hoc, algoritmos simple-instancia actualizados y envoltorios para algoritmos simple-instancia. Estos últimos son algoritmos que aplican una transformación a los datos del problema multi-instancias de forma que puedan aplicarse los algoritmos del aprendizaje supervisado tradicional para obtener la solución.

```
+-miFilters.jar
+-Description.props
+-build_package.xml
+-src
|    +-main
|    |     +-java
|    |           +-weka
|    |                 +-filters
|    |                       +-unsupervised
|    |                                 +-instance
|    |                                           +-AverageMapping.java
|    |                                           +-MinMaxMapping.java
|    |                                           +-MomentsMapping.java
|    +-test
|          +-java
|                +-weka
|                      +-filters
|                            +-unsupervised
|                                      +-instance
|                                                +-AverageMappingTest.java
|                                                +-MinMaxMappingTest.java
|                                                +-MomentsMappingTest.java
+-lib
+-doc
```

Figura 10. Estructura de directorios para los filtros.

En este trabajo se incorporan al Weka métodos envoltorios que permiten transformar datos que representan problemas multi-instancias en datos simple-instancia. Así, es posible aplicar a cualquier problema multi-instancias los métodos de aprendizaje incorporados al Weka. Para ello, hemos identificado que los filtros en Weka constituyen una estructura que facilita la incorporación de métodos envoltorio. Los filtros, como se explicó en el subepígrafe 2.1.3.2, se encuentran agrupados en el paquete *filters*, por lo cual, los envoltorios o filtros para algoritmos

simple-instancia también se incorporan en este mismo paquete. En este caso, los filtros que se adicionan son métodos basados en el mapeo de bolsas. El proceso de mapeo se realiza mediante medidas estadísticas de las bolsas y el procedimiento que se sigue es transformar cada bolsa en una instancia. Los filtros incorporados mediante el paquete *miFilters* son: Average mapping, Min-Max mapping y Moments mapping. Para ello, como primer paso se debe crear una estructura de directorios que permite otorgarle la ubicación a dichos filtros en la plataforma Weka, como se muestra en la Figura 10. Posteriormente, se implementa una clase en Java por cada uno de estos filtros las cuales heredan de una clase padre común llamada *Filter*, la cual es la encargada de proveer la estructura para cualquier esquema de filtro y se redefinen los métodos necesarios para realizar la transformación de las bolsas de nuestro conjunto de datos en instancias. La Figura 11 muestra el diagrama de clases correspondiente.

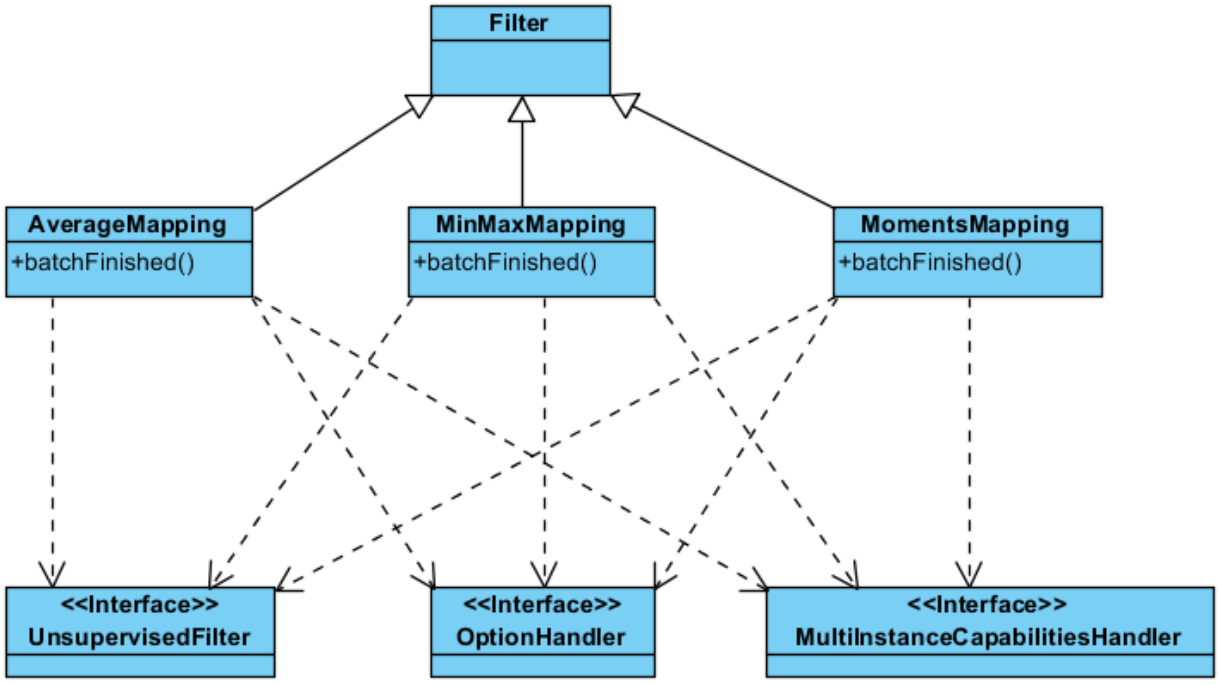

Figura 11. Diagrama de clases para los filtros.

El primer envoltorio mencionado, *Average mapping*, es un método que convierte las bolsas en instancias a partir de la media de los atributos. Este proceso se realiza de la siguiente forma: se toma el primer atributo de cada una de las instancias dentro de la bolsa, se halla el promedio de los valores del atributo para cada una de las instancias de la bolsa en cuestión, y el valor

resultante va a ser el valor correspondiente al primer atributo del objeto que se está creando a partir de la bolsa en análisis. Luego, se toma el segundo atributo y se promedian los valores de este para cada instancia de la bolsa. Así, el valor resultante será el valor asignado a este atributo para el objeto vector que se está formando correspondiente a esta bolsa. Este proceso se ejecuta hasta concluir con todos los atributos que describen la bolsa y así formar el vector simple-instancia resultante de la transformación aplicada a la bolsa. Finalmente, queda transformada la bolsa en una instancia con la misma cantidad de atributos predictores, y se le asocia la clase que tenía la bolsa original. Luego, se prosigue con las restantes bolsas hasta tener nuestro conjunto de datos totalmente transformado. Toda esta tarea de transformación se ejecuta en el método *batchFinished()*, el cual es uno de los métodos principales de la clase.

Para ilustrar el funcionamiento de los filtros se ha desarrollado el ejemplo que se muestra en la Tabla 1. En la Tabla 2 se muestra el resultado de aplicar este filtro a partir del ejemplo de la Tabla 1.

Tabla 1. Ejemplo de un conjunto de datos multi-instancias.

	A_1	A_2	A_3	A_4	Clase
Bolsa 1	3	4	2	7	C_1
	2	1	2.5	5	
	2	4	7	1	
Bolsa 2	4.2	8	5.3	6	C_2
	1	3	9	2.5	
Bolsa 3	4	1.9	8	3	C_1
	3.3	2	6	7.4	
	8	3	1.5	6	
	7	3	9	2	

Tabla 2. Resultados de aplicar el filtro *Average mapping* al conjunto de datos de ejemplo.

	A_1	A_2	A_3	A_4	Clase
Objeto 1	2.3	3	3.8	4.3	C_1
Objeto 2	2.6	5.5	7.1	4.2	C_2
Objeto 3	5.5	2.4	6.1	4.6	C_1

El segundo filtro, denominado *Min-Max mapping*, transforma las bolsas en instancias a partir de los valores mínimos y máximos de los atributos. Este proceso se realiza como se explica a continuación. Se toma el primer atributo de cada instancia de la bolsa y se hallan los valores mínimo y máximo alcanzados por las instancias de la bolsa para este atributo. Estos dos

valores se asocian a los nuevos atributos creados, correspondientes el mínimo y al máximo del atributo original, que caracterizan al objeto simple-instancia que se está obteniendo a partir de la bolsa original en análisis. Luego se toma el segundo atributo de cada instancia de la bolsa y se realiza la misma operación. Este proceso se ejecuta hasta concluir con todos los atributos de las instancias pertenecientes a la bolsa. Al finalizar, obtenemos la bolsa transformada en una instancia con el doble de la cantidad de atributos. Después se continúa con las bolsas restantes del conjunto de datos. De la misma forma, este proceso toma lugar en el método *batchFinished()*. En Tabla 3 se muestra el resultado de aplicar este filtro a partir del ejemplo de la Tabla 1.

Tabla 3. Resultados de aplicar el filtro *Min-Max mapping* al conjunto de datos de ejemplo.

	A_{11}	A_{21}	A_{31}	A_{41}	A_{12}	A_{22}	A_{32}	A_{42}	Clase
$Objeto_1$	2	1	2	1	3	4	7	7	C_1
$Objeto_2$	1	3	5.3	2.5	4.2	8	9	6	C_2
$Objeto_3$	3.3	1.9	1.5	2	8	3	9	7.4	C_1

El tercer filtro, nombrado *Moments mapping*, convierte las bolsas en instancias a partir de la media, la varianza, la métrica skewness y la métrica kurtosis de los valores de los atributos. Este proceso se desarrolla como se explica a continuación. Primeramente, se halla la media de los valores asociados a cada atributo que describe las instancias de la bolsa, tal como se explicó en el primer filtro. A continuación, se genera un nuevo bloque de atributos, esta vez correspondientes a la varianza de los valores de las instancias para la bolsa en análisis. A continuación, se ubica la misma cantidad de atributos, esta vez calculando la métrica skewness de la misma forma que se realizó el promedio, y finalmente se coloca nuevamente la misma cantidad de atributos, pero calculando le métrica kurtosis de la misma manera que el promedio y las anteriores métricas. Al concluir tenemos la bolsa transformada en una instancia con cuatro veces la cantidad original de atributos. Luego, se prosigue con las bolsas que restan hasta reformar enteramente el conjunto de datos. Esta tarea también toma lugar en el método *batchFinished()*. En Tabla 4 se muestra el resultado de aplicar este filtro a partir del ejemplo de la Tabla 1.

Tabla 4. Resultados de aplicar el filtro *Moments mapping*.

	A_{11}	A_{21}	A_{31}	A_{41}	A_{12}	A_{22}	A_{32}	A_{42}	A_{13}	A_{23}	A_{33}	A_{43}	A_{14}	A_{24}	A_{34}	A_{44}	C
O_1	2.3	3	3.8	4.3	0.2	2	5	6.2	1	-0.7	0.7	-0.3	2.1	1.4	1.5	1.4	C_1
O_2	2.6	5.5	7.1	4.2	2.5	6.2	3.4	3	0	0	0	0	1	1	1	1	C_2
O_3	5.5	2.4	6.1	4.6	3.8	0.2	8.2	4.7	0.1	0.6	-0.7	0	1.2	2.1	1.9	1.3	C_1

2.4 Incorporación de algoritmos ad-hoc de agrupamiento multi-instancias a Weka

Los algoritmos de agrupamiento, como su nombre lo indica, se utilizan para descubrir grupos en un conjunto de datos dado. Existen varias clasificaciones de estos algoritmos, una de ellas los agrupan en seis categorías, las cuales se denominan de la siguiente forma: los métodos particionales, los métodos jerárquicos, los métodos basados en densidad, los métodos basados en celdas, los métodos basados en modelos y los métodos basados en un margen máximo (Wong, 2015; Xu and Tian, 2015; Sajana, Rani and Narayana, 2016; Carbonneau *et al.*, 2017). En la herramienta Weka se encuentran incorporados algoritmos clásicos de agrupamiento multi-instancias que constituyen los principales exponentes de estas categorías previamente mencionadas. Sin embargo; en Weka no aparecen disponibles los principales algoritmos de agrupamiento multi-instancias, ya sean algoritmos ad-hoc o creados a partir de la transformación de algoritmos de agrupamiento simple-instancia. En este trabajo de diploma nos hemos propuesto la incorporación de algunos de estos algoritmos a Weka y así incrementar las potencialidades de la herramienta para el MIL.

De los principales algoritmos de agrupamiento multi-instancias descritos en el Capítulo 1 (BAMIC, COSMIC, UC-kNN, MIEM y M3IC-MBM), se incorporaron al Weka los algoritmos BAMIC (Zhang and Zhou, 2009) y COSMIC (Kriegel *et al.*, 2006). Se seleccionó BAMIC por ser el primer algoritmo de agrupamiento multi-instancias publicado, además es un particional. COSMIC fue seleccionado porque pertenece a otra clasificación y así logramos diversidad, ya que éste es basado en densidad y jerárquico. BAMIC particiona las bolsas de entrenamiento no etiquetadas en *k* grupos disjuntos de bolsas. Las bolsas son consideradas elementos atómicos y los grupos son formados mediante el cálculo de las distancias entre ellos. COSMIC es un método basado en densidad, el cual modela los grupos como regiones densas en el espacio de datos, separado por regiones poco densas (Herrera *et al.*, 2016).

La adición de algoritmos de agrupamiento multi-instancias al Weka se efectúa mediante la creación del paquete *miClusterers*, éste a su vez sigue la estructura del paquete *clusterers*, diseñado para incorporar los métodos de agrupamiento en Weka. Para llevar a cabo la incorporación de estos algoritmos es necesario construir una estructura de directorios específica que permite conceder la ubicación a los mismos. En la Figura 12 se muestra esta estructura de directorios. Posteriormente, se crea una clase en Java por cada uno de los algoritmos. Estas clases heredan de una clase común llamada *AbstractClusterer*, que es la encargada de definir la estructura de cualquier esquema de agrupamiento, ya sea multi-instancias o simple-instancia. En la Figura 13 se muestra el diagrama de clases correspondiente.

```
+-miClusterers.jar
+-Description.props
+-build_package.xml
+-src
|    +-main
|    |     +-java
|    |           +-weka
|    |                 +-clusterers
|    |                             +-BAMIC.java
|    |                             +-COSMIC.java
|    |
|    +-test
|          +-java
|                +-weka
|                      +-clusterers
|                                  +-BAMICTest.java
|                                  +-COSMICTest.java
|
+-lib
+-doc
```

Figura 12. Estructura de directorios para los algoritmos ad-hoc de agrupamiento.

Para realizar la implementación del algoritmo BAMIC se creó una clase de igual nombre, la cual redefine los métodos de la clase padre *AbstractClusterer* e incorpora otros. El método principal de esta clase se denomina *buildClusterer()*, en el cual se lleva a cabo el proceso de agrupamiento. También, la clase *BAMIC* implementa ciertas interfaces que permiten manejar

los parámetros que necesita el algoritmo para su correcto funcionamiento como, por ejemplo, el número de grupos a ser creado, ya que este puede ser modificado por el usuario.

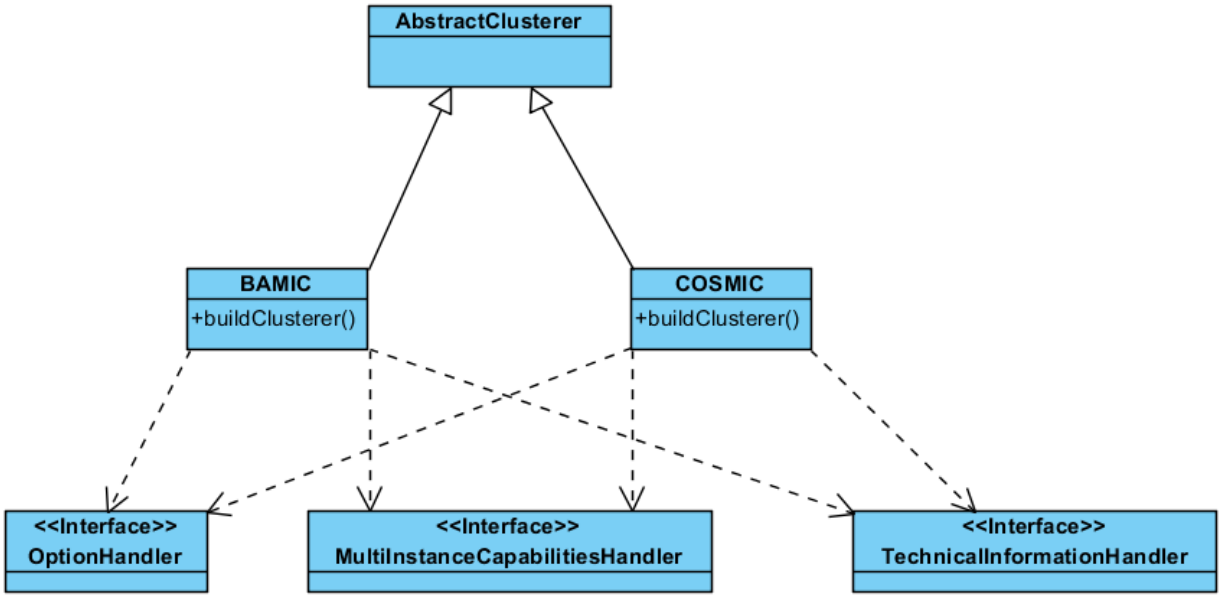

Figura 13. Diagrama de clases para los algoritmos BAMIC y COSMIC.

En cuanto a la implementación del algoritmo COSMIC, también se creó una clase con el mismo nombre, y en ésta se redefinen los métodos de la clase *AbstractClusterer* y se adicionan otros. De la misma manera, el método principal de esta clase constituye *buildClusterer()*, en donde se efectúa el proceso de agrupamiento. Además, la clase *COSMIC* implementa determinadas interfaces que permiten manejar los parámetros que necesita el algoritmo para su adecuado funcionamiento como, por ejemplo, la cantidad mínima de puntos que debe contener un grupo y el radio *épsilon* de cada región, ya que modificando estos valores el usuario puede determinar la cantidad de grupos que se generan.

Ambos algoritmos emplean la distancia de Hausdorff para realizar el cálculo de la distancia entre las bolsas del conjunto de datos. No obstante, si otras medidas para comparar bolsas fueran incorporadas, entonces pudieran ser utilizadas en estos métodos.

2.5 Adaptación de algoritmos simple-instancia al agrupamiento de datos multi-instancias

La adaptación de algoritmos de agrupamiento simple-instancia a multi-instancias es sumamente importante ya que los primeros no están capacitados para manejar conjuntos de

datos con información multi-instancias. Como se hizo referencia previamente, estos algoritmos en Weka se encuentran situados en el paquete *clusterers*, por lo cual, las modificaciones que se llevaron a cabo también se incorporan en este mismo paquete.

De los algoritmos clásicos de agrupamiento que posee Weka, existen algunos que son basados en distancias y otros no. En esta investigación se decidió adaptar los métodos *Canopy* y *HierarchicalClusterer* al manejo de datos multi-instancias ya que son algoritmos basados en distancias y uno de ellos crea una partición, y el otro, forma una jerarquía, lo que los hace diversos.

```
+-miClusterers.jar
+-Description.props
+-build_package.xml
+-src
|    +-main
|    |     +-java
|    |           +-weka
|    |                 +-clusterers
|    |                             +-Canopy_MIL.java
|    |                             +-HierarchicalClusterer_MIL.java
|    |
|    +-test
|          +-java
|                +-weka
|                      +-clusterers
|                                  +-Canopy_MILTest.java
|                                  +-HierarchicalClusterer_MILTest.java
|
+-lib
+-doc
```

Figura 14. Estructura de directorios para algoritmos de agrupamiento.

El algoritmo Canopy es usado frecuentemente como una estrategia de inicialización para el algoritmo *k-means* o como un método de agrupamiento de aproximación rápida para utilizarlo en grandes conjuntos de datos, donde la aplicación directa de otros algoritmos sería impráctica. El algoritmo particiona el conjunto de datos de entrada en regiones de proximidad (*canopies*) en forma de hiperesferas definidas por una distancia t_1 desde el centro de la región. Una segunda distancia ajustada t_2, donde $t_2 < t_1$, se utiliza para controlar la cantidad de

canopies o grupos que son formados por el algoritmo. Al menos se requiere ejecutar dos pasadas sobre los datos (McCallum, Nigam and Ungar, 2000; Frank, Hall and Witten, 2016).

El algoritmo HierarchicalClusterer es un método jerárquico aglomerativo. La ejecución del algoritmo comienza considerando cada instancia como un grupo en sí mismo, luego busca los dos grupos más cercanos, los combina y continúa haciendo esta operación hasta que quede un solo grupo. El registro de fusiones crea una estructura de agrupación jerárquica llamada dendrograma jerárquico (Ian H. *et al.*, 2011; Frank, Hall and Witten, 2016).

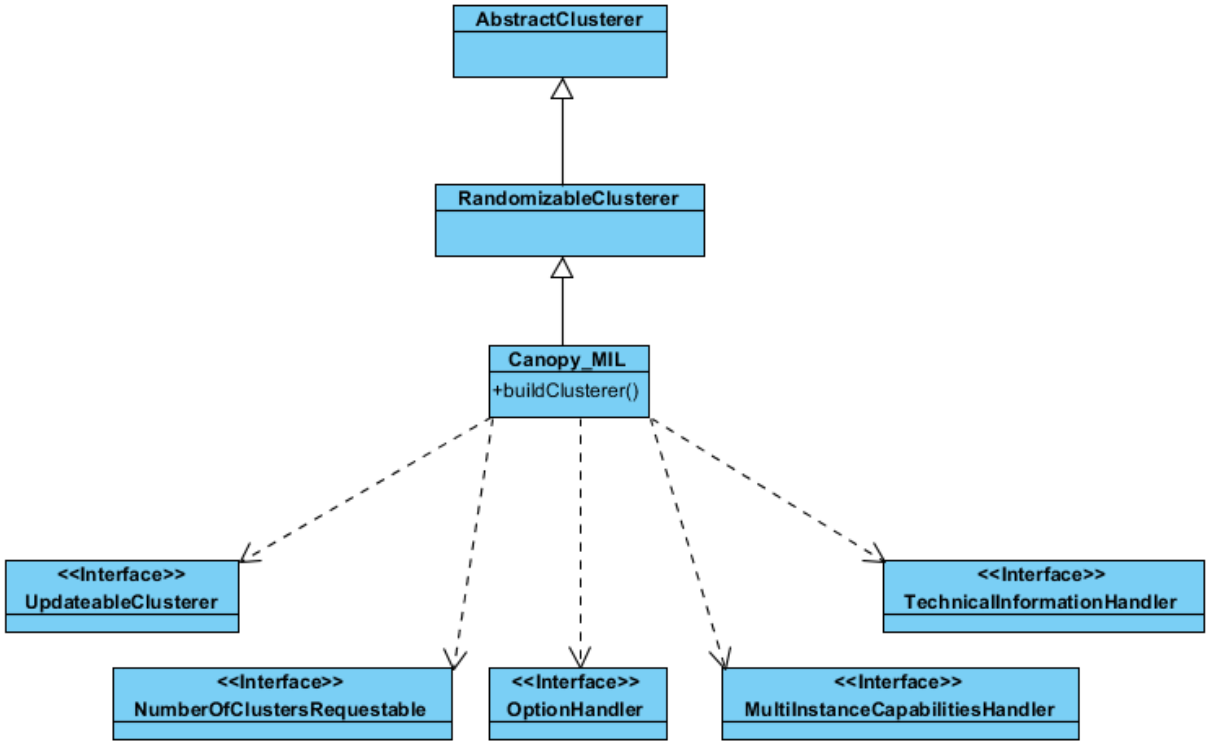

Figura 15. Diagrama de clases del algoritmo Canopy_MIL.

La transformación de estos algoritmos para que logren manejar datos multi-instancias requiere que primeramente se cree una estructura de directorios que permita conceder la ubicación a dichos algoritmos modificados en la plataforma Weka, como se muestra en la Figura 14. Para adaptar el algoritmo Canopy se implementó una clase en Java llamada *Canopy_MIL*, que contiene los métodos necesarios para realizar el proceso de agrupamiento y otros adicionales que proporcionan información acerca del algoritmo. En la Figura 15 se muestra el diagrama de clases correspondiente. Una de las primeras transformaciones que se efectuó corresponde al

cambio de la distancia Euclidiana por la distancia de Hausdorff, ya que esta última está diseñada para calcular distancias entre bolsas. Luego se agregaron los métodos *getMultiInstanceCapabilities()* y *setDoNotCheckCapabilities()* los cuales permiten dotar al algoritmo de ciertas propiedades para manejar problemas multi-instancias.

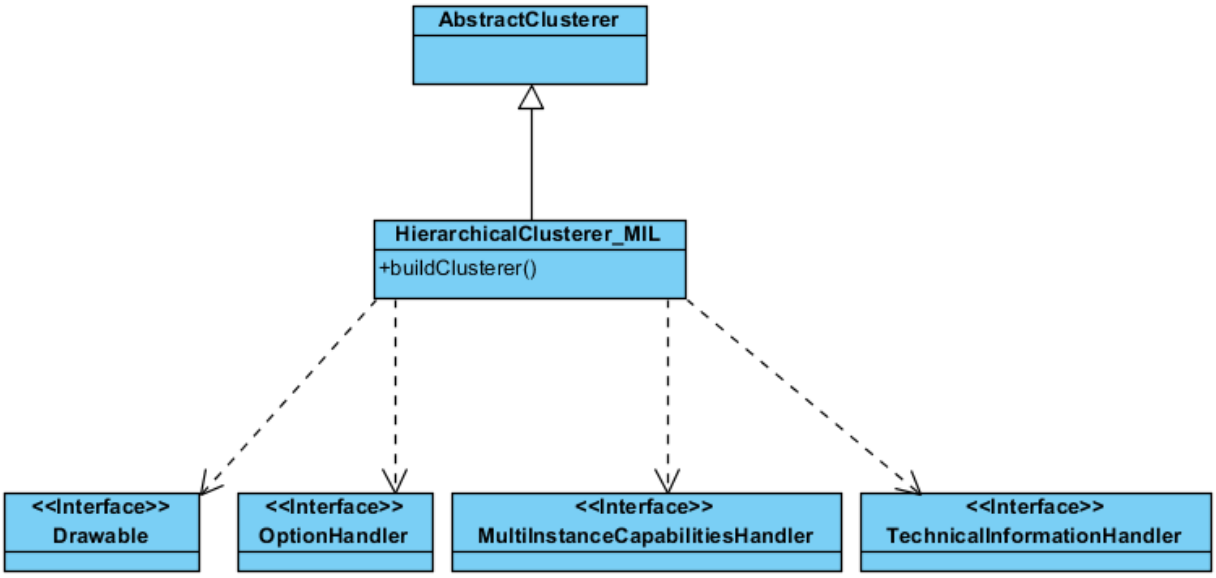

Figura 16. Diagrama de clases del algoritmo HierarchicalClusterer_MIL.

Para lograr la adaptación del algoritmo HierarchicalClusterer se implementó una clase en Java llamada *HierarchicalClusterer_MIL*, que contiene los métodos necesarios para realizar el proceso de agrupamiento y otros adicionales que aportan información acerca del algoritmo. En la Figura 16 se muestra el diagrama de clases correspondiente. Una de las primeras transformaciones que se efectuó corresponde al cambio de la distancia Euclidiana por la distancia de Hausdorff, ya que esta última está diseñada para calcular distancias entre bolsas. Luego, los métodos *getMultiInstanceCapabilities()* y *setDoNotCheckCapabilities()* fueron agregados para conceder al algoritmo ciertas propiedades para manejar problemas de tipo multi-instancias.

2.6 Conclusiones parciales del capítulo

La clase *HausdorffDistance* permitió la incorporación a Weka de las distancias mínima, máxima y promedio de Hausdorff, mediante la implementación de tres interfaces, una de ellas *DistanceFunction*, la cual es la encargada de proveer el esquema general de las distancias.

Los filtros Average mapping, Min-Max mapping y Moments mapping permiten aplicar a cualquier problema multi-instancias los métodos de aprendizaje simple-instancia incorporados al Weka. La incorporación estos envoltorios, clasificados como métodos de mapeo basados en bolsas, se realizó en el paquete *filters*, a partir de la implementación de la clase *Filter* de Weka.

Los algoritmos de agrupamiento multi-instancias BAMIC y COSMIC fueron incorporados al paquete *clusterers* de Weka mediante la creación de una clase para cada algoritmo que hereda de la clase *AbstractClusterer*, que es la encargada de definir la estructura de cualquier esquema de agrupamiento, ya sea multi-instancias o simple-instancia. BAMIC y COSMIC fueron seleccionados porque pertenecen a distintas categorías de agrupamiento.

Los algoritmos de agrupamiento simple-instancia Canopy y HierarchicalClusterer fueron adaptados para que sea capaces de agrupar datos multi-instancias. Para ello, se reemplazó la distancia Euclidiana que empleaban para comparar objetos por la distancia de Hausdorff para comparar bolsas, y se agregaron los métodos *getMultiInstanceCapabilities()* y *setDoNotCheckCapabilities()* para dotar los algoritmos de ciertas propiedades para manejar problemas multi-instancias.

Capítulo 3 Estudio experimental del agrupamiento multi-instancias

En este capítulo se describe cómo se efectuó la validación de los algoritmos y los filtros incorporados a Weka para manejar el agrupamiento multi-instancias. Primeramente, se describe el conjunto de datos utilizado y los subconjuntos obtenidos a partir de este, ya que el primero es muy extenso y dificulta la ejecución de los algoritmos multi-instancias y de los algoritmos simple-instancia actualizados. Se presenta una descripción detallada de cómo se llevó a cabo el proceso de validación de los algoritmos de agrupamiento multi-instancias, de los algoritmos simple-instancia actualizados y de los envoltorios para algoritmos simple-instancia.

3.1 Descripción de los conjuntos de datos

El conjunto de datos utilizado en esta investigación se denomina Corel20 y está almacenado en un archivo de tipo arff. El mismo está compuesto por un gran número de imágenes las cuales están agrupadas en 20 clases o categorías diferentes. Estas clases son: African people and villages, Beach, Historical buildings, Buses, Dinosaurs, Elephants, Flowers, Horses, Mountains and glaciers, Food, Dogs, Lizards, Fashion, Sunsets, Cars, Waterfalls, Antiques, Battle ships, Skiing y Desserts. Cada una de ellas se corresponde con una etiqueta para su respectiva identificación que son números enteros que van desde el 0 hasta el 19. Cada imagen constituye una bolsa, conformada por varias instancias que se corresponden con las regiones obtenidas de la segmentación de la imagen.

Para llevar a cabo la realización de pruebas a los distintos algoritmos, el conjunto de datos Corel20 fue dividido en subconjuntos más pequeños de diferentes tamaños, los cuales contienen categorías de imágenes totalmente diferentes. Se elaboraron un total de 25 subconjuntos de datos conformados por diversas categorías (100 bolsas por cada categoría), los cuales se describen a continuación.

Conjuntos de datos con dos categorías:

- Clases 0 y 3 (0: African people and villages, 3: Buses)
- Clases 0 y 14 (0: African people and villages, 14: Cars)

- Clases 0 y 17 (0: African people and villages, 17: Battle ships)
- Clases 1 y 7 (1: Beach, 7: Horses)
- Clases 3 y 19 (3: Buses, 19: Desserts)
- Clases 6 y 10 (6: Flowers, 10: Dogs)
- Clases 6 y 18 (6: Flowers, 18: Skiing)
- Clases 11 y 12 (11: Lizards, 12: Fashion)

Conjuntos de datos con tres categorías:

- Clases 0, 3 y 11 (0: African people and villages, 3: Buses, 11: Lizards)
- Clases 2, 4 y 6 (2: Historical buildings, 4: Dinosaurs, 6: Flowers)
- Clases 3, 7 y 12 (3: Buses, 7: Horses, 12: Fashion)
- Clases 5, 6 y 14 (5: Elephants, 6: Flowers, 14: Cars)
- Clases 5, 16 y 18 (5: Elephants, 16: Antiques, 18: Skiing)
- Clases 6, 7 y 17 (6: Flowers, 7: Horses, 17: Battle ships)
- Clases 10, 12 y 14 (10: Dogs, 12: Fashion, 14: Cars)
- Clases 12, 14 y 15 (12: Fashion, 14: Cars, 15: Waterfalls)

Conjuntos de datos con cinco categorías:

- Clases 1, 3, 5, 9 y 16 (1: Beach, 3: Buses, 5: Elephants, 9: Food, 16: Antiques)
- Clases 2, 6, 10, 12 y 14 (2: Historical buildings, 6: Flowers, 10: Dogs, 12: Fashion, 14: Cars)
- Clases 3, 4, 9, 12 y 13 (3: Buses, 4: Dinosaurs, 9: Food, 12: Fashion, 13: Sunsets)
- Clases 3, 6, 7, 12 y 15 (3: Buses, 6: Flowers, 7: Horses, 12: Fashion, 15: Waterfalls)
- Clases 9, 10, 12, 14 y 19 (9: Food, 10: Dogs, 12: Fashion, 14: Cars, 19: Desserts)

Conjuntos de datos con siete categorías:

- Clases 0, 3, 4, 6, 9, 15 y 16 (0: African people and villages, 3: Buses, 4: Dinosaurs, 6: Flowers, 9: Food, 15: Waterfalls, 16: Antiques)
- Clases 1, 2, 3, 5, 6, 9 y 16 (1: Beach, 2: Historical buildings, 3: Buses, 5: Elephants, 6: Flowers, 9: Food, 16: Antiques)

- Clases 2, 3, 4, 6, 9, 12 y 16 (2: Historical buildings, 3: Buses, 4: Dinosaurs, 6: Flowers, 9: Food, 12: Fashion, 16: Antiques)
- Clases 2, 6, 9, 10, 12, 13 y 14 (2: Historical buildings, 6: Flowers, 9: Food, 10: Dogs, 12: Fashion, 13: Sunsets, 14: Cars)

La conformación de estos subconjuntos de datos permite estudiar las variantes de agrupamiento multi-instancias implementadas teniendo en cuenta diversidad de imágenes y diferentes cantidades de grupos a descubrir.

3.2 Resultados experimentales

Los resultados experimentales obtenidos se derivaron de la ejecución de cada uno de los algoritmos de agrupamiento multi-instancias, de los algoritmos simple-instancia actualizados, así como de los envoltorios para algoritmos simple-instancia sobre los conjuntos de datos mencionados en el epígrafe 3.1. Posteriormente, a través de la utilización de la herramienta MOA se elaboró un ranking para determinar el orden de los algoritmos atendiendo a los resultados alcanzados por cada uno de ellos. Este ranking se confeccionó mediante el desarrollo de pruebas estadísticas basadas en la precisión de cada algoritmo para realizar el proceso de agrupamiento. Primeramente, se efectuó el test de Friedman para determinar si existen diferencias significativas entre los algoritmos que son de nuestro interés, y luego se realizó el test de Holm para determinar entre cuáles de estos algoritmos existen diferencias significativas.

Se realizó la ejecución del algoritmo BAMIC con las tres variantes de la distancia de Hausdorff, la mínima, la máxima y la promedio, sobre los conjuntos de datos obtenidos previamente. En la Figura 17 se aprecia que el algoritmo BAMIC, al ser ejecutado utilizando la distancia promedio de Hausdorff para comparar bolsas obtiene los mejores resultados al realizar el agrupamiento. También se observa que posee diferencias significativas con respecto a la aplicación de dicho algoritmo con la distancia mínima de Hausdorff y la distancia máxima de Hausdorff, mientras que estas últimas no poseen diferencias significativas entre ellas.

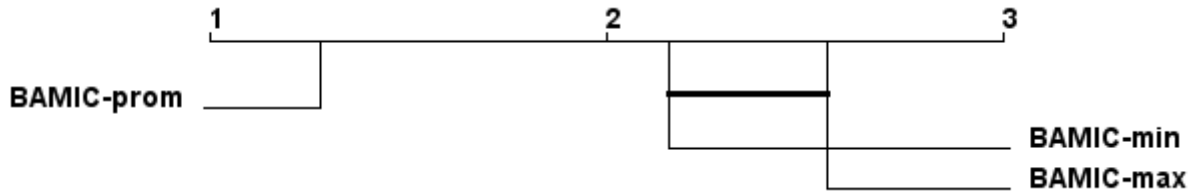

Figura 17. Resultados de la experimentación del algoritmo BAMIC variando distancias.

Al realizar la ejecución del algoritmo COSMIC empleando las tres variantes de la distancia de Hausdorff, la mínima, la máxima y la promedio, sobre los conjuntos de datos obtenidos previamente, se obtuvo como resultado que el algoritmo COSMIC utilizando la distancia mínima de Hausdorff para comparar bolsas alcanza los mejores resultados al realizar el proceso de agrupamiento. Además, en la Figura 18 se observa que el algoritmo COSMIC aplicando la distancia mínima de Hausdorff no presenta diferencias significativas con respecto a la aplicación de dicho algoritmo con la distancia máxima de Hausdorff y la distancia promedio de Hausdorff.

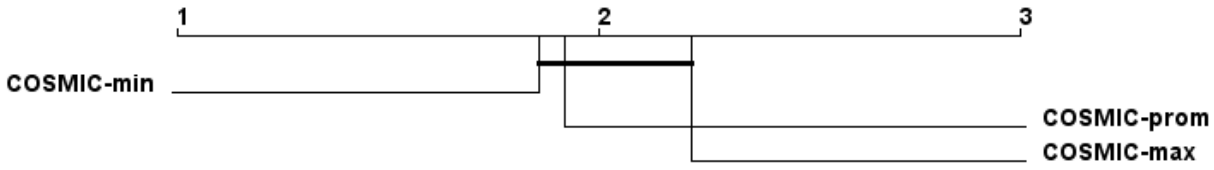

Figura 18. Resultados de la experimentación del algoritmo COSMIC variando distancias.

Realizando las ejecuciones de los algoritmos BAMIC y COSMIC con las tres variantes de la distancia de Hausdorff, mínima, máxima y promedio, para comparar bolsas, es posible apreciar en la Figura 19 que el algoritmo BAMIC alcanza los mejores resultados con las tres variantes de la distancia de Hausdorff con respecto al algoritmo COSMIC. Ambos algoritmos presentan diferencias significativas entre ellos. Por tanto, se puede concluir que BAMIC tiene mejor rendimiento que COSMIC para los conjuntos de datos analizados y que las variantes de

la distancia de Hausdorff no arrojan diferencias significativas en los resultados de los algoritmos respecto a la precisión.

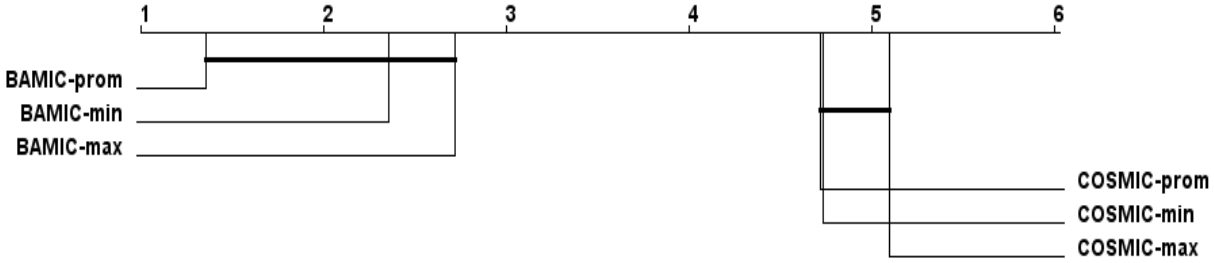

Figura 19. Resultados de la experimentación del algoritmo BAMIC vs COSMIC.

La transformación de los datos multi-instancias en simple-instancia se llevó a cabo mediante la aplicación de los filtros AverageMapping, Min-MaxMapping y MomentsMapping. Luego de aplicar cualquiera de estos filtros es posible ejecutar los algoritmos simple-instancia que posee Weka para realizar tareas de agrupamiento. En la Figura 20 se muestran los resultados derivados de la aplicación de los filtros y la posterior utilización del algoritmo simple-instancia Canopy para realizar el agrupamiento. Con la aplicación del envoltorio Min-MaxMapping y la ejecución del algoritmo Canopy se logró el mejor agrupamiento posible. La aplicación del filtro Min-MaxMapping con Canopy no presenta diferencias significativas con respecto a la aplicación del filtro AverageMapping con este mismo algoritmo, en cambio, ambas variantes sí presentan diferencias significativas con respecto a la aplicación del filtro MomentsMapping y posterior ejecución del algoritmo Canopy.

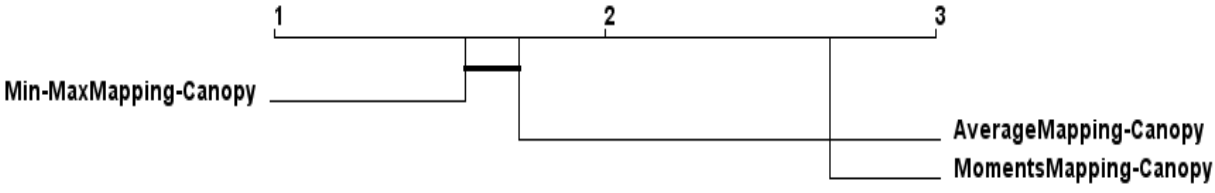

Figura 20. Resultados de la experimentación al aplicar los filtros y utilizar el algoritmo simple-instancia Canopy.

Al realizar la transformación de los datos multi-instancias en simple-instancia mediante la aplicación de los filtros AverageMapping, Min-MaxMapping y MomentsMapping, y la posterior ejecución del algoritmo simple-instancia HierarchicalClusterer se lograron los resultados mostrados en la Figura 21. Con la aplicación del filtro AverageMapping y la ejecución del algoritmo HierarchicalClusterer se logró el mejor agrupamiento posible. Esta composición del filtro y el algoritmo no presenta diferencias significativas con respecto a la aplicación del filtro Min-MaxMappinng con el algoritmo HierarchicalClusterer y tampoco con el filtro MomentsMapping y la ejecución del mismo algoritmo. Por tanto, podemos concluir que los tres filtros implementados producen un comportamiento similar en HierarchicalClusterer, no así en Canopy.

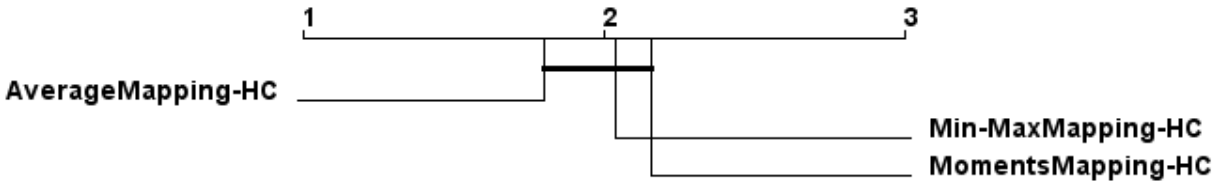

Figura 21. Resultados de la experimentación al aplicar los filtros y utilizar el algoritmo simple-instancia HierarchicalClusterer.

El algoritmo Canopy_MIL constituye la modificación del algoritmo simple-instancia Canopy, mientras que el algoritmo HierarchicalClusterer_MIL constituye la transformación del algoritmo simple-instancia HierarchicalClusterer. Al realizar la ejecución del algoritmo Canopy_MIL con la distancia promedio de Hausdorff como medida para comparar bolsas se alcanzaron mejores resultados en el agrupamiento que con respecto a la aplicación del algoritmo HierarchicalClusterer_MIL con la distancia promedio de Hausdorff; sin embargo, ambos algoritmos no presentan diferencias significativas.

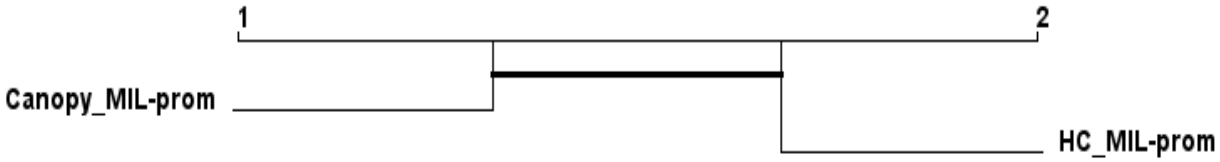

Figura 22. Resultados de la experimentación del algoritmo Canopy_MIL vs HierarchicalClusterer_MIL.

En la Figura 23 se muestran los resultados obtenidos a partir de la aplicación de una selección de los algoritmos y filtros que aportan los mejores resultados en el proceso de agrupamiento, atendiendo a las tres alternativas posibles para enfrentar un problema de tipo multi-instancias. A partir de los resultados obtenidos por estos algoritmos y filtros seleccionados, se elaboró un ranking para determinar cuál o cuáles son las mejores alternativas para manejar los problemas multi-instancias. El algoritmo BAMIC se ejecutó calculando la distancia promedio de Hausdorff para comparar bolsas, se seleccionó el filtro Min-MaxMapping para transformar los datos a simple-instancia y poder aplicar el algoritmo Canopy clásico, el algoritmo COSMIC se ejecutó calculando la distancia mínima de Hausdorff para comparar bolsas, se seleccionó el filtro AverageMapping para transformar los datos a simple-instancia y poder aplicar el algoritmo simple-instancia HierarchicalClusterer, y finalmente, se seleccionaron las transformaciones Canopy_MIL y HierarchicalClusterer_MIL, las cuales compararon las bolsas utilizando la distancia promedio de Hausdorff. Como resultado de este estudio experimental, se obtuvo que el algoritmo multi-instancias BAMIC con la distancia promedio de Hausdorff fue el que logró agrupamientos con mayor precisión, aunque no presenta diferencias significativas con los resultados arrojados por el algoritmo Canopy al aplicar previamente el filtro Min-MaxMapping, ni con la modificación Canopy_MIL. En cambio, sí presenta diferencias significativas con respecto al algoritmo multi-instancias COSMIC, la transformación HierarchicalClusterer_MIL y el algoritmo simple-instancia HierarchicalClusterer al aplicar previamente el filtro AverageMapping.

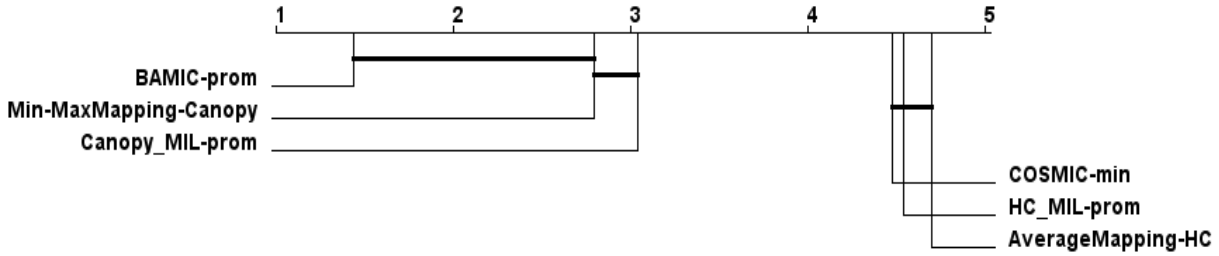

Figura 23. Resultados de la experimentación de los mejores algoritmos desde las tres perspectivas para afrontar un problema MIL.

3.3 Conclusiones parciales del capítulo

La conformación de diversos subconjuntos de datos a partir de la colección Corel20 permitió explorar el comportamiento de los filtros y los algoritmos implementados al tener que agrupar imágenes de diversas naturalezas y pertenecientes a una gama de categorías (grupos de tamaño dos, tres, cinco y siete).

Las tres variantes implementadas de la distancia de Hausdorff arrojan comportamientos similares en los métodos BAMIC y COSMIC, aunque se destacan diferencias significativas en el caso de BAMIC con la distancia promedio de Hausdorff. Por otro lado, podemos concluir que los tres filtros implementados producen un comportamiento similar en HierarchicalClusterer, no así en Canopy, donde el filtro MomentsMapping genera diferencias significativas respecto a los resultados obtenidos al aplicar los filtros Min-MaxMapping y AverageMapping. Tampoco se detectan diferencias significativas entre los resultados reportados por los dos métodos de agrupamiento clásicos, Canopy y HierarchicalClusterer, transformados a manejo de datos multi-instancias. De manera general, BAMIC resultó ser un algoritmo ganador y exitoso respecto a la precisión que logra en el agrupamiento, seguido de Canopy tanto en su variante simple-instancia como en su variante multi-instancias, marcando diferencias respecto a HierarchicalClusterer y COSMIC.

Conclusiones

Como resultado de esta investigación se incorporaron al Weka los algoritmos BAMIC y COSMIC, tres filtros que permiten transformar problemas multi-instancias en problemas simple-instancia y se adaptaron los métodos Canopy y HierarchicalClusterer para el manejo de datos multi-instancias, permitiendo realizar un estudio del agrupamiento siguiendo los tres enfoques del manejo de problemas multi-instancias; cumpliéndose de esta forma el objetivo general planteado, ya que:

1. El agrupamiento ha sido poco abordado en el aprendizaje multi-instancias, donde solo cinco algoritmos se han encontrado en la literatura; no obstante, gran parte del desarrollo de las técnicas de clasificación multi-instancias pueden ser aplicables al entorno no supervisado, tales como las taxonomías existentes para categorizar los métodos, así como los enfoques que existen para abordar este tipo de aprendizaje a través del desarrollo de algoritmos ad-hoc, envoltorios o métodos de mapeo.
2. El diseño extensible de Weka permitió la incorporación de métodos envoltorios mediante la creación del paquete *miFilters* que implementa la clase *Filter*, los algoritmos de agrupamiento multi-instancias mediante la creación del paquete *miClusterers* y la transformación de los algoritmos de agrupamiento simple-instancia mediante la sustitución de las distancias para comparar vectores por las variantes de la distancia de Hausdorff para comparar bolsas.
3. BAMIC resultó ser un algoritmo ganador y exitoso respecto a la precisión que logra en el agrupamiento, seguido de Canopy clásico y multi-instancias, marcando diferencias respecto a HierarchicalClusterer y COSMIC. Las distancias de Hausdorff arrojan comportamientos similares, aunque se destacan diferencias significativas de la variante promedio en el caso de BAMIC, mientras que los tres filtros implementados producen un comportamiento similar en HierarchicalClusterer, no así en Canopy, donde MomentsMapping genera resultados significativamente superiores.

Recomendaciones

Derivadas del estudio realizado, así como de las conclusiones generales emanadas del mismo, se recomienda:

- Incorporar al Weka otros algoritmos para el agrupamiento multi-instancias, tales como: UC-KNN, MIEM y M3IC-MBM.
- Continuar adaptando el resto de los algoritmos de agrupamiento simple-instancia disponibles en Weka para que sean capaces de agrupar datos multi-instancias.

Referencias bibliográficas

Amores, J. (2013) 'Multiple instance classification: review, taxonomy and comparative study', *Artificial Intelligence*. Elsevier B.V., 201, pp. 81–105. Available at: http://dx.doi.org/10.1016/j.artint.2013.06.003.

Ankerst, M. *et al.* (1999) 'OPTICS: ordering points to identify the clustering structure', in *Proceedings of the International Conference on Management of data (SIGMOD 1999)*, pp. 49–60.

Campbell, A. T., Ping, W. and Xu, Y. (2011) 'Multi-instance metric learning', in *11th IEEE International Conference on Data Mining*, pp. 874–883. doi: 10.1109/ICDM.2011.106.

Carbonneau, M. (2016) *Introduction to multiple instance learning*. Available at: http://www.etsmtl.ca/Unites-de-recherche/LIVIA/Seminars/Introduction-to-Multiple-Instance-Learning.pdf.

Carbonneau, M. A. *et al.* (2017) 'Multiple instance learning: a survey of problem characteristics and applications', *Pattern Recognition*, 77, pp. 1–25. doi: 10.1016/j.patcog.2017.10.009.

Chen, Y. and Wu, O. (2012) 'Contextual hausdorff dissimilarity for multi-instance clustering', in *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012)*. IEEE, Los Alamitos (2012), pp. 870–873.

Devaney, R. L. (2007) *Measure, topology, and fractal geometry (Gerald A. Edgar)*. 2nd ed. Edited by Springer. doi: 10.1137/1033153.

Deza, M. M. and Deza, E. (2012) *Encyclopedia of distances*. 2nd ed. Edited by Springer.

Dietterich, T. G., Lathrop, R. H. and Lozano-Pérez, T. (1997) 'Solving the multiple instance problem with axis-parallel rectangles', *Artificial Intelligence*, 89(1–2), pp. 31–71. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0004370296000343.

Dong, L. (2006) *A comparison of multi-instance learning algorithms, Science*. The University of Waikato, New Zeland. Available at: http://adt.waikato.ac.nz/uploads/approved/adt-uow20081023.115544/public/01front.pdf.

Eiter, T. and Mannila, H. (1997) 'Distance measures for point sets and their computation', *Acta Informatica*, 34(2), pp. 109–133. Available at: http://link.springer.com/10.1007/s002360050075.

Foulds, J. (2008) *Learning instance weights in multi-instance learning, University of Waikato*. University of Waikato.

Foulds, J. and Frank, E. (2010) 'A review of multi-instance learning assumptions', *Knowledge Engineering Review*, 25(1), pp. 1–25. doi: 10.1017/S026988890999035X.

Frank, E., Hall, M. A. and Witten, I. H. (2016) *The WEKA workbench, Morgan Kaufmann, Fourth Edition*. Available at: https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf.

Gärtner, T. *et al.* (2002) 'Multi-instance kernels', in *Proceedings of the 19th International Conference on Machine Learning (ICML 2002)*, pp. 179–186.

Hajimirsadeghi, H. and Mori, G. (2017) 'Multi-instance classification by max-margin training of cardinality-based markov networks', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9), pp. 1839–1852.

Haripriya, H. *et al.* (2016) 'Multi label prediction using association rule generation and simple k-means', in *2016 International Conference on Computational Techniques in Information and Communication Technologies, ICCTICT 2016 - Proceedings*, pp. 159–163. doi: 10.1109/ICCTICT.2016.7514571.

Henegar, C., Clément, K. and Zucker, J.-D. (2006) 'Unsupervised multiple-instance learning for functional profiling of genomic data', in *Proceedings of the 17th European Conference on Machine Learning (ECML 2006)*, pp. 186–197.

Herrera, F. *et al.* (2016) *Multiple instance learning*. Available at: http://link.springer.com/10.1007/978-3-319-47759-6.

Ian H., W. *et al.* (2011) *Data mining practical machine learning tools and techniques*. 3rd ed. Morgan Kaufmann.

Jin, R., Wang, S. and Zhou, Z.-H. (2009) 'Learning a distance metric from multi-instance

multi-label Data', in. Miami, FL, USA: IEEE. Available at: https://www.cse.msu.edu/~rongjin/publications/cvpr-1208.pdf.

Kriegel, H. P. *et al.* (2006) 'COSMIC: Conceptually specified multi-instance clusters', in *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 917–921.

Kriegel, H., Pryakhin, A. and Schubert, M. (2006) 'An EM-approach for clustering multi-instance objects', in *10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'06), Singapore, 2006*. Singapore, pp. 1–10.

Li, B. *et al.* (2015) 'Multi-view multi-instance learning based on joint sparse representation and multi-view dictionary learning', 14(8), pp. 1–8. Available at: http://eprints.bbk.ac.uk/18136/1/18136.pdf.

Macqueen, J. (1967) 'Some methods for classification and analysis of multivariate observations', 233(233), pp. 281–297.

McCallum, A., Nigam, K. and Ungar, L. H. (2000) 'Efficient clustering of high-dimensional data sets with application to reference matching', in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, pp. 169–178. Available at: http://portal.acm.org/citation.cfm?doid=347090.347123.

Melki, G., Cano, A. and Ventura, S. (2018) 'MIRSVM: multi-instance support vector machine with bag representatives', *Pattern Recognition*. Elsevier Ltd, 79(February), pp. 228–241. Available at: https://doi.org/10.1016/j.patcog.2018.02.007.

Mitchel, T. M. (1997) *Machine learning*. McGraw-Hill Science/Engineering/Math; (March 1, 1997).

Roy, A., Banerjee, B. and Murino, V. (2017) 'A novel dictionary learning based multiple instance learning approach to action recognition from videos', in *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, pp. 519–526. Available at: http://www.scitepress.org/DigitalLibrary/Link.aspx?

Sajana, T., Rani, C. M. S. and Narayana, K. V (2016) 'A survey on clustering techniques for big data mining', *Indian Journal of Science and Technology*, 9(3), pp. 1–12.

Sørensen, L. *et al.* (2010) 'Dissimilarity-based multiple instance learning', *Structural, Syntactic, and Statistical Pattern Recognition (SSPR & SPR)*, 6218 LNCS, pp. 1–10. doi: 10.1007/978-3-642-14980-1_12.

Tarel, J. P., Boughorbel, S. and Boujema, N. (2005) 'Intermediate matching kernel for image local features', in *Proceedings of International Joint Conference on Neural Networks, Montreal, Canada*, pp. 889–894.

Tax, D. M. J. *et al.* (2011) 'Bag dissimilarities for multiple instance learning', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7005 LNCS, pp. 222–234. doi: 10.1007/978-3-642-24471-1_16.

Tran, Q. N. *et al.* (2017) 'Multiple instance learning with the optimal sub-pattern assignment metric'. Available at: http://arxiv.org/abs/1703.08933.

Wagstaff, K. L., Lane, T. and Roper, A. (2008) 'Multiple-instance regression with structured data', in *Proceedings - IEEE International Conference on Data Mining Workshops, ICDM Workshops 2008*, pp. 291–300. doi: 10.1109/ICDMW.2008.31.

Wang, H., Nie, F. and Huang, H. (2013) 'Robust and discriminative self-taught learning', *Icml*, 28, pp. 298–306.

Wong, K.-C. (2015) 'A short survey on data clustering algorithms', pp. 1–30. doi: 10.1109/ISCMI.2015.10.

Wu, J. *et al.* (2015) 'Deep multiple instance learning for image classification and auto-annotation', in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3460–3469. doi: 10.1109/CVPR.2015.7298968.

Xu, D. and Tian, Y. (2015) 'A comprehensive survey of clustering algorithms', *Annals of Data Science*. Springer Berlin Heidelberg, (October), pp. 1–30.

Xu, X. (2003) *Statistical learning in multiple instance problems*. University of Waikato, New Zeland.

Zhang, C. and Chen, X. (2005) 'Region-based image clustering and retrieval using multiple instance learning', in *Proceedings of the 4th International Conference on Image and Video*

Retrieval (CIVR), pp. 194–204.

Zhang, D. *et al.* (2011) 'Maximum margin multiple instance clustering with applications to image and text clustering', *IEEE Transactions on Neural Networks*, 22(5), pp. 739–751.

Zhang, M. and Zhou, Z. (2009) 'Multi-instance clustering with applications to multi-instance prediction', *Applied Intelligence*, 31(1), pp. 47–68.