Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

"Plataforma para Algoritmos de Agrupamiento con Traslape"

TESIS

Que para obtener el título de

Licenciado en Ciencias de la Computación

Presenta

Argenis Aldair Aroche Villarruel

Asesores:

Dr. José Arturo Olvera López
BUAP
Dr. José Francisco Martínez Trinidad
INAOE
Dr. Jesús Ariel Carrasco Ochoa
INAOE
Dr. Airel Pérez Suárez
CENATAV

Junio 2013

Agradecimientos

He concluido este trabajo de tesis no solo gracias a mi esfuerzo, sino también gracias al apoyo de varias personas, quienes me brindaron sus consejos y estuvieron al pendiente de mi avance durante mi desarrollo de tesis y durante mi formación mientras estuve en la universidad.

Quiero agradecer a mi mamá Amada Villarruel Solano, por todo el amor que me has brindado, por todo el apoyo que me otorgaste en todas las etapas de mi vida y también por el gran esfuerzo que represento para ti el poder apoyarme al realizar mis estudios universitarios fuera de Izúcar. Espero que te sientas orgullosa de mí.

Agradezco a mi papá Agustín Aroche Orzuna, por todos tus consejos y enseñanzas que me has dado desde pequeño, además de lo mucho que has trabajado por mí. Espero que también te sientas orgulloso de lo que he logrado gracias a tu apoyo.

Quiero agradecer a mi hermano Agustín Andrés y a mi cuñada Lidia Luna, por brindarme sus consejos y por apoyarme de distintas maneras durante el transcurso que estudie en la universidad. También un agradecimiento a mi hermana Andrea Abigail, por tu apoyo incondicional y por el cariño que me brindas a pesar de nuestras diferencias.

Agradezco mi tía Teresa Aroche Orzuna, por sus consejos y el apoyo que me brindo en muchos aspectos no solo durante mis estudios en la universidad, sino desde que era apenas un niño.

Agradezco al Dr. Arturo Olvera, porque fue mi mentor durante los últimos dos años de carrera, además le agradezco por permitirme participar con usted en diversos proyectos del área de Reconocimiento de Patrones, área que me gustó mucho y que aprendí gracias a usted. Además le agradezco por aceptar ser mi asesor de tesis y por introducirme en el área de investigación.

Quiero agradecer a mis asesores del INAOE Dr. Francisco Martínez Trinidad y Dr. Ariel Carrasco Ochoa, por la confianza que tuvieron en mí para realizar parte de uno de sus proyectos,

a partir del cual realice esta tesis, agradezco por la paciencia que me tuvieron en todas las revisiones y correcciones de tesis, también agradezco sus consejos.

Agradezco también a mi asesor del CENATAV Dr. Airel Pérez Suarez, por su ayuda en la planeación del proyecto y por todas las dudas que me resolvió, le agradezco también por el tiempo que le dedico a la revisión de mi tesis y por todo el apoyo que me brindo a pesar de encontrarse en otro país.

Quiero agradecer a mis sinodales el Dr. Manuel Martín Ortiz y el Dr. Ivan Olmos Pineda, por haber aceptado ser parte de mi jurado y por el tiempo que dedicaron a la revisión de mi tesis.

Agradezco a mis amigos de la facultad Javier Cervantes y Jesús Cruz, por todo el apoyo, la confianza y los consejos que me brindaron, sé que puedo contar con ustedes para cualquier cosa, juntos compartimos muchas experiencias, sepan que los considero a ambos como mis hermanos.

Dedico un agradecimiento a mis amigos Inamic Meza, Andy Cruz y Marisol Peña, por las momentos que pase con ustedes, les agradezco por acompañarme en esta etapa de mi vida. También agradezco a mis amigos Joshua Betancourt, Jave Jael y a mi primo Roberto Vidales con quienes he pasado y sigo pasando ratos de mucha distracción para liberar el estrés.

Agradezco a la VIEP por las becas que me proporciono en los diferentes programas de apoyo que brinda cada año, así como a la DIIE por el programa de beca Académica. Agradezco también al CONACyT (Proyecto Referencia – CB2008/106366) por la beca que me proporciono durante el desarrollo de mi tesis y al INAOE por brindar el espacio donde se realizaron las reuniones con mis asesores de tesis.

Sin duda alguna sé que he omitido a varias personas que me apoyaron de alguna manera para poder concluir con mis estudios en la BUAP, y me disculpo por no tener sus nombres escritos en estas hojas, pero eso no les resta importancia y les agradezco por todo.

Resumen

En la actualidad, diversos problemas requieren soluciones computacionales del área de Reconocimiento de Patrones (recuperación de información, biometría, etc.). En la literatura se han reportado varias plataformas, las cuales permiten la utilización de algoritmos de reconocimiento de patrones. Sin embargo, las plataformas reportadas no permiten la evaluación de algoritmos de agrupamiento con traslape. Adicionalmente, estas plataformas no permiten la inclusión de medidas de evaluación, limitando al usuario a utilizar solo las que están incluidas en la plataforma.

Este trabajo presenta los resultados de una plataforma que incluye algunos algoritmos de agrupamiento con traslape, además, se proporcionan al usuario funcionalidades de validación para llevar a cabo comparaciones experimentales. La plataforma además permite la inclusión de algoritmos de Reconocimiento de Patrones, los cuales deben seguir ciertos requerimientos para establecer comunicación con la plataforma, sin embargo, estos requerimientos son mínimos, también permiten que los algoritmos sean totalmente independientes de la plataforma en sí.

Finalmente, se presenta un estudio comparativo entre tres algoritmos de agrupamiento con traslape, para la evaluación de dichos algoritmos se utilizaron dos medidas de evaluación específicas para este tipo de algoritmos.

Índice

1. Introducción

El agrupamiento es una de las técnicas fundamentales en el Aprendizaje Automático y la Minería de Datos. Esta técnica se encarga de organizar una colección de objetos en clases o agrupamientos, de tal forma que los objetos pertenecientes a un mismo agrupamiento sean lo suficientemente similares para poder inferir que son del mismo tipo y los objetos pertenecientes a agrupamientos distintos sean lo suficientemente diferentes para poder afirmar que son de tipos diferentes.

Dentro de los algoritmos de agrupamiento, una clase importante la constituyen aquellos que permiten formar agrupamientos con traslape; es decir, que permiten que los objetos puedan pertenecer a más de un agrupamiento. Los algoritmos de agrupamiento con traslape resultan importantes para aquellas aplicaciones en las que los objetos pueden pertenecer a más de un agrupamiento [Aslam, 2000, 2004], por ejemplo: recuperación de información, análisis de noticias, y bioinformática, entre otras. En la literatura existen diversos algoritmos de agrupamiento con traslape [Cleuziou, 2008-2009], éstos difieren en cuanto a sus bases matemáticas y estrategias de agrupamiento, así como en el tipo de colecciones que pueden procesar.

Cuando en un problema real se requiere utilizar un algoritmo de agrupamiento como parte de la solución, un paso importante lo constituye la selección del algoritmo a utilizar. En muchas ocasiones, las características propias del problema se encargan de reducir el número de algoritmos a analizar. En otras, es necesario realizar pruebas con muestras controladas de datos del problema a resolver y, posteriormente, seleccionar el algoritmo que mejor resultado mostró en estas evaluaciones.

En la literatura se encuentra reportadas, entre otras, las plataformas [Orange], [WEKA], [jHelpWork], [KNIME], [ELKI], [GNU R] las cuales permiten la utilización de algoritmos de reconocimiento de patrones. Para el caso no supervisado se encuentra la plataforma [CLUTO], desarrollada por Karypis LAB. Esta plataforma permite la evaluación de varios

algoritmos de agrupamiento jerárquicos, pero no de algoritmos de agrupamiento con traslape. Adicionalmente, muchos de los autores de los algoritmos de agrupamiento con traslape no tienen disponible su implementación y por lo tanto, en muchas ocasiones, estos algoritmos deben ser implementados por las personas que los necesitan, perdiéndose mucho tiempo en este proceso, sin garantía de que los resultados sean fiables.

A partir de lo explicado hasta el momento, es evidente la necesidad de tener una biblioteca de algoritmos de agrupamiento con traslape, así como una plataforma que permita utilizar los algoritmos incluidos en esta biblioteca sobre diferentes colecciones de prueba. Otro aspecto importante es poder evaluar los resultados obtenidos por los algoritmos, de acuerdo a las diferentes medidas de evaluación reportadas en la literatura. Adicionalmente, es deseable que esta biblioteca de algoritmos pueda ser ampliada fácilmente y que la plataforma de experimentación sea capaz de utilizar los nuevos algoritmos que se agreguen.

La plataforma antes mencionada constituye uno de los objetivos centrales de este trabajo de tesis. Cabe mencionar que el desarrollo de esta plataforma debe ser lo suficientemente extensible, ya que debe permitir la inclusión de algoritmos para distintos tipos de problemas de reconocimiento de patrones y minería de datos. Los algoritmos a implementar en el marco de esta tesis son: OKM [Cleuziou, 2008] y OKMED [Cleuziou, 2009], que son extensiones al caso del agrupamiento con traslape, de los algoritmos k-Means [MacQueen, 1967] y k-medoids (PAM) [Kaufman, 1987], respectivamente. Además, se implementará el algoritmo WOKM [Cleuziou, 2009], que es una variante de OKM que incluye el uso de pesos en los atributos de los objetos.

El algoritmo k-Means es uno de los algoritmos de agrupamiento más utilizados, fundamentalmente por su simplicidad y sus buenos resultados. Al ser estos algoritmos (OKM, WOKM y OKMED) extensiones de k-means al caso de agrupamiento con traslape, resulta interesante implementarlos para poder hacer un estudio comparativo mediante medidas de evaluación que tomen en cuenta que estos algoritmos permiten el agrupamiento con traslape.

El presente documento de tesis está organizado de la siguiente manera: En el capítulo 2 se introduce el marco teórico, necesario para entender los distintos conceptos utilizados en los capítulos posteriores, el capítulo 3 detalla los algoritmos OKM, WOKM y OKMED, así como las medidas GFM y FBCubed, lo anterior fue implementado dentro del marco de esta tesis para formar parte de la plataforma, en el capítulo 4 se presenta todo lo referente al desarrollo de la plataforma para algoritmos de agrupamiento con traslape, la cual también ha sido desarrollada como parte de este trabajo de tesis, en el capítulo 5 se muestra el funcionamiento de la plataforma así como un pequeño estudio comparativo de los algoritmos OKM, WOKM y OKMED, en el capítulo 6 se encuentran las conclusiones y el trabajo a fututo, posteriormente se encuentran las referencias y un apéndice con ejemplos para una mejor comprensión de algunas secciones.

2. Marco Teórico

En este capítulo se presenta el problema de la clasificación no supervisada que es uno de los problemas más comúnmente estudiados dentro del Reconocimiento de Patrones. También se describen algunas de las plataformas computacionales que se han propuesto en la literatura, junto con sus principales características y limitaciones; estas plataformas han ayudado en la evaluación y aplicación de las técnicas de Reconocimiento de Patrones y Minería de Datos más relevantes propuestas hasta el momento.

2.1 Reconocimiento de Patrones

El proceso de análisis, extracción de patrones y la construcción de modelos a partir de datos es esencial en diferentes aplicaciones del mundo moderno. A partir de este proceso se puede extraer de los datos conocimiento útil que permita reconocer patrones para la toma de decisiones en diferentes aplicaciones como vigilancia, comercialización, descubrimientos científicos, detección de fraudes, biometría, entre muchas otras [Jain, 2000].

Dentro del Reconocimiento de Patrones (RP) se abordan varios problemas, siendo el problema de la clasificación uno de los más estudiados. La clasificación consiste en la asignación de un objeto a una o varias de las diversas categorías o clases (se hace referencia a clase como una agrupación de objetos que tienen características comunes) existentes en un contexto determinado.

Los problemas de clasificación se pueden dividir en:

Clasificación supervisada: En este caso, se tiene por cada clase una muestra de objetos que fueron previamente etiquetados (conjunto de entrenamiento), y con base en este conocimiento previo, lo que se pretende es clasificar objetos nuevos.

Clasificación no supervisada: En este tipo de clasificación no se conocen las clases del conjunto de datos, por tanto, el problema consiste en encontrar dichas clases, para esto,

generalmente se buscan similitudes entre conjuntos de objetos que tienen entre sí características muy parecidas [Jain, 1999]. Hay que mencionar que esta tesis se ubica en este tipo de clasificación, en particular, cuando se permite traslape entre las clases; es decir, que un objeto pueda pertenecer a más de un agrupamiento.

En la literatura, el problema de la clasificación no supervisada se aborda siguiendo distintos paradigmas, los cuales se describen a continuación:

Paradigma del conjunto cociente: Es quizás el caso más común al que se enfrenta la clasificación no supervisada. En este paradigma los agrupamientos construidos tienen la característica de que son conjuntos duros disjuntos, es decir, en los agrupamientos no puede haber objetos en común; por lo que cada objeto debe pertenecer únicamente a un agrupamiento.

Paradigma del Traslape: En este paradigma los agrupamientos también son conjuntos duros como en el paradigma anterior pero los objetos pueden pertenecer a más de un agrupamiento.

Paradigma difuso: En este paradigma los agrupamientos son conjuntos difusos y los objetos pertenecen a todos los agrupamientos, pero en diferente grado.

El presente trabajo de tesis se enmarca en el paradigma del agrupamiento con traslape. Por esta razón, se implementaron algoritmos de este tipo y se evaluaron, dentro de una plataforma de experimentación, para estudiar qué tan bueno es en términos de medidas de evaluación para este tipo de agrupamiento; la plataforma antes mencionada constituye uno de los objetivos centrales de este trabajo de tesis. Los algoritmos implementados son: OKM [Cleuziou, 2008] y OKMED [Cleuziou, 2009], que son extensiones al caso del agrupamiento con traslape, de los algoritmos k-Means [MacQueen, 1967] y k-medoids (PAM) [Kaufman, 1987], respectivamente. Además, se implementó el algoritmo WOKM [Cleuziou, 2009], que es una variante de OKM que incluye el uso de pesos en los atributos de los objetos.

Como parte del marco teórico es necesario definir lo que es un cubrimiento y una partición, ya que estos conceptos serán utilizados para la descripción de los algoritmos de agrupamientos estudiados en este documento.

El concepto de cubrimiento es utilizado en el paradigma del traslape y el concepto de partición en el paradigma del conjunto cociente. La diferencia está en que en una partición la intersección de dos o más debe ser el vacío, mientras que en un cubrimiento la intersección de dos o más puede ser distinta del vacío. Dicho en términos de un problema de agrupamiento un objeto no puede pertenecer a más de un agrupamiento, en cambio si hablamos de cubrimientos un objeto puede pertenecer a más de un agrupamiento.

Formalmente:

Sea X el conjunto de objetos $\{x_1, x_2, \dots\ x_n\}$.

Sea $\pi = \{\pi_1,\ \dots, \pi_k\}$ un conjunto de agrupamientos sobre X, π forma un cubrimiento de X si

$$\forall x_i \in X: \exists \pi_j \in \pi | x_i \in \pi_j$$

Sea $\pi = \{\pi_1,\ \dots, \pi_k\}$ un conjunto de agrupamientos sobre X, π forma una partición de X si

$$\forall x_i \in X: \left(\exists \pi_j \in \pi \middle| x_i \in \pi_j\right) \wedge (\nexists \pi_l \in \pi | x_i \in \pi_l) \wedge (j \neq l)$$

Se puede notar que una partición es un caso particular de un cubrimiento.

Independientemente de la distinción entre los diferentes paradigmas de agrupamiento mencionados anteriormente, los métodos desarrollados para la clasificación no supervisada se pueden dividir atendiendo a diferentes criterios. Una de las categorizaciones más comunes es la siguiente:

Métodos Jerárquicos: los objetos no se dividen en agrupamientos de una sola vez, sino que se van haciendo divisiones sucesivas en "distintos niveles de agrupamiento". Fundamentalmente, los métodos jerárquicos suelen subdividirse en métodos aglomerativos

(ascendentes), que van sucesivamente fusionando agrupamientos en cada paso; y métodos divisivos (descendentes), que van desglosando en agrupamientos cada vez más pequeños el conjunto total de datos.

- ***Métodos jerárquicos aglomerativos***: En el primer paso cada objeto forma un agrupamiento y a partir de este punto, en cada etapa del algoritmo se unen los agrupamientos más semejantes o cercanos. Este proceso continua hasta que se llega a un único agrupamiento con todos los objetos. Un ejemplo de este tipo de métodos es UPGMA [Sokal, 1958].

- ***Métodos jerárquicos divisivos***: En este caso, el algoritmo comienza con un único agrupamiento que contiene a todos los objetos. Posteriormente, en cada etapa se divide un agrupamiento, hasta que cada objeto forma un agrupamiento unitario. Un ejemplo de este tipo de métodos es Bisecting k-means [Steinbach, 2000].

Métodos de reagrupamiento: A diferencia de los métodos jerárquicos, en los que cada nivel de jerarquía constituye una solución al problema, estos métodos construyen solo una solución, en la cual los agrupamientos responden a un criterio de agrupamiento previamente definido. Un ejemplo de método de reagrupamiento es k-Means [MacQueen, 1967]. Este algoritmo recibe como entrada el número de agrupamientos a formar, a partir de este número se genera aleatoriamente la misma cantidad de centroides, y los objetos se agrupan en el agrupamiento del centroide más cercano. Cuando se han agrupado todos los objetos se recalculan o ajustan los centroides teniendo en cuenta a los objetos clasificados en cada agrupamiento; este proceso se repite hasta que los agrupamientos obtenidos sean estables.

2.2 Plataformas para Minería de Datos

Dentro del Reconocimiento de Patrones y la Minería de Datos existen varias plataformas que permiten aplicar diversos algoritmos de estas áreas de conocimiento y, comparar los resultados de los mismos. A continuación se describen brevemente las características de las plataformas más utilizadas:

Orange[1] es una plataforma desarrollada en la facultad de informática de la Universidad de Ljubljana, que permite realizar minería de datos y análisis predictivo. Esta plataforma consta de una serie de componentes desarrollados en C++ que implementan algoritmos de minería de datos, así como operaciones de preprocesamiento y representación gráfica de datos. Los componentes de Orange pueden ser manipulados desde programas desarrollados en Python o a través de un entorno gráfico. Se distribuye bajo licencia GPL.

Weka[2] (Waikato Environment for Knowledge Analysis - Entorno para Análisis del Conocimiento de la Universidad de Waikato) es una plataforma de software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato. Weka es un software libre distribuido bajo licencia GNU-GPL. El paquete Weka contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades.

jHepWork[3] es un framework libre y de código abierto, que fue creado usando paquetes de código abierto y con una interfaz de usuario comprensible, como un intento de hacer un entorno de análisis de datos que fuera competitivo con los programas comerciales. Este framework contiene bibliotecas científicas numéricas implementadas en Java para funciones matemáticas, números aleatorios, así como algoritmos de minería de datos; además, permite

[1] http://orange.biolab.si/
[2] http://www.cs.waikato.ac.nz/ml/weka/
[3] http://jwork.org/jhepwork/

crear gráficos científicos interactivos en 2D y 3D. jHepWork se basa en un lenguaje de programación de alto nivel llamado Jython, pero también puede usarse código en java para llamar librerías numéricas y gráficas de jHepWork.

KNIME[4] es una plataforma de minería de datos que permite el desarrollo de modelos en un entorno visual. Está programado en java. Concebido como una herramienta gráfica y dispone de una serie de nodos (que encapsulan distintos tipos de algoritmos) y flechas (que representan flujos de datos) que se despliegan y combinan de manera gráfica e interactiva.

Los nodos implementan distintos tipos de acciones que pueden ejecutarse sobre un conjunto de datos:

- Manipulación de filas, columnas, etc., como muestreos, transformaciones, agrupaciones, etc.
- Visualización (histogramas, etc.).
- Creación de modelos estadísticos y de minería de datos, como árboles de decisión, máquinas de vectores de soporte, regresiones, etc.
- Validación de modelos, como curvas ROC, etc.
- Scoring o aplicación de dichos modelos sobre conjuntos nuevos de datos.
- Creación de informes a la medida; gracias a su integración con BIRT.

El carácter abierto de la herramienta hace posible su extensión mediante la creación de nuevos nodos que implementen algoritmos a la medida del usuario. Además, existe la posibilidad de llamar directa y transparentemente a Weka y/o de incorporar de manera sencilla, códigos desarrollados en R o en python/jython.

El proyecto Business Intelligence and Reporting Tools[5] (BIRT) es software de código abierto que proporciona capacidades de creación de informes y de inteligencia de negocio para clientes y aplicaciones web, especialmente aquellas aplicaciones basadas en Java y Java EE.

[4] http://www.knime.org/
[5] http://www.eclipse.org/birt/phoenix/intro/

Los objetivos del proyecto son cubrir un amplio rango de necesidades de creación de informes dentro de una aplicación típica. Inicialmente, el proyecto se ha enfocado en capacidades que permitan a los desarrolladores diseñar e integrar fácilmente informes dentro de aplicaciones.

BIRT, utiliza los datos para construir modelos predictivos de forma rápida. A partir de técnicas de Minería de Datos, BIRT permite descubrir relaciones, extraer la información implícita en los datos y anticiparse a situaciones no planificadas.

Sin embargo, a pesar de que algunas de las plataformas mencionadas anteriormente son software de código abierto, éstas tienen ciertas limitaciones. Entre sus principales limitaciones se encuentran que no reportan todos los resultados que el usuario pudiera necesitar ya que las métricas de evaluación vienen predefinidas en la plataforma y se puede dar el caso de que no sean suficientes para todos los experimentos prácticos que requiere el usuario. Un ejemplo de esto es que las medidas de evaluación de algoritmos de agrupamiento disjunto no son aplicables para los algoritmos de agrupamiento con traslape, que es el caso que se contempla en este trabajo. En algunos casos, no se tiene la posibilidad de agregar nuevos algoritmos, lo cual atenta contra la aceptación de dicha herramienta por parte de los usuarios. Además de lo antes mencionado, muchas de estas plataformas requieren que los algoritmos se adicionen en lenguaje Java, lo cual puede atentar contra la eficiencia del algoritmo, sobre todo si se requiere trabajar con grandes volúmenes de información.

3. Algoritmos de Agrupamiento con Traslape

En el presente capítulo se describen los algoritmos de agrupamiento con traslape implementados en la plataforma propuesta, así como las métricas para la evaluación de este tipo de agrupamientos. Los algoritmos de agrupamiento incluidos en la plataforma son OKM [Cleuziou, 2008], WOKM [Cleuziou, 2009] y OKMED [Cleuziou, 2009], los cuales, como ya se ha comentado en el capítulo 1, se basan en los algoritmos k-Means [MacQueen, 1967], WKM [Chan, 2004] y k-Medoides (PAM) [Kaufman, 1987] respectivamente. En OKM, WOKM y OKMED, si el conjunto de datos a agrupar no tiene traslape, entonces los algoritmos se comportan de manera similar a los algoritmos originales, de los cuales fueron generalizados.

El problema de agrupamiento intenta responder cómo es que ciertos objetos (casos) pertenecen naturalmente a un cierto número de clases o agrupamientos, de tal manera que estos objetos compartan ciertas características. Éste es el objetivo principal del problema de agrupamiento.

Como se mencionó anteriormente, el presente trabajo realiza un estudio de algunos algoritmos de agrupamiento con traslape. En este tipo de agrupamiento, los objetos pueden pertenecer a más de una clase. Más específicamente, en este trabajo se analizan algoritmos que son variantes del algoritmo OKM; el cual es una generalización del algoritmo k-Means.

El algoritmo k-Means es el algoritmo de agrupamiento más conocido y utilizado en la literatura. Este algoritmo sigue un procedimiento simple de clasificación de un conjunto de objetos en un determinado número k de agrupamientos; donde, k es definido a priori por el usuario.

El nombre de k-Means se debe a que este algoritmo busca clasificar a los objetos en k agrupamientos, representando a cada uno de los agrupamientos por su respectivo centroide;

el centroide de un agrupamiento es calculado como la media de los objetos pertenecientes a su agrupamiento.

En k-Means, el problema de agrupar un conjunto de objetos $X = \{x_1, x_2, \dots x_n\}$ en k agrupamientos puede formularse como un problema de optimización. La función objetivo utilizada en k-Means es la inercia intra-clase (también llamada error cuadrático), la cual está definida por:

$$Q(\pi) = \sum_{j=1}^{k} \sum_{x_i \in \pi_j} d^2(x_i, z_j) \qquad \textbf{(3.0.1)}$$

donde:

$Z = \{z_1, \dots z_k\}$ son los centroides de los agrupamientos $\pi_1, \dots, \pi_k$ respectivamente.

d es la distancia Euclidiana que hay entre dos objetos.

La partición final depende de los centroides considerados inicialmente.

3.1 OKM

Este algoritmo, al igual que k-Means, calcula centroides de cada agrupamiento pero para agrupar, toma en cuenta los centros de gravedad entre centroides para contemplar el traslape. Un centro de gravedad (CG) es un objeto generado a partir de los valores de un conjunto de objetos, éste es calculado utilizando la siguiente expresión:

$$CG(T) = \frac{\sum_{i=1}^{|T|} x_i}{|T|} \qquad \textbf{(3.1.1)}$$

donde:

T es un conjunto de objetos $\{x_1, x_2, \dots, x_m\}$ a partir de los cuales será calculado el centro de gravedad.

Función Objetivo para OKM

Al igual que en el caso del algoritmo k-Means, el problema de agrupar un conjunto de objetos $X = \{x_1, x_2, \dots x_n\}$ en k agrupamientos con traslape puede formularse como un problema de optimización, sin embargo debe utilizarse una función objetivo que contemple el traslape.

En OKM, el autor propone una función objetivo Q' que es una extensión del error cuadrático Q.

Para determinar la pertenencia de los objetos a los agrupamientos se tendrá en cuenta para cada objeto x_i, el conjunto de centroides de los agrupamientos a los que pertenece $\{z_j | m_{j,i} = 1\}$ y se calculará el centro de gravedad de este conjunto. En adelante, a este centro de gravedad se le llamará "imagen" de x_i en π y será denotado por $\phi(x_i)$.

$$\phi(x_i) = CG(A_i) \qquad \textbf{(3.1.2)}$$

donde:

A_i es el conjunto de centroides a los cuales x_i es asignado.

Por lo cual la función objetivo para OKM está definida por

$$Q'(M, Z) = \sum_{i=1}^{n} d^2(x_i, \phi(x_i)) \qquad \textbf{(3.1.3)}$$

donde:

$M\ (k \times n)$ es una matriz Booleana de pertenencias, $m_{i,j} = 1\ si\ x_i \in \pi_j\ (0\ en\ otro\ caso)$, esta matriz es utilizada para saber a qué agrupamientos es asignado x_i.

$Z = \{z_1,\ \dots\ z_k\}$ son los centroides de los agrupamientos $\pi_1,\ \dots, \pi_k$ respectivamente.

$\phi(x_i)$ con $A_i = \{z_j | x_i \in \pi_j\}$

Nótese que $Q'(.,.)$ generaliza a $Q(.)$, dado que cuando no hay traslape $\phi(x_i)$ es el centroide más cercano z_j, en lugar del centro de gravedad entre los centroides a los que pudiera ser asignado x_i.

Algoritmo

El algoritmo OKM (figura 3.1) es similar a k-Means, las diferencias se encuentran en la asignación de los objetos a uno o más agrupamientos y en el cálculo de centroides para cada agrupamiento.

OKM $(\boldsymbol{X}, \boldsymbol{k}, \boldsymbol{t_{max}}, \boldsymbol{\varepsilon})$***:***

Entrada:

- *X conjunto de objetos*
- *k número de agrupamientos esperados*
- t_{max} *máximo número de iteraciones (opcional)*
- *ε parámetro de convergencia (opcional)*

Salida:

- *π conjunto de agrupamientos con traslape que cubren X (cubrimiento)*

1. t = 0

2. Elegir aleatoriamente k centroides $Z^t = \{z_1^t, z_2^t, \dots, z_k^t\}$ *de X,*

3. Para cada $x_i \in X$*: Asignar*(x_i, Z^t)*,* $es\ decir, construir\ M_i^t$

4. Construir un primer cubrimiento $\pi^t = (M^t, Z^t)$.

Hacer

5. *t = t+1*
6. *Actualizar*$(Z^{t-1}, M^{t-1})(construir\ Z^t)$*,*
7. *Para cada* $x_i \in X$*: Asignar*(x_i, Z^t)
8. *Construir el cubrimiento* $\pi^t = (M^t, Z^t)$.

Mientras $([M^t \neq M^{t-1}]\ o\ [Q'(M^{t-1}, Z^{t-1}) - Q'(M^t, Z^t) > \epsilon]\ o\ [t < t_{max}])$

Regresar el cubrimiento final π^t

Fig. 3.1. Pseudo código de OKM

Para la construcción de la matriz de pertenencias *M*, se toma en cuenta la pertenencia a cada agrupamiento de cada objeto x_i, es decir, aquellos agrupamientos de los cuales x_i forma parte, $m_{i,j} = 1$ si x_i está en el agrupamiento j, con $j \leq k$.

Asignar$(\boldsymbol{x_i, Z})$

Entrada:

- *Z conjunto de centroides de los agrupamientos*
- x_i *objeto a asignar*

1. Sea $z*$ *el centro más cercano de* x_i *en* $Z\left(\forall z_j \in Z, d(x_i, z*) \leq d\left(x_i, z_j\right)\right)$

2. $A = \{z*\}$*(con A la lista de asignaciones para* x_i*),*

3. $Z = Z \backslash \{z*\}$.

Hacer

4. *Caclular* $\phi(x_i)^A$ *denotando el centro de gravedad de A*
5. *Sea* $z*$ *el centro más cercano de* x_i *en* Z,
6. *Si* $d\left(x_i, \phi(x_i)^{A \cup \{z*\}}\right) \leq d(x_i, \phi(x_i)^A)$ *entonces*

$A \leftarrow A \cup \{z*\}, Z = Z \backslash \{z*\}$

Mientras una nueva asignación se lleve a cabo

Decisión Final:

7. Sea A' el conjunto de asignaciones anterior para x_i, calcular $\phi(x_i)^{A'}$

8. Si $d(x_i, \phi(x_i)^A) \leq d(x_i, \phi(x_i)^{A'})$ *entonces*

asignar x_i a los centroides de A

En otro caso

mantener las asignaciones anteriores A'.

Fig. 3.2. Proceso de Asignación de OKM

Sea $Z = \{z_1, z_2, \ldots, z_k\}$ un conjunto de centroides de agrupamientos y x_i un objeto a asignar a uno o más agrupamientos. El proceso de asignación de x_i, en OKM, está descrito en la figura 3.2. Consiste en recorrer la lista de centroides del más cercano al más lejano, y

asignar x_i al agrupamiento respectivo si la imagen $\phi(x_i)$ (permite que $(d(x_i, \phi(x_i))$ decrezca). La nueva asignación se conserva solo si es mejor que la anterior.

En el paso de actualización el nuevo centroide z_j para el agrupamiento π_j está definido en OKM por

$$z_{j,v} = \frac{1}{\sum_{x_i \in R_j} \frac{1}{\delta_i^2}} \times \sum_{x_i \in R_j} \frac{1}{\delta_i^2} . {\tau_i}_v^j \qquad \textbf{(3.1.4)}$$

donde:

$z_{j,v}$ Denota el $v - ésimo$ atributo del objeto z_j.

δ_i es el número de agrupamientos a los que pertenece x_i ($\delta_i = \sum_{j=1}^{k} m_{j,i}$).

${\tau_i}_v^j$ denota el $v - ésimo$ atributo del centro z_j de acuerdo a x_i. El valor de este objeto se calcula de la siguiente manera:

$${\tau_i}_v^j = \delta_i . x_{i,v} - \sum_{z_j \in A_i / \{z_i\}} z_{j,v} \qquad \textbf{(3.1.5)}$$

A_i es el conjunto de agrupamientos a los que x_i es asignado.

La definición anterior permite que cada uno de los objetos pertenecientes a cada agrupamiento sea ponderado, de manera que aquellos objetos que pertenecen a más agrupamientos, tienen menos impacto en la nueva posición del centroide.

3.2 WOKM

En esta sección se explica la extensión del algoritmo Weighting-K-Means (WKM) propuesto por [Chan, 2004] para el paradigma del traslape. WKM generaliza la función objetivo utilizada en k-Means asociando un peso para cada atributo, el cual es diferente para cada agrupamiento. Sea X el conjunto de objetos $\{x_1, x_2, \dots x_n\}$, $Z = \{z_1, \dots z_k\}$ son los

centroides de los agrupamientos $\pi_1, \ldots, \pi_k$ respectivamente, la función objetivo utilizada en WKM se describe a continuación:

$$Q(\pi) = \sum_{j=1}^{k} \sum_{x_i \in \pi_j} \sum_{v=1}^{p} \lambda_{j,v}^{\beta} |x_{i,v} - z_{j,v}|^2 \qquad \textbf{(3.2.1)}$$

$$\forall j, \sum_{v=1}^{p} \lambda_{j,v} = 1$$

donde:

$x_{i,v}$ denota el valor del atributo *v* en el objeto x_i.

$z_{j,v}$ es el valor del atributo *v* en el centroide z_j.

$\lambda_{j,v}$ denota el peso asociado al atributo v en el agrupamiento j.

β es un parámetro (> 1) que regula la influencia del peso en el algoritmo.

Función Objetivo para WOKM

Es necesario definir la imagen de un objeto usando pesos. La imagen de x_i se define como un promedio de los centroides pesados asociados de los agrupamientos para x_i:

$$\phi_{WOKM}(x_i) = (\phi_1(x_i), \ldots, \phi_p(x_i))$$

$$\phi_v(x_i) = \frac{\sum_{z_j \in A_i} \lambda_{j,v}^{\beta} z_{j,v}}{\sum_{z_j \in A_i} \lambda_{j,v}^{\beta}} \qquad \textbf{(3.2.2)}$$

La imagen del objeto x_i caracteriza a un objeto que es representativo de la intersección[6] de los agrupamientos de A_i.

Teniendo en cuenta que al centroide del agrupamiento π_j se le asocia un vector de pesos λ_j, se tiene entonces que proponer un peso para los traslapes, en otras palabras, se debe

[6] En el algoritmo OKM este objeto es el centro de gravedad entre centroides, sin embargo para WOKM los pesos influyen en la construcción de este objeto, por lo cual no necesariamente es un centro de gravedad.

proponer un vector de pesos γ_i para las imágenes $\phi(x_i)$. Este vector está definido a continuación:

$$\gamma_{i,v} = \frac{\sum_{z_j \in A_i} \lambda_{j,v}}{|A_i|} \quad \textbf{(3.2.3)}$$

A partir de esta definición, se obtiene la siguiente función objetivo para el algoritmo WOKM:

$$Q'(M,Z) = \sum_{x_i \in X} \sum_{v=1}^{p} \gamma_{i,v}^{\beta} |x_{i,v} - \phi_v(x_i)|^2 \quad \textbf{(3.2.4)}$$

donde:

M $(k \times n)$ es una matriz Booleana de pertenencias.

$Z = \{z_1, \ \dots \ z_k\}$ son los centroides de los agrupamientos $\pi_1, \ \dots, \pi_k$ respectivamente

Se debe tomar en cuenta la restricción $\forall j, \sum_{v=1}^{p} \lambda_{j,v} = 1$ en los pesos de los agrupamientos, estos pesos están encapsulados en la definición de los pesos de imagen $\{\gamma_i\}$. Este algoritmo generaliza los algoritmos previos de la siguiente manera:

- En caso de asignaciones simples, si $x_i \in \pi_j$ entonces $\phi_v(x_i) = z_{j,v}$ y $\gamma_{i,v} = \lambda_{j,v}$; la función objetivo es entonces equivalente a la utilizada en WKM.
- En caso de pesos uniformes $\left(\forall j, v: \lambda_{j,v} = \frac{1}{p}\right)$, $\gamma_{i,v} = 1/p \ \ y \ \phi_{WOKM}(x_i) = \phi_{OKM}(x_i)$; la función objetivo es entonces equivalente a la utilizada en OKM.

Algoritmo

La principal diferencia con respecto al algoritmo OKM es que se deben actualizar los pesos que influyen en este algoritmo, el algoritmo se describe en la figura 3.3.

WOKM*$(\boldsymbol{X}, \boldsymbol{k}, \boldsymbol{t_{max}}, \boldsymbol{\varepsilon})$*:

Entrada:

- *X conjunto de objetos*
- *k número de agrupamientos esperados*
- t_{max} *máximo número de iteraciones (opcional)*
- *ε parámetro de convergencia (opcional)*

Salida:

- *π conjunto de agrupamientos con traslape que cubren X (cubrimiento)*

1. t = 0

2. Elegir aleatoriamente k centroides $Z^t = \{z_1^t, z_2^t, \dots, z_k^t\}$ *de X*

3. Inicializar los pesos $\{\lambda_{j,v}^{(0)}\}$ *de manera uniforme* $(\lambda_{j,v}^{(0)} = 1/p)$

4. Para cada $x_i \in X$*: Asignar*(x_i, Z^t), $es\ decir, construir\ M_i^t$

5. Construir un primer cubrimiento $\pi^t = (M^t, Z^t)$.

Hacer

6. *t = t + 1*
7. *Actualizar*$(Z^{t-1}, M^{t-1})(construir\ Z^t)$,
8. *CalcularPesos(*π^{t-1}*)*
9. *Para cada* $x_i \in X$*: Asignar*(x_i, Z^t)
10. *Construir el cubrimiento* $\pi^t = (M^t, Z^t)$.

Mientras $([M^t \neq M^{t-1}]\ o\ [Q'(M^{t-1}, Z^{t-1}) - Q'(M^t, Z^t) > \epsilon]\ o\ [t < t_{max}])$

Regresar el cubrimiento final π^t

Fig. 3.3. Algoritmo WOKM

El paso de asignación (4 y 9) es similar al paso correspondiente del algoritmo OKM (figura 3.2), pero en este caso los pesos influyen en la asignación y creación de la imagen, un objeto es asignado a sus agrupamientos más cercanos mientras $\sum_{v=1}^{p} \gamma_{i,v}^{\beta} |x_{i,v} - \phi_v(x_i)|^2$ decrezca.

El paso en el que se actualizan los centroides de los agrupamientos es realizado para cada agrupamiento, considerando los otros centroides fijos; el nuevo centroide óptimo z_j^* para el agrupamiento π_j se obtiene a partir del conjunto $\{(\tau_i{}^j, w_i) | x_i \in \pi_j\}$; $\tau_i{}^j$ denota el centroide

del agrupamiento π_j que al igual que en OKM permite que aquellos objetos que pertenecen a más agrupamientos, tengan menos impacto en la nueva posición del centroide y w_i denota el vector de pesos asociado definido a continuación:

$$w_{i,v} = \frac{\gamma_{i,v}^{\beta}}{(\sum_{z_l \in A_i} \lambda_{l,v}^{\beta})^2}$$

El paso 7 actualiza los vectores de pesos locales $\{\lambda_j\}_{j=1}^{k}$; el problema de optimización con la restricción ($\sum_{v=1}^{p} \lambda_{j,v} = 1$) no puede ser resuelto directamente porque los vectores $\lambda_{j,v}$ son mutuamente dependientes. Debido a esto, se propone una heurística para el cálculo de pesos basada en el teorema de Bezdeck [Bezdek, 1981]; la heurística consiste en, para cada clase:

1. Calcular un nuevo peso $\lambda_{j,v}$ para el agrupamiento π_j como una estimación de la varianza de cada atributo de los objetos que pertenecen solo a π_j:

$$\lambda_{j,v} = \frac{(\sum_{\{x_i \in \pi_j |\ |A_i|=1\}} (x_{i,v} - z_{j,v})^2)^{1/(\beta-1)}}{\sum_{u=1}^{p} (\sum_{\{x_i \in \pi_j |\ |A_i|=1\}} (x_{i,v} - z_{j,v})^2)^{1/(\beta-1)}} \quad \textbf{(3.2.5)}$$

2. Guardar el peso calculado solo si mejora la función objetivo asociada al algoritmo WOKM.

3.3 OKMED

Los métodos basados en medoides consisten en la asignación de objetos alrededor de objetos representativos de los agrupamientos, los representativos denotados como medoides son elegidos del conjunto de objetos. Es aquí donde difieren de los métodos basados en centroides, en los cuales los representativos de los agrupamientos no necesariamente pertenecen al conjunto de objetos X.

El algoritmo PAM [Kaufman, 1987] es considerado como una referencia en este campo de estudio. PAM construye una partición de objetos con la iteración de dos pasos: asignación de cada objeto a su medoide más cercano y la actualización del medoide para cada agrupamiento.

Durante el segundo paso, la actualización de medoides consiste en buscar dentro del conjunto de objetos pertenecientes al agrupamiento, aquel que minimice la suma de las distancias con todos los objetos dentro de su agrupamiento.

Las dos principales ventajas de estos métodos son:

1. La robustez ante los objetos outliers.
2. La posibilidad de usar cualquier métrica dado que solo requiere de una matriz de proximidad sobre el conjunto de objetos.

Función Objetivo para OKMED

En [Cleuziou, 2009] se propone el algoritmo OKMED, para generalizar la función objetivo del algoritmo original OKM, a cualquier función de distancia o disimilitud entre objetos. Sea X el conjunto de objetos $\{x_1, x_2, \dots x_n\}$ y d una función de disimilitud de $X \times X \rightarrow \mathbb{R}^+$, la función objetivo para OKMED está dada por:

$$Q(M, Z) = \sum_{x_i \in X} d^2(x_i, \phi(x_i)) \qquad \textbf{(3.3.1)}$$

De donde:

$\phi(x_i)$ es la imagen de x_i en π.

$M\ (k \times n)$ es una matriz Booleana de pertenencias.

$Z = \{z_1,\ \dots\ z_k\}$ son los medoides de los agrupamientos $\pi_1,\ \dots, \pi_k$ respectivamente

Una vez más, la función objetivo minimiza la inercia de los objetos con respecto a su imagen. La noción de imagen ha sido redefinida usando medoides en lugar de centroides: la imagen $\phi_{OKMED}(x_i)$ del objeto x_i en el agrupamiento $\{\pi_j\}_{j=1}^{k}$ es entonces definida como el objeto de X que minimiza la suma de las disimilitudes con todos los medoides de los agrupamientos de los cuales x_i forma parte:

$$\phi_{OKMED}(x_i) = \arg min_{x_j \in X} \sum_{z_l \in A_i} d^2(x_j, z_l) \quad \textbf{(3.3.2)}$$

Con esta nueva definición, el cálculo de una imagen requiere de probar todos los objetos en X.

En el caso de asignaciones a un solo agrupamiento (sin traslape), para cada objeto x_i perteneciente a un solo agrupamiento π_j, la imagen $\phi(x_i)$ es exactamente el medoide z_j. Entonces, k-Medoids puede considerarse como un caso especial de OKMED, con la función objetivo previamente definida.

Algoritmo

Este algoritmo minimiza la función objetivo con la iteración de dos pasos: asignación de objetos y actualización de medoides.

La asignación de un objeto a uno o varios agrupamientos se lleva a cabo por la función Asignar, la cual usa la heurística propuesta por [Cleuziou, 2008] (sección 3.1). Su adaptación para OKMED consiste en que, para cada objeto x_i, se consideran los medoides en un orden específico (del más cercano al más lejano a x_i de acuerdo a D) y se asigna a x_i al agrupamiento asociado mientras la inercia $d(x_i, \phi(x_i))$ decrezca. La nueva asignación $A_i^{(t+1)}$ es guardada solo si mejora la asignación previa $A_i^{(t)}$ de acuerdo a la función objetivo. En la figura 3.4 se describe el algoritmo OKMED.

La actualización consiste en la búsqueda de nuevos medoides para cada agrupamiento, de tal forma que mejoren la función objetivo. La heurística que se propone para esta búsqueda está formalizada por la función Medoide (figura 3.5); la cual busca un medoide relevante en lugar del mejor (u óptimo) medoide, en términos de la función objetivo. Esto por lo costoso del cálculo del óptimo, el cual, en el caso del agrupamiento con traslape, es aún más costoso.

OKMED*$(\boldsymbol{D}, \boldsymbol{k}, \boldsymbol{t_{max}}, \boldsymbol{\varepsilon})$*:

Entrada:

- *D matriz de disimilitud de* $(n \times n)$ *sobre el conjunto X*
- *k número de agrupamientos esperados*
- t_{max} *máximo número de iteraciones (opcional)*
- ε *parámetro de convergencia (opcional)*

Salida:

- π *conjunto de agrupamientos con traslape que cubren X (cubrimiento)*

1. $t = 0$

2. Elegir aleatoriamente k medoides $Z^t = \{z_1^t, z_2^t, \dots, z_k^t\}$ *de X*

3. Para cada $x_i \in X$*: Asignar*(x_i, Z^t), $es\ decir, construir\ M_i^t$

4. Construir un primer cubrimiento $\pi^t = (M^t, Z^t)$.

Hacer

5. $t = t + 1$
6. *Para cada agrupamiento* π_j^{t-1}

$$z_j^t = Medoide\left(\pi_j^{t-1}, k\right)$$

7. *Para cada* $x_i \in X$*: Asignar*(x_i, Z^t)
8. *Construir el cubrimiento* $\pi^t = (M^t, Z^t)$.

Mientras $([M^t \neq M^{t-1}]$ *o* $[Q'(M^{t-1}, Z^{t-1}) - Q'(M^t, Z^t) > \epsilon]$ *o* $[t < t_{max}])$

Regresar el cubrimiento final π^t

Fig. 3.4. Algoritmo OKMED

***Medoide*$(\boldsymbol{\pi_j, k})$**

Entrada:

- π_j *agrupamiento sobre el conjunto X*
- *k número de agrupamientos esperados*

Salida:

- z_j *medoide para el agrupamiento* π_j

1. Calcular la inercia de los objetos para π_j*:*

$$inercia = \sum_{x_i \in \pi_c} d^2(x_i, \phi(x_i))$$

2. Para b desde 1 hasta k hacer:

Para cada objeto $x_l \in \pi_j$ *tal que* $|A_l| = b$ *hacer:*

Calcular las imágenes $\phi(x_i)'$ *con* $z_j = x_l$ *para cada* $x_i \in \pi_j$

Calcular la nueva inercia para π_j

$$inercia' = \sum_{x_i \in \pi_c} d^2(x_i, \phi(x_i)')$$

Si $inercia' < inercia$

devolver x_l *(nuevo medoide para* π_j *)*

Fig. 3.5. Actualización de los medoides para los agrupamientos

3.4 Medidas de Evaluación para Algoritmos de Agrupamientos con Traslape

Para determinar qué tan bueno es un algoritmo de agrupamiento se debe hacer uso de alguna métrica que mida su calidad. En la literatura han sido reportadas tres tipos de medidas de calidad: externas, internas y relativas [Pfitzner, 2009].

Las medidas externas utilizan un conjunto de datos, previamente etiquetado. Mientras mayor sea la similitud entre las clases originales del conjunto de datos y los agrupamientos generados por el algoritmo de agrupamiento, mejor será el algoritmo con el que se construyeron dichos agrupamientos. Algunos ejemplos de este tipo de medidas son Purity [Zhao, 2001], Jaccard coefficient [Halkidi, 2001] y F1-measure [Larsen, 1999].

Las medidas relativas evalúan un algoritmo de agrupamiento comparando los agrupamientos obtenidos, utilizando distintos valores en los parámetros de entrada. De esta comparación se determina qué tan bueno es el algoritmo para generar buenos agrupamientos significativos con distintos parámetros iniciales. Algunos ejemplos de este tipo de medidas son Figure of Merit (FOM) [Yeung, 2000] y Stability index [Roth, 2002].

Las medidas internas evalúan un algoritmo de agrupamiento utilizando la información contenida en los agrupamientos a evaluar, como por ejemplo: la compacidad y separación de los agrupamientos, la conectividad de los mismos, etc. Algunos ejemplos de este tipo de medidas son los índices David-Bouldin y Silhouette [Halkidi, 2001], la medida Λ [Stein, 2003].

De los tres tipos de medidas mencionadas, las más utilizadas en la literatura son las medidas externas [Amigó, 2009].

Existen trabajos en los que se han utilizado medidas basadas en conteo de pares de objetos, como Fmeasure y Jaccard coefficient, para evaluar agrupamientos con traslape [Banerjee, 2005] [Pérez, 2007, 2009] [Gago, 2007]. Otros trabajos, en los cuales se aborda el problema del agrupamiento con traslape, han utilizado medidas basadas en cotejo de

conjuntos como F1-measure para evaluar sus resultados [Gil, 2003, 2010]. Sin embargo, estas medidas no contemplan el caso de que, en un agrupamiento con traslape, los objetos que comparten *n* clases en el conjunto de datos etiquetado deberían compartir *n* agrupamientos en el conjunto resultante del algoritmo de agrupamiento con traslape.

En el marco de esta tesis se optó por utilizar las medidas externas Generalized Fowlkes-Mallows Index [Ramirez, 2012] y FBCubed [Amigó, 2009] que explícitamente contemplan el caso en que los agrupamientos están traslapados, la primera es una generalización de la medida Fowlkes-Mallows Index [Fowlkes, 1983], a continuación se describe cómo está estructurada ésta para llegar a la generalización.

3.4.1 FM

El índice de Fowlkes and Mallows (FM) fue inicialmente introducido como una medida para la comparación de algoritmos de agrupamiento. Sin embargo teniendo un conjunto previamente etiquetado, se puede hacer la evaluación de un algoritmo de agrupamiento comparando las etiquetas y los resultados obtenidos al aplicar un algoritmo de agrupamiento al conjunto de datos.

El objetivo de esta medida es cuantificar cuán similar es el conjunto etiquetado con el conjunto de agrupamientos obtenido por el algoritmo de agrupamiento, para esto, se toman pares de objetos y se calcula la probabilidad de que esos objetos pertenezcan al mismo agrupamiento y a la misma clase, se considera también la probabilidad de que esos objetos pertenezcan a la misma clase, pero a distintos agrupamientos, por último se calcula la probabilidad de que esos objetos pertenezcan a distintas clases, pero al mismo agrupamiento.

Para poder explicar con más detalle esta medida de evaluación se deben tener en cuenta ciertas definiciones:

- Sea *X* un conjunto de objetos $\{x_1, \ \dots \ x_n\}$.
- Sea π un conjunto de particiones $\{ \pi_1, \dots, \pi_k\}$.
- Sea *C* un conjunto de clases $\{c_1, \dots, c_s\}$.

		Clases				
		c_1	c_2	...	c_s	Σ
Agrupamientos	π_1	t_{11}	t_{12}	...	t_{1s}	t_{1*}
	π_2	t_{21}	t_{22}	...	t_{2s}	t_{2*}
	...	...	...	...	...	...
	π_k	t_{k1}	t_{k2}	...	t_{ks}	t_{k*}
	Σ	t_{*1}	t_{*2}	...	t_{*s}	t_{**}

Fig. 3.6. Matriz de contingencia

En el caso de esta medida de evaluación es necesaria la construcción de una matriz de tamaño $|\pi| \times |C|$ llamada matriz de contingencia (figura 3.6). El contenido de la posición t_{ij} representa el número de objetos pertenecientes al agrupamiento π_i y a la clase c_j. En el caso que no se tengan agrupamientos traslapados y el conjunto esté etiquetado con una clase para cada objeto, se mantienen las siguientes propiedades:

- $\bigcup_i^k \pi_i = X$.
- $\pi_i \cap \pi_j = \emptyset \; \forall i, j = 1, \ldots, k \; con \; i \neq j$: no hay traslape en los agrupamientos.
- $c_i \cap c_j = \emptyset \; \forall i, j = 1, \ldots, s \; con \; i \neq j$: no hay traslape en las clases.

Haciendo uso de la notación de la matriz de contingencia el índice FM está definido por la siguiente expresión:

$$FM = \frac{\sum_{i,j}\binom{t_{ij}}{2}}{\sqrt{\sum_i \binom{t_{i*}}{2} \sum_j \binom{t_{*j}}{2}}} \qquad \textbf{(3.4.1)}$$

La fórmula 3.4.1 también puede expresarse en términos de una distribución de probabilidad hipergeométrica, definida como la probabilidad de obtener exactamente *y* casos en una muestra de tamaño *m*, obtenidos sin reemplazo de una población de *n* objetos donde *z* posee el valor de interés:

$$h(n, z, m, y) = \frac{\binom{z}{y}\binom{n-z}{m-y}}{\binom{n}{m}} \qquad \textbf{(3.4.2)}$$

Utilizar una función de distribución hipergeométrica ayuda a comprender las propiedades de la medida FM, además de que el uso de esta distribución resulta conveniente cuando se desean evaluar conjuntos de datos grandes, debido a que cuando $n > 50$ y $\frac{m}{n} \leq 0.10$ ésta tiende a aproximarse a una distribución binomial [Brunk, 1968] la cual tiene un menor costo computacional.

Utilizando la distribución de probabilidad hipergeométrica en 3.4.1 la expresión equivalente está dada por:

$$FM = \frac{\sum_{i,j} h(t_{**}, t_{ij}, 2, 2)}{\sqrt{\sum_i h(t_{**}, t_{i*}, 2, 2) \sum_j h(t_{**}, t_{*j}, 2, 2)}} \quad \textbf{(3.4.3)}$$

Esta nueva fórmula ayudará a generalizar esta medida de evaluación para el caso de agrupamiento con traslape.

Mientras más alto es el valor obtenido por esta medida, quiere decir que mayor es la similitud entre los agrupamientos obtenidos y el conjunto etiquetado, y por lo tanto el algoritmo evaluado es mejor.

3.4.2 GFM

Antes de describir esta medida se mostrará con un ejemplo por qué la medida FM no funciona para el caso de los agrupamientos con traslape.

Supóngase el conjunto de objetos $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_{9,} x_{10}\}$.
Las clases c_1, c_2 donde $\{x_1, x_2, x_3, x_6\} \subset c_1$ y $\{x_4, x_5, x_7, x_8, x_{9,} x_{10}\} \subset c_2$
Los agrupamientos π_1, π_2 donde $\{x_1, x_2, x_3, x_4, x_5\} = \pi_1$ y $\{x_6, x_7, x_8, x_9, x_{10}\} = \pi_2$.
La matriz de contingencia construida para este caso es:

		Clases		
		c_1	c_2	Σ
Agrupamientos	π_1	3	2	$t_{1*} = 5$
	π_2	1	4	$t_{2*} = 5$
	Σ	$t_{*1} = 4$	$t_{*2} = 6$	$t_{**} = 10$

Ahora, a partir de esta matriz se calcula la probabilidad de obtener dos objetos que pertenezcan a la misma clase con la formula $\sum_j h(t_{**}, t_{*j}, 2, 2) = \binom{4}{2}/\binom{10}{2} + \binom{6}{2}/\binom{10}{2} = 21/45$. Hasta este punto los cálculos son correctos puesto que se trata de un problema en el que no hay traslape. Si a este caso le agregamos una clase c_3 donde $\{x_1, x_4, x_8, x_{9,} x_{10}\} \in c_3$, la matriz correspondiente sería:

		Clases			
		c_1	c_2	c_3	Σ
Agrupamientos	π_1	3	2	2	$t_{1*} = 7$
	π_2	1	4	3	$t_{2*} = 8$
	Σ	$t_{*1} = 4$	$t_{*2} = 6$	$t_{*3} = 5$	$t_{**} = 15$

Intuitivamente se esperaría que al haber objetos repetidos la probabilidad de obtener dos objetos que pertenezcan a la misma clase fuera mayor. Sin embargo, $\sum_j h(t_{**}, t_{*j}, 2, 2) = \binom{4}{2}/\binom{15}{2} + \binom{6}{2}/\binom{15}{2} + \binom{5}{2}/\binom{15}{2} = 31/105$ es mucho menor que $21/45$. Esto se debe al hecho de que el muestreo sin reemplazo ya no se mantiene. Entonces, no hay $\binom{15}{2} = 105$ formas de seleccionar dos objetos, ya que eso implicaría que se puede extraer el mismo objeto más de una vez.

Lo anterior muestra que esta medida no es apta para el caso en el que el agrupamiento es traslapado, por lo cual se deben hacer algunos ajustes en las fórmulas para corregir este problema.

Para calcular la probabilidad de que dos objetos pertenezcan a la misma clase, la expresión ahora será:

$$\sum_{j=1}^{|C|} h\left(|X|, |c_j|, 2, 2\right) - J(C) \tag{3.4.4}$$

Donde $J(C)$ es la probabilidad de dos objetos seleccionados pertenezcan a dos clases simultáneamente y está dada por:

$$J(C) = \sum_{j=1}^{|C|} \sum_{v=j+1}^{|C|} h\left(|X|, c_j \cap c_v, 2, 2\right) \tag{3.4.5}$$

Para calcular la probabilidad de que dos objetos pertenezcan al mismo agrupamiento sucede algo parecido:

$$\sum_{i=1}^{|\pi|} h(|X|, |\pi_i|, 2, 2) - J(\pi) \tag{3.4.6}$$

Donde $J(\pi)$ es la probabilidad de dos objetos seleccionados pertenezcan a dos agrupamientos simultáneamente y la expresión para calcularla es:

$$J(\pi) = \sum_{i=1}^{|\pi|} \sum_{u=i+1}^{|\pi|} h(|X|, \pi_i \cap \pi_u, 2, 2) \tag{3.4.7}$$

También, es necesario calcular la intersección de otro posible evento que se genera en el caso de que haya traslape en las clases y en los agrupamientos. La probabilidad de seleccionar dos objetos que pertenezcan simultáneamente a dos clases y a dos agrupamientos:

$$J(\pi, C) = \sum_{i=1}^{|\pi|} \sum_{j=1}^{|C|} \sum_{u=i+1}^{|\pi|} \sum_{v=j+1}^{|C|} h\big(|X|, \{\pi_i \cap c_j\} \cap \{\pi_u \cap c_v\}, 2, 2\big) \quad \textbf{(3.4.8)}$$

Finalmente, la versión generalizada del índice FM, está dada por:

$$GFM = \frac{\sum_{i=1}^{|\pi|} \sum_{j=1}^{|C|} h\big(|X|, t_{ij}, 2, 2\big) - J(\pi, C)}{\sqrt{\{\sum_{i=1}^{|\pi|} h(|X|, |\pi_i|, 2, 2) - J(\pi)\}\{\sum_{j=1}^{|C|} h\big(|X|, |c_j|, 2, 2\big) - J(C)\}}} \quad \textbf{(3.4.9)}$$

Aun cuando esta medida fue generalizada para contemplar el agrupamiento con traslape, en casos de conjuntos de datos con alto índice de traslape esta medida podría no ser precisa. Puede notarse que la poca precisión en conjuntos con mucho traslape se debe a que el cálculo de $J(\pi, C), J(\pi)\ y\ J(C)$ solo contempla el hecho de que el traslape esté dado por dos agrupamientos y/o clases, es decir, un solo objeto no puede pertenecer a más de dos agrupamientos y/o clases. Asimismo, una posible extensión debería considerar que puede haber combinaciones de $|\pi|$ agrupamientos y $|C|$ clases, lo cual, a simple vista no pareciera ser un gran problema, sin embargo, una extensión de esta métrica no está contemplada en el marco de esta tesis, por lo tanto se utilizará la medida tal cual se ha descrito anteriormente. Al igual que FM, un valor mayor significa que los agrupamientos son más parecidos a las clases, y por lo tanto el algoritmo con el que fueron generados los agrupamientos es mejor.

3.4.3 FBcubed

FBcubed se calcula utilizando variaciones de las medidas Bcubed precision y Bcubed recall, propuestas en [Bagga, 1998].

Para calcular la FBcubed precisión y FBcubed Recall se utilizan las fórmulas de MultPre y MultRec, respectivamente. A continuación se describen estas medidas:

$$MultPre\big(x_i, x_j\big) = \frac{Min(\big|\pi(x_i) \cap \pi\big(x_j\big)\big|, \big|C(x_i) \cap C\big(x_j\big)\big|)}{\big|\pi(x_i) \cap \pi\big(x_j\big)\big|} \quad \textbf{(3.4.10)}$$

$$MultRec(x_i, x_j) = \frac{Min(|\pi(x_i) \cap \pi(x_j)|, |C(x_i) \cap C(x_j)|)}{|C(x_i) \cap C(x_j)|} \quad \textbf{(3.4.11)}$$

donde:

x_i y x_j son dos objetos.

$C(x_i)$ son las clases asociadas a x_i.

$\pi(x_i)$ son los agrupamientos asociados a x_i.

Estas fórmulas solo están definidas cuando x_i y x_j comparten al menos un agrupamiento (3.4.10) y cuando comparten al menos una clase (3.4.11).

Sea $D(x_i)$ el conjunto de objetos que comparten al menos un agrupamiento con el objeto x_i incluyendo a x_i. La medida Bcubed-Precision de x_i se calcula a partir de la siguiente expresión:

$$Bcubed_{pre}(x_i) = \frac{\sum_{x_j \in D(x_i)} MultPre(x_i, x_j)}{|D(x_i)|} \quad \textbf{(3.4.12)}$$

Sea $H(x_i)$ el conjunto de objetos que comparten al menos una clase con el objeto x_i incluyendo a x_i. La medida Bcubed-Recall de x_i se calcula a partir de la siguiente expresión:

$$Bcubed_{rec}(x_i) = \frac{\sum_{x_j \in H(x_i)} MultRec(x_i, x_j)}{|H(x_i)|} \quad \textbf{(3.4.13)}$$

Finalmente la medida FBcubed se obtiene a partir de la siguiente expresión:

$$FBcubed = \frac{2(\frac{1}{n}\sum_i^n Bcubed_{pre}(x_i))(\frac{1}{n}\sum_i^n Bcubed_{rec}(x_i))}{\left(\frac{1}{n}\sum_i^n Bcubed_{pre}(x_i)\right) + (\frac{1}{n}\sum_i^n Bcubed_{rec}(x_i))} \quad \textbf{(3.4.14)}$$

Esta medida satisface una serie de restricciones formales que fueron propuestas en [Amigó, 2009], las cuales ayudan a determinar qué tan buena es una medida de evaluación

para algoritmos de agrupamiento. Un valor mayor supone una mejor evaluación para los agrupamientos, y por lo tanto mejor es el algoritmo con el que fueron generados esos agrupamientos.

Las dos medidas mencionadas en este capítulo ayudarán en la evaluación de los algoritmos de agrupamiento con traslape implementados, además se podrá determinar qué tan buena es la medida GFM para evaluar algoritmos de agrupamiento con traslape.

4. Construcción de la Plataforma

En el presente capítulo se detalla el análisis, diseño e implementación de la Plataforma para Algoritmos con Traslape. En la subsección 4.1 se discuten los requerimientos y especificaciones que deben tomarse en cuenta para el diseño de la plataforma. En la subsección 4.2 se detallan los módulos que conforman la plataforma. En la subsección 4.3 se hace mención de los diferentes lenguajes de programación utilizados y la razón por la cual se utilizaron en lugar de otros lenguajes.

4.1 Análisis

Como ya se ha mencionado uno de los objetivos centrales de esta tesis es el desarrollo de una plataforma computacional para la evaluación de algoritmos del área de Reconocimiento de Patrones (más específicamente algoritmos de Agrupamiento con Traslape). Una de las características de la plataforma es que debe tener un mecanismo de comunicación que permita que los algoritmos que se incluyan en la plataforma no necesariamente estén implementados en la misma. Esta característica permitirá que en un futuro se integren otros algoritmos de clasificación no supervisada, pero también se deberían poder incluir algoritmos de clasificación supervisada, de selección de variables, de selección de prototipos, entre otros que no se encuentran implementados en las plataformas actualmente disponibles. Esto obliga a que la plataforma sea lo suficientemente general para hacer la menor cantidad de cambios o, si es posible, no hacer cambios en ella cuando se incluyan nuevos algoritmos.

La plataforma deberá incluir:

1) Una biblioteca que permita el mecanismo para la comunicación entre la plataforma y los algoritmos de minería de datos.
2) Los algoritmos OKM [Cleuziou, 2008], WOKM [Cleuziou, 2009], OKMED [Cleuziou, 2009] para agrupamiento con traslape.
3) Medidas de evaluación para los algoritmos antes mencionados.
4) Un módulo que permita generar diferentes tipos de gráficos.

4.2 Diseño de la Plataforma

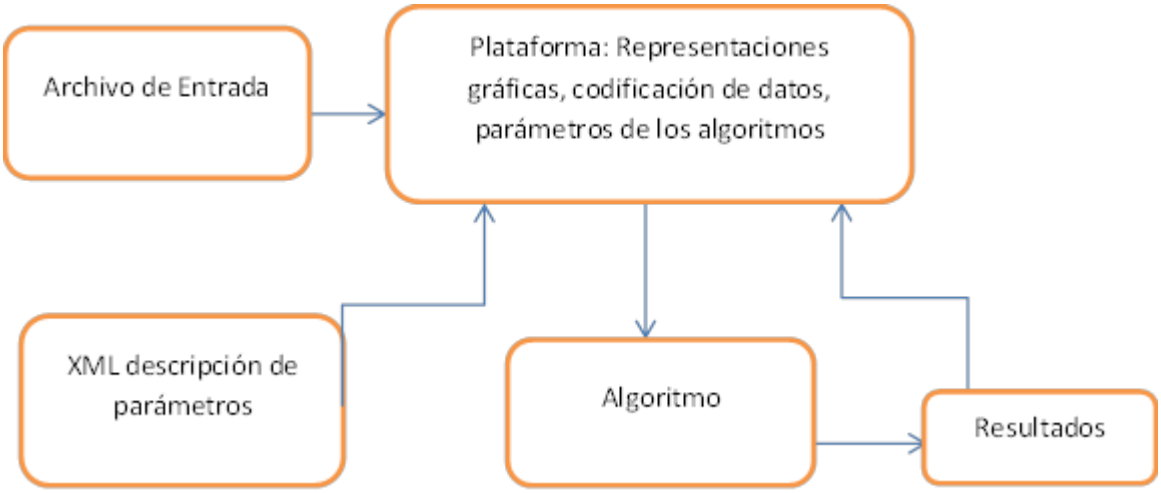

Fig. 4.1. Flujo de datos de la plataforma.

En la Fig. 4.1 se hace una representación de cómo están conectados todos los módulos de los cuales está constituida la plataforma, las flechas representan el flujo de la información durante la ejecución de la plataforma, se propuso esta división de módulos ya que facilitará futuros cambios en caso de ser requeridos.

El mecanismo de comunicación de la plataforma es una parte fundamental ya que proporciona toda la información acerca de la estructura de los algoritmos que se integrarán y de esta manera se mantiene comunicada la plataforma con los algoritmos.

En el módulo de archivos de entrada se contemplan varios formatos comúnmente utilizados en las plataformas actualmente disponibles, de esta manera no se limita a la compatibilidad con un único formato de entrada, sin embargo para la entrada de los algoritmos que se quieran incluir en la plataforma si se maneja un formato único.

Los algoritmos no se consideran como un módulo tal cual, ya que éstos son independientes de la plataforma y pueden estar implementados en cualquier lenguaje; por lo tanto podrán ser integrados siempre y cuando puedan ser llamados desde línea de comandos y devuelvan sus resultados en un formato que la plataforma pueda interpretar. Los resultados de los algoritmos deben seguir un formato propuesto para su lectura desde la plataforma, es así como se pueden mostrar los resultados en gráficas y en forma textual, pero de manera comprensible para el usuario.

Cada uno de los módulos mencionados anteriormente se describe de manera detallada en las siguientes secciones del presente capítulo.

4.2.1 Comunicación XML

Para que la plataforma tenga los detalles acerca de cómo están estructurados los algoritmos (nombre, tipo, parámetros, archivos resultantes), se propone utilizar archivos en formato XML. Estos archivos, contendrán toda la información que la plataforma necesitará para poder ejecutar los algoritmos que se deseen integrar a la misma.

La estructura de los archivos debe contemplar en su contenido las características de los algoritmos que se pudieran integrar a la plataforma. Se propone la siguiente estructura:

- Todos los archivos empiezan con una etiqueta que indica qué tipo de algoritmo se va a cargar, estos pueden ser de agrupamiento, clasificación o selección de prototipos. Si en el futuro se requieren agregar otros tipos de algoritmos se debe agregar el nombre a un archivo de configuración (config.dat) que se encuentra en la carpeta raíz de la plataforma, además de crear un directorio con el mismo nombre para ese nuevo tipo de algoritmos.

- El cuerpo del archivo contiene otras etiquetas que indican el número y tipo de parámetros que utiliza el algoritmo, como el nombre del archivo ejecutable y el nombre del archivo de salida que se obtendrá de la ejecución es ese algoritmo.

Para indicar qué tipo de algoritmo se está describiendo, se utilizarán las etiquetas <Clustering></Clustering> para indicar que es de agrupamiento, <Classifier></Classifier> para indicar que es de clasificación supervisada y <Prototype-Selection></Prototype-Selection> para indicar que es de selección de prototipos.

En el cuerpo, lo primero que se describe es el nombre del algoritmo. Este nombre es el que la plataforma mostrará en la interfaz de usuario, las etiquetas utilizadas son <Name>Nombre de Ejemplo</Name>

El nombre del archivo ejecutable del algoritmo debe indicarse con la etiqueta <Binary></Binary>, sin embargo como el algoritmo puede estar compilado para distintas plataformas esto se debe especificar. La forma en la que se especifica el sistema operativo al que corresponden es la siguiente:

<Windows>Ejecutable.ext</Windows>	Para sistemas Microsoft Windows
<Linux>Ejecutable.ext</Linux>	Para sistemas basados en Linux
<Apple>Ejecutable.ext</Apple>	Para sistemas OS X de Mac

En este caso la plataforma verificará en qué sistema se está ejecutando para saber a qué archivo debe hacer la llamada para su ejecución, en caso de que no se tenga un ejecutable para el sistema actual no se mostrará en la plataforma para evitar errores. Entonces la etiqueta Binary incluyendo su contenido quedaría como en la figura 4.2.

```
<Binary>
   <Windows>Ejecutable.ext</Windows>
   <Linux>Ejecutable.ext</Linux>
   <Apple>Ejecutable.ext</Apple>
</Binary>
```

Fig. 4.2. Ejemplo de uso de la etiqueta binary y sus subetiquetas

La siguiente etiqueta en el cuerpo en el archivo XML <Input> describe la entrada que se dará al algoritmo. Para esto, se contará con otras sub-etiquetas que permitirán describir, entre otras cosas, el tipo de parámetro que se recibe, su significado y un posible valor por omisión. La estructura se ejemplifica en la figura 4.3.

```
<Input>
  <Parameter>                Aquí se indica el inicio de la descripción de un parámetro
    <Type>Integer</Type>     Aquí se indica el tipo de dato de este parámetro
    <Default>3</Default>           Se debe poner un valor por default para este parámetro
    <Name>Neighbors</Name>   Por último se indica el texto que se debe mostrar para
  </Parameter>               describir este parámetro en la plataforma
</Input>
```

Fig. 4.3. Ejemplo del cuerpo de la etiqueta Input.

Los tipos de datos que se pueden utilizar en la etiqueta <Type> son: File (archivo), Integer (entero), Float (punto flotante), Character (caracter), List (Una lista de valores).

La última etiqueta del fichero XML es la de salida, ésta se utiliza para indicar los resultados que devolverá el algoritmo cuando termina su ejecución.

```
<Output>
  <Result>NombreArchivoSalida.ext</Result>
</Output>
```

Para los archivos de salida se tiene un formato dependiendo de qué tipo de algoritmo se haya ejecutado; de esta manera, la plataforma podrá interpretar los resultados para mostrarlos al usuario. Sin embargo, se ha pensado también en permitir otro archivo de salida que se muestre tal cual en la plataforma, por si el autor del algoritmo quiere mostrar el resultado de una manera distinta a la estándar. Para esto último, el autor deberá incluir 2 archivos de salida. El primero es el que la plataforma puede interpretar y el segundo es el que el autor quiere mostrar a los usuarios en la plataforma.

Para integrar nuevas medidas de evaluación de acuerdo al tipo de algoritmo que se desee evaluar, se debe seguir el mismo formato XML descrito anteriormente, solo que se incluirá también una etiqueta que indique que es una medida de evaluación. La etiqueta utilizada es: <Metric></Metric>. No es necesario indicar algún valor entre estas etiquetas, solo son para indicar que es una medida de evaluación.

Lo que determina si es una medida de evaluación para un algoritmo de agrupamiento o de clasificación supervisada, etc., es dependiendo de la etiqueta de cabecera, si se utilizó la etiqueta Clustering quiere decir que la medida solo funciona para ese tipo de algoritmos, lo mismo para las demás tipos de algoritmos.

Un aspecto a considerar es que, en estos casos, la plataforma dará como primera entrada el conjunto de datos que se utilizó en el algoritmo a evaluar y como segunda entrada el archivo de resultados del algoritmo ejecutado, además de esto si se requieren más parámetros se pueden colocar las etiquetas antes descritas.

Finalmente el resultado se tiene que enviar a un archivo y el texto que contenga se mostrará en texto plano en la plataforma.

Un ejemplo de cómo agregar un nuevo tipo de algoritmos, es como sigue: primero se tiene un archivo llamado config.dat en la raíz de donde se encuentra la plataforma, éste archivo contiene nombres en cada línea, en un principio son: Clustering, Classifier, Prototype-Selection, esto hará que la plataforma busque carpetas con esos nombres dentro de la carpeta Algorithm. Después verificará que cada cabecera de estos archivos tenga el nombre que corresponda según a la carpeta en la que está contenido. Finalmente los tendrá separados por categorías.

4.2.2 Archivos de Entrada

En esta etapa de la plataforma se tienen contemplados 5 formatos de archivo los cuales se describen a continuación:

1. Archivo con una solo línea de cabecera que describe los tipos de atributos. Los atributos numéricos se describen con valor 0, en caso de nominal se coloca la cantidad de valores que puede tomar, cada valor separado por coma y al final el número de clases.

2. Archivo con 3 líneas de cabecera, en la primera línea se describe la cantidad de objetos, en la segunda la cantidad de atributos y en la tercera los tipos de atributos, en este caso la tercera línea es el mismo formato usado en la descripción anterior.

3. Archivo con formato de la plataforma WEKA[7].

4. Variante de archivo WEKA con etiquetas múltiples.

5. Archivo con 4 líneas de cabecera, en la primera línea se describe la cantidad de objetos, en la segunda la cantidad de atributos, en la tercera la cantidad de etiquetas, la cual tiene valor uno si los objetos pertenecen solo a una clase o mayor que uno si el archivo tiene objetos pertenecientes a varias clases, y en la cuarta línea se indican los tipos de atributos, en este caso la cuarta línea sigue el mismo formato usado en el punto 1.

Después del encabezado o cabecera de los archivos se encuentra el contenido de la base de conocimiento, cada fila representa un objeto con sus atributos separados por coma, éstos atributos pueden ser numéricos o nominales, en el caso de los nominales se pueden incluir caracteres simples, cadenas de caracteres o números para describirlos.

Al utilizar diferentes formatos de entrada y al tener permitidas las cadenas de caracteres, es necesario hacer una codificación a un archivo sencillo. El formato elegido está descrito por 4 líneas de cabecera:

[7] http://www.cs.waikato.ac.nz/ml/weka/arff.html

- La primera para el número de objetos.
- La segunda para el número de atributos incluyendo la clase.
- La tercera para la cantidad de etiquetas usadas para las clases, si el valor es mayor que uno quiere decir que el conjunto tiene objetos pertenecientes a más de una clase, en caso de que el valor sea uno los objetos pertenecen a una sola clase.
- La cuarta para los tipos de atributos (0 numérico, para nominales se escribe el número de posibles valores).

En la descripción de los objetos para los atributos nominales no hay caracteres ni cadenas de caracteres, aun cuando sean números en su descripción se hace un cambio a valores que empiezan en 0 y se incrementa dependiendo los posibles valores que se encuentren en ese atributo. Esto para facilitar su lectura y manejarlos de manera fácil en los algoritmos implementados o por implementar.

4.2.3 Algoritmos a Integrar

Los algoritmos que se pueden integrar en esta plataforma pueden estar implementados en cualquier lenguaje de programación cuyo archivo ejecutable pueda ser llamado desde la línea de comandos. Además de eso, debe poder recibir los parámetros al momento de hacer la llamada a su ejecución. Si bien se aceptan archivos ejecutables de cualquier lenguaje es recomendable la implementación en un lenguaje como el ANSI C, ya que su manejo de recursos (memoria y CPU) es de los más eficientes. Para la implementación de los algoritmos incluidos se utilizó ANSI C.

Además de lo mencionado anteriormente, los algoritmos a implementar deben recibir el formato codificado que se detalló en el subcapítulo 4.2.2 o, en caso del formato especial utilizado para el agrupamiento de documentos se puede detallar en el archivo XML para que se reciba en el formato original. El algoritmo debe tener un archivo XML en el que se describan sus características como se menciona en la sección 4.2.1.

Los resultados de los algoritmos deben ser escritos en un archivo cuyo nombre también está descrito en su correspondiente archivo XML y este archivo de resultados debe seguir un formato según el tipo de algoritmo que sea, de esta manera la plataforma leerá los resultados y los mostrará en alguna representación que sea de fácil comprensión para el usuario.

4.2.4 Archivos de Resultado

Para los resultados es necesario cumplir con cierto formato ya que éste será interpretado por la plataforma para posteriormente ser mostrados al usuario. El formato propuesto para los tipos de algoritmos soportados en esta etapa se describe a continuación:

1. Clasificación Supervisada (Classifier): En la primer línea se colocan las clases que predijo el algoritmo para cada objeto del conjunto de prueba, cada clase obtenida separada por coma y deben estar en el mismo orden en el que se encuentran los objetos en el archivo de prueba, en la segunda línea se debe dar la precisión que se obtuvo con respecto a las clases que originalmente contenía el archivo de prueba, en caso de que el archivo de prueba no contenga las clases originales se coloca un 0. En la tercera línea se especifica expresado en segundos el tiempo que tardó en ejecutarse el algoritmo.

En la figura 4.4, se muestra un ejemplo del formato de salida definido para este tipo de algoritmos, asumiendo que se utilizó como clasificador el algoritmo k-NN y como conjunto de datos, el repositorio iris; para la ejecución, se extrajeron al azar del repositorio varios objetos que luego fueron utilizados como prueba.

```
0,0,0,0,0,1,1,1,1,1,2,2,2,2,2
100.000000
1.000000
```

Fig. 4.4. Formato de Resultado para algoritmos de Clasificación Supervisada

2. Clasificación no Supervisada (Clustering): Las primeras líneas representan los agrupamientos que se obtuvieron después de la ejecución del algoritmo (si son 3 agrupamientos son las primeras 3 líneas si son 4 serían 4 líneas, etc.), cada una de estas

líneas contiene índices separados por coma, los índices corresponden a los objetos que pertenecen a cada agrupamiento. En el caso de agrupamientos con traslape los índices pueden estar repetidos en varias líneas, después de las líneas habrá:

a. Un número -2 indicando que hay objetos sintéticos representativos (centroides)
b. Un número -3 indicando que hay objetos representativos que pertenecen al conjunto (medoides)
c. Un número -1 para indicar que no hay algún tipo de objeto representativo

En caso de haber representativos, estos se encontrarán en las líneas siguientes a este número separados por un salto de línea. El primer objeto representativo corresponderá al primer agrupamiento, el segundo representativo al segundo agrupamiento y así sucesivamente. En caso de que los objetos sean centroides se devolverá el valor de cada uno de sus atributos, además de esto se agregará una etiqueta a cada uno de estos objetos como representación del agrupamiento al que pertenecen (0, 1, …, k). En caso de que el objeto representativo si pertenezca al conjunto de datos se devolverá el índice de cada uno. Finalmente en la última línea se tendrá expresado en segundos el tiempo que tardó la ejecución del algoritmo.

En el siguiente ejemplo se utilizó el algoritmo k-means en la base iris, se pidió calcular 3 agrupamientos en este caso los objetos representativos son centroides y los resultados se muestran en la figura 4.5.

```
50,51,52,56,65,70,76,77,85,86,100,102,103,104,105,107,108,109,110,111,112,114,115,116,117,118,120,122,123,124,125,126,127,128,129,130,131,132,135,136,137,138,139,140,141,143,144,145,146,147,148,149
53,54,55,57,58,59,60,61,62,63,64,66,67,68,69,71,72,73,74,75,78,79,80,81,82,83,84,87,88,89,90,91,92,93,94,95,96,97,98,99,101,106,113,119,121,133,134,142
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49
-2
6.703846, 3.063462, 5.467309, 1.976923, 0
5.783333, 2.664583, 4.297917, 1.350000, 1
5.006000, 3.418000, 1.464000, 0.244000, 2
1.000000
```

Fig. 4.5. Formato de Resultado para algoritmos de Clasificación no Supervisada (Agrupamiento)

3. Selección de Prototipos (Prototype-Selection): La primera línea corresponde a los objetos seleccionados del conjunto de entrenamiento éstos están descritos por sus índices separados por coma, en la segunda línea se tiene el porcentaje de retención con respecto del conjunto original, y en la última línea el tiempo de ejecución del algoritmo.

Para el caso de los algoritmos de selección de prototipos se ejemplificará con el algoritmo PSR en este caso utilizando 30% de retención como parámetro y en la misma base iris Fig. 4.6.

```
7,39,49,17,40,28,11,27,26,4,35,24,29,2,99,71,55,96,97,82,94,78,63,92,91,73,61,95,74,147,112,104,128,116,139,132,103,137,110,145,111,124,115,102,98,133,119,138,77,83,70
34.000000
1.000000
```

Fig. 4.6. Formato de Resultado para algoritmos de Selección de Prototipos

Si se agregará un nuevo tipo de algoritmo, la plataforma mostrará el resultado en texto plano tal cual se describa en su archivo de resultados, hasta que se añada una interpretación gráfica o de otro tipo para el nuevo tipo de algoritmo.

4.3 Implementación de la Plataforma

Para la implementación de la plataforma se utilizó el lenguaje de programación Java[8] el cual es un lenguaje que sigue el paradigma orientado a objetos. Esta facilidad de Java es una de las razones por las cuales se eligió, pues permite que la plataforma esté modelada de modo que se pueda extender realizando pocos cambios en el código original. Otra de las razones por las cuales se eligió este lenguaje es el de hecho de ser multiplataforma. Sin embargo los algoritmos que se pueden añadir a la plataforma pueden estar escritos en diferentes lenguajes de programación, siempre y cuando se cumplan los siguientes requisitos:

a) Que la llamada se pueda hacer pasándoles los parámetros por línea de comandos
b) Que los resultados se devuelvan en un archivo de texto cuyo formato ya fue explicado anteriormente.

Aún cuando pareciera que esto hace que la plataforma se limite a ejecutarse en ciertos sistemas operativos, no es así, ya que en la descripción del algoritmo a añadir a la plataforma, la cual está hecha en un archivo con formato XML, se especifica para qué plataformas se ha compilado el algoritmo, además del nombre que lleva el archivo ejecutable dependiendo del sistema para el que está compilado. Lo mencionado anteriormente permite a la plataforma saber si se puede ejecutar el algoritmo en el sistema en el que se encuentra al momento de su ejecución.

Para la descripción de los algoritmos se utilizó el lenguaje XML porque es un metalenguaje, lo cual lo hace muy adecuado para este tipo de aplicaciones en las que se requiere generar una especie de lenguaje, que en este caso, es de etiquetas que se acoplen a los requerimientos de esta plataforma y de los algoritmos, los cuales serán descritos en archivos con este formato.

Los algoritmos implementados en la plataforma fueron desarrollados en lenguaje ANSI C, esto porque su manejo de recursos es muy bueno con respecto a otros lenguajes que dependen de máquinas virtuales o que no se pueden compilar en todos los sistemas. Debido a

[8] http://www.oracle.com/us/technologies/java/overview/index.html

que este ANSI C sigue el paradigma estructurado, se utilizaron estructuras y funciones tratando de hacer el código lo más entendible posible.

La plataforma está modelada siguiendo el paradigma orientado a objetos para facilitar futuros cambios y para comprender de manera más fácil la estructura de la misma. Con la abstracción de los conceptos de archivos de entradas, archivos de salida, formatos de representación gráfica, además del manejo codificado de datos, se pueden incluir otros módulos internos haciendo muy pocos cambios, un ejemplo es el módulo de comunicación con los archivos XML. Si se necesitara el uso de etiquetas distintas a las ya soportadas por este módulo solo se tiene que modificar el módulo que verifica la estructura de los archivos y que recupera los parámetros de los algoritmos y si es necesario agregar el nuevo atributo a la clase Algorithm, que es la que lleva todos los parámetros que los algoritmos necesitan para su ejecución.

Para mantener organizados los archivos de trabajo que se utilizan en la plataforma o en los algoritmos se maneja una estructura de directorios, la cual se describe en la figura 4.7.

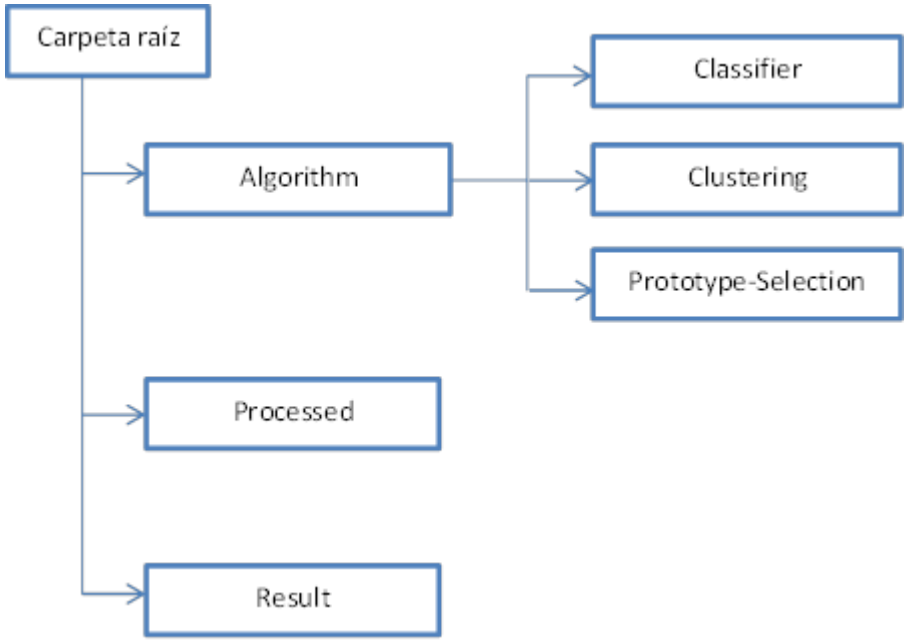

Fig. 4.7. Estructura de Directorios

La carpeta raíz es la carpeta donde se encuentra la plataforma, en la carpeta ***Algorithm*** se tienen tres carpetas en las que se encuentran los archivos ejecutables de todas las plataformas. La carpeta ***Processed*** contiene los archivos de entrenamiento ya codificados en el formato que los algoritmos pueden leer. La carpeta ***Result*** contiene los archivos de resultado que devuelven los algoritmos cuando terminan la ejecución de algún algoritmo.

Si se requiriera de algoritmos de distinto tipo al que la plataforma soporta actualmente en términos de comunicación, solo se tiene agregar el nuevo tipo de algoritmos a un archivo de texto (config.dat); de este texto es de donde la plataforma lee las cabeceras de los archivos XML de los que obtendrá la información de los algoritmos. Además el nombre dado al nuevo tipo de algoritmos es el que debe llevar la carpeta de los nuevos algoritmos a agregar, ya que cada tipo de algoritmo tiene su propio formato de resultados. Donde lo único que se tendría que agregar es un nuevo módulo para que la plataforma atienda a las necesidades específicas de ese tipo de algoritmos en la representación gráfica de resultados.

Si por ejemplo, ahora se deseara agregar algoritmos de Selección de Características, lo único que hay que hacer es agregar una nueva línea en el archivo de configuración (config.dat), con el nombre que representará a este nuevo tipo de algoritmos se le podría llamar Feature-Selection. Ahora, la próxima vez que se inicie la plataforma, leerá todos los archivos con esa cabecera en una carpeta con el nombre Feature-Selection y creará también una categoría para este nuevo tipo de algoritmos. Si no se hace modificación alguna al código solo mostrará una interfaz muy genérica en estos algoritmos, si se desea que la plataforma interprete los nuevos resultados deberá agregarse una clase nueva que pueda leer los mismos.

En el caso de los formatos de archivos que se pueden leer, sucede algo similar a la comunicación con XML, ya que solo se necesita crear una clase que herede de la clase abstracta encargada de la codificación de archivos para que el nuevo formato a aceptar sea codificado de la misma manera que los archivos que actualmente están soportados, en este caso no se tiene que modificar la interfaz, solo agregar unas cuantas líneas de código del módulo encargado de seleccionar el formato adecuado.

5. Evaluación de los Algoritmos en la Plataforma

En el presente capítulo se muestra de qué manera la plataforma ayuda al usuario a realizar sus experimentos, posteriormente se realizará un análisis comparativo entre los algoritmos de agrupamiento con traslape OKM, WOKM y OKMED mencionados en el capítulo 3, utilizando las medidas de evaluación GFM y FBcubed.

5.1 Plataforma

La plataforma ayuda al usuario a realizar experimentos con los algoritmos incluidos en la misma, además si el usuario así lo decide puede experimentar con sus propios algoritmos incorporándolos a la plataforma como se explicó en el capítulo 4.

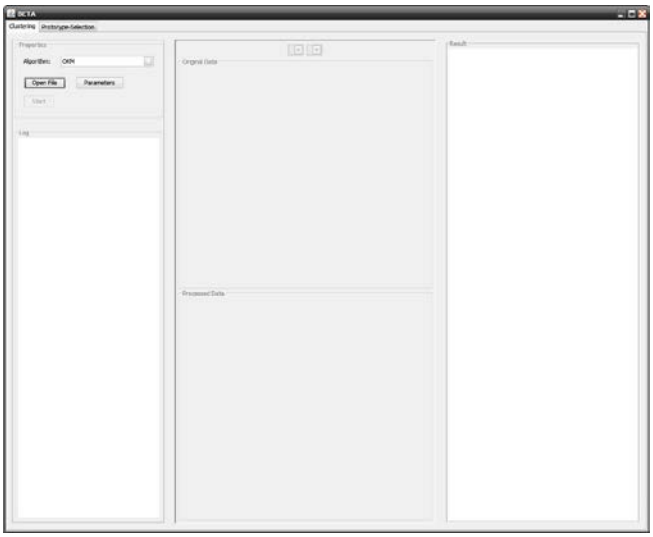

Fig. 5.1. Interfaz Gráfica al ejecutar la Plataforma.

En la figura 5.1 se muestra la interfaz principal de la plataforma para algoritmos de agrupamiento, analizando un poco se pueden notar los paneles de Propiedades, Log, Graficado (Original y Processed) y Result. Cada uno de estos componentes facilita al usuario la aplicación de cierto algoritmo, así como la interpretación de los resultados.

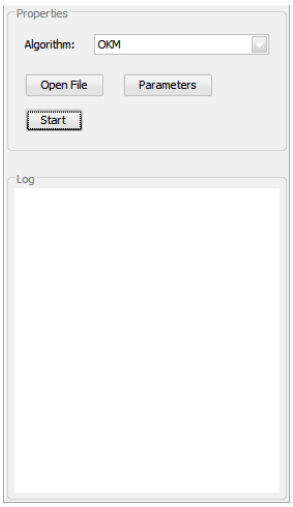

Fig. 5.2. Panel Properties y Log.

En la figura 5.2 se muestra el panel *Properties* y *Log*, en *Porperties* se muestran los algoritmos que se encuentran disponibles, además el botón Parameters permite al usuario cambiar los parámetros del algoritmo. En *Log* se muestra un log con los algoritmos ejecutados desde que se inició el programa, al dar clic en alguno de los experimentos registrados se muestran sus resultados.

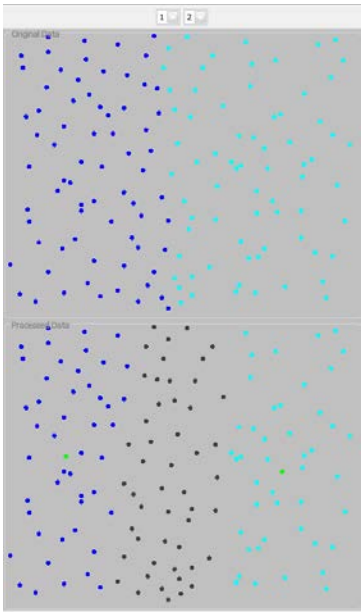

Fig. 5.3. Panel de graficado.

En la figura 5.3 se puede ver cómo es que se puede comparar un conjunto de datos con el resultado después de aplicarle un algoritmo.

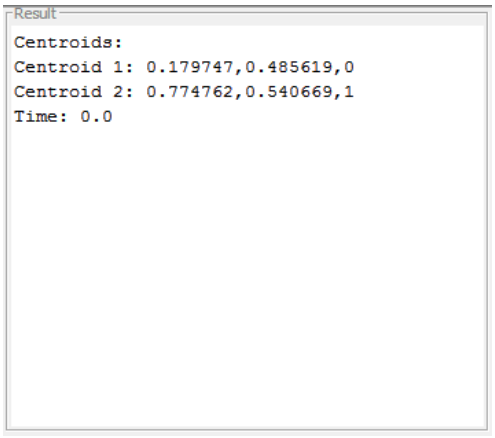

Fig. 5.4. Panel de resultado en texto plano.

En la figura 5.4 se muestra el panel *Result*, este es el encargado de mostrar los resultados textualmente, especialmente útil cuando se hace alguna evaluación, ya que en este caso el resultado se visualiza en este panel.

Al abrir un conjunto de datos, la plataforma verificará si es posible darle al usuario una representación gráfica en R^2, seleccionando dos atributos, de esta manera se puede tener una idea de cómo están distribuidos espacialmente los objetos, además se asigna un color distinto para cada clase. La gráfica no se genera si el conjunto de datos contiene más de 1500 objetos, lo anterior por cuestiones de costo computacional. En la figura 5.5 se puede ver una gráfica de un conjunto de 142 objetos sintéticos en dos dimensiones con dos clases.

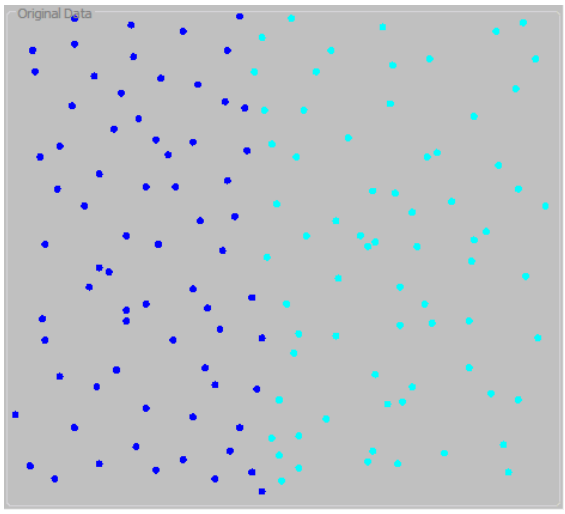

Fig. 5.5. Gráfica de un conjunto de datos sintético con dos clases.

Una vez abierto el archivo se procede a ejecutar un algoritmo, para esto se pueden dejar los parámetros por default o cambiarlos. En la figura 5.3 se muestra el resultado gráfico de haber ejecutado OKM con k = 2 en el conjunto de datos sintético del cual se hizo mención anteriormente. Al tratarse de un algoritmo de agrupamiento que obtiene objetos representativos, estos se graficarán de otro color. Como OKM obtiene traslapes, estos también se pintan de otro color para poder distinguirlos. En la figura 5.6 el color negro representa a los objetos traslapados, el color azul representa el agrupamiento uno, el color cyan representa el agrupamiento dos y los objetos en color verde son los centroides de agrupamientos.

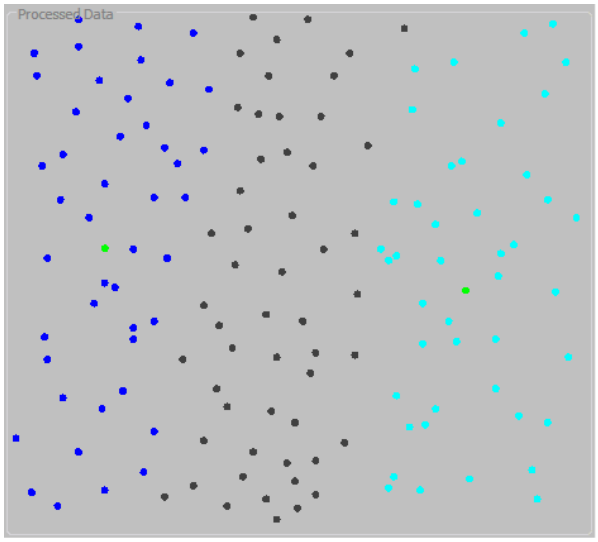

Fig. 5.6. Gráfica del resultado de aplicar OKM en un conjunto de datos.

Una vez ejecutado un algoritmo también se muestra un texto con ciertos datos con respecto al resultado del algoritmo sobre el conjunto de datos, siguiendo con la ejecución del algoritmo OKM, su resultado textual hace referencia a los centroides que obtuvo para cada agrupamiento, además del tiempo de ejecución (figura 5.7).

```
Centroids:
Centroid 1: 0.174213,0.472421,0
Centroid 2: 0.776236,0.553620,1
Time: 0.1 seconds
```

Fig. 5.7. Resultado textual al aplicar OKM en un conjunto de datos.

En el caso de que el conjunto de datos tenga más de dos atributos, en la parte superior del módulo de graficado se puede elegir cuál de los atributos disponibles se desea que represente las coordenadas X y cual las coordenadas Y, en las opciones se descartan los atributos no numéricos. En la figura 5.8 se muestra el conjunto iris [Bache, 2013], tomando su primer atributo como las coordenadas X y el cuarto atributo representa las coordenadas Y.

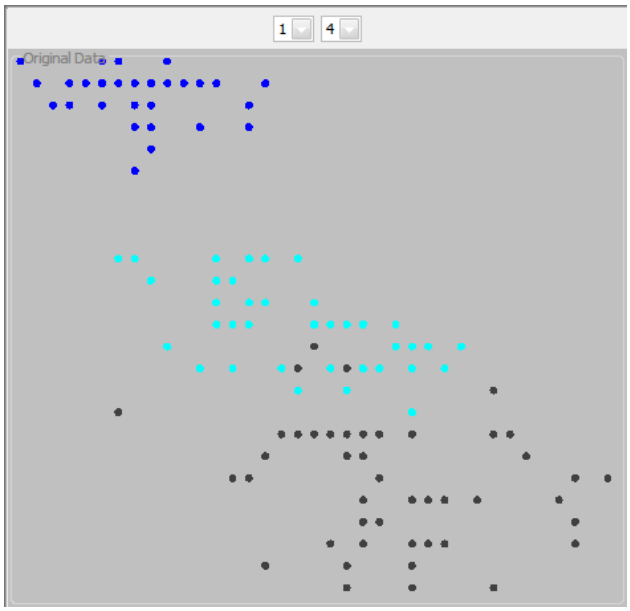

Fig. 5.8. Gráfica de iris con su atributo 1 como X y su atributo 4 como Y.

Hasta este punto se han mostrado las principales características que permiten que la plataforma facilite la aplicación de algoritmos, la interpretación visual y textual de los resultados así como su comparación.

Algo que facilita la comparación de dos o más algoritmos en un mismo conjunto es el log, el cual permite cambiar con un simple clic entre experimentos, con esto fácilmente se puede visualizar un experimento de un algoritmo y compararlo con un experimento de otro algoritmo. La plataforma es en donde se pueden evaluar los algoritmos con distintas medidas de evaluación.

5.2 Experimentos

Habiendo explicado los principales componentes que facilitan la evaluación de los algoritmos, ahora se procede a evaluar los algoritmos OKM, WOKM y OKMED. Por esto, se utilizarán las medidas GFM y FBcubed, de esta manera se podrá analizar el desempeño de estas métricas en problemas con agrupamientos traslapados.

Los conjuntos de datos (tabla 5.1) procesados para los experimentos fueron:

El conjunto de datos *emotions* [Trohidis, 2008] contiene 593 sonidos con una duración de 30 segundos, descritos con 72 atributos y etiquetados manualmente por expertos en 6 etiquetas emocionales (feliz, triste, relajado, sorprendido, tranquilo, enojado).

El conjunto de datos *scene* [Boutell, 2004] contiene imágenes a color descritas con atributos de color y espacio. Hay 6 posibles etiquetas (playa, puesta de sol, follaje de otoño, campo, montaña y urbano).

El conjunto de datos *yeast* [Elisseeff, 2001] está formado por micro-arreglos, datos de expresión y perfiles filogenéticos. El número de atributos de este conjunto es de 103. Cada gen está asociado a un conjunto de clases funcionales.

Nombre	Dominio	Objetos	Atributos	Etiquetas
Emotions	Música	593	72	6
Scene	Multimedia	2407	294	6
Yeast	Biología	2417	103	14

Tabla 5.1. Descripción de conjunto de datos utilizados en los experimentos

Para cada uno de los conjuntos de datos se aplicaron los algoritmos OKM, WOKM y OKMED, en cada conjunto se hicieron 5 ejecuciones con el parámetro *k* (número de agrupamientos) igual a la cantidad de etiquetas del conjunto de datos. Los índices para los representativos iniciales fueron seleccionados al azar para cada ejecución y se utilizaron los mismos índices en la inicialización de cada algoritmo, de esta manera se garantiza que fueron ejecutados bajo las mismas condiciones. En estos experimentos también se reporta[9] el tiempo de ejecución para los tres algoritmos.

#	OKM			WOKM			OKMED		
	GFM	FBcubed	Tiem.	GFM	FBcubed	Tiem.	GFM	FBcubed	Tiem.
1	**1.18842**	**0.487909**	**2**	**1.18842**	**0.487909**	**23**	**0.87515**	**0.458175**	**150**
2	**0.866503**	**0.485431**	**1**	**0.866503**	**0.485431**	**8**	**0.74657**	**0.445587**	**156**
3	**1.02524**	**0.521459**	**2**	**1.02524**	**0.521459**	**22**	**0.77648**	**0.442798**	**129**
4	**0.981738**	**0.483411**	**1**	**0.981738**	**0.483411**	**15**	**0.978262**	**0.507741**	**189**
5	**1.05219**	**0.500188**	**1**	**1.05219**	**0.500188**	**8**	**0.713043**	**0.439814**	**57**
Prom.	**1.0228182**	**0.4956796**	**1.4**	**1.0228182**	**0.4956796**	**15.2**	**0.817901**	**0.458823**	**136.2**

Tabla 5.2. Resultados experimentales en el conjunto de datos Emotions

En las Tablas 5.2, 5.3 y 5.4 se reportan los resultados obtenidos con los diferentes algoritmos de agrupamiento con traslape y las medidas de evaluación para estos algoritmos. Cada algoritmo tiene valores asignados en tres columnas, la primera columna representa el valor obtenido con la medida GFM, la segunda columna es el valor que reporto la medida FBcubed y la tercera columna es el tiempo que transcurrió durante la ejecución del algoritmo, éste último reportado en segundos.

[9] Se utilizó un equipo con procesador Intel® Atom™ N280 1.66 GHz y 2GB en RAM DDR2 a 800MHz de frecuencia, con Windows XP.

En la tabla 5.2 se puede notar que para OKM y WOKM, GFM y FBcubed obtienen los mismos valores, esto puede ser debido a que realmente no hubo mejora ajustando pesos para los atributos y el algoritmo en esos casos es equivalente a OKM. En el caso de OKMED obtuvo valores menores (peores) con ambas medidas, sin embargo, en lo que respecta a FBcubed no fue tan grande la diferencia.

Aún cuando pareciera que los valores mayores a uno que GFM obtiene en OKM y WOKM son un error de cálculo, realmente no es así, en el capítulo 3 se mencionó, que ésta medida pudiera no ser buena para conjuntos con alto índice de traslape, dado que su formulación solo contempla que los objetos se traslapen a lo más en dos agrupamientos. Siendo Emotions un conjunto en donde incluso se encuentran casos de objetos que pertenecen a más de 3 clases, por lo cual es entendible encontrarse con este problema. Este problema no se presentó en OKMED, posiblemente debido a que el índice de traslape que encuentra es menor respecto al de OKM.

#	OKM			WOKM			OKMED		
	GFM	FBcubed	Tiem.	GFM	FBcubed	Tiem.	GFM	FBcubed	Tiem.
1	**0.972719**	**0.268283**	**20**	**0.972719**	**0.268283**	**189**	**0.588849**	**0.357736**	**11438**
2	**0.983523**	**0.247385**	**37**	**0.983523**	**0.247385**	**342**	**0.573796**	**0.413552**	**9221**
3	**0.86842**	**0.275395**	**35**	**0.86842**	**0.275395**	**287**	**0.540317**	**0.351983**	**12065**
4	**0.912981**	**0.256847**	**46**	**0.912801**	**0.256856**	**433**	**0.540912**	**0.376878**	**10789**
5	**0.777673**	**0.324312**	**45**	**0.782873**	**0.314635**	**212**	**0.586433**	**0.395032**	**15000**
Prom.	**0.9030632**	**0.2744444**	**36.6**	**0.9040672**	**0.2725108**	**292.6**	**0.5660614**	**0.3790362**	**11702.6**

Tabla 5.3. Resultados experimentales en el conjunto de datos Scene

En la tabla 5.3 pareciera que las medidas se contradicen, por un lado se tienen valores altos en la medida GFM para OKM y WOKM, a la vez que la medida FBcubed reporta valores bajos en estos algoritmos, por otra parte OKMED tiene valores mayores en la medida FBcubed y GFM reportó valores mucho menores que en OKM y WOKM.

El problema en este caso es que OKM y WOKM encuentran un alto índice de traslape, lo cual provoca una sobreestimación de probabilidades, como se mencionó en el experimento

anterior. Dado que este conjunto no tiene un alto índice de traslape FBcubed penaliza su estimación de calidad. En OKMED no se encuentra un alto índice de traslape, esto hace que aún se mantenga en el rango de probabilidades de GFM y por eso no hay tanta sobreestimación en su medida, también se ve compensado en el valor que estima FBcubed, el cual es mejor que en los otros dos algoritmos.

#	OKM			WOKM			OKMED		
	GFM	FBcubed	Tiem.	GFM	FBcubed	Tiem.	GFM	FBcubed	Tiem.
1	**4.61388**	**0.679942**	**19**	**4.62983**	**0.682119**	**517**	**2.60602**	**0.105068**	**428**
2	**5.56438**	**0.651571**	**12**	**5.56438**	**0.651571**	**138**	**2.61913**	**0.100048**	**182**
3	**5.36726**	**0.675373**	**47**	**5.36726**	**0.675373**	**501**	**2.74584**	**0.113204**	**553**
4	**5.14707**	**0.676744**	**47**	**5.14707**	**0.676744**	**451**	**2.69073**	**0.109168**	**434**
5	**4.9986**	**0.663089**	**35**	**4.9986**	**0.663089**	**375**	**2.89692**	**0.123697**	**769**
Prom.	**5.138238**	**0.6693438**	**32**	**5.141428**	**0.6697792**	**396.4**	**2.711728**	**0.110237**	**473.2**

Tabla 5.4. Resultados experimentales en el conjunto de datos Yeast

En la tabla 5.4 se hace notorio que la medida GFM no resulta ser muy buena para conjuntos traslapados, a menos que se generalice a otro nivel, es clara la exagerada sobreestimación que hace. Este conjunto de datos es el que tiene el más alto índice de traslape de los tres conjuntos que analizamos.

Con base a la medida FBcubed, se puede ver que tanto OKM como WOKM obtuvieron un muy buen resultado, mientras que OKMED está muy por debajo en cuanto a calidad. Como se mencionó anteriormente, OKM y WOKM encuentran un alto índice de traslape, al ser este un conjunto con alto índice de traslape, es normal pensar que obtendrían una evaluación buena, en cambio OKMED al obtener un índice menor de traslape se vio penalizado.

Conjunto de Datos	OKM		WOKM		OKMED	
	FBcubed	Tiempo	FBcubed	Tiempo	FBcubed	Tiempo
Emotions	**0.4956796**	**1.4**	**0.4956796**	*15.2*	*0.458823*	136.2
Scene	0.2744444	**36.6**	*0.2725108*	292.6	**0.3790362**	*11702.6*
Yeast	0.6693438	**32**	**0.6697792**	396.4	*0.110237*	*473.2*
Promedio	**0.4798226**	**23.3333333**	0.4793232	234.733333	*0.31603207*	*4104*

Tabla. 5.5. Promedio de los valores obtenidos con FBcubed y tiempo de ejecución reportado en segundos.

Descartando los valores de la medida GFM tenemos en la tabla 5.5 un promedio de tiempos de ejecución y de calidad estimada por FBcubed. Algo muy notorio es el elevado tiempo de ejecución obtenido con OKMED, ya que sus tiempos exceden por mucho a OKM y WOKM. En base a la medida FBcubed se podría decir que para estos experimentos el mejor algoritmo es OKM seguido por WOKM y finalmente OKMED.

El hecho de que OKMED haya obtenido valores menores en estos experimentos no necesariamente quiere decir que sea peor, ya que en Scene claramente se vio que puede haber casos en donde tiene mejor desempeño, y se requiere un estudio experimental más amplio.

En WOKM no se vio un caso donde realmente se mejorará el resultado con el ajuste de pesos, quizás se necesiten conjuntos de datos con más atributos, de hecho el ajuste de peso está pensado para conjuntos de datos grandes.

Finalmente, se puede concluir que OKM es un buen algoritmo de traslape en conjuntos con alto índice de traslape, pero no recomendable en conjuntos con un índice bajo de traslape, ya que encontrará traslape donde no lo hay.

6. Conclusiones y Trabajo Futuro

En el área de Reconocimiento de Patrones resulta importante poder evaluar los algoritmos propuestos en la literatura, esta evaluación permite saber qué algoritmo conviene utilizar bajo ciertas circunstancias. Para hacer la evaluación o comparación se encuentran disponibles diversas plataformas, sin embargo, estas plataformas tienen ciertas limitantes en cuanto a la forma en que se integran los algoritmos, además de que por lo general no se pueden incorporar nuevas medidas de evaluación. Estas limitantes fueron contempladas para el desarrollo de una nueva plataforma, la cual permite la integración de algoritmos de agrupamiento y medidas de evaluación de una manera simple e independiente del lenguaje de programación que se desea usar.

Para evaluar la eficacia de esta plataforma fueron incorporados los algoritmos de agrupamiento con traslape OKM, OKMED y WOKM, ya que las plataformas reportadas en la actualidad no contienen este tipo de algoritmos, por lo cual resulta muy útil tenerlos disponibles para experimentos, así como para otras aplicaciones. Esta plataforma además permite la inclusión de algoritmos de clasificación supervisada, selección de prototipos, agrupamiento (conjunto cociente), y prácticamente cualquier algoritmo de reconocimiento de patrones y/o minería de datos que siga las especificaciones del mecanismo de comunicación que utiliza la plataforma. También es posible incorporar medidas de evaluación de acuerdo a la familia de algoritmos que se deseen evaluar, dentro de esta plataforma fueron incorporadas las medidas GFM y FBcubed para la evaluación de algoritmos de agrupamiento con traslape. Se evaluaron los algoritmos disponibles en la plataforma con esta medida y se reportaron los resultados obtenidos.

En los experimentos realizados fue notoria la mala calidad de la medida GFM, la cual debería extenderse a un nivel más generalizado para poder trabajar con casos en los que los conjuntos de datos tengan un alto índice de traslape. En cambio, FBcubed resulta ser una buena medida que contempla muy bien los casos con bajo y alto índice de traslape. El algoritmo que en promedio tuvo un mejor desempeño fue OKM, sin embargo, para hacer un análisis más detallado con respecto al rendimiento de los algoritmos resultaría adecuado

probar más conjuntos de datos con traslape y además de probar distintas funciones de distancia para OKMED. Adicionalmente, se deberían contemplar también conjuntos con gran cantidad de atributos, así se podría ver qué tanto mejora la calidad el ajuste de pesos en WOKM.

Finalmente, como trabajo a futuro se contempla el desarrollo de un módulo interno para la plataforma que se encargue de todo el pre-procesamiento de los conjuntos de datos de entrada, esto para proporcionar más opciones al usuario a la hora de analizar sus conjuntos de datos. También se contempla la extensión paralela de los algoritmos OKM, WOKM y OKMED ya sea usando varios núcleos de CPU o unidades de procesamiento de GPU. En cuanto al crecimiento de la plataforma, gracias a su capacidad de incorporar casi cualquier algoritmo de reconocimiento de patrones y minería de datos, su crecimiento estará dado por la cantidad de alumnos, docentes, investigadores y usuarios de otra índole que deseen añadir sus algoritmos, lo cual deja abierto a un crecimiento muy amplio. En base a la velocidad de este crecimiento podría también contemplarse el crear interfaces gráficas aún más completas, un editor de archivos de entrada, configuración, además incorporar nuevas representaciones gráficas, como puntos en 3D, etc.

7. Referencias

[Aslam, 1998] J. Aslam, K. Pelekhov and D. Rus, Static and dynamic information organization with star clusters. In Proceedings of the Seventh International Conference on Information and knowledge Management, 1998, pages 208–217.

[Aslam, 2000] J. Aslam, K. Pelekhov and D. Rus, Using star clusters for filtering. In Proceedings of the Ninth International Conference on Information and Knowledge Management, USA, 2000, pages 306–313.

[Bache, 2013] Bache, K. & Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Bagga, 1998] Bagga A, Baldwin B (1998) Entity-based cross-document coreferencing using the vector space model. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98), pp 79–85

[Banerjee, 2005] Banerjee A, Krumpelman C, Basu S, Mooney R, Ghosh J (2005) Model-based overlapping clustering. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD2005), pp 532–537.

[Bezdek, 1981] Bezdek, J. C. (1981). Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York.

[Boutell, 2004] M.R. Boutell, J. Luo, X. Shen, and C.M. Brown. Learning multi-labelscene classiffication. Pattern Recognition, 37(9):1757-1771, 2004.

[Brunk, 1968] H. D. Brunk, J. E. Holstein, F. Williams, A comparison of binomial approximations to the hypergeometric distribution, American Statistician 22 (1) (1968) 24–26.

[Chan, 2004] Chan, E. Y., Ching, W.-K., Ng, M. K., and Huang, J. Z. (2004). An optimization algorithm for clustering using weighted dissimilarity measures. Pattern Recognition, 37(5):943–952.

[Cleuziou, 2007] Guillaume Cleuziou, A Generalization of k-Means for Overlapping. Clustering. Université d'Orléans, LIFO. Rapport No RR-2007-15.

[Cleuziou, 2008] Guillaume Cleuziou, "An extended version of the k-means method for overlapping clustering," Pattern Recognition, 2008. ICPR 2008. 19th International Conference on , vol., no., pp.1-4, 8-11 Dec. 2008.

[Cleuziou, 2009] Guillaume Cleuziou: Two Variants of the OKM for Overlapping Clustering. EGC (best of volume) 2009: 149-166.

[Elisseeff, 2002] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In T.G. Dietterich, S. Becker, and Z. Ghahramani, (eds), Advances in Neural Information Processing Systems 14, 2002.

[Fowlkes, 1983] E.B. Fowlkes, C. L. Mallows, A method for comparing two hierarchical clusterings, Journal of the American Statistical Association 78 (383) (1983) 553–569. ISSN 01621459.

[Gago, 2007] Gago Alonso A, Pérez Suárez A, Medina Pagola J (2007) Acons: a new algorithm for clustering documents. In: Proceedings of the 12th Iberoamerican Congress on Pattern Recognition (CIARP2007), LNCS 4756, pp 664–673.

[Gil, 2010] Gil García R, Pons Porrata A (2010) Dynamic hierarchical algorithms for document clustering. Pattern Recognition Letters 31(6):469–477.

[Gil, 2003] Gil García RJ, Badía Contelles JM, Pons Porrata A (2003) Extended star clustering algorithm. In: Proceedings of the 8th Iberoamerican Congress on Pattern Recognition (CIARP2003), LNCS 2905, pp 480–487.

[Halkidi, 2001] Halkidi M, Batistakis Y, Vazirgiannis M (2001) On clustering validation techniques. Journal of Intelligent Information Systems 17(2-3):107–145.

[Jain, 2000] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: a review. IEEE transactions on Pattern Analysis and Machine Intelligence, 22(1), pp. 4-37, 2000.

[Jain, 1999] A.K. Jain, M.N. Murty and P.J. Flynn, Data clustering: a review, ACM Computing Surveys 31(3) (1999), 264–323.

[Kaufman, 1987] Kaufman, L. and Rousseeuw, P. J. (1987). Clustering by means of medoids. In Dodge, Y. (Ed.) Statistical Data Analysis based on the L1 Norm, pages 405–416.

[Larsen, 1999] Larsen B, Aone C (1999) Fast and effective text mining using linear-time document clustering. Knowledge Discovery and Data Mining pp 16–22.

[MacQueen, 1967] MacQueen, J. B. (1967). "Some Methods for classification and Analysis of Multivariate Observations". 1. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.

[Pérez, 2007] Pérez Suárez A, Medina Pagola J (2007) A clustering algorithm based on generalized stars. In: Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2007), LNAI 4571, pp 248– 262.

[Pérez, 2009] Pérez-Suárez A, Martínez Trinidad J, Carrasco Ochoa J, Medina Pagola J (2009) A new incremental algorithm for overlapped clustering. In: Proceedings of the 14th Iberoamerican Congress on Pattern Recognition (CIARP2009), LNCS 5856, pp 497–504.

[Pfitzner, 2009] Pfitzner D, Leibbrandt R, Powers D (2009) Characterization and evaluation of similarity measures for pairs of clusterings. Knowledge and Information Systems 19(3):361–394.

[Ramirez, 2012] Eduardo H. Ramirez, Ramon Brena, Davide Magatti, Fabio Stella, Topic model validation, Neurocomputing, Volume 76, Issue 1, 15 January 2012, Pages 125-133.

[Roth, 2002] Roth V, Braun ML, Lange T, Buhmann JM (2002) Stability-based model order selection in clustering with applications to gene expression data. In: Proceedings of the International Conference on Artificial Neural Networks, pp 607–612.

[Sokal, 1958] Sokal R and Michener C (1958). "A statistical method for evaluating systematic relationships". University of Kansas Science Bulletin 38: 1409–1438.

[Stein, 2003] Stein B, Meyer S, Wissbrock F (2003) On cluster validity and the information need of users. In: Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA'03), pp 216–221.

[Steinbach, 2000] Steinbach, M., Karypis, G., y Kumar, V. (2000). A comparison of Document Clustering Techniques. Technical Report #00-034. University of Minnesota. In KDD Workshop on Text Mining.

[Trohidis, 2008] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas. "Multilabel Classification of Music into Emotions". Proc. 2008 International Conference on Music Information Retrieval (ISMIR 2008), pp. 325-330, Philadelphia, PA, USA, 2008.

[Yeung, 2000] Yeung KY, Haynor DR, Ruzzo WL (2000) Validating clustering for gene expression data. Bioinformatics 17:309–318.

[Zhao, 2001] Zhao Y, Karypis G (2001) Criterion functions for document clustering: Experiments and analysis. Tech. Rep. 01–40, Department of Computer Science, University of Minnesota, Minneapolis, MN.

Apéndice

Ejemplos Sección 4.2.1

A continuación se muestra un ejemplo de cada tipo de algoritmo:

Clustering: Para el ejemplo de clasificación no supervisada se utilizó el algoritmo k-Means compilado para las plataformas Windows, Linux y Mac. Los parámetros son el archivo del conjunto de datos, los k agrupamientos que se requieren y el nombre del archivo de resultado (Figura 8.1).

```
<Clustering>
  <Name>k-Means</Name>
  <Binary>
    <Windows>kmeans.exe</Windows>
    <Linux>kmeans</Linux>
    <Apple>kmeans.o</Apple>
  </Binary>
  <Input>
    <Parameter>
      <Type>File</Type>
      <Name>Training</Name>
    </Parameter>
    <Parameter>
      <Type>Integer</Type>
      <Name>K Clusters</Name>
      <Default>3</Default>
    </Parameter>
  </Input>
  <Output>
    <Result>FileName.txt</Result>
  </Output>
</Clustering>
```

Fig. 8.1. Ejemplo de Descripción XML para k-Means con 2 parámetros de entrada

Classifier: Para el ejemplo de clasificación supervisada utilizamos el algoritmo k-Nearest Neighbors compilado para las plataformas Windows, Linux y Mac. Los parámetros son el archivo del conjunto de datos de entrenamiento, el archivo del conjunto a clasificar, los k vecinos que se obtendrán y el nombre del archivo de resultado (Figura 8.2).

```
<Cassifier>
  <Name>k-NN</Name>
  <Binary>
    <Windows>knn.exe</Windows>
    <Linux>knn</Linux>
    <Apple>knn.o</Apple>
  </Binary>
  <Input>
    <Parameter>
      <Type>File</Type>
      <Name>Training</Name>
    </Parameter>
    <Parameter>
      <Type>File</Type>
      <Name>Test</Name>
    </Parameter>
    <Parameter>
      <Type>Integer</Type>
      <Name>K Neighbors</Name>
      <Default>3</Default>
    </Parameter>
  </Input>
  <Output>
    <Result>FileName.txt</Result>
  </Output>
</Classifier>
```

Fig. 8.2. Ejemplo de Descripción XML para k-NN con 3 parámetros de entrada

Prototype Selection: Para el ejemplo de selección de prototipos utilizamos el algoritmo Prototype Selection By Relevance compilado para las plataformas Windows, Linux y Mac. Los parámetros son el archivo del conjunto de datos de entrenamiento, el porcentaje de retención y el nombre del archivo de resultado (figura 8.3).

```
<Prototype-Selection>
  <Name>PSR</PSR>
  <Binary>
    <Windows>psr.exe</Windows>
    <Linux>psr</Linux>
    <Apple>psr.o</Apple>
  </Binary>
  <Input>
    <Parameter>
      <Type>File</Type>
      <Name>Training</Name>
    </Parameter>
    <Parameter>
      <Type>Integer</Type>
      <Name>% Retention</Name>
      <Default>30</Default>
    </Parameter>
  </Input>
  <Output>
    <Result>FileName.txt</Result>
  </Output>
</Prototype-Selection>
```

Fig. 8.3. Ejemplo de Descripción XML para PSR con 2 parámetros de entrada

Ejemplos Sección 4.2.2

A continuación se ejemplifican los formatos de archivo de entrada, aceptados por la plataforma, en el ejemplo se utilizó la base weather y al final se encuentra el resultado después de codificar cualquiera de los formatos:

1. Formato de una cabecera.

```
3,0,0,2,2
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
```

2. Formato de 3 cabeceras.

```
14
5
3,0,0,2,2
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
```

3. Formato WEKA.

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}
```

4. Resultado después de la codificación.

```
14
5
1
3,0,0,2,2
0,85,85,0,0
0,80,90,1,0
1,83,86,0,1
```

@data
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

2,70,96,0,1
2,68,80,0,1
2,65,70,1,0
1,64,65,1,1
0,72,95,0,0
0,69,70,0,1
2,75,80,0,1
0,75,70,1,1
1,72,90,1,1
1,81,75,0,1
2,71,91,1,0

A continuación se ejemplifican los formatos de archivo de entrada multi-etiquetados, aceptados por la plataforma, al final se encuentra resultado después de codificar cualquiera de los formatos:

1. Formato de 4 cabeceras.

14
5
2
3,0,0,2,1,1
sunny,85,85,FALSE,0,1
sunny,80,90,TRUE,0,1
overcast,83,86,FALSE,1,0
rainy,70,96,FALSE,1,0
rainy,68,80,FALSE,1,0
rainy,65,70,TRUE,0,1
overcast,64,65,TRUE,1,0
sunny,72,95,FALSE, 0,1
sunny,69,70,FALSE,1,0
rainy,75,80,FALSE,1,0
sunny,75,70,TRUE,1,0
overcast,72,90,TRUE,1,0
overcast,81,75,FALSE,1,0
rainy,71,91,TRUE, 0,1

2. Formato WEKA multi-etiqueta.

@multilabelrelation weather
@labels 2
@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play-yes {1, 0}
@attribute play-no{1, 0}

@data
sunny,85,85,FALSE,0,1
sunny,80,90,TRUE,0,1
overcast,83,86,FALSE,1,0
rainy,70,96,FALSE,1,0
rainy,68,80,FALSE,1,0
rainy,65,70,TRUE,0,1
overcast,64,65,TRUE,1,0
sunny,72,95,FALSE, 0,1
sunny,69,70,FALSE,1,0
rainy,75,80,FALSE,1,0
sunny,75,70,TRUE,1,0
overcast,72,90,TRUE,1,0
overcast,81,75,FALSE,1,0
rainy,71,91,TRUE, 0,1

3. Resultado después de la codificación.

14
5
2
3,0,0,2,2
0,85,85,0,0,1
0,80,90,1,0,1
1,83,86,0,1,0
2,70,96,0,1,0
2,68,80,0,1,0
2,65,70,1,0,1
1,64,65,1,1,0
0,72,95,0,0,1
0,69,70,0,1,0
2,75,80,0,1,0
0,75,70,1,1,0
1,72,90,1,1,0
1,81,75,0,1,0
2,71,91,1,0,1

En estos ejemplos aun siendo multi-etiquetados no están traslapados, ya que no se asigna un 1 en más de una de sus etiquetas, por lo cual un conjunto no traslapado también puede representarse con este formato, sin embargo este formato se hizo pensando en que la plataforma pueda evaluar conjuntos traslapados, esta es la manera en la cual se le puede dar como entrada un conjunto de datos con traslape.