UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO UAEM TEXCOCO

ESCALABILIDAD DE ALGORITMOS ON-LINE CON OPTIMIZACIÓN DE RAM

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

ING. MÓNICA SANTIBÁÑEZ SÁNCHEZ

TUTOR ACADÉMICO:

DRA. EN C. ROSA MARÍA VALDOVINOS ROSAS

TUTORES ADJUNTOS:

DR. ADRIÁN TRUEBA ESPINOSA

DRA. ERÉNDIRA RENDÓN LARA

TEXCOCO, ESTADO DE MÉXICO NOVIEMBRE 2013

Dedicatoria

Con todo cariño y amor a mis padres y hermana, quienes nunca han dejado de apoyarme y ser un aliento para continuar.

Agradecimientos

Quiero agradecer enormemente el apoyo que mis padres y mi hermana me han brindado para lograr alcanzar este objetivo que hoy se ve concluido con mucha satisfacción, y que a lo largo del camino para alcanzarlo tuvo algunos contratiempos que gracias a su apoyo salieron adelante, y que sin el soporte que me brindaron el camino habría sido muy diferente. Gracias mamá por seguir confiando en mí y ayudarme a mantener este sueño cuando ya no podía.

También quiero hacer un muy especial agradecimiento y reconocimiento a Rosa María Valdovinos como profesionista y como persona, porque es en mucho gracias a ella que el día de hoy me encuentro en el final de esta aventura, que comenzó como una sugerencia suya y que me alegro de haber seguido porque que me ha traído grandes satisfacciones y enseñanzas. De usted he aprendido que todo hay que hacerlo con pasión y dedicación para hacerlo bien. Le agradezco el que me haya señalado mis aciertos y errores cuando fue necesario y que me haya ayudado a ver las oportunidades y motivado a tomarlas.

En definitiva Dios pone en nuestro camino a la gente correcta, normalmente aquella que puede darnos grandes enseñanzas, pero con un plus cuando no sólo nos enseña si no que comparte con nosotros. Agradezco el que haya compartido conmigo Dra.

Viví muy diferentes experiencias, conocí muchas personas y lugares, de los cuales me quedo con amigos y grandes recuerdos. Dos de esas personas son Soco y Laura, a quienes quiero agradecer su apoyo y paciencia… a Soquito por ayudarme a integrarme a la experiencia de vivir en un lugar nuevo y a Laura su paciente dedicación con mis dudas. Gracias.

A mi amiga Alba, que a pesar de la distancia siempre has estado presente en todo momento. A Dulce quien también siempre ha seguido de cerca mis pasos aunque estemos lejos.

De igual forma quiero agradecer el gran apoyo del Dr. Adrián Trueba para continuar el trabajo con la Dra. Rosa María y por compartirme parte de su conocimiento y experiencia. Agradezco el apoyo y aliento del M. Sergio Ruíz, la Dra. Eréndira Rendón, el Dr. Joel Ayala, la Dra. Yulia Ledeneva y el Dr. Jair Cervantes, de quienes aprendí que el conocimiento es la corona del esfuerzo y la dedicación.

Por último pero no por ello menos importante al M. Roberto Carlos Garduño por su apoyo durante mi estancia en el Tecnológico de Estudios Superiores de Chalco como docente en el primer año de maestría. Al Dr. Raymundo y a la Facultad de Ingeniería de la UAEM por brindarme el espacio para desarrollar este proyecto.

ÍNDICE

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE ALGORITMOS

Pág.

ÍNDICE DE FIGURAS

Pág.

PARTE I

INTRODUCCIÓN

CAPÍTULO 1 INTRODUCCIÓN

A través de los años, el ser humano ha tratado de dotar de cierta inteligencia a las máquinas para que realicen tareas propias del hombre de forma automática con la finalidad de alcanzar objetivos como su implementación en ambientes reales, en los que se requiere de una alta fiabilidad y rendimiento. Una de estas áreas de investigación es el Reconocimiento de Patrones (RP), en el que se intenta reproducir el proceso biológico a través del uso de algoritmos que ayuden a la máquina a identificar objetos y clasificarlos como lo hace el ser humano.

A partir de la automatización, el incremento continuo de los datos y el requerimiento de información, el uso de los sistemas de RP se ha extendido, de forma que en la actualidad, la capacidad para reconocer patrones es en muchas áreas de gran importancia, de manera que las tareas puedan realizarse de forma rápida y eficaz [Ripley, 2008]. La visión por computadora, el reconocimiento de voz, caracteres e imágenes, el diagnostico asistido por computadora y la minería de datos, con la que puede descubrirse información relevante escondida en el conjunto de datos. Son algunas de las aplicaciones del reconocimiento de patrones.

En concreto, se pueden mencionar como ejemplos en seguridad y automatización el reconocimiento biométrico como huella digital, el reconocimiento facial y de voz [Espinosa, 2001], reconocimiento de firma y escritura [Mendaza, 2010]. En medicina, el diagnóstico de enfermedades y reconocimiento de imágenes médicas como el análisis de electrocardiogramas para la detección de enfermedad coronaria [Clavijo, 2006], electroencefalogramas [Guarnizo, 2008]. Otra aplicación es el reconocimiento de hábitos de compra y la predicción del mercado [Sanjurjo, 2009]. En geología, reconocimiento de suelos y análisis de imágenes telescópicas [Rullán, 2011], análisis sísmico [Fernández, 2001], recursos geológicos [Irigoyen, 2011]. Tiene aplicaciones en previsión meteorológica [Paraschivescu, 2011]. Por lo anterior el RP es un área multidisciplinar que impacta en una gran cantidad de espacios en los que mejora el rendimiento [Marques de Sá, 2001].

En este sentido, en el proceso RP se pueden encontrar dos enfoques de realizar el aprendizaje del clasificador: aprendizaje *off-line* y aprendizaje *on-line*. Por un lado, el aprendizaje *off-line* aprende a través de un Conjunto de Entrenamiento (CE, patrones etiquetados), que le enseña al sistema a reconocer los patrones que corresponden a cada clase, sin la posibilidad de incorporar casos nuevos, una vez finalizado éste. Por otro lado, el aprendizaje *on-line*, como lo dice su nombre, trabaja con un flujo constante de datos, el cual debe ser procesado y clasificado en tiempo real. Para que esto sea posible, se hace uso de una metodología que tiene la capacidad de obtener resultados óptimos de la clasificación del conjunto de datos en tiempo real de forma automática [Ferrer, 2005a].

Con el avance tecnológico y el incremento de información que se ha desarrollado en los últimos años, la obtención de datos evidentemente aumenta, así como su área de aplicación. Por tanto, el almacenamiento y tratamiento de dichos datos requiere de formas que los procesen conforme al tiempo de llegada al sistema, pero no sólo eso, si no que el sistema debe ser capaz de reconocer nuevos grupos que puedan desprenderse de datos nuevos que no pueden ser clasificados y que deben ser integrados como nuevo conocimiento; o que de otra forma representen ruido para el conjunto de datos y puedan ser rechazados para evitar la corrupción de los valores y extraer el mayor provecho posible del conjunto.

Este tipo de sistemas requiere de una respuesta en tiempo real, por lo que el conjunto de datos debe ser tratado de forma dinámica, tomando en cuenta los recursos disponibles para dicha tarea. De lo contrario los nuevos ejemplos no podrían ser integrados al conocimiento en tiempo real, retrasando el procesamiento de los mismos, siendo necesario generar un nuevo conjunto de entrenamiento que integre los nuevos ejemplos para re-entrenar el algoritmo, consumiendo de esta forma, mayor tiempo, recursos y costo.

Por lo antes expuesto, en problemas reales de RP la utilización del aprendizaje *on-line* ha cobrado mayor fuerza pues los sistemas *off-line* asumen, como se había mencionado, un aprendizaje estático que no facilita la integración de nuevos ejemplos al CE y tienen que ser integrados a través de un nuevo CE diseñado por el especialista, con lo que el sistema tiene que ser reentrenado para tomar en cuenta el nuevo conocimiento. Esto no siempre es fácil de llevar a cabo, pues puede que el especialista no esté disponible en el momento que se requiere de su conocimiento.

Lo anterior podría orillar al sistema a llegar a un punto en el que, por no ser actualizado constantemente, conforme al cambio en el contexto del problema, no se adapte y el aprendizaje obtenido sea considerado obsoleto, teniendo implícito, desde el inicio de su desarrollo, un límite de aprendizaje o en su defecto, que contemplar a largo plazo el costo de actualización y los contratiempos que podrían existir para ello.

Es por tanto que el aprendizaje *on-line* ha sido aplicado a diversas áreas y tareas en las que la generación y el análisis de datos continuos es de gran importancia, como menciona [Beringer, 2003], en sistemas multisensores, como el monitoreo de redes, los sistemas de telecomunicación, los flujos generados por los clics de los clientes, los mercados de valores, entre otras tareas, pues de esta forma el procesamiento constante de cada dato aporta al sistema la información necesaria para mantenerse actualizado y poder responder generando una respuesta en cualquier instante que le sea requerido.

Algunos trabajos relacionados al aprendizaje *on-line* son [Ortega, 2010] para la comprensión del habla a través de la comparación de unidades semánticas que representan frases de un corpus no alineado, implementado para la comprensión de un dialogo hablado. Otro trabajo relacionado puede ser consultado en [Schmitt, 2008] donde se plantean la aplicación del algoritmo FLORA-MC para servicios que se adapten de acuerdo a la información que se colecta en tiempo real de diferentes sensores, para adaptar los servicios en tiempo real de la informática ambiental y computación ubicua. Mairal [Mairal,2009] propone un algoritmo para el problema de aprendizaje de diccionarios. Hoi [Hoy, 2013] propone la investigación de un nuevo problema denominado *on-line* Multiple Kernel Classification(OMKC). El aprendizaje *on-line* es aplicado en [Santibáñez, 2013] junto con la administración de RAM. Mientras que en el trabajo de Kidera [Kidera, 2006] se expone un ensemble de clasificadores incremental, basado en una extensión del algoritmo AdaBoost.M1.Siendo estos sólo algunos de los trabajos recientes que se han desarrollado con el enfoque *on-line*.

Muchos de los datos mencionados que se generan en tiempo real, carecen de cualquier descripción que indique de forma evidente su clasificación, y siendo que la generación de datos tiende a ser cada vez mayor y más rápida, se destaca la necesidad de contar con un método que tome en cuenta estos aspectos, tanto de las características de los datos como del entorno en el que deben ser procesados. Es por ello que se adopta el aprendizaje *on-line* como una propuesta para el procesamiento de los datos conforme llegan, para dotar a un sistema de la capacidad de dar una respuesta en todo momento, tomando en cuenta el proceso desde la llegada del patrón hasta su clasificación. Por lo que en esta investigación se destacan tres elementos principales, el aprendizaje *on-line,* el *clustering*, y la optimización de RAM.

1.1 Objetivo

Con base a lo expuesto anteriormente, el objetivo principal de esta Tesis es desarrollar una metodología que contemple la incorporación de nuevos casos *on-line* durante el proceso de aprendizaje, de tal modo que según los recursos disponibles de memoria RAM el aprendizaje se realice de forma dinámica. En específico, la investigación persigue:

- ∂ Manejo dinámico de memoria RAM. Determinar la cantidad de patrones que pueden ser leídos en la RAM del equipo y con base en ello, construir la matriz que los contenga.
- ∂ Adquisición de datos *on-line.* Aplicar un algoritmo heurístico incremental en donde utilizando principios de no supervisado y radio de vecindad, crear los grupos existentes en los datos leídos.
- ∂ Análisis de *cluster.* Garantizar la calidad de los grupos creados mediante la aplicación de índices de validación.

- ∂ Integración del clasificador. Establecer una metodología que integre de forma eficiente los diferentes métodos, proporcionando un rendimiento igual o superior al observado con tratamiento de datos *off-line.*

1.2 Esquema de contenido

Esta Tesis está estructurada en tres partes, en la primera parte se exponen las razones y los objetivos del desarrollo de la investigación. La parte dos está integrada por la fundamentación teórica y consta de tres capítulos, en los cuales se expone de forma general el funcionamiento de la memoria en el lenguaje de programación Java, y plantea algunas alternativas para mejorar el rendimiento de una aplicación, así como describir la mejor forma de utilizar estas alternativas en el sistema que cubre la metodología propuesta. La parte tres, consta de dos capítulos integrados por la descripción de la estrategia de solución y el desarrollo experimental. Por último se incluye una sección de conclusiones y líneas abiertas de estudio.

De igual modo, se incluye una sección de apéndices, donde pueden encontrarse los resultados experimentales a detalle.

PARTE 2

FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 2 RECONOCIMIENTO DE PATRONES

Dentro de la Inteligencia Artificial (IA) se encuentran varias áreas, una de ellas, el Reconocimiento de Patrones (RP), que se apoya de un conjunto de disciplinas como el aprendizaje de máquina, la estadística, la minería de datos y la neuro-computación [Méndez, 2006] (Figura 1). El RP se basa en el proceso natural ejecutado por el hombre, ya que en el mundo real el ser humano es capaz de reconocer objetos, aún en sus formas más complejas.

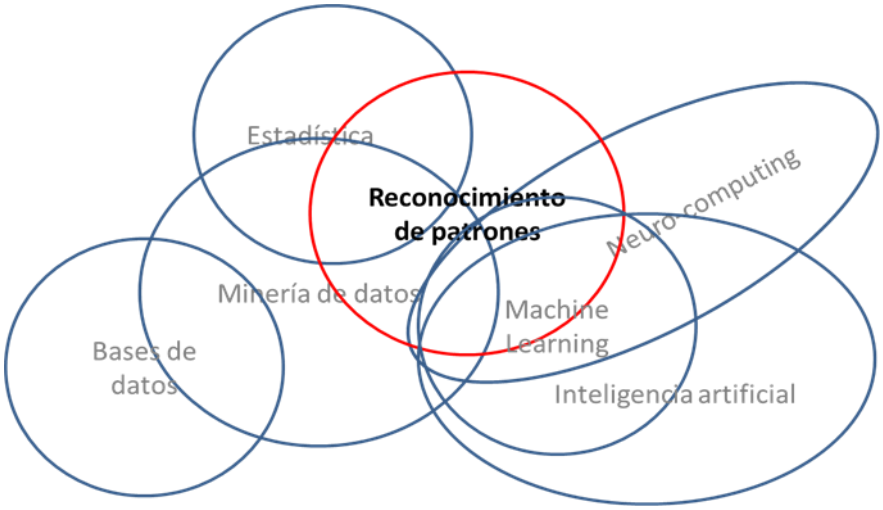


Figura 1. Disciplinas que intervienen en el reconocimiento de patrones. Extraído de [Méndez, 2006].

El proceso biológico del reconocimiento de patrones, proporciona al ser humano la capacidad de distinguir entre diversos objetos y ubicarlos en algún rubro conocido. El reconocimiento inicia con un estímulo, que el cerebro debe verificar con la información conocida que se tiene en memoria y determinar si las características corresponden o no con esta información para identificar el objeto (Braidot, 2008]. El mismo autor menciona tres enfoques del reconocimiento de patrones, que son:

1. Modelo perceptivo por igualación: en el que recibido un estímulo se procede a igualarlo con algún modelo que sea exactamente igual al que tenemos en memoria.
2. Prototipos: en este caso no se compara con características de un objeto específico, si no con aquellas que comparten la mayoría de los mismos.
3. Análisis de rasgos: se obtiene la información a través de atributos específicos, llamados rasgos, y se comparan con la información almacenada previamente, para verificar similitudes.

Por lo tanto, un ser humano es capaz de reconocer a un conejo de una liebre de acuerdo a sus diferentes atributos, como son: el tamaño, el largo de las orejas, la longitud y forma de las patas, como puede observarse en la Figura 2. De acuerdo a los enfoques del reconocimiento de patrones en el ser humano, pueden determinarse maneras en las que éste reconoce patrones de forma más especializada, como por ejemplo rostros humanos a través de rasgos específicos, o clasificar un objeto desconocido de acuerdo a su parecido con una clase de objetos conocidos.

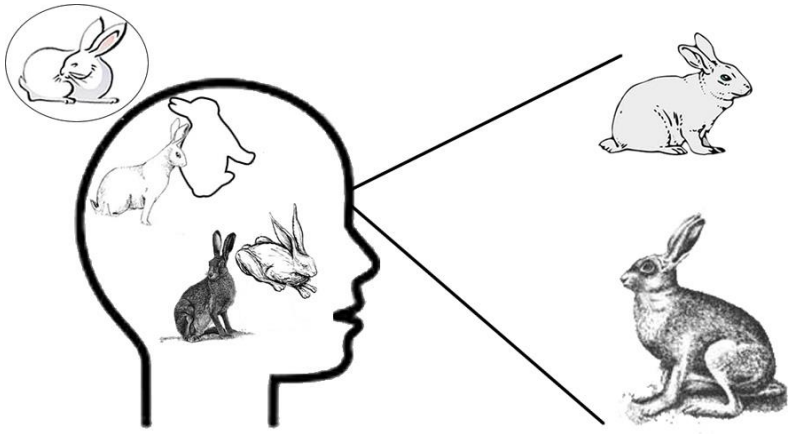


Figura 2. Proceso del ser humano en el reconocimiento de patrones.

Como se puede observar, el reconocimiento de patrones biológico responde a estímulos o percepciones, y es un proceso de categorización por medio del cual identificamos objetos. En computación el RP cumple un objetivo principal, emular dicho proceso biológico de forma automática y con la menor intervención humana posible mediante el uso de algoritmos de clasificación o agrupamiento, que permitan identificar objetos a través de sus características más representativas, por lo que de acuerdo a [Salcedo, 2007] un sistema de RP es más que un clasificador, ya que para realizar esta tarea el sistema debe tomar una decisión.

2.1 Enfoques del RP

El reconocimiento de patrones puede ser aplicado en diferentes áreas y problemas, por lo que se reconocen diferentes enfoques, de los cuales no existe, en la literatura, un consenso sobre su clasificación. En este trabajo se menciona la organización de los enfoques más recurridos y aunque no se ha consensado una clasificación de dichos enfoques, existen cuatro de ellos en los que, después de revisar la bibliografía recabada, coinciden los autores, estos son: el enfoque estadístico, estructural, de redes neuronales y *clustering*, Por lo tanto se propone una clasificación basada en la revisión bibliográfica.

Partiendo de la clasificación que hace [Marques de Sá, 2001], los diferentes enfoques pueden derivarse conforme al modo en que se mide la similitud entre patro-

nes, la tarea de que se realiza, ya sea clasificación o regresión y el método de aprendizaje que es aplicado. Teniendo en cuenta lo anterior, las formas de medir la similitud que pueden ser aplicadas son [Marques de Sá, 2001]:

- Distancia. A través de una fórmula matemática, como la distancia Euclidea o Mahalanobis, por mencionar algunas, se calcula que tan cercano es un patrón de otro a través de sus vectores de características.
- Ponderación (Estructura primitiva). Es la asignación de un valor, ya sea estático o dinámico, sobre un elemento del clasificador.
- Regla sintáctica (Estructura primitiva). Establece una regla compuesta por estructuras, generalmente cadenas, que describen el comportamiento de los patrones.

Así como existen diversas formas de medir la similitud, también existen tres tareas diferentes que el sistema puede desarrollar, la clasificación, la regresión y la predicción. La primera de ellas corresponde a la acción de etiquetar un patrón y asignarlo a alguna de las clases existentes. La regresión está ampliamente ligada a la inferencia, permite establecer una relación entre los patrones y realizar una predicción. Estas dos tareas se basan en la medición, por lo que son evaluadas de forma numérica, a diferencia del último enfoque, la descripción, que utiliza la estructura de los objetos para determinar la similitud entre estos por medio de reglas estructurales donde dos objetos son similares si cubren la misma regla estructural. Por el enfoque de esta Tesis en adelante se abordara la tarea de clasificación.

En base a los tres niveles establecidos por [Marques de Sá, 2001], y al método de aprendizaje que puede ser utilizado (ver Capitulo CAPÍTULO 3), pueden establecerse ahora los diferentes enfoques, que se listan a continuación, y que se muestran en la Figura 3.

2.1.1 Enfoque estadístico

Por sus siglas en inglés, (SC, *Statistic Clasification*), se basa en métodos y fórmulas de estadística con una base matemática sólida. Aplica modelos probabilísticos para determinar las funciones de distribución y clasificación del vector de patrones con respecto a alguna de las clases. Este enfoque utiliza el método de aprendizaje supervisado, pues parte de una muestra de entrenamiento, de la cual se conoce su clasificación. Dentro de esta clasificación pueden encontrarse los métodos paramétricos y no paramétricos [Hernandez, 2004] y de acuerdo a [Alppaydin, 2010] se puede ubicar también el método semi-paramétrico.

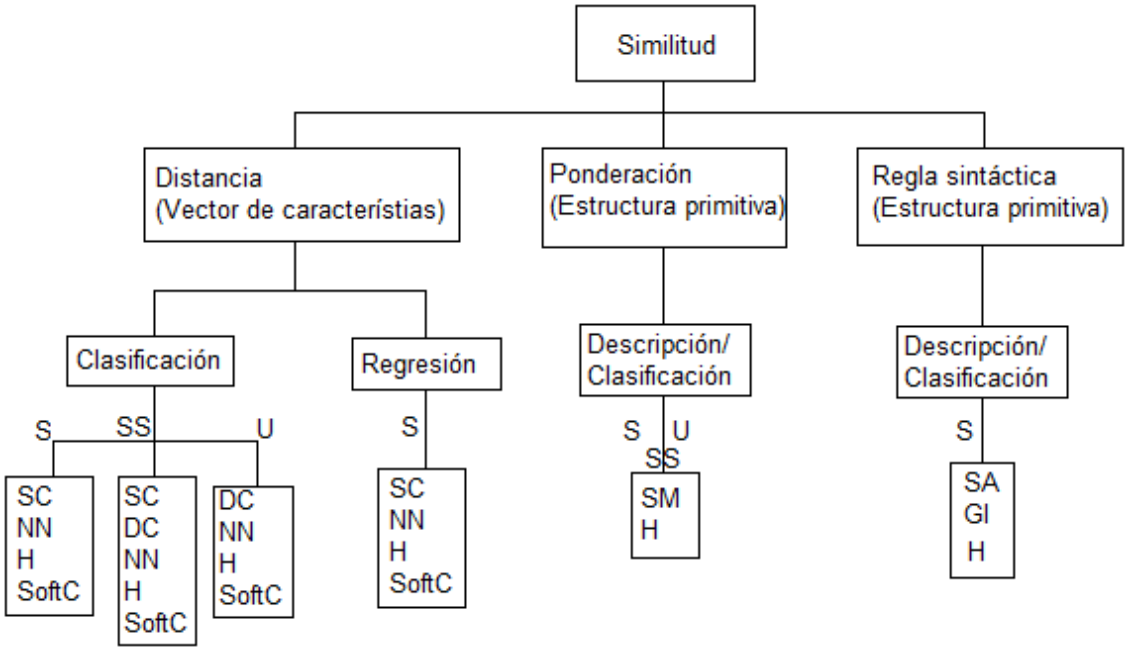


Figura 3. Enfoques del RP; S-supervisado, U-no supervisado, SS-Semi-Supervisado, SC- clasificación estadística, NN-redes neuronales, DC-agrupamiento de datos, SM-adaptación estructural, SA-análisis sintáctico, GI-inferencia gramática, H-Hibrido, SoftC-Soft Computing. Extraído de [Marques de Sá, 2001].

El primer método asume una distribución conocida para todo el modelo, que se define a partir de una cantidad pequeña y finita de parámetros, estimados en base a una muestra dada, para conocer toda la función de distribución [Bishop, 2006]. Entre las principales ventajas de este tipo de métodos es posible mencionar que la función puede ajustarse cuando se trabaja con una cantidad pequeña de datos, y que los valores de los parámetros muchas veces son intuitivos; aunque asumir una distribución particular no siempre ajusta con todos los conjuntos de datos, lo que genera un modelo poco flexible [Hernández, 2004].

Los métodos no paramétricos, son conocidos también como *instance-based* o *memory-based learning*, porque almacenan en memoria los ejemplos de entrenamiento, no cuentan con información a priori, por lo que crean modelos locales basado en las instancias pasadas para obtener una salida que sea similar a estas [Alppaydin, 2010]. Por lo tanto estos métodos, también pueden tener parámetros desconocidos, pero a diferencia de los paramétricos estos no controlan la forma de la distribución sino la complejidad del algoritmo [Bishop, 2006], entonces la forma de la distribución se ajusta completamente a los patrones encontrados en los datos [Hernández, 2010]. La desventaja más significativa de estos métodos es la necesidad de memoria y la cantidad de cálculos requeridos [Alppaydin, 2010].

Por último los métodos semi-paramétricos que describe [Alpayddin, 2010] permiten realizar la combinación de las distintas distribuciones a partir de las que se asumen del modelo paramétrico asignado a cada grupo o clase.

2.1.2 Enfoque de Redes Neuronales

Las redes neuronales artificiales, por sus siglas en inglés (NN, *Neural Network*), se fundamentan en el modelo biológico de las redes neuronales, las cuales se encuentran interconectadas y envían señales en ambos sentidos. Este enfoque puede trabajar con el aprendizaje de forma supervisada y no supervisada, a través de una interconexión de neuronas, a las cuales se les asigna un peso sináptico que indica la intensidad de cada entrada, para obtener, después del proceso de los datos, sólo una salida [Marín, 2008].

Sobre las NN se puede decir que donde reside el conocimiento de la red es en los pesos, que están asignados a cada una de las conexiones que existen entre las neuronas, ya que estos valores son los que se modifican en cada una de las evaluaciones de la red a un patrón, siendo éste el proceso de aprendizaje de la red. Existen diversos modelos que pueden ser implementados, el más sencillo de ellos, y con el que iniciaron las NN, es el perceptron, que consta sólo de una capa de entrada y una de salida, pensado para la clasificación de dos clases linealmente separables, es decir, implementa una función lineal [Dunne, 2007].

Atendiendo a las deficiencias del perceptron, surgió el perceptron multicapa, con el que es posible la clasificación entre más de dos clases, aunque su costo computacional es mayor, ya que el número de neuronas de entrada y de salida dependen totalmente del número total de patrones y el número de clases existentes en el CE. A partir de este tipo de redes surgen diversas modificaciones que pueden ser implementadas de acuerdo al problema que se trate [Theodoridis, 2006].

2.1.3 Enfoque de agrupamiento de datos

Por sus siglas en inglés, (DC, *Data Clustering*). También conocido como aprendizaje no supervisado. Consiste en agrupar los datos de acuerdo a la similitud que existe entre ellos. Este tipo de enfoque utiliza aprendizaje no supervisado, pues no requiere de información previa sobre las clases [Marques de Sá, 2001].

Algunos autores no incluyen al *clustering* como un enfoque del RP, aunque el consenso entre la bibliografía recabada, indica que puede ser reconocido de ambas formas, como un método de aprendizaje y como un enfoque del reconocimiento de patrones.

Con este enfoque implica trabajar con un conjunto de datos no etiquetado y que carece de toda descripción, a partir del cual deben formarse un número desconocido de grupos que integren a los patrones en conjuntos, cuyas características sean compartidas por todos. Esto se lleva a cabo mediante algoritmos que a través de medidas de similitud se encargan, ya sea de crear grupos a partir de un patrón, o de encontrar grupos más pequeños dentro de alguno existente, aunque no son los únicos métodos, a estos algoritmos se les conoce como algoritmos de partición y jerárquicos, respectivamente.

La mayoría de los algoritmos de *clustering* requiere de parámetros que indiquen principalmente la distancia entre los grupos y el grado de confianza que hay dentro de estos para determinar que un patrón pertenece al grupo. La asignación de estos parámetros no es una tarea trivial, pues de ello depende que tan bien se comporte el algoritmo al momento de formar los grupos. Por lo tanto, encontrar los parámetros óptimos, no sólo para estos algoritmos, si no para cualquiera, es una de las tareas cruciales implícitas en toda experimentación de RP.

2.1.4 Enfoque Estructural o Sintáctico

Abarca la teoría de lenguajes formales [Pal, 2001], dentro de este podemos encontrar la Correlación Estructural (SM por sus siglas en inglés, *Structural Matching*), que toma una parte de la estructura de un patrón previamente seleccionado, llamado prototipo y determinar cuando la estructura de otro patrón coincide con dicho prototipo.

Normalmente el prototipo del SM se realiza tomando las partes más representativas del patrón, codificándolas generalmente en cadenas que describen el comportamiento del patrón a través de letras, que se formalizan como reglas, y que son conocidas como reglas sintácticas o gramaticales. Por lo tanto, el objetivo es identificar cuando la secuencia de un nuevo patrón obedece a alguna de las reglas establecidas [Marques de Sá, 2001].

Dentro de este enfoque pueden encontrarse dos tipos particulares de RP, el Análisis Sintáctico (SA por sus siglas en inglés, *Sintactical Analisis*), que “relaciona la estructura de los patrones con la sintaxis de un lenguaje definido formalmente” y la inferencia gramatical (GI por sus siglas en inglés, *Grammatical Inference*), para “deducir de forma automática una gramática a partir de un conjunto de patrones muestra” [Sánchez, 2001].

2.1.5 Enfoque de Soft Computing

Dentro de este se encuentran los Conjuntos Borrosos o *Fuzzy set theory*, cuya integración con las redes neuronales y los algoritmos genéticos fueron el fundamento de este enfoque. La teoría de Conjuntos Borrosos permite modelar la incertidumbre e imprecisión de los conjuntos [Pal, 2001], y consiste principalmente en no tomar una decisión determinante sobre la clasificación de un patrón, ya que existe solapamiento de los *clusters* o clases, los cuales no son disjuntos y es necesario generar una asignación más flexible. El fin de este enfoque es brindar mayor flexibilidad que permita trabajar con situaciones reales, en las que las condiciones no son las ideales. La combinación de este tipo de metodologías da como resultado sistemas híbridos. Por lo tanto el siguiente enfoque parte de la idea de combinar los métodos existentes.

2.1.6 Enfoque hibrido

Propone la combinación de técnicas de RP para "construir un sistema con capacidades de clasificación mejoradas", que compense las debilidades de cada clasificador entre si y aproveche sus fortaleza individuales [Bunke, 2002]. Este tipo de enfoque surge con la necesidad de encontrar un clasificador que de forma general pueda clasificar de manera óptima cualquier muestra que se le presente, pero a lo largo de la búsqueda por el desarrollo, optimización y mejora de los algoritmos de forma particular, por razones propias del diseño de los algoritmo y de los conjuntos de datos esto es poco probable de alcanzar; es entonces cuando se propone la unión de diversos algoritmos que puedan complementarse unos a otros, y generen diversidad en las respuestas, apuntando hacia un sistema de clasificación más preciso [Alppaydin, 2010].

2.2 Etapas de un Sistema de Reconocimiento de Patrones

Un sistema de RP consta de tres etapas, la adquisición de datos, la parametrización y la clasificación, que se muestran a continuación en la

Figura 4 y se describen posteriormente.

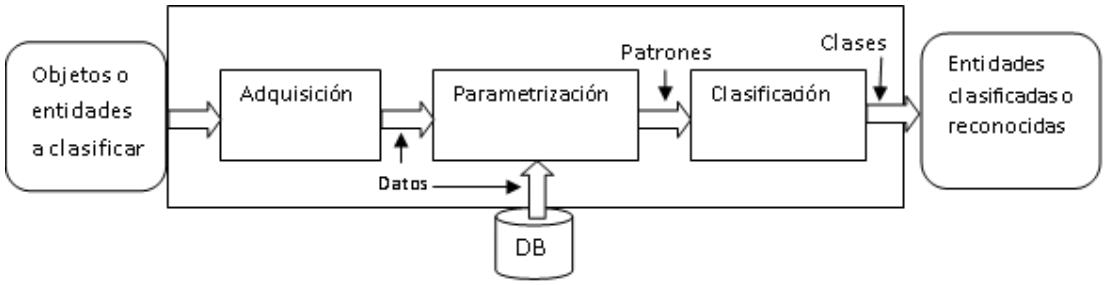


Figura 4. Esquema general de un sistema de RP. Extraído de [Salcedo, 2007].

2.2.1 Adquisición de datos

En esta etapa, se obtienen los datos directamente del objeto, a través de algún dispositivo que capture sus mediciones u otras señales.

Por dato se entiende, de acuerdo a [Taylor, 2004], "la salida de cualquier observación, medición o aparato de grabación". Dichos datos pueden entonces encontrarse en diversos formatos, es por ello que deben ser representados de forma tal que la máquina pueda procesarlos. Por tanto deben tomarse en cuenta los tipos de características que pueden existir. En la Figura 5 se muestra el esquema de dichas características.

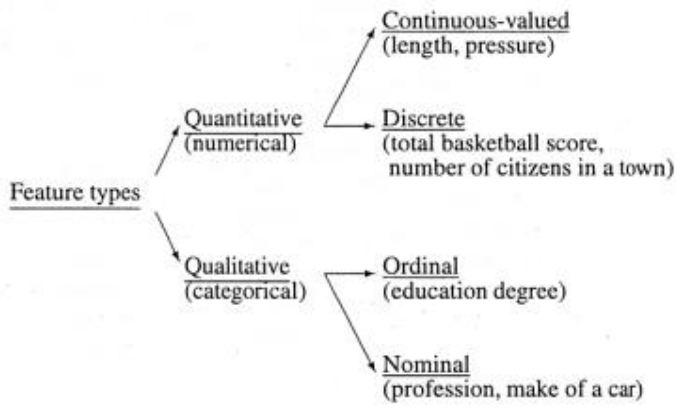


Figura 5. Tipos de características. Extraído de [Kuncheva, 2004].

La primera clasificación de los tipos de datos se da en dos ramas, los datos cualitativos y los datos cuantitativos, los primeros son aquellos que pueden tomar una cantidad reducida de valores, y los cuantitativos son datos con una posibilidad grande de valores. Dentro de los datos cuantitativos se encuentran los de valor continuo, que pueden tomar un cualquier valor dentro de un rango específico, y los discretos, que sólo pueden tomar valores determinados, como el total de estudiantes.

Los atributos cualitativos o categóricos se clasifican en ordinales, en los que existe un orden natural, y nominales, son atributos que no guardan ninguna relación estrecha entre sí.

Existen diversas técnicas para representar los datos de forma tal que la máquina pueda procesarlos, dichos métodos se conocen como normalización, algunos ejemplos de estás son: discretización de igual anchura, discretización de igual frecuencia, discretización difusa, discretización iterativa, entre otros [Yang, 2002].

2.2.2 Parametrización

En esta etapa, se extraen, a través de algoritmos, las características más relevantes del objeto, a partir de los datos de entrada. Éstos pueden estar contenidos en una Base de Datos (BD) y no ser propiamente de la etapa de adquisición. Esta fase corresponde a la extracción de datos, a través de técnicas que permiten obtener las características más relevantes del patrón, para generar una muestra representativa del conjunto de patrones y estandarizar los datos a través de algoritmos diseñados para tal fin.

La clasificación se desarrolla sobre un conjunto de datos, que la máquina debe ser capaz de comprender, por lo que dicho conjunto debe ser procesado previamente para poder ser entendido por la computadora. Este conjunto está formado por los datos obtenidos a partir de la observación, medición o grabación, y cada una de las características obtenidas son conocidas como atributos.

El preprocesado, también llamado extracción de características, es una tarea que permite mejorar el conjunto de datos, ya que en muchas ocasiones, los datos de entrada no presentan la calidad necesaria para que la clasificación sea óptima, por lo tanto es un paso importante transformar el espacio de entrada [Bishop, 2006]. Algunos de los principales inconvenientes que pueden presentarse en el conjunto de datos son: patrones ruidosos o atípicos, omisiones, atributos o patrones que son poco discriminantes, conocido como solapamiento de clases, otro factor es el desbalance del CE, así como la extensión del conjunto de datos, que en ocasiones supera los recursos de memoria disponibles [Valdovinos, 2006].

2.2.2.1 Problemas del conjunto de datos

Diferentes aspectos inherentes al conjunto de datos podrían afectar el resultado y desempeño del clasificador, este tipo de problemas han sido conocidos como complejidad de datos [Alejo, 2008] En los siguientes apartados se explican las más ampliamente estudiadas.

a) Patrones atípicos o ruidosos

Dentro del conjunto de datos cabe la posibilidad de que existan vectores que corrompen los datos, a esto se le conoce como *ruido,* y dos tipos de ruido pueden ser definidos, de acuerdo a [Nielson, 1998], el ruido en los atributos que altera los valores y el ruido de clase que altera el valor de la función.

En sí los atípico son patrones que se encuentran muy lejos de su media, es decir, que a pesar de formar parte de una clase, sus características son diferentes a las

del resto de los patrones que pertenecen a dicha clase, y puede guardar parecido con los patrones de otras clases, por lo que confunde al clasificador. Este tipo de patrones puede ser generado por errores de procesamiento, captura, etiquetado o porque pertenece a una clase que no está representada en el CE [Valdovinos, 2006].

Si el número de patrones ruidosos o atípicos es muy bajo, normalmente son descartados, en caso contrario debe adoptarse una función que sea sensible a la detección de este tipo de patrones [Theodoridis, 2006].

b) Solapamiento entre clases

Este se da cuando en un conjunto de clases $C = \{C_1, \dots, C_n\}$ existen patrones en los que al menos dos clases comparten algunas características, lo cual hace que la distinción de las clases sea menos sencilla. Un ejemplo puede verse en la Figura 6, donde se observa el solapamiento en el espacio de proyección [Valdovinos, 2006].

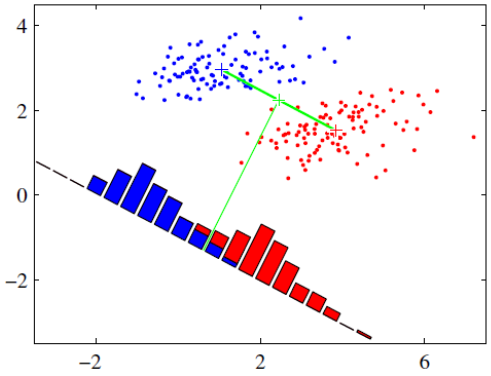


Figura 6. Solapamiento de clases

c) Desbalance de clases

Este se refiere a la presencia de un mayor número de patrones pertenecientes a la clase C_i, siendo esta la clase mayoritaria y el resto clases minoritarias, por lo que la clase C_i tiene mayor representación, lo que provoca que al entrenar el algoritmo este reconozca de forma óptima los patrones pertenecientes a esta clase, mientras que los patrones de las clases minoritarias son mal clasificados o no reconocidos.

Cuando el CE cuenta con más de dos clases es difícil saber cuáles de ellas serán detectadas como minoritarias y cuáles como mayoritarias, por lo que se han dedi-

cado investigaciones a revisar cómo afecta este comportamiento al clasificador, siendo comprobado que esto puede deteriorar en grado importante la precisión del clasificador. En base a lo anterior, se han desarrollado técnicas que permitan balancear las clases, la primera de ellas es el *under-sampling* que elimina patrones de la clase mayoritaria, ya sea de forma aleatoria o seleccionando aquellos patrones que pudieran deteriorar la clasificación y el *over-sampling* que replica patrones en la clase minoritaria, conocidos como patrones *sintéticos*. Por último, internamente predisponer el proceso de discriminación para compensar el desequilibrio del CE [Alejo, 2008].

Es común observar este tipo de comportamiento en problemas reales en los que es importante que el clasificador reconozca aquellos eventos de la clase minoritaria, como por ejemplo: en diagnósticos médicos, la detección de fraude en las tarjetas de crédito, clasificación de textos, entre otros [Valdovinos, 2006].

d) Tamaño de la muestra

En esta época, existen bases de datos de un tamaño que puede llegar hasta los *petabyets*, pues el avance tecnológico ha permitido que la generación de datos en ambientes comerciales y científicos, en los que se maneja una gran cantidad de información, sea posible de obtener y almacenar. Por lo tanto se requiere de medios que cuenten con la capacidad para guardar toda esta información, así como su análisis, el primer caso ha sido resuelto en gran medida, mientras que el segundo se encuentra aún en estudio, pues el procesamiento de tal cantidad de información requiere de una gran capacidad de recursos computacionales.

En respuesta a lo anterior, se han generado alternativas para el análisis de grandes cantidades de datos, una de ellas es a través de la *escalabilidad de algoritmos*, en los que se propone el uso de metodologías que permitan tratar de forma eficiente este tipo de conjuntos a través de algoritmos existentes. Los métodos de reducción del tamaño del conjunto de datos están diseñados para realizar el preprocesado de los datos, con lo que se pretende obtener un CE de menor tamaño que pueda ser procesado con los recursos disponibles. Esto puede lograrse ya sea mediante la reducción del CE a través de la eliminación de patrones atípicos o a través de un subconjunto representativo del CE que no considera a los patrones que son poco útiles o redundantes [Valdovinos, 2006].

2.2.3 Clasificación

Esta etapa va de la mano con el entrenamiento, en el cual se enseña al clasificador a reconocer los patrones. Posterior al entrenamiento se realiza la clasificación, asignando la etiqueta correspondiente a una clase al patrón que está siendo identificado.

A través del uso de un clasificador se puede asignar un patrón a una clase determinada, es decir se pueden agrupar o clasificar los patrones de acuerdo al enfoque que se esté utilizando. Para este proceso se consideran diversos métodos. En algunos, dependiendo del método de aprendizaje que se esté implementando el clasificador es entrenado con la muestra de entrenamiento, por medio de este proceso el clasificador puede identificar más tarde un nuevo Patrón y asignarlo a la clase correspondiente, generalizando la función de decisión. Posteriormente se tratará con mayor detenimiento esta etapa en el enfoque no supervisado.

Como se mencionó, la clasificación de los objetos se lleva a cabo en base a las características más significativas de éste, dicho conjunto de atributos se denomina *patrón*, que de manera formal, se representa a través de un vector de *N* variables aleatorias.

$$p = \{x_1, x_2, x_3, \dots, x_N\} \qquad (1)$$

Cada patrón representa un punto en el espacio de representación de los patrones *P*, *d*-dimensional, donde *d* es el número de variables consideradas y *P* es el producto cartesiano de los conjuntos de valores X_i. Así se tiene:

$$p \in P = \{X_1\} x \{X_2\} x \{X_3\} x \dots x \{X_i\} \qquad (2)$$

donde X_i es el conjunto de valores que puede tomar la variable aleatoria de la característica *i* [Salcedo, 2007].

Por lo general, aquellos patrones que comparten características en común, se encuentran agrupados o separados en una misma región de decisión (Figura 7). Dependiendo de la forma como se entrena el clasificador, a estas regiones se le conoce como clase (supervisado) o grupo (no supervisado). De este modo, un patrón *p* pertenece a alguna de *M* clases/grupos ω, por lo que formalmente se define el conjunto de las clases/grupos como $\Omega = \{\omega_1, \omega_2, \omega_3, \dots, \omega_M\}$. En su totalidad, el conjunto de clases o grupos forman lo que se conoce como Conjunto de Entrenamiento o Muestra de Entrenamiento (ME), matemáticamente la asociación de un patrón a una de las regiones de decisión se puede definir como el conjunto *T*, conformado por parejas (X_i, c_i), donde $X_i \in P$ y $c_i \in \Omega$ [Juárez, 2003].

$$T = \{(X_1, c_1), (X_2, c_2), \dots, (X_N, c_N)\} \qquad (3)$$

Cada una de las clases/grupos consta de un límite de clase, que determina el límite de la distribución de los patrones en el espacio *d*-dimensional de la clase la cual tiene asignada una región de decisión. De igual modo, el área que separa estas regiones es conocida como frontera de decisión.

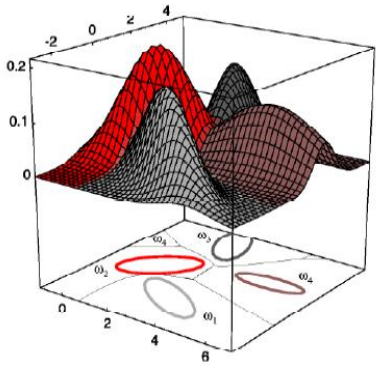


Figura 7. Frontera de decisión. Extraído de [Duda, 2000].

Para ejemplificar un poco más, la división de un espacio *d*-dimensional en 1, 2 y 3 clases se muestra en la Figura 8, donde se puede observar que el número de regiones crece exponencialmente de acuerdo a la dimensionalidad del espacio.

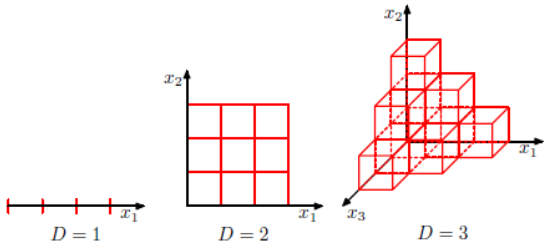


Figura 8. División de la dimensionalidad del espacio. Extraído de [Bishop, 2006].

2.2.3.1 Medidas de similaridad y discimilaridad

Para realizar la división de un espació *d*-dimensional, pueden aplicarse diferentes medidas, que pueden ser clasificadas en tres grandes grupos, de acuerdo a la separabilidad de los datos. El primer grupo corresponde a la separabilidad lineal que introducen una función lineal que se encarga de dividir a los patrones en dos clases o grupos que se diferencian claramente. Tomando en cuenta que en la realidad los datos pueden tener una separabilidad menor y estar integrados por más de dos clases, se introducen otras medidas como la separabilidad geométrica, que ajusta a otro tipo de patrones, que no son lineales y cuya separabilidad depende del número de ejemplos que se tomen.

Por último se introduce la separabilidad geométrica radial, que es una mejora de las anteriores, aplicable para datos de baja separabilidad, pues esta se basa más en las relaciones existentes entre los patrones, que en la distancia geométrica entre ellos, de acuerdo al radio asignado para determinar la relación entre dos patrones, la distancia calculada entre ellos forma parte de la relación [Hernández, 2004].

De acuerdo a lo anterior, las métricas permiten calcular la distancia entre dos puntos, asumiendo que el espacio de representación es un espacio métrico [Vázquez, 2008]. Por tanto, una métrica se define como una función real entre dos puntos en dicho espacio de representación, tal que para todos los puntos *x, y* y *z,* que satisface las siguientes propiedades [Valdovinos, 2006]:

a) $d(x,y) \geq 0$ y $d(x,y) = 0$ si y solo si $x = y$ (positividad)
b) $d(x,y) = d(x,y)$ (simetría)
c) $d(x,y) + d(y,z) \geq d(x,z)$ (desigualdad triangular)

La distancia determina que tan similares son dos objetos, mientras más cercanos la similitud es mayor. Por lo tanto, para determinar cuando dos patrones comparten características parecidas se utilizan medidas de similitud, mientras que las medidas de discimilaridad se utilizan para saber cuándo no son parecidos [Vázquez, 2008].

a) Medidas de discimilaridad para valores reales. Una de las medidas más difundidas es la distancia Euclidea [Theodoridis, 2006]:

$$\delta_E(X,Y) = \sqrt{\sum_{j=1}^{n}(x_j - y_j)^2} \qquad (4)$$

donde n es el número de características.

- Distancia de Manhattan:

$$\delta_M(X,Y) = \sum_{j=1}^{n} |x_j - y_j| \qquad (5)$$

- Distancia de Minkowski:

$$\delta_{Mk}(X,Y) = \sqrt[q]{\sum_{j=1}^{n}(x_j - y_j)^q} \text{ con } q > 0 \qquad (6)$$

Estas tres distancias son variantes de la distancia de Minkowski, Mientras *q* sea más grande la diferencia entre las variables será más marcada, cada una de las variantes implementa a q = 1,2 e ∞, respectivamente [Valdovinos, 2006]. Una característica de estas métricas es que son invariante a traslaciones y rotaciones, pero ante transformaciones lineales o variaciones de escala muestran un comportamiento irregular, por lo tanto se recomienda estandarizar los datos de forma que las medidas de las variables sean compatibles.

- Una de las distancias que es invariante a las transformaciones lineales y cambios de escala es la distancia de Mahalanobis, esta toma en cuenta la interdependencia de las características.

$$\delta_{Mh} = \sqrt{(\eta - \mu)^T \sum^{-1} (\eta - \mu)} \qquad (7)$$

donde:

$\eta \approx \frac{1}{m} \sum_{i=1}^{m} x_i$ Media de los m vectores de entrenamiento

$\Sigma \approx \frac{1}{m} \sum_{i=1}^{m} (x_i - \eta)(x_i - \eta)^T$ Matriz de covarianza de los vectores de entrenamiento x

$\mu \approx \frac{1}{p} \sum_{i=1}^{p} y_i$ Medida de los p vectores de prueba

b) Medidas de discimilaridad para valores discretos.

- Distancia *Hamming:* Cuando se dispone de datos binarios una de las métricas más utilizadas es la distancia de Hamming que busca identificar la semejanza de dos vectores binarios.

$$\delta_H(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (8)$$

Distancia l_1:

$$\delta_{l_1}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \qquad (9)$$

b) Medidas de similitud de valores reales

- La medida más común de discimilaridad es el Producto interno [Theodoridis, 2006]:

$$\delta_{inner}(x, y) = x^T y = \sum_{i=1}^{n} (x_i y_i) \qquad (10)$$

Normalmente el producto interno se utiliza cuando los vectores de entrenamiento x_i y de entrada y_i están normalizados y tienen la misma longitud, y depende del ángulo existente entre *x* y *y*.

- Medida de similitud del coseno. Esta métrica que es invariante a rotaciones, aunque no a transformaciones lineales:

$$\delta_{cosine}(x,y) = \frac{x^T y}{||x||||y||} \qquad (11)$$

donde la longitud de los vectores x y y se representa por:

$$\big||x|\big| = \sqrt{\sum_{i=1}^{n} x^2}$$

$$\big||y|\big| = \sqrt{\sum_{i=1}^{n} y_i^2}$$

- Medida del coeficiente de correlación de *Pearson:*

$$\delta_{Pearson}(x,y) = \frac{x_d^T y_d}{||x_d||||y_d||} \qquad (12)$$

donde x_d y y_d son normalmente conocidos como vectores de diferencia, siendo:

$$x_d = [x_1 - \bar{x}, \dots, x_n - \bar{x}]^T$$

$$y_d = [y_1 - \bar{y}, \dots, y_n - \bar{y}]^T$$

$$y$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Esta medida toma valores entre -1 y +1.

- Métrica *Tanimoto* (distancia *Tanimoto*) [Theodoridis, 2006]:

$$\delta_T(x,y) = \frac{1}{1 + \dfrac{(x-y)^T(x-y)}{x^T y}} \qquad (13)$$

$$Normalizada: \ \delta_T(x,y) = \frac{1}{-1+2\frac{a^2}{x^T}}$$

donde a es la longitud de los vectores

Esta medida puede ser utilizada para valores reales y discretos.

2.3 Métodos de validación y evaluación

Cuando se implementa un algoritmo de *clustering* o de cualquier método de aprendizaje, es necesario saber la eficiencia con la que éste desarrolla su tarea, en cualquiera de sus etapas; para ello existen diversos métodos que pueden ser utilizados para medir su desempeño. A continuación se presentan algunos de los métodos utilizados para la validación de *cluster*, así como para la evaluación del clasificador y de la clasificación.

2.3.1 Validación de *cluster*

Una de las características del *clustering*, es que normalmente no se conoce el número de grupos que serán formados a partir de un conjunto de datos. Para evaluar de una forma cuantitativa los resultados del *clustering*, existen métodos de validación de *cluster*, que pueden ayudar a saber si la asignación de parámetros y las restricciones establecidas para las formas han sido bien establecidas.

En general la validación de *cluster* puede verse desde tres grandes enfoques de acuerdo a [Thedoridis2006], el *criterio externo*, el *criterio interno* y el *criterio relativo*. El primer criterio corresponde a la validación a través de las estructuras independientes, es decir refleja la intuición que se tenía de ante mano sobre la estructura de los grupos, así como la aleatoriedad de los mismos.

El criterio interno, también conocido como índice de validación de *cluster* (*cluster validity index*, CVI), evalúa el *clustering* a través de cantidades obtenidas a partir de los propios vectores, como la matriz de proximidad. Cabe mencionar que los casos en los que puede ser utilizado es cuando la estructura del agrupamiento es jerárquica o de un solo grupo [Arizmendi, 2010]. En consecuencia, la aplicación de las pruebas en base al criterio interno puede revisarse detalladamente en [Theodoridis, 2006].

El último criterio relativo valora el agrupamiento a través de la comparación del resultado de los algoritmos y sus parámetros, por ejemplo puede compararse el resultado que se obtiene de un mismo algoritmo, pero aplicando diferentes pará-

metros, o puede compararse el resultado que se obtiene de aplicar un mismo conjunto de datos a diferentes algoritmos de *clustering* [Theodoridis, 2006].

De acuerdo a [Arbelaitz, 2010] no existe un criterio que asegure al cien por ciento la calidad de la agrupación, ya que por la naturaleza de la tarea, no se cuenta con mayor información sobre el resultado que debe obtenerse, y eso hace que la tarea de validar el número de *clusters* óptimo y su calidad sea más complejo, pues menciona que el criterio externo asume que se tiene conocimiento sobre la partición correcta (*ground truth* o *standard gold*) que debe obtenerse como resultado, cuestión que es poco probable en tareas reales, pero que puede llevarse a cabo en ambientes experimentales.

Por su lado la validación interna utiliza los datos propios de la partición, es decir evalúa su calidad de agrupamiento en base a los datos que la conforman y sus distancias internas, no haciendo necesario ningún conocimiento previo sobre la partición correcta que debe obtenerse, por lo que es más factible de implementar en tareas reales.

Lo antes mencionado conlleva a que los grupos sean evaluados sobre los mismos datos con los que se han construido, lo que provoca, de acuerdo a [Arbelaitz, 2010] que esto sea “equivalente a evaluar los *clusters* bajo una diferente función objetivo”, y menciona como ejemplo la comparación de la calidad de una partición con *k-means* y con *single-linkage*, donde el cálculo del error cuadrático medio beneficia a *k-means*. De acuerdo a esto [Arbelaitz, 2010] considera los métodos de validación desde un punto de vista relativo, es decir, de acuerdo a la comparación de las particiones y su clasificación de acuerdo a su calidad.

Como se observa, existen diferentes enfoques y métodos que pueden ser empleados para la validación de los agrupamientos, por la importancia que tienen en el enfoque de esta Tesis se menciona la validación enfocada a la partición, tanto para la validación interna como externa. Los criterios de validación externos se basan en la *Prueba de Hipótesis en Validación de Cluster*, y pueden utilizarse tanto en el criterio externo como en el interno. Esta prueba se basa, a su vez, en el test de hipótesis, pero en este caso la hipótesis nula está orientada a descubrir la aleatoriedad de la estructura del conjunto de datos.

2.3.2 Criterio externo

Prueba de Hipótesis en Validación de Cluster

Esta prueba tiene dos objetivos: generar un conjunto de datos que modele una estructura aleatoria, de acuerdo a [Theodoridis, 2006] y elegir una estadística que

permita encontrar los valores que indican la estructura del conjunto, para comparar ambos valores.

a) Recuento de pares

En este caso se puede partir de dos enfoques, de la jerarquía establecida para un *cluster*, como para un *cluster* especifico, es decir creado a partir de un algoritmo. Por la orientación que tiene esta investigación, se mencionarán los métodos del segundo enfoque. Por lo tanto, la técnica consiste en generar un conjunto *P* de *m* grupos $P = \{C'_1, \dots, C'_K\},\}$ y los *clusters* creados por el algoritmo, denotados por $C = \{C_1, \dots, C_m\}$, obsérvese que no necesariamente *C* y *P* deben tener el mismo número de grupos.

Se compara la matriz de contingencia de las dos particiones, donde las filas representan un *cluster* $X = \{C_1, \dots, C_K\}$ y las columnas el otro *cluster* $P = \{C'_1, \dots, C'_K\}$, por lo que cada celda contiene el número de patrones que ambos *clusters* tienen en común (Figura 9).

	C_1	C_2	$\cdots$	C_K	
C'_1	n_{11}	n_{12}	$\cdots$	n_{1K}	$n_{1.}$
C'_2	n_{21}	n_{22}	$\cdots$	n_{2K}	$n_{2.}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$C'_{K'}$	$n_{K'1}$	$n_{K'2}$	$\cdots$	$n_{K'K}$	$n_{K'.}$
	$n_{.1}$	$n_{.2}$	$\cdots$	$n_{.K}$	$n_{..} = N$

Figura 9. Matriz de contingencia para el recuento de pares. Extraído de [Arbelaitz, 2010]

Una vez que se cuenta con estos valores, se realiza la comparación del resultado de ambos conjuntos, se trata de generar índices estadísticos que apliquen al test de hipótesis. Para ello se toman dos vectores del conjunto de datos, y se compara si ambos se encuentran en el mismo *cluster C* y en el mismo conjunto *P*, si es así se identifica al valor con *SS*. Si los dos vectores seleccionados se encuentran en el mismo *cluster C*, pero en diferente grupo *P*, entonces corresponde al índice *SD*. Si ambos vectores se encuentran en diferente *cluster C* y diferente grupo *P* entonces se debe incrementar el índice *DD*. Por último, si los vectores se encuentran en diferente *cluster*, y en el mismo grupo *P* entonces el índice que debe incrementarse es el *DS*. Estos valores se obtienen mediante las siguientes fórmulas:

$$SS = \left(\frac{1}{2}\right)\sum_{i=1}^{k}\sum_{j=1}^{k'} n_{ij}^2 - ({}^{N}/_{2}) \qquad (14)$$

$$SD = \left(\frac{1}{2}\right)\sum_{j=1}^{k\prime} n_j^2 - \left(\frac{1}{2}\right)\sum_{i=1}^{k}\sum_{j=1}^{k\prime} n_{ij}^2 \quad (15)$$

$$DS = \left(\frac{1}{2}\right)\sum_{i=1}^{k\prime} n_i^2 - \left(\frac{1}{2}\right)\sum_{i=1}^{k}\sum_{j=1}^{k\prime} n_{ij}^2 \quad (16)$$

$$DD = \frac{N(N+1)}{2} - \left(\frac{1}{2}\right)\left[\sum_{i=1}^{k} n_i^2 + \sum_{j=1}^{k\prime} n_j^2\right] \quad (17)$$

Donde n_{ij} corresponde al número de instancias que los clusters resultado del algoritmo y originales tienen en común y N corresponde al número de instancias.

Estos valores se aplican en los siguientes índices para obtener el grado de similitud que existe entre ambos conjuntos, entre mayor sea ésta, el resultado será más aceptable. Los primeros dos índices tienen un valor entre 0 y 1, y calculan el total de pares que hay en SS y DD, aunque el coeficiente Jaccard excluye el conjunto DD. A continuación se muestran los índices [Arbelaitz, 2010]:

- Estadística Rand. "Representa la fracción de pares de casos en el mismo estado en ambas particiones" [Ingaramo, 2007], es decir Su valor se encuentra entre 0 y 1, mientras más se acerque a 1 las particiones son más similares [Xuan, 2009].

$$R = \frac{(SS + DD)}{(SS + SD + DS + DD)} \quad (18)$$

- Índice Rand Ajustado (ARI por sus siglas en inglés). Es un ajuste al índice anterior, que toma en cuenta el espacio hypergeométrico. De acuerdo a [Yeung, 2001] en un estudio de previo de Milligan se recomienda el ARI como un buen índice para determinar la similitud que existe entre dos agrupamientos con diferente número de grupos. Surge para solventar las deficiencias de RI, que consisten en incrementar su valor a más de uno conforme la cantidad de grupos aumenta, o que su valor no es constante. Por lo que el resultado de ARI se encuentra entre -1 y 1, siendo cero cuando el índice es igual a su valor esperado [Hubert, 1985].

$$ARI = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \quad (19)$$

Donde $\binom{n}{2}$ es el número de posibles combinaciones de pares.

- Coeficiente Jaccard. Ignora los casos que se han asignado a distintos *clusters* en las dos particiones. Este índice calcula la proporción de patrones asignados correctamente con respecto al número total de aquellos que fueron asignados a algún cluster [Campello, 2007]. Es adecuado cuando existen gran número de *clusters* y este valor puede ser muy alto.

$$J = \frac{SS}{(SS+SD+DS)} \quad (20)$$

- Índice Fowlkes y Mallows. Al igual que el índice anterior ignora los patrones asignados a diferentes grupos en ambas particiones. De acuerdo a [Tantrum, 2004] es la media geométrica de la probabilidad de que las dos instancias seleccionadas de forma aleatoria se encuentren en el mismo *cluster,* así como en el mismo grupo, por lo que el agrupamiento es mejor en tanto el resultado se acerque a la unidad. Aunque la desventaja de este índice es que con un número pequeño de *clusters* su valor suele ser muy alto [Wagner, 2007].

$$FM = \frac{SS}{\sqrt{m_1 m_2}} = \sqrt{\frac{SS}{SS+SD}\frac{SS}{SS+DS}} \quad (21)$$

- Por último la estadística *Γ* de Hubert, mide la correlación existente entre las matrices de proximidad de *C* y *P* [Theodoridis, 2006].

$$\Gamma = \left(\frac{1}{M}\right)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)Y(i,j) \quad (22)$$

b) Otra forma de medir la calidad del agrupamiento es mediante la prueba de Pureza, cuya principal aportación es calcular el valor óptimo de grupos. Se calcula de forma individual mediante [Arizmendi, 2010]:

$$pureza = (C_j) = \frac{1}{|C_j|} max(|C_j|)_{class=1} \quad (23)$$

donde C_j es el tamaño del grupo, $|C_j|_{class} = i$ denota el número de objetos de la clase i que se asignaron al grupo j.

Para calcular la pureza global se calcula la suma de la pureza individual de los grupos, como sigue:

$$pureza = \sum_{j=1}^{k} \frac{|C_j|}{D} pureza\ (C_j) \qquad (24)$$

donde D es el total de observaciones.

c) Teoría de la información.

Otra medida para validar la tarea de agrupamiento es la *Entropía* que permite conocer el grado de desorden que puede existir dentro del grupo, por lo tanto entre mayor sea la entropía menor será la calidad del agrupamiento, en caso contrario la calidad será mejor, pues lo que se busca al crear los grupos es que los patrones internos sean lo más similares entre sí, pero lo más diferentes posibles de aquellos que integran otros grupos.

Para calcular la entropía se define la siguiente ecuación:

$$E(X) = -\sum_{x \in X} p(x)\ log\ p(x) \qquad (25)$$

donde x es una variable aleatoria discreta que expresa el conjunto de posibles valores y p(x) es la función de probabilidad de la variable x, es decir,

$$p(x) = \frac{|X|}{N}$$

Tomando en cuenta que cada patrón es un vector multidimensional $\vec{x} = x_1, \dots, x_n$, para calcular la entropía se obtienen las probabilidades de cada atributo, de tal forma que pueda calcularse la entropía global, que se define en la siguiente ecuación:

$$E(\vec{X}) = E(x_1) + E(x_2) +, \dots, +E(x_n) \qquad (26)$$

donde x es una variable aleatoria discreta que expresa el conjunto de posibles valores que puede tomar X

Dos medidas de similitud entre particiones son: la entropía conjunta y la información mutua. La primera de ellas se calcula de la siguiente forma [Arbelaitz, 2010]:

$$H(P, P') = -\sum_{C_i \in P} \sum_{C'_i \in P'} p(x_i, x'_i)\ log\ p(x_i, x'_i) \qquad (27)$$

$$\text{donde } p(x_i, x'_i) = \frac{|x_i \cap x'_i|}{N}$$

Por su parte la Información mutua, se expresa de la siguiente forma:

$$I(P, P') = \sum_{C_i \in P} \sum_{C'_i \in P'} p(C_i, C'_i) \frac{log\, p(C_i, C'_i)}{p(C_i) p(C'_i)} \quad (28)$$

d) Medida F.

Ayuda a determinar que tanto los grupos resultantes del *clustering* se asemejan a los que pudieran haberse logrado a través de la clasificación manual, por lo tanto requiere de información sobre la clasificación real del agrupamiento [Ingaramo, 2007]. Esta medida combina las medidas de *precisión* y *recall*. Por lo tanto se tiene un conjunto de agrupamientos $C = \{C_1, \dots, C_k\}$ y la clasificación real $C' = \{C'_1, \dots, C'_l\}$, las medidas *precisión* y *recall* se integran como sigue:

$$F_{i,j} = \frac{2}{\frac{1}{prec(i,j)} + \frac{1}{rec(i,j)}} \quad (29)$$

donde $prec(i,j) = |C_j \cap C'_i| / |C_j|$ y $rec(i,j) = |C_j \cap C'_i| / |C'_i|$

Mientras que la medida global de F obedece a la siguiente ecuación:

$$F = \sum_{i=1}^{l} \frac{|C'_i|}{|N|} \cdot \max_{j=1,\dots,k} \{F_{i,j}\} \quad (30)$$

2.3.3 Criterio interno

En las siguientes medidas, para Calinski-Harabasz y Dunn un valor alto denota una mejor partición, mientras que para el resto de las medidas lo hace un valor bajo.

- Calinski-Harabasz (CH). También conocido como Criterio de Porcentaje de Variación (VRC). Evalúa la calidad de los agrupamientos a través del uso de la varianza de los patrones dentro del *cluster* y entre los *clusters* [Vendramin, 2009]. Esta distancia hace uso de los centros [Lewis, 2012]. Su interpretación se obtiene comparando el cálculo resultante del índice variando los parámetros del algoritmo y seleccionando como mejor agrupamiento aquel que tenga el valor más alto, que puede observarse como un pico en

la gráfica resultante. Aunque si los resultados tienen una tendencia lineal, ascendente o descendente entonces no hay una razón para preferir una solución u otra.

$$CH(C) = \frac{(N - |P|)inter_{CH}(P)}{(|P| - 1)intra_{CH}(P)} \quad (31)$$

donde $inter_{CH}(P) = \sum_{C \in P} |C|\, d(C, \bar{X})$ e $intra_{CH}(P) = \sum_{C \in P} \sum_{x \in C} d(x, \bar{C})$

- C-index [Arbelaitz, 2010].

$$CI(C) = \frac{S(C) - S_{min}(C)}{S_{max}(C) - S_{min}(C)} \quad (32)$$

donde $S(P) = \sum_{C \in P} \sum_{x_i, x_j \in C} d(x_i, x_j)$ representa la suma de las distancias de todos los pares (n_w) de casos en el mismo cluster. S_{min} es la suma las n_w menores distancias para todos los patrones. S_{max} es entonces la suma de las n_w mayores.

- Gamma. Es la medida con mejor desempeño, de acuerdo al estudio realizado por Milligan sobre treinta criterios internos [Lewis, 2012].

$$Gamma(C) = \frac{\sum_{C \in P} \sum_{x_i, x_j \in C} dl(x_i, x_j)}{n_w \left(\frac{N(N-1)}{2} - n_w \right)} \quad (33)$$

donde $dl(x_i, x_j)$ es la cantidad de pares de patrones cuya distancia es menor que $d(x_i, x_j)$ y se encuentran en diferentes grupos. El resultado se normaliza mediante el denominador, obteniendo así un resultado entre 0 y 1.

- Davies-Bouldin [Arbelaitz, 2010]. Se basa en la distancia interna de los patrones de un grupo y entre los grupos, es decir que cuantifica la proporción de dispersión [Lewis, 2012].

$$DB(C) = \frac{1}{|P|} \sum_{C_k \in P} \max_{C_l \in P / C_k} \left\{ \frac{S(C_k) + S(C_l)}{d(\bar{C}_k, \bar{C}_l)} \right\} \quad (34)$$

$$\text{donde } S(C) = 1/|C| \sum_{x \in C} d(x, \bar{C})$$

- Índice Dunn [Lewis, 2012]. Sirve para comparar la máxima distancia dentro del *cluster* con la mínima distancia entre *clusters*.

$$Dunn(C) = \frac{inter_{Dunn}(P)}{intra_{Dunn}(P)} \qquad (35)$$

donde $inter_{Dunn}(P) = mín_{C_k \in P}\{min_{C_l \in P/C_k}\{\delta(C_k, C_l)\}\}$

$$\delta(C_k, C_l) = \min_{x_i \in C_k} \left\{ \min_{x_j \in C_l} d(x_i, x_j) \right\}$$

$$intra_{Dunn}(P) = \max_{C \in P} \left\{ \max_{x_i, x_j \in C} d(x_i, x_j) \right\}$$

- Promedio entre y promedio dentro [Lewis, 2012]. Estas medidas calculan la separación entre los grupos y la homogeneidad dentro de los mismos, respectivamente.

$$AVG_B(C, Y) = avg_{x \nsim cy}\, d(x, y) \qquad (36)$$

$$AVG_W(C, Y) = avg_{x \sim cy} d(x, y)$$

donde $x \sim cy$ cuando x y y pertenecen al mismo cluster, de otra forma,
$x \nsim cy$

2.3.4 Evaluación del clasificador

Para poder medir la proporción de aciertos y errores que comete un clasificador, es necesario emplear alguna técnica que nos permita cuantificar su eficiencia mediante los resultados que arroja éste. Para ello se emplean los métodos conocidos como estimadores del error, que calculan la cantidad de errores cometidos en la clasificación [Micheli, 2000]. Para cada uno de estos estimadores de error es necesario contar con un conjunto *R*, que dado un patrón *y* pueda compararse con cada patrón x_i del CE, siendo x_i y *y* las etiquetas de la clase, por tanto se tiene [Valdovinos, 2006]:

$$E(x_i, y) = \begin{cases} 0 \; si \; T(x_i) = T(y) \\ 1 \; si \; T(x_i) \neq T(y) \end{cases} \qquad (37)$$

De esta forma es posible realizar una evaluación más óptima del resultado final, ya que si se evalúa al clasificador con la misma de entrenamiento, este tendrá un sobreajuste, pues aprenderá de forma muy fina sólo los datos conocidos y aquellos que no se encontraban en la muestra de entrenamiento no serán reconocidos por el clasificador. Por tanto, se exponen los métodos principales para evaluar al clasificador [Kuncheva, 2004]. Cabe mencionar que los conjuntos de datos son divididos en dos conjuntos, una parte para entrenamiento y otra para prueba. Normal-

mente se asignar la mayor parte para entrenamiento y el resto para prueba, en minería de datos se recomienda un 70% para entrenamiento y 30% para test [Microsoft, 2012]. Bishop [Bishop, 2006] menciona que la mayor cantidad de datos es usada para entrenamiento y el resto para prueba.

2.3.4.1 Re sustitución o Reclasificación (Método R)

A partir de una muestra *M* dada, de tamaño *N,* se construye un conjunto *R,* de tamaño *N,* para la validación. A partir de esta muestra se clasifican todos los patrones contenidos en *M* utilizando *R*, así es posible observar el número de patrones que han sido mal clasificados [Valdovinos, 2006]. Este método de evaluación tiene un sesgo optimista por sobreajuste [Pérez, 2005], pues utiliza muestras idénticas, además el recurso necesario y el tiempo aumentan de acuerdo al tamaño del conjunto de datos. La estimación del error (PE) se realiza mediante la siguiente fórmula [Valdovinos, 2006]:

$$PE = \frac{1}{m}\sum_{x \in M} E(x, y) \qquad (38)$$

2.3.4.2 Partición mediante un conjunto de prueba (Hold-out o método-H)

Consiste en dividir el conjunto de datos en dos conjuntos disjuntos, uno para entrenamiento R_1 (CE) y otro para prueba R_2 (*test*) [Theodoridis, 2006]. La estimación del error, en este caso tiene un sesgo pesimista, debido a que todos los patrones presentados en el conjunto de prueba son desconocidos por el clasificador y al porcentaje de datos asignado al conjunto de prueba, ya que a través de este se estima el error, y un conjunto de prueba pequeño entregaría una estimación poco fiable. Normalmente se realiza la selección de los patrones, para cada conjunto, de forma aleatoria; por tanto, siendo $R_1 \cup R_2 = M$ y $R_1 \cap R_2 = \emptyset$, la estimación está dada por [Valdovinos, 2006]:

$$PE = \frac{1}{R_2}\sum_{x \in R_2} E(x, y) \qquad (39)$$

2.3.4.3 Validación cruzada con k conjuntos (método de rotación o método-π)

El conjunto de datos, de tamaño *N* es dividido, de forma aleatoria, en *k* conjuntos disjuntos $R_1, R_2, \ldots, R_v$ de tamaño $\frac{N}{k}$. Cada uno de los conjuntos R_v se toma como conjunto de *test*, siendo el CE *M*-$\{R_v\}$, y se estima el error correspondiente [Valdovinos, 2006]:

$$PE_{R_v} = \frac{1}{R_v} \sum_{x \in R_v} E(x, y) \qquad (40)$$

Este proceso se repite hasta concluir con los *k* conjuntos de *test* y al final se realiza el promedio de la estimación del error de la siguiente forma:

$$PE = \frac{1}{V} \sum_{v=1}^{V} PE_{R_v} \qquad (41)$$

2.3.4.4 Validación cruzada con k = N (leave-one-out o método-U)

Este método parte del anterior, en éste, uno de los patrones x_i es dejado fuera del CE, siendo éste el patrón de prueba, y el resto de los datos *M-*$\{x_i\}$, el CE, por lo tanto, cada patrón de prueba que es mal clasificado se cuenta como un error, y el total de errores permite estimar el error del clasificador, que está dada por [Micheli, 2000]:

$$PE = \frac{1}{m} \sum_{x \in M} E(x, y) \qquad (42)$$

Este es un estimador para conjuntos con pocas observaciones, pues tiene un alto costo computacional ya que utiliza para el entrenamiento casi la totalidad del conjunto de datos y cada patrón para prueba y para estimar el error [Valdovinos, 2006], la ventaja de este método, además de entrenar al clasificador con todo el conjunto de datos, es que mantiene la independencia entre el CE y el de prueba [Theodoridis, 2006].

En la actualidad, con el incremento de datos y el avance tecnológico, los conjuntos de datos se vuelven cada vez más grandes, por lo que es posible realizar el entrenamiento y prueba de los clasificadores con una porción de estos conjuntos, por tanto, en conjuntos de datos grandes es común la división de éstos, no sólo en dos conjuntos, si no en tres, uno para entrenamiento, otro para validación y uno más para prueba; de esta forma se realiza el entrenamiento hasta que la mejora del rendimiento del CE ya no se corresponde con la mejora del conjunto de validación, para evitar el sobre ajuste [Kuncheva, 2004].

2.3.5 Evaluación de la clasificación

Para evaluar que los elementos estén correctamente clasificados, se han desarrollado diversas técnicas que permiten saber de forma cuantitativa el porcentaje de

acierto y error que es realizado por el clasificador. La forma más común de medir éste desempeño es con la precisión general, que consiste en obtener la media aritmética de los valores bien clasificados y dividirlo entre el total de patrones [Valdovinos, 2006]:

1) Precisión general (PG)

$$P.G. = \frac{\sum_e x}{p} \qquad (43)$$

donde e=patrones correctamente clasificados y p=total de patrones presentados en la clasificación.

A pesar de ser uno de los criterios más utilizados, no siempre es el más adecuado, pues debe tenerse en cuenta el desbalance que pudieran presentar las clases, ya que este criterio está influenciado por la clase mayoritaria, provocando que la clase minoritaria sea mal clasificada y entregando un porcentaje mayor de acierto a la clase más influyente. Por ende, es importante mencionar criterios que tomen en cuenta este tipo de situaciones, para evitar imprecisiones en el momento de la evaluación.

2) Media geométrica

Se obtiene a partir de la precisión para cada clase, de forma separada. De esta manera, si algún número es 0 o negativo, la media geométrica es inexistente o es negativa. La media se expresa de la siguiente forma [Valdovinos, 2006]:

$$g = \sqrt{a^+ \cdot a^-} \qquad (44)$$

donde $a^+ = 1 - \frac{errores_{a^+}}{patrones_{a^+}}$ (precisión de la clase menos representada) y
$a^- = 1 - \frac{errores_{a^-}}{patrones_{a^-}}$ (precisión de la clase más representada).

Su generalización se describe a continuación:

$$\bar{x} = \sqrt[q]{\prod_{i=1}^{q} x_i} = \sqrt[q]{x_1 * x_2, \dots, x_q} \qquad (45)$$

3) Varianza y desviación estándar

La varianza es una medida estadística que permite calcular la dispersión de los datos en torno a la media a través de la diferencia de sus valores, es decir, de

acuerdo a [Duda, 2000] “mide la coincidencia o alineación del algoritmo de aprendizaje al problema de clasificación”. Por lo tanto evalúa la precisión o especificidad de la coincidencia, define de acuerdo a [Alppaydin, 2010] que tanto varia *x* del valor esperado.

Para calcular la media de la diferencia de sus valores se aplica la siguiente fórmula:

$$s = \sum_{i=1}^{N} (x_i - \bar{x}) \qquad (46)$$

$$\text{donde } \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Mientras que para datos que caen por debajo o están por encima de la media, y cuyas variables son estadísticas, se calcula entonces el cuadrado de las diferencias y se divide el resultado de la sumatoria entre *N*-1, esto se conoce como *varianza muestral*, y permite reparar el sesgo generado por los valores mencionado; de tal forma que ésta es representada por [Valdovinos, 2006]:

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 \qquad (47)$$

Para variables aleatorias se calcula la *varianza estimada*, en la cual la media y la varianza no son conocidas, por lo que la suma se divide entre el total de observaciones. Una varianza grande implica una coincidencia débil, mientras que una varianza baja implica una coincidencia fuerte, de este modo, para obtener un menor error de generalización, la varianza debe ser baja [Duda, 2000].

Por su parte, la desviación estándar o típica, de igual forma, permite evaluar la dispersión de los resultados de clasificación, de manera más sencilla que la varianza [Alppaydin, 2010], pues es la medida de cuanto se aleja una observación de su media. Implícitamente representa la fiabilidad de la estimación del error [Micheli, 2000]. En el caso de la desviación también se debe distinguir el tipo de variable, en este caso se muestra la *desviación estándar muestral*:

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \qquad (48)$$

4) Matriz de confusión

También llamada matriz de contingencia, ayuda a mostrar con precisión no solo los errores y aciertos de la tarea de clasificación [Alppaydin, 2010], sino que también puede observarse la confusión que puede darse entre las clases, así como la distribución del error entre las mismas [Kuncheva, 2004], además pueden agregarse columnas con información adicional como los totales, la asignación real de patrones por clase, la asignación correcta y el porcentaje de bien clasificados [Valdovinos, 2006].

La matriz de confusión se representa a través de una matriz cuadrada de orden *K* x *K*, cuyos elementos (*i,j*) contienen los elementos que pertenecen a la clase C_i, pero que se asignan a la clase C_j. Los valores de la diagonal principal indican el porcentaje de patrones bien clasificados [Kuncheva, 2004]. Los valores de las filas representan los patrones que fueron asignados a la clase sin pertenecer a ella (valores de referencia), mientras que los valores de las celdas por columna indican el número de patrones que perteneciendo a esa clase han sido asignados a una clase diferente e indica a que clase fueron asignados.

Las columnas de asignación por clase y asignación correcta muestran el total de patrones asignados a la clase y el porcentaje de patrones asignados correctamente a dicha clase, respectivamente. Por último, el porcentaje de bien clasificados muestra el porcentaje total de patrones que fueron bien clasificados en la clase (Figura 10). Una vez que se cuenta con los resultados de la matriz de confusión es posible aplicar diversas técnicas estadísticas para el análisis de estos, como la precisión general, que se calcula dividiendo los patrones bien clasificados entre los mal clasificados, o la precisión por clase, que se obtienen de dividir el total de bien clasificados de una clase entre el total de patrones. Un estudio más que puede realizarse es el cálculo del coeficiente y la varianza capa, para analizar la diferencia entre los valores de referencia y los datos determinados por la diagonal principal [Valdovinos, 2006].

Clases	1	2	3	4	5	6	...	*c*	Asignación por clase	Asignación correcta
1	**399**	62	0	0	6	0	...	0	467	0.85
2	0	**58**	113	0	0	0	...	0	174	0.33
3	0	0	**148**	45	0	0	...	0	193	0.77
4	1	0	0	**92**	0	0	...	6	149	0.62
5	2	0	16	0	**17**	0	...	1	36	0.47
6	1	0	0	0	4	**124**	...	0	332	0.37
...	…	…	…	…	…	…	**...**	…	…	…
c	1	0	0	0	0	0	...	**355**	359	0.99
Total	404	120	277	137	27	124	...	362	1733	5.41
% Bien clasificados	0.99	0.48	0.53	0.67	0.63	1.00	...	0.98	1216	0.70

Figura 10. Matriz de confusión. Extraído de [Valdovinos, 2006].

5) Coeficiente *Kappa* [Cerda, 2008].

Es una medida estadística utilizada frecuentemente para medir la concordancia entre dos observaciones, por lo tanto puede medirse con mayor precisión la exactitud de la clasificación. Este coeficiente toma valores entre -1 y 1, donde 1 indica una total concordancia, 0 indicaría que la concordancia es únicamente la que se espera por cuestiones de azar, mientras que el valor de kappa se acerque más hacia -1 la discordancia sería mayor. Este coeficiente cuenta con una escala que permite expresar de forma cualitativa la concordancia encontrada por el indicador, como se muestra en la Tabla 1.

Tabla 1. Grado de concordancia del coeficiente Kappa. Extraído de [Cerda, 2008]

Coeficiente Kappa	Grado de concordancia
0,00	Pobre
0,01 – 0,20	Leve
0,21 - 0,40	Aceptable
0,41 - 0,60	Moderada
0,61 – 0,80	Buena
0,81 – 1,00	Casi perfecta

La ecuación para calcular el coeficiente se muestra a continuación:

$$\widehat{K} = \frac{N\sum_{i=1}^{r} x_{i,i} - \sum_{i=1}^{r}(x_{i+} * x_{+i})}{N^2 - \sum_{i=1}^{r}(x_{i+} * x_{+i})} \qquad (49)$$

donde N es el número de patrones evaluados, r es el número de filas, $x_{i,i}$ el número de patrones en una fila i y una columna i, x_{i+} y x_{+i} son los totales marginales de una fila i y una columna i respectivamente.

Este coeficiente puede medir la exactitud de clasificación de forma más precisa que la matriz de confusión gracias a que incluye todos los valores que integran a la matriz y no sólo sus extremos. Por lo tanto, si todos los valores que no se encuentran en la diagonal fueran iguales a 0, la concordancia sería completa, es decir $\widehat{K} = 1$, por el contrario si ahora los valores que contiene la diagonal fueran nulos entonces $\widehat{K} = -1$, lo que indica una completa discordancia [Arenas, 2011].

CAPÍTULO 3 APRENDIZAJE Y CLASIFICACIÓN

El aprendizaje de máquina se refiere a la posibilidad de optimizar en las computadoras la capacidad de incrementar su conocimiento o aprender a partir de ejemplos o experiencia pasada. El aprendizaje de máquina está basado en modelos, los cuales pueden ser: predictivo, descriptivo o ambos, el primero se refiere a realizar predicciones en el futuro y el segundo a ganar conocimiento a partir de los datos [Ethem, 2004].

Siendo la parte medular del proceso general de RP, el tipo de aprendizaje está directamente relacionado con la forma en que los datos son ingresados al clasificador. De este modo, se pueden establecer dos enfoques que influyen en la forma de aprendizaje del clasificador; estos son: el aprendizaje *off-line* y *on-line*. Dentro del primero, de acuerdo al grado de supervisión que puede ser aplicado al modelo, o como menciona [Dipti, 2010] “de acuerdo a la salida esperada del algoritmo” se encuentran el aprendizaje supervisado y no supervisado, ambos trabajan fuera de línea, es decir no procesan flujos de datos constantes, y se implementan en ambientes más bien estáticos, en los que el conjunto de datos es tratado por completo en un sólo momento.

Mientras que el aprendizaje *on-line*, o incremental, permite trabajar con flujos de datos constantes que son ingresados de forma dinámica al clasificador, requiriendo de un proceso en tiempo real que sea capaz de adaptar el modelo de acuerdo a los nuevos patrones que se van clasificando [Anagnostopoulos, 2010]. En este protocolo se presenta el enfoque *on-line* más a fondo, por ser el tema principal a tratar. La Figura 11 muestra un esquema general de las dos primeras aproximaciones, correspondientes al aprendizaje *off-line,* en el RP [Kuncheva, 2004], las cuales se revisan por separado en las secciones siguientes.

3.1 Aprendizaje supervisado

En este tipo de aprendizaje el clasificador es entrenado para reconocer y clasificar los patrones, de acuerdo a un conjunto de entrenamiento desarrollado por un experto, que cuenta con la información necesaria para que el sistema de RP aprenda a reconocer nuevos patrones y asignarles la etiqueta correcta, de acuerdo a la información que conoce.

Como se observa en la Figura 11 las etapas del clasificador supervisado consisten primero en obtener el conjunto de datos, directamente de las mediciones, posteriormente se realiza la selección y extracción de características. Una vez que se cuenta con el conjunto de datos se selecciona el modelo del clasificador a utilizar,

esto es, se determina el algoritmo que será utilizado para modelar el problema. A continuación viene la etapa de entrenamiento, en la que se ingresa la muestra con los patrones que han de servir para enseñar al clasificador a identificar la clase a la que pertenecen los patrones; una vez terminada esta fase, se considera que el aprendizaje ha concluido.

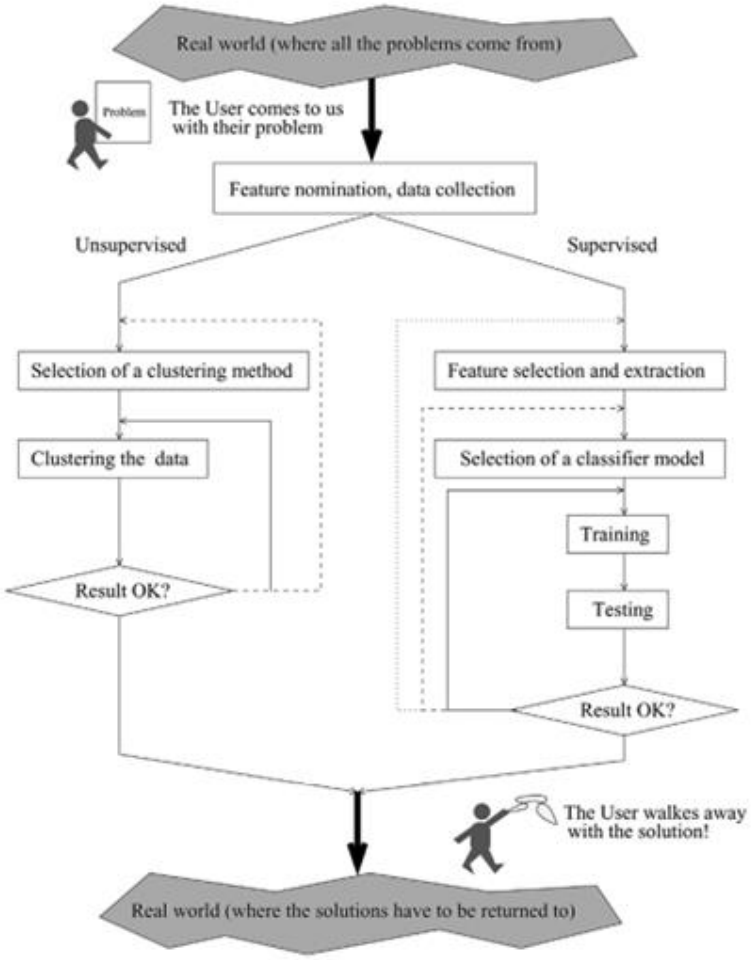


Figura 11. Aprendizaje supervisado y no supervisado en el reconocimiento de patrones. Extraído de [Kuncheva, 2004].

En este tipo de aprendizaje, la muestra de entrenamiento o CE corresponde al conjunto de patrones previamente seleccionados y etiquetados con la clase adecuada, que será utilizado para enseñar al clasificador a reconocer las clases de objetos existentes. Este CE es tratado previamente mediante las técnicas descritas en el capítulo anterior, para que pueda ser procesada por el sistema de RP.

Entre algunos de los principales algoritmos del aprendizaje supervisado que pueden encontrarse, están los siguientes: la regla del vecino más cercano (KNN por sus siglas en inglés *K-Nearest Neighbor*), las Redes Neuronales Artificiales (NN por sus siglas en inglés Neural Networks)

3.1.1 Regla de los *k*-vecinos más cercanos

Es una de las reglas más sencillas para clasificación, consiste en implementar una medida de distancia entre pares, para obtener la distancia de un patrón *x* y los patrones existentes, obteniendo sus *k* vecinos más cercanos, donde *k* determina el número de vecinos que serán tomados en cuenta para determinar la clase mayoritaria a la que pertenece el patrón *x*. La medida estándar aplicada en la regla del vecino más cercano es la distancia euclidiana, entonces, teniendo una instancia *x* arbitraria, se describe el vector de distancias como sigue: $[\delta_1(x, a_1), \delta_2(x, a_2), \dots, \delta_k(x, a_k)]$, donde a indica uno de los *k* vecinos cercanos al patrón *x* [Micheli, 2000]. En el Algoritmo 1 se describe la regla de los *K* vecinos más cercanos.

Algoritmo 1. Algoritmo de la regla del vecino más cercano Extraído de [Smola, 2008]

Clasificar(**X**, **Y**, *x*) {leer documentos **X**, etiquetas **Y** y buscar *x*)

For *i* =1 **to** *m* **do**

Calcular distancia $d(x_i, x)$

End for

Calcular conjunto *I* conteniendo índices para la *k*-esima distancia $d(x_i, x)$ más pequeña.

Regresar etiqueta de la mayoría de $\{y_i$ donde $i \in I\}$

En la Figura 12 se muestra un ejemplo gráfico de la regla *K-NN,* donde el patrón a clasificar se encuentra encerrado en un cuadrado, y se indica mediante líneas verdes la distancia hacia sus patrones más cercanos, siendo el punto rojo el patrón más cercano, por lo tanto la clase a la que pertenece el nuevo punto. También se muestra en el supuesto de que el ejemplo tomara un valor de *k*=1 cuál sería la frontera de decisión resultante.

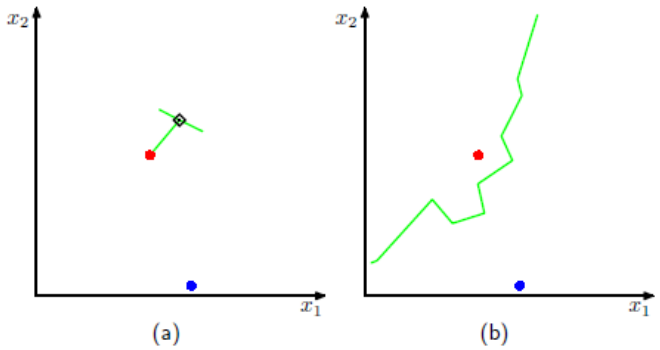


Figura 12. Aplicación de la regla del vecino más cercano, donde (a) el patrón a clasificar se encuentra encerrado en un cuadro, teniendo el valor de *k*=3, el patrón se clasifica de acuerdo a la mayoría de clase de esos tres patrones más cercanos. (b) Con *k*=1, la frontera de decisión que se obtiene se compone de hiperplanos que forman mediatrices perpendiculares de pares de puntos de diferentes clases. Extraído de [Bishop, 2006].

3.2 Aprendizaje no supervisado

El aprendizaje no supervisado es aquel en el que existe un conjunto de datos de entrada, pero se desconoce el resultado a obtener. Este tipo de aprendizaje pretende clasificar los patrones en grupos con características similares a través de medidas de similitud o disimilitud entre las características de los objetos de estudio. La métrica utilizada depende del tipo de datos del que se trate, siendo en datos numéricos continuos las basadas en la distancia de Minkousqui [Marin, 2008] y la distancia de Gower [Pavoine, 2009] la combinación de ambos, numéricos y no numéricos.

Tal como se aprecia en la Figura 11, las etapas del clasificador no supervisado inician con la obtención del conjunto de datos, seguido de la selección del método de clustering. Los algoritmos basados en este tipo de aprendizaje, se pueden clasificar principalmente en dos enfoques, que de acuerdo [Murty, 2012], son *hard clustering*, y soft *clustering* (

Figura 13).

3.2.1 Hard clustering

Este enfoque se basa en generar los grupos sin solapamiento, y para su estudio se divide a su vez en dos grandes grupos, los algoritmos de partición y los algoritmos jerárquicos.

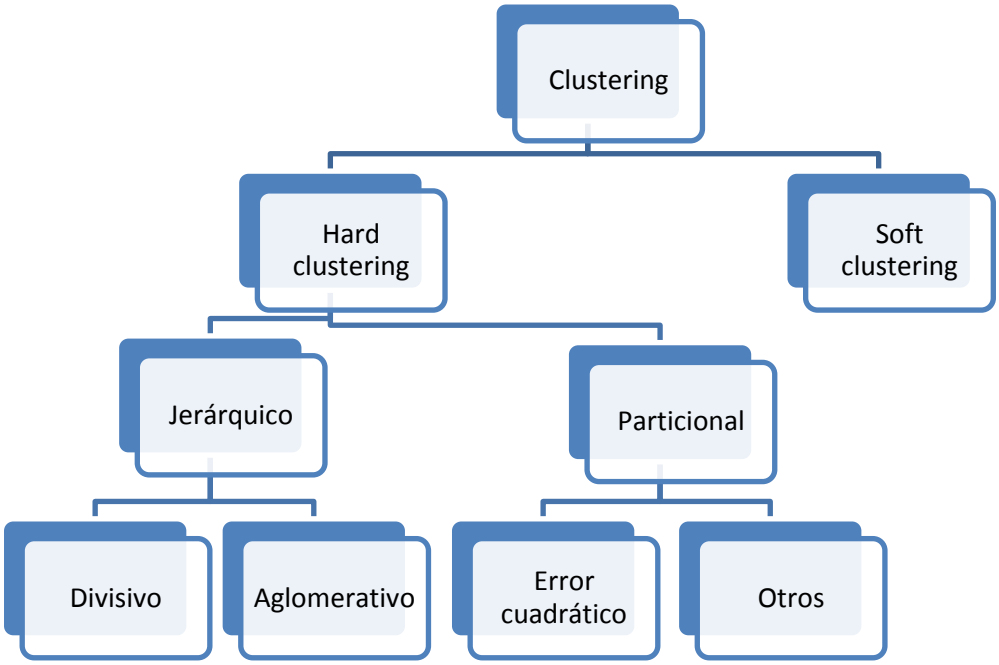


Figura 13. Enfoques de *clustering*. Extraído de [Murty, 2012].

3.2.1.1 Algoritmos de partición

En estos algoritmos se debe determinar el número de grupos que se desea generan, en base a la función de densidad. Existen dos enfoques que pueden tomar los algoritmos de partición, el primero de ellos es utilizan un prototipo que representa al *cluster*, y de acuerdo al tipo de prototipo seleccionado se pueden observar tres variantes de la regla del vecino más cercano [Ankerst, 1999].

- *K-means*. Es una técnica no probabilística iterativa (para minimizar el costo de la función), es útil para recuperar grupos (*clusters*) compactos. Para la identificación de los grupos, parte del cálculo de la distancia a los centros de los grupos. En este caso se toma como prototipo aquel punto que represente el valor medio de todos los elementos que componen al *cluster*.
- *K-modes*. Es *k-means* con una variante para ser aplicado a datos categóricos.
- *K-medoid*. En este algoritmo no se toma el centro del *cluster* como prototipo, sino uno de sus elementos cercanos.

Dada la importancia que el algoritmo tiene en el desarrollo de la Tesis, a continuación se proporciona una explicación un poco más detallada del funcionamiento del algoritmo *k*-medias, así como del algoritmo heurístico incrementale Batchelor y Wilkins.

a) K-medias

En el algoritmo *k*-medias, el valor de *k* indica el número de grupos que han de ser generados, y se utilizan puntos representativos de la clase, que se toman centros de referencia. Así, a través de la medición de la distancia euclidiana se evalúa la disimilitud que existe entre un patrón x_i y el punto representativo de la clase. De acuerdo a esta evaluación se asigna el nuevo patrón a la clase cuyo punto representativo este más próximo a éste (Algoritmo 2).

Algoritmo 2. K-medias

Elegir las estimaciones iniciales arbitrarias θ_j (0) para los $\theta_{j's}$, j = 1,…,m
Repetir
    De i = 1 hasta N
        Determinar el representante más cercano, es decir θ_j, para x_i.
        b(i) = j
        fin del {for}
        de j = 1 a m
        actualización de parámetro: determinar θ_j como la media de los vectores $x_i \epsilon X$ con b(i) = j.
    fin del {for}
Fin de {while} Hasta que no haya cambio en $\theta_{j's}$ entre dos iteraciones sucesivas.

b) Batchelor y Wilkins

Este algoritmo, se basa en determinar la distancia máxima existente entre dos patrones, y mediante un parámetro definido por el usuario determina si un nuevo grupo debe crearse o no. El primer paso es asignar el primer patrón del conjunto de datos como centro del primer grupo, a partir de este se obtiene la distancia de cada uno de los patrones al centro, mediante la distancia Euclídea [Cortijo, 2001] (ver sección 2.2.3.1)

De las distancias resultantes se selecciona aquel patrón que sea el más lejano al primer centro y se asigna como centro del segundo grupo, este procedimiento corresponde a la etapa de inicialización. Una vez que se cuenta con dos grupos se realiza la segunda etapa del algoritmo, que consiste en determinar si se crean nuevos grupos. Esto consiste en la construcción del conjunto ***T***, donde se almacenarán los patrones que aún no han sido procesados, es decir que no han sido asignados como centros. El conjunto se construye al calcular la distancia de cada patrón a cada uno de los centros existentes y determinar cuál es el centro más

cercano, para así asignarlo de forma provisional al grupo al que pertenece dicho centro.

Una vez construido el conjunto ***T*** y en base a su grupo se determina el patrón más alejado. A continuación se calcula el umbral de distancia multiplicando θ por la sumatoria de la distancia de los centros existentes, hasta el momento, a través de la siguiente formula [Cortijo, 2001]:

$$umbral = \theta \sum_{i=1}^{k} \delta(Z_i, Z_{i+1}) \qquad (50)$$

Donde *N* es el número de centros existente y Z_i el centro correspondiente.

Una vez calculado el umbral de distancia, se compara la distancia obtenida del conjunto ***T*** con este umbral; si este último es superado entonces se crea un nuevo grupo, asignando como centro al patrón cuya distancia corresponde a la seleccionada, y se repite nuevamente el proceso. Si la distancia no supera el umbral entonces se da por terminada la etapa de generación de grupos y se continúa con la agrupación libre (etapa 3).

En la agrupación libre se asignan de forma definitiva las etiquetas de grupo a cada uno de los patrones, se parte del conjunto de datos que aún no han sido procesados (es decir, asignados como centro grupo). Para esto se calcula la distancia del patrón *i-esimo* a cada uno de los centros, se asigna al más cercano y se recalcula el centro del grupo *j* a través de la siguiente fórmula:

$$C_j = \left(\frac{x_i + x_{i+1} + \cdots + x_t}{t}\right) \qquad (51)$$

Donde *i* corresponde al patrón, *t* es el número de elementos en el grupo.

El proceso se repite para cada uno de los patrones restantes, utilizando los nuevos centros. Se detalla su implementación en el Algoritmo 3. Las funciones GrupoMasCercano() y RecalcularCentro() se describen en el Algoritmo 4.

Algoritmo 3. Batchelor y Wilkins. Extraído de [Cortijo, 2001]

Entradas:
X Conjunto de *M* patrones $\{X_1, X_2, \ldots, X_M\}$

Θ ϵ [0,1] Fracción de la distancia media entre agrupamientos

Salidas:

A ***ϵ*** *N* Número de agrupamientos encontrados

S1,S2,…,S_A *A* conjuntos de patrones (agrupamientos)

$Z_1, Z_2, \ldots, Z_A$ Los centros de los *A* agrupamientos

Auxiliares:

L Conjunto de patrones que faltan por procesar

T Conjunto de parejas (patrón-centro más cercano)

Inicialización

L←X

$Z_1 \leftarrow X_1 \quad A \leftarrow 1 \quad S_1 \leftarrow \{X_1\} \quad \boldsymbol{L} \leftarrow \boldsymbol{L} - \{X_1\}$

$$\text{Sea } m \text{ tal que } \delta(X_m, Z_1) = \max_{X_i \epsilon \boldsymbol{L}} \{\delta(X_i, Z_1)\}$$

$Z_2 \leftarrow X_m \quad A \leftarrow 2 \quad S_2 \leftarrow \{X_m\} \quad \boldsymbol{L} \leftarrow \boldsymbol{L} - \{X_m\}$

¿Crear agrupamientos?

Repetir

$\boldsymbol{T} \leftarrow \emptyset$

Repetir para cada *X* ***ϵ*** ***L***

$$m \leftarrow \textbf{GrupoMasCercano}(X) \; \{Guardar\ en\ \boldsymbol{T}\ la\ pareja\}$$

$$\boldsymbol{T} \leftarrow \boldsymbol{T} \cup \{(X, Z_m)\} \; \{formada\ por\ X\ y\ Z_m$$

Fin-para // cada ***X*** ***ϵ*** ***L***

$$\text{Sea } n \text{ tal que } \delta(X_n, Z) = \max_{(X,Z) \epsilon \boldsymbol{T}} \{\delta(X, Z)\}$$

umbral←CalcularUmbral(θ)

Si $\delta(X_n, Z) > umbral$ {Crear agrupamiento}

$Z_{A+1} \leftarrow X_n \quad A \leftarrow A + 1 \quad S_A \leftarrow \{X_n\} \quad \boldsymbol{L} \leftarrow \boldsymbol{L} - \{X_n\}$

terminar←FALSO

Si-no // En este caso $\delta(X_n, Z) \leq umbral$

terminar←VERDAD

Fin-si // Si $\delta(X_n, Z) > umbral$

Hasta *terminar*=VERDAD

Agrupar libremente

Repetir para cada *X* ***ϵ*** ***L***

$m \leftarrow \boldsymbol{GrupoMasCercano}(X)$

$S_A \leftarrow S_m \cup \{X\}$

$Z_m \leftarrow \mathbf{RecalcularCentro}(m)$

Fin-para // cada *X* ***ϵ L***

FIN DEL ALGORITMO

Función CalcularUmbral(θ){

$sum = 0$

Repite para *i* = 1,2,…, *A* – 1

$sum = sum + \ \delta(Z_i, Z_{i+1})$

Fin-para

$umbral = \ \theta \dfrac{sum}{A-1}$

Devuelve (*umbral*)

}

Algoritmo 4. Funciones GrupoMasCercano() y RecalculaCentro()

Función GrupoMasCercano(X){

dTemp = 0 //

dActual ← $\delta(X_m, Z_1)$

Mientras ($l \ < \ l-1$){

Si dTemp < dActual {actualiza más cercano}

dTemp = dActual

}

Fin-mientras //

dTemp = $\min_{X_i \epsilon \, \boldsymbol{L}}\{\delta(X_i, Z_1)\}$

X ← dTemp

}

Función RecalculaCentro(i){

Repetir para j = 1…n

Repetir para k = 1…m

$z_i = \sum_1^n x_{j,k} \ \{x_{j,k} \ \epsilon \ S_i\}$

Fin-repetir

Fin-repetir

}

3.2.1.2 Algoritmos de región de densidad

Otros algoritmos dentro de este mismo enfoque son los basados en región de densidad, donde se parte de un punto y se determina, por medio de criterios locales, las regiones de alta densidad, separadas por regiones con baja densidad, que son tratadas como ruido. Este tipo de algoritmos son capaces de descubrir regiones de formas muy complejas. Entre los algoritmos más representativos pueden distinguirse los siguientes [Theodoridis, 2006]:

- DBSCAN (por sus siglas en inglés *Density-Based Spatial Clustering of Applications with Noise*). Es un algoritmo determinístico, que a partir de un punto determina un radio de vecindad, conocido como ε, y en el que deben encontrarse un mínimo de puntos (MinPts) para que a partir del punto más alejado dentro de ese radio de vecindad se pueda llevar a cabo el mismo procedimiento, hasta que no se cumpla el criterio de MinPts, siendo este y el parámetro ε definidos por el usuario. En la Figura 14 puede verse un ejemplo de cómo se determinan los grupos mediante el algoritmo.

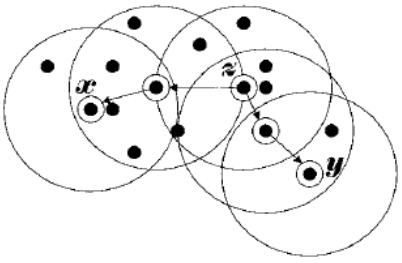


Figura 14. Ejemplo con el número mínimo de puntos igual a 5.

- DENCLUE (por sus siglas en inglés Density-Based CLUstering). Se basa en una malla, a través de la cual almacena cada punto, mientras que utiliza la función de densidad para identificar los máximos locales significantes y poder crear un *clusters,* ya que si existe un máximo local se espera que la región alrededor de él sea densa, a esto se le denomina paso de *precluste-ring*. Posteriormente se determinan aquellos máximos locales que son significativos para los cuales se debe crear un *cluster*.

Dentro de este enfoque también puede encontrarse la partición suave de los grupos, dentro de las estrategias para resolver este tipo de partición se encuentra la estrategia “dividir y mezclar”, un ejemplo de algoritmo de este tipo es ISODATA [Theodoridis, 2006]. Otra estrategia es usar el “aprendizaje competitivo”, mediante técnicas de *clustering Fuzzy* [Al-Sultan, 1993],[Chiang, 2003], *Rough clustering* [Tripathy, 2011] o Redes Neuronales para *clustering* [Kumar, 2007]. Por último, el *soft clustering* puede resolverse a través de técnicas de búsqueda estocástica,

como el recocido simulado [Theodoridis, 2006], la búsqueda tabú [Sung, 2000] o los algoritmos evolutivos [Taher, 2008].

3.2.1.3 Algoritmos jerárquicos

Estos algoritmos son conocidos por generar agrupamientos internos a uno existente, esto es, grupos dentro de grupos, de los cuales se genera un árbol conocido como *dendrogram*, que contiene la jerarquía de los *clusters* [Ripley, 2008]. Los algoritmos jerárquicos pueden ser de dos tipos, el primero aglomerativo, donde cada patrón forma un grupo, y a partir de la similitud existente entre cada uno de estos patrones, éstos se van uniendo hasta formar grupos mayores, que al final reducen la partición del conjunto de datos [Murty, 2012]. Algunos de los principales algoritmos para el *clustering* jerárquico son: GAS (*Generalized Agglomerative Scheme*), MUAS (*Matrix Updating Algorithmic Scheme),* WPGMA (*Weighted Pair Group Method Averange)*, UPGMA (*Unweighted Pair Group Method Averange)*, algoritmo de mínima varianza, entre otros [Theodoridis, 2006].

Los de tipo divisivo, al contrario de los anteriores, que implementan una estrategia *bottom-up,* usan una estrategia *top-down* que crea las particiones en los datos. Dentro de estos se pueden encontrar dos tipos que dividen los grupos de diferente forma, los *politéticos* dividen el grupo basados en más de una característica, mientras que los *monotéticos* cuando en la división se considera sólo una característica [Murty, 2012]. Existen diversos algoritmos para realizar esta tarea de agrupamiento, para algoritmos *politéticos* puede consultarse [Theodoridis, 2006], mientras que para algoritmos monotéticos puede consultarse [Everit, 01].

3.2.2 Soft clustering

Los algoritmos de *soft clustering* trabajan con el solapamiento de grupos, están basados en los conjuntos difusos, los llamados "*Rough sets"*, redes neuronales artificiales y algoritmos genéticos AG [Murty, 2012]. En los algoritmos de partición se han mencionado algunas de las estrategias y métodos que pueden ser aplicadas en conjuntos difusos, "*Rough sets"* y NN, en AG pueden encontrarse algunas técnicas propuestas en [Maulik, 2000] y algunas aplicaciones en [Yolis, 2003], [Casills, 2003], [Díez, 2004], [Vélez, 2001].

No obstante a los tipos de aprendizaje presentados, existe un enfoque intermedio, que es el aprendizaje semi-supervisado, que tiene el objetivo principal de combinar datos etiquetados y no etiquetados, suponiendo que la distribución de los datos etiquetados es la misma para los no etiquetados, y así poder generar modelos más eficientes.

Se tiene un conjunto de datos sin etiqueta y otro etiquetado, el aprendizaje semi-supervisado puede ser aplicado de dos formas [Chapelle, 2006]: la primera para obtener información adicional del conjunto de datos no etiquetado a través del conjunto de datos etiquetado. La segunda como tarea de agrupamiento, utilizando el conjunto de datos etiquetado como restricciones y formando puntos que incluyan o excluyan cierto tipo de características.

Dentro del aprendizaje *on-line* el mismo autor menciona diferentes técnicas de este enfoque con las cuales puede trabajar. A continuación se describen algunas de ellas [Chapelle, 2006]:

Co-entrenamiento (*co-training*): el conjunto de datos es dividido en dos y se entrena con cada conjunto un clasificador diferente, al finalizar el entrenamiento cada clasificador re-entrena al otro con los ejemplos sin clasificar que son más fiables, y el proceso sigue hasta que ambos clasificadores coincidan en los resultados, tanto de los patrones no clasificados como los clasificados.

Máxima Entropía (Expectation-Maximization): se entrena al clasificador a través del CE, es decir, de datos etiquetados; en base al entrenamiento recibido el clasificador debe asignar pesos a las etiquetas de cada uno de los patrones no etiquetados, con lo que se entrenara nuevamente otro clasificador para volver a asignar pesos.

3.3 Aprendizaje *on-line*

Actualmente, el incremento de información y por tanto, de datos disponibles, requiere de técnicas capaces de dar soporte a su procesamiento y análisis en tiempo real; es el aprendizaje *on-line* quien cubre estas necesidades, pues con su implementación se procesan flujos continuos de datos, siendo capaz de dar una respuesta en todo momento [Ferrer, 2005b]. Esta última aproximación resulta ventajosa principalmente cuando se requiere mantener un conocimiento del clasificador actualizado y dinámico, no obstante, la complejidad de implementación es una tarea no trivial de resolver.

De acuerdo a [Ferrer, 2005a] los principios de los esquemas inductivos en los que se basa el aprendizaje *off-line* son:

- Los ejemplos deben estar disponibles previamente al aprendizaje
- Todos los ejemplos pueden ser cargados en memoria
- El aprendizaje finaliza al terminar el proceso de entrenamiento.

Por tanto, tal esquema es usado principalmente porque puede ser aplicado en contextos en los que el conjunto de datos es pequeño y no se requiere del procesamiento de estos en tiempo real. Con el avance tecnológico de los últimos años, esto ha cambiado y ahora se requiere modelar grandes bases de datos, así como su procesamiento en tiempo real, por lo que poco a poco esta técnica está volviéndose impracticable en muchas áreas [Anagnostopoulos, 2010].

La forma de trabajar *off-line* en el aprendizaje supervisado es a partir de un conjunto de datos que es dividido en dos muestras, una de entrenamiento y otra de test. A partir de la muestra de entrenamiento se entrena al clasificador para reconocer ciertos objetos como parte de alguna clase especifica. Una vez que el algoritmo está entrenado para identificar objetos, se comprueba mediante la muestra de test que tan buena es la precisión del algoritmo para identificarlos [Kuncheva, 2004].

En el aprendizaje no supervisado de forma *off-line* también se parte de un conjunto de datos establecidos, la diferencia es que al no contar con información sobre la clasificación de este conjunto de datos se procesa todo el conjunto para buscar similitud entre los objetos y con ellos crear grupos. En tanto que para el aprendizaje semi-supervisado el proceso es similar al seguido en el aprendizaje supervisado.

Entre los algoritmos más representativos del aprendizaje *off-line* se encuentran las redes neuronales artificiales (RNA) [Ripley, 1996], [Dunne, 2007], [Ramírez, 2011], las máquinas de soporte vectorial (SVM) [Betamcourt, 2005], [Steinward, 2008], [Abe, 2010], [Hsu, 2003], [Cristianini, 2000], [Campbell, 2011] y los árboles de decisión [Escolano, 2003], [Marín, 2008], [Hernández, 2004], entre otros.

Actualmente, el incremento de información y por tanto, de datos disponibles, requiere de técnicas capaces de dar soporte a su procesamiento y análisis en tiempo real; es el aprendizaje *on-line* es quien cubre estas necesidades, pues con su implementación se procesan flujos continuos de datos, siendo capaz de dar una respuesta en todo momento [Fernández, 2005b]. Esta última aproximación resulta ventajosa principalmente cuando se requiere mantener un conocimiento del clasificador actualizado y dinámico, no obstante, la complejidad de implementación es una tarea no trivial de resolver.

El aprendizaje *on-line*, de acuerdo a Alber [Alber, 2006] recibe una “secuencia de porciones de entrada”, $\sigma = \sigma(1), \ldots, \sigma(m)$, por lo que el sistema debe dar una respuesta a cada una de estas porciones $\sigma(t)$, de tal forma que dada la secuencia el algoritmo no conoce la cadena futura $\sigma(t')$, donde $t' > t$, y se entiende que las secuencias son ingresadas de acuerdo a su orden de ocurrencia.

Al respecto Krumke [Krumke, 2006] menciona que existen dos paradigmas para determinar cómo se presentan los ejemplos a un algoritmo *on-line*; estos son: el modelo de secuencia y el modelo de marca de tiempo.

El primer paradigma hace referencia a que los objetos son ingresados conforme se generan mientras que en el segundo modelo se especifica un tiempo en el que el objeto se hace disponible para su procesamiento, por lo que no necesariamente los objetos deben ingresar en el orden en que se generan. Por su parte Maloof [Maloof, 2003] menciona dos estrategias para almacenar los ejemplos, el *on-line* batch o temporal batch, que corresponde a procesar la instancia actual sin tomar en cuenta la instancia previa pero mantiene en memoria conocimiento sobre estas, y el aprendizaje incremental, que modifica la instancia actual en base a la anterior y al conjunto de instancias sin mantener conocimiento en memoria, es decir el modelo se actualiza con cada instancia. Por otro lado existe otro enfoque que es el almacenamiento parcial de instancias que permite mantener en memoria una parte representativa de instancias, lo que ayuda a mejorar la precisión del clasificador [Schmitt, 2008].

Para evaluar el desempeño de los algoritmos *on-line* Sleator y Tarjan [Sleator, 1985] propusieron comprar el resultado del algoritmo *on-line* con el resultado del algoritmo *off-line* más óptimo para el mismo problema, por su parte Krumke [Krumke, 2006] menciona que este análisis competitivo solo toma en cuenta la información, no el recurso disponible.

Dentro de este enfoque, se pueden encontrar diversos algoritmos de aprendizaje supervisado que incluyen un enfoque de aprendizaje continuo, uno de los más conocidos es el perceptron, que ajusta los pesos de acuerdo a las nuevas entradas, sumando cuando es positivo y restando en caso contrario [Zubiaga, 2008]. Aun cuando fue desarrollado bajo un enfoque de aprendizaje *off-line*, es posible su incorporación en un esquema *on-line*. *K-medias* es otro algoritmo ampliamente conocido para realizar aprendizaje no supervisado, no obstante, con una pequeña adecuación, también es posible utilizarlo como un algoritmo de aprendizaje *on-line*. La modificación consiste en mover el *cluster* más cercano hacia el patrón de entrada en un factor de η. La actualización se puede expresar como [Anagnostopoulos, 2010]:

$$c_i = c_i + \eta \cdot b_{i,j} \cdot (u_i - c_i) \qquad (52)$$

donde $b_{i,j} \in \{0,1\}$ denota que cluster será modificado
$\eta \in [0,1]$ especifica que tanto es movido el cluster hacia el nuevo patrón
$(u_i - c_i)$ expresa la distancia que será aprendida

Como se había mencionado en la introducción, algunos trabajos relacionados al aprendizaje *on-line* son [Ortega, 2010] para la comprensión del habla a través de la comparación de unidades semánticas que representan frases de un corpus no alineado, implementado para la comprensión de un dialogo hablado. Otro trabajo relacionado puede ser consultado en [Schmitt, 2008] donde se plantean la aplicación del algoritmo FLORA-MC para servicios que se adapten de acuerdo a la información que se colecta en tiempo real de diferentes sensores. Este algoritmo puede diferenciar entre múltiples categorías, no solo categorías binarias. Soporta datos numéricos en vez de nominales. El desarrollo de FLORA-MC se basa en el algoritmo FLORA-2, que tiene un consumo bajo de memoria, y permite agregar un manejo flexible del tamaño de la ventana que contiene los ejemplos.

FLORA-MC trabaja con reglas de clasificación llamadas Items descriptivos (DI, Description Items), los cuales son asignados a alguno de los tres subconjuntos posibles, ADES (Accepted DEScriptor) para las reglas positivas, PDES (Potential DEScriptor) para aquellas reglas que son neutrales y NDES (Negative DEScriptor) para las reglas negativas. Los ejemplos se mantienen en la ventana siempre y cuando estén dentro del rango de la misma, si lo rebasan son removidos. Esto permite ajustar el modelo, de acuerdo a la vigencia de los patrones que son introducidos.

Mairal [Mairal,2009] propone un algoritmo para el problema de aprendizaje de diccionarios, que consisten en un conjunto base a partir de la descomposición lineal de una señal. También propone su uso para el aprendizaje del diccionario de conjuntos pequeños y grandes de imágenes y audio.

Hoi [Hoi, 2013] propone la investigación de un nuevo problema denominado *on-line* Multiple Kernel Classification (OMKC) que es la unión del aprendizaje *on-line* y mediante kernel, además propone el desarrollo de algoritmos para este problema, que consta de la combinación de clasificadores que tienen que determinar de entre un subconjunto de kernels la función más adecuada para la resolución un problema de forma *on-line*. En los resultados Hoi muestra que entre mayor sea el número de ejemplos presentados al algoritmo la precisión mejora, y en cuanto al tiempo menciona que este puede ser mejorado con dos de sus algoritmos. También presenta una revisión extensa de los trabajos desarrollados en el aprendizaje *on-line*.

El aprendizaje *on-line* es aplicado en [Santibáñez, 2013] junto con la administración de RAM como una propuesta para tratar grandes conjuntos de datos en base al recurso de memoria disponible. Los resultados obtenidos muestran una precisión muy cercana a la obtenida en el tratamiento de los conjuntos de forma *off-line* y una reducción significativa en el tiempo de procesamiento.

En el trabajo de Kidera [Kidera, 2006] se expone un ensemble de clasificadores incremental, basado en una extensión del algoritmo AdaBoost.M1. El número de clasificadores es predeterminado y modificado durante el aprendizaje, fase en la que todos los clasificadores son actualizados. Esta propuesta expone un algoritmo que puede aprender de forma incremental a partir de subconjuntos de entrenamiento cuyo tamaño no afecta el resultado. Por lo tanto el desempeño se puede comparar con el de un algoritmo *off-line* en un modelo multi-clasificador.

Otros trabajos de aprendizaje *on-line* en la tarea de agrupamiento son: [Forestiero, 2013], [Schaighofer, 2009], [Guan, 2011], [Jiang, 2010], [Khaleghi, 2012], [Mashayekhi, 2012] y [Vadrevu, 2011], mientras que [Jain, 2006] y Giraud, 2000] presentan su aportación en el aprendizaje incremental.

En [Barbakh, 2008] se presenta un conjunto de algoritmos de agrupamiento para compensar la sensibilidad de las condiciones de inicialización del algoritmo K-medias, y a partir de estos se deriva la versión *on-line* que es capaz de converger a una solución que K-medias no. Los algoritmos son K-medias, IWK01, IWK02 y K-Harmonic. Otro estudio importante es el de [Singh, 2010] que propone aplicar el aprendizaje *on-line* para la actualización de los clasificadores biométricos desarrollando un algoritmo basado en una 2 v-SVM que integra computación granular y soft labels. Esta propuesta indica que la precisión y tiempo son al menos tres veces mejores que los obtenidos con la SVM convencional. Entre otras propuestas, se encuentran [Cao, 2006] y [Diehl, 2003].

El trabajo de Tu [Tu, 2009] plantea un marco de trabajo basado en el enfoque de densidad para realizar el agrupamiento de flujos continuos de datos. Este marco de trabajo consiste en dos componentes el *on-line* y el *off-line*, siendo el primero el que asigna el ejemplo en una malla, y el componente *off-line* se encarga de procesar la maya para agruparla según la densidad, teniendo como resultado un agrupamiento de alta calidad ya que puede reconocer formas arbitrarias en los datos y mantener un seguimiento en la evolución del flujo de datos. En esta misma vertiente se encuentran los trabajos de [Younis, 2005] y [Kifer, 2004].

CAPÍTULO 4 ADMINISTRACIÓN DE MEMORIA RAM

La memoria RAM o Memoria de Acceso Aleatorio (por sus siglas en inglés, *Random Access* Memory) es la unidad de almacenamiento principal de la computadora y es indispensable en su funcionamiento ya que en ella se almacenan los programas y datos que están en uso. La RAM se clasifica como una memoria volátil, pues funciona con el suministro de corriente eléctrica, por lo que al no existir ninguna corriente los datos contenidos en memoria se pierden [Durán, 2007].

Es posible tener acceso a los datos almacenados en la RAM mediante acceso directo a través de una dirección de memoria, las acciones que pueden realizarse en la RAM son la lectura de datos almacenados y la escritura de nuevos datos [Niño, 2011], estas se realizan a través de un método que implica la utilización de la memoria caché. Para conocer mejor el funcionamiento de la RAM, primero debe exponerse la jerarquía de memoria, compuesta en primer instancia por los registros, la caché, la memoria RAM y el disco duro.

En general, una memoria entre más pequeña suele ser más costosa y tener menor espacio de almacenamiento, aunque una mayor velocidad de acceso [Niño, 2011]. Por ejemplo, una memoria RAM tiene mayor capacidad que una memoria caché, pero la caché es más rápida que la RAM, por ello para obtener el mayor provecho de estos componentes y por ende del sistema se implementa la jerarquía de memoria [Berral, 2010].

4.1 Estructura

La jerarquía de memoria es una forma de distribuir los datos y programas en el sistema [Berral, 2010], apelando al rendimiento y buen funcionamiento de este. Se compone en primera instancia por los registros, que se encuentran dentro del procesador y no tienen mucha capacidad de almacenamiento, La caché, es una memoria pequeña de alta velocidad que almacena la información que se utiliza más frecuentemente, la memoria RAM, que ya se ha definido, y el disco duro, que es un dispositivo con una capacidad de almacenamiento muy superior a la de la RAM, este dispositivo es no volátil y tiene un costo menor a la RAM, aunque también su velocidad de acceso es inferior [Niño, 2011].

4.2 Funcionamiento

El funcionamiento de La RAM requiere de una jerarquía de memoria diseñada para el mejor funcionamiento y aprovechamiento de las capacidades de esta.

Por lo tanto, para que la escritura y lectura de datos funcione adecuadamente, existen dos buses que hacen de conexión entre la RAM y el procesador. El bus de direcciones, a través del cual se envía la dirección exacta de memoria en la que ha de ser escrito o leído un dato, y el bus de datos, que transporta los datos hacia la dirección indicada previamente.

Tomando en cuenta, que el tiempo de acceso a la RAM actualmente es muy importante para que el equipo tenga un buen rendimiento, es necesario utilizar una memoria intermedia llamada caché [Durán, 2007], que almacena los datos que se utilizan frecuentemente, y cuya velocidad de acceso es de 5 a 10 veces superior a la de la RAM [Morris, 1994].

Para llevar a cabo el proceso de lectura y escritura en la RAM es necesario hacer uso de la caché, como se ha mencionado, por lo que ambas se dividen en bloques de igual tamaño. Esto permite trabajar con un sistema de paginación que hace más eficiente estos procesos organizado en un área especial del disco duro la tabla de paginación, estas páginas permiten que la información sea llamada por bloques, aquellos que son de uso frecuente, o que son requeridos, son almacenados en la chache; este tema se describe con mayor profundidad en [Morris, 1994].

En base a lo anterior, cuando se requiere algún dato de la memoria principal (RAM) se hace una referencia a una palabra de memoria (dirección), con esto, el procesador busca primero en la caché, si encuentra la referencia especificada, ocurre un *hit* y se utiliza la palabra encontrada, pero si no encuentra la palabra ocurre un *miss* y es necesario que el bloque que la contiene sea traído de la RAM a la caché.

4.3 Interacción de java con la memoria física

Java interactúa con la RAM a través de la Máquina virtual de Java (JVM por sus siglas en inglés, *Java Virtual Machine*) [Lindhom, 2011], quien es la responsable de que Java sea multiplataforma, eficiente y segura [Menchaca, 2000] pues es una computadora abstracta, que funciona como si fuera una computadora real ya que cuenta con un conjunto de instrucciones y maneja varias áreas de memoria [Lindholm, 2011].

4.4 Estructura de la memoria en java

La memoria en la JVM se divide en dos áreas principales (Figura 15), el área *Heap* o generación joven, que es la memoria volátil y el Área de código, conocida también como generación vieja o comúnmente como *No-Heap* o memoria permanente [Lindhom, 2011]. En la primera se almacenan las instancias, variables y arrays que están siendo utilizadas en tiempo de ejecución, es decir datos dinámicos, mientras que en la segunda se almacenan los datos permanentes de la aplicación, como las clases, interfaces y metadatos.

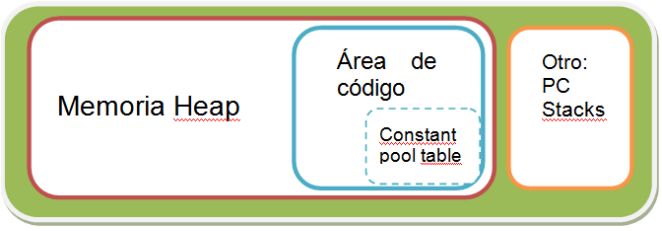


Figura 15. Estructura de memoria de la JVM. Extraído de [Yourkit, 2003]

Ambas áreas son compartidas por todos los hilos creados hasta el momento, pueden ser de tamaño fijo o dinámico, es decir que su tamaño puede ser asignado a través de instrucciones propias de la JVM. Tomando en cuenta lo anterior, si se sobrepasa el límite máximo asignado para cualquiera de estas dos áreas se generará la excepción *outOfMemoryError* [Lindhom, 2011].

A partir del área de código se asigna otra, que es un conjunto de constantes de tiempo de ejecución (*Constant Pool Area*), esta área se crea con cada clase o interfaz y contiene desde constantes hasta referencias, cuando por alguna razón no puede asignarse el espacio de área de código a la tabla de constantes, se generará la excepción *outOfMemoryError* [Lindhom, 2011].

Además de estas áreas, también se encuentra el área que contiene el código de la máquina virtual, es decir las estructuras internas y datos propios de la JVM [Lindhom, 2011], dentro de esta puede ubicarse el registro PC (por sus siglas en inglés *Program counter*), que se genera para cada hilo de ejecución; la pila o *Stack*, almacena los marcos que contienen la información sobre los métodos, lo cual incluye las variables e instancias locales.

La memoria para la pila no necesita ser contigua, lo que permite que su asignación pueda ser dinámica, si algún cálculo en un hilo requiere mayor espacio que el asignado en la pila ocurrirá una excepción, *StackOverFlowError* u *OutOfMemoryError* si la pila puede ser expandida dinámicamente y no hay suficiente memoria.

4.5 Funcionamiento de la memoria JVM

Una vez que se conoce la estructura de la memoria de la JVM, es posible explicar su funcionamiento. La memoria *Heap* se crea al iniciar la máquina virtual de Java, esta se carga inicialmente con espacio de 64 MB, los cuales pueden aumentar o contraerse, de acuerdo a las necesidades de la aplicación. Dentro de esta área de memoria se implementa el administrador automático de almacenamiento (*automatic storage management*) [Menchaca, 2000] que mantiene un historial de los objetos y al momento de no ser referenciados libera el espacio, es conocido como el recolector de basura o *Garbage Collector* y puede ser implementado a través de diversas técnicas [Lindhom, 2011].

Como el área de método se encuentra dentro del área *Heap* se entiende que el recolector de basura es también el encargado de liberar espacio en memoria de instancias de clases, interfaces o métodos que ya no son referenciados, aunque en algunas implementaciones puede no utilizarse esta técnica [Lindhom, 2011].

El área de métodos se crea con el inicio de la máquina virtual y puede también ser de tamaño fijo o dinámico, de acuerdo a los requerimientos en tiempo de ejecución, aunque también implementa un límite máximo, en este caso la memoria tampoco necesita ser contigua.

Java no provee mayor información sobre la memoria, ya que estos datos se dejan a discreción del implementador, pero dentro de la información que si puede ser consultada, se encuentran las instrucciones para manipular el tamaño de las dos áreas de memoria. A continuación se describe su funcionamiento y se ejemplifica cada una de ellas de acuerdo al manual publicado por [Sun, 2006].

En la Tabla 2 y Tabla 3 se muestran las instrucciones para la asignación de tamaño y expansión de la memoria *Heap.*

Tabla 2. Instrucciones de asignación de tamaño de memoria *Heap*

-Xms<*n*>. Indica el tamaño inicial del área.
-Xms256m<nombre de la aplicación>, donde el tamaño inicial del Heap será de 256Mb
-Xmx<n>m. Indica el tamaño máximo que puede tener el Heap.
-Xmx256m<nombre de la aplicación>, donde el tamaño máximo del Heap será de 256Mb
Por tanto en total el tamaño del Heap será de 512Mb

Tabla 3. Instrucciones para manejo de expansión de memoria *Heap*

MinHeapFreeRatio=<*n*>. Especifica el tamaño mínimo que puede existir de espacio libre a partir del tamaño total del *Heap*. Si alguna de las generaciones o áreas existentes es menor que el límite asignado entonces se expande hasta cubrir el porcentaje especificado. Su sintaxis se muestra a continuación: -XX:MinHeapFreeRatio=30, donde 30 es el porcentaje mínimo que debe existir de espacio libre. MaxHeapFreeRatio=<n>. Especifica el tamaño máximo que puede existir de espacio libre a partir del total del Heap. Si alguna de las generaciones o área existentes ocupa mayor espacio libre del que se ha determinado entonces este espacio se reduce hasta cubrir el porcentaje asignado. A continuación el ejemplo: -XX: MaxHeapFreeRatio=60, donde 60 es el porcentaje máximo de espacio libre que puede existir

En la Tabla 4 se exponen las instrucciones para la asignación del tamaño del Área de código.

Tabla 4. Instrucciones de asignación de tamaño del Área de código

PermSize=<*n*>. Asigna el tamaño mínimo que tendrá el área. A continuación se describe su sintaxis: -XXPermSize=128M<nombre de la aplicación>, Fija el tamaño mínimo a 128 Mb MaxPermSize. Para asignar el tamaño máximo que tendrá el área, esto es más útil si se cargan dinámicamente muchas clases. Su sintaxis en línea de comandos se escribe a continuación: -XX:MaxPermSize=*256m*, donde el tamaño máximo del área de código será de 256Mb.

PARTE 3
DESARROLLO

CAPÍTULO 5 METODOLOGÍA

La metodología comprendió los diferentes pasos para desarrollar la investigación, desde la exploración documental del tema, hasta el planteamiento de la metodología para procesar los datos. Mediante la investigación documental se hizo una revisión exhaustiva de la literatura, para seleccionar la información existente que mejor describía y analizaba el tema de investigación, y de igual forma permitía establecer los antecedentes que existen hasta el momento.

Una vez recolectada la información relevante para el desarrollo de la investigación se realizó el prototipo que ayudaría a llevar a la práctica la metodología propuesta. Hecho lo anterior, se utilizó el método experimental para realizar las pruebas necesarias, permitiendo la medición y comprobación del funcionamiento de dicha metodología, así como la extracción de información. A través del método analítico se realizó la comprobación de la hipótesis planteada. Lo anterior se efectuó apoyado en la estadística descriptiva para organizar y clasificar los indicadores cuantitativos, así como en la estadística inferencial para dar la interpretación y valoración cuantitativa de las magnitudes del desempeño de la metodología.

Las variables de estudio que se plantean son las siguientes: la precisión, que debe ser lo más alta posible para asegurar el buen desempeño en la tarea de clasificación; el costo computacional, que a través del conocimiento de los recursos de memoria RAM disponibles y la implementación de la administración de este recurso para buscar que el costo sea el mínimo posible. Por último en el tiempo de ejecución, que requiere el mínimo posible para asegurar la respuesta en un tiempo razonable.

5.1 Propuesta

Una vez planteada la forma en que ha de desarrollarse el trabajo de investigación, se plantea la metodología propuesta (Figura 16), que inicia una vez seleccionado el conjunto de datos, (el cual es numérico), y debido a su tamaño, será ingresado de acuerdo al espacio disponible de RAM. Para lo anterior se identificará la capacidad de memoria del equipo, así, a partir de ésta se determinará el tamaño que los subconjuntos han de tener para ser procesados de forma consecutiva. Una vez considerados los elementos anteriores se construye la matriz de patrones del tamaño determinado, la cual residirá en memoria y sobre la cual se aplicará un algoritmo heurístico incremental para encontrar los grupos de datos que comparten características similares.

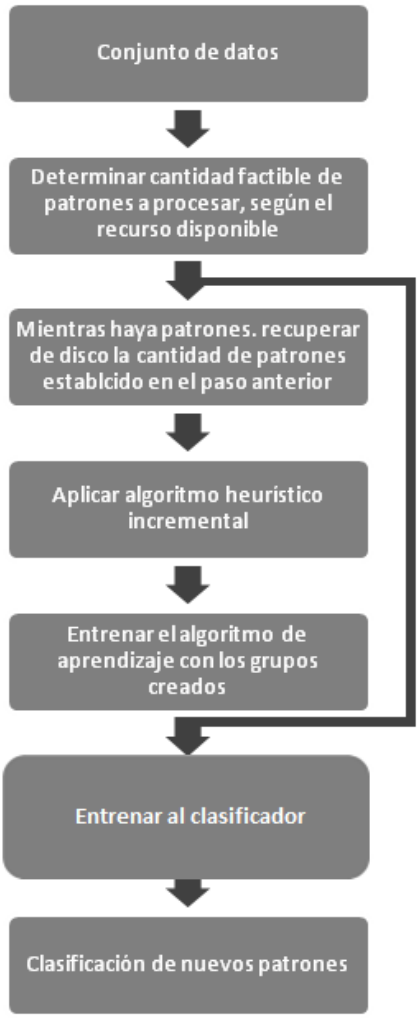


Figura 16. Metodología propuesta

A partir de los grupos creados en los pasos anteriores, se entrenará al algoritmo de clasificación, para que obtenga el conocimiento necesario para reconocer nuevos patrones y clasificarlos correctamente, tomando en cuenta que alguno de los patrones podrían no ser agregados al conocimiento, dependiendo si éstos representan ruido o un valor atípico, caso en el cual serán rechazados.

Estos pasos, a partir de determinar la cantidad de patrones para cada subconjunto, habrán de realizarse de forma iterativa hasta concluir con el conjunto de datos, con lo que cada una de las reglas de decisión seleccionadas para el siguiente paso tendrán el espacio de conocimiento correspondiente. Por lo tanto, con los algoritmos entrenados se formará el Sistema Múltiple de Clasificación que integrará el conocimiento de cada uno de éstos para determinar la clasificación del nuevo patrón que haya sido ingresado.

5.2 Conjuntos de datos

Los conjuntos de datos utilizados para esta metodología fueron tomados del repositorio de la SIPU, (por sus siglas en *inglés Speech and Image Processing Unit*) de donde se toman el conjunto de formas D31, dos conjuntos de ubicaciones de usuarios y tres conjuntos sintéticos de cien mil instancias.

La primer muestra corresponde al conjunto con formas arbitrarias D31 (Tabla 5) (Figura 17 a).

Tabla 5. Conjunto D31

No. de vectores	3100
No. de clusters	31
Dimensiones	2
Tamaño de archivo	49.5KB

En la Tabla 6 se describe el cuarto conjunto de datos correspondiente a la base de datos de las ubicaciones de usuarios (Figura 17 b).

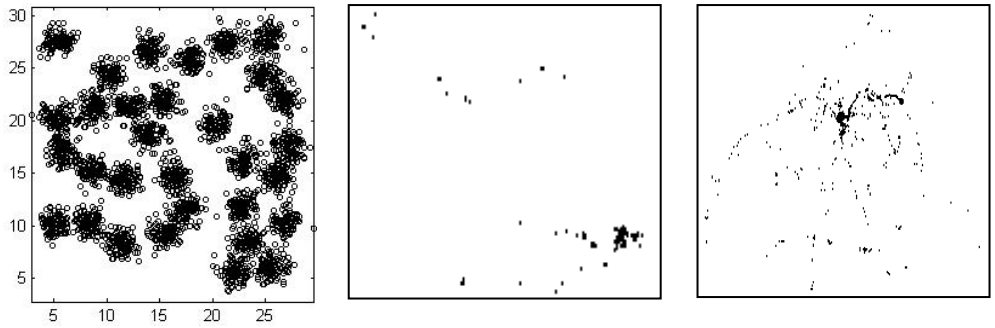
Tabla 6. Conjunto ubicaciones de usuarios

No. de vectores	8589
No. de clusters	-
Dimensiones	2
Tamaño de archivo	155KB

El conjunto de la Tabla 7, corresponde a la ubicación de usuarios en la base de datos Joensuu (Figura 17 c).

Tabla 7. Ubicaciones de usuarios de la base de datos Joensuu

No. de vectores	6014
No. de clusters	-
Dimensiones	2
Tamaño de archivo	110KB



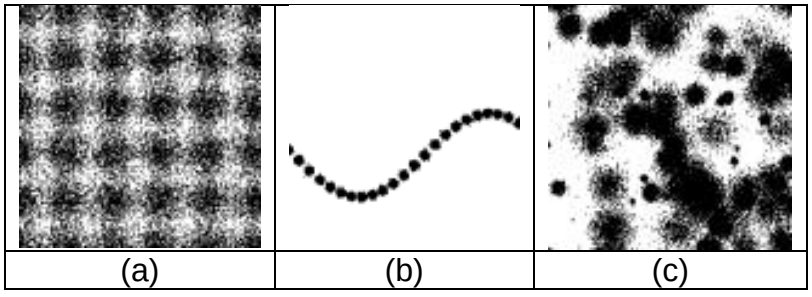
(a) (b) (c)

Figura 17. (a) Conjunto D31. (b) Conjunto Ubicación de usuarios MOPI. (c) Conjunto Ubicación de usuarios Joensuu BD. Extraído de [SIPU, 2012].

Por último en la Tabla 8 se especifican las características de las bases de datos Birch1, Birch2 y Birch3, de los cuales sólo se conoce el número de grupos que contiene, y puede observarse su resultado en la Figura 18a, Figura 18b y Figura 18c.

Tabla 8. Bases de datos Birch

	Birch1	Birch2	Birch3
No. de vectores	100,000	100,000	100,000
No. de clusters	100	100	100
Dimensiones	2	2	2
Tamaño de archivo	2MB	2MB	2MB



(a) (b) (c)

Figura 18. Conjuntos Birch

5.3 Manejo de RAM

Para su aplicación en la metodología, se determina la cantidad de memoria RAM disponible dinámicamente, ya que esta depende del equipo en el que se implemente. Como se mencionó en el marco teórico quien interactúa directamente con la RAM es la JVM, por lo que se utilizó una librería Java de libre distribución, *libSigar*, que obtiene mediante la implementación de un pequeño código el tamaño to-

tal de la RAM, así como el espacio disponible, entre otros datos del sistema. Esta librería maneja el tamaño de memoria en MB, mientras que el espacio disponible lo maneja en kB.

La expansión del tamaño de memoria volátil de la JVM se realizó de acuerdo al espacio disponible de memoria RAM menos un 10%, que puede ser requerido por el sistema durante la ejecución de la aplicación. De acuerdo a este espacio disponible se asigna el 80% a la memoria volátil o *Heap* de la JVM y el 20% a la memoria permanente.

Lo anterior se debe a que en la memoria volátil se almacenan las variables dinámicas, que serán utilizadas por un "periodo de tiempo", mientras que la memoria permanente almacena aquellos datos que estarán presentes durante toda la ejecución del programa, entre ellas se encuentran las librerías, clases, interfaces, entre otras. Como no existe un criterio establecido para determinar la cantidad de memoria en la JVM y esto se deja al programador [Sun, 2006], se considera que al procesar un conjunto, aprovechando la mayor capacidad del equipo, es importante que se tome en cuenta la cantidad de patrones que se cargaran, pero también el uso de memoria por parte de la aplicación.

Pese a que se descargaron conjuntos de datos de gran tamaño, no fue posible conseguir uno que de forma real superara los recursos disponibles de RAM. Por esta situación, se introdujo a la aplicación un valor que simula que el tamaño de la memoria es menor al real, para poder trabajar con la división del conjunto de datos de acuerdo a la capacidad de RAM.

El conjunto de datos se divide de acuerdo al tamaño asignado a la memoria volátil de la JVM. Por lo tanto se obtienen *n* subconjuntos de un tamaño no mayor al especificado. Cada uno de estos subconjuntos es procesado de forma iterativa hasta concluir con toda la muestra.

Para fines de experimentación con los conjuntos de datos obtenidos, el tamaño fijado en la simulación para la memoria, se realiza a partir del tamaño del conjunto de datos en Bytes. Si la el tamaño del conjunto en B no es un número entero entonces este se redondea al siguiente entero, ya que para calcular el número de Bytes cargados en memoria se manejan sólo cantidades enteras. Enseguida la cantidad obtenida como tamaño del conjunto de datos es dividida entre el número de conjuntos con los que se desea realizar la prueba, a esta cantidad se suma el número de patrones existentes en el conjunto, ya que a cada línea hay que agregar el espacio que ocupa el carácter de salto de línea. De esta forma es posible especificar en los experimentos el tamaño total de RAM.

5.4 Algoritmo de agrupamiento

Para realizar la tarea de agrupamiento y recuperación *on-line* de los patrones, se implementó el algoritmo de agrupamiento, primero Batchelor y Wilkins o de máxima distancia. Para su funcionamiento se requiere de la especificación del parámetro θ que, indica la distancia media entre los agrupamientos existentes, y ayuda a calcular el umbral de distancia para reconocer cuando un nuevo grupo debe ser creado (ver 3.2). No existe un valor específico para este parámetro en la literatura, pero se recomienda que su valor sea 0.5, aunque esto depende de las características del conjunto de datos. En la aplicación de este algoritmo en la metodología propuesta se utilizaron los valores para θ de 0.1, 0.2, 0.3, 0.5, 0.7 y 0.9, con RAM definida para simular el procesamiento del conjunto de forma *on-line* y de forma *off-line*.

Para construir el conjunto $\boldsymbol{T}$, que contiene la relación del patrón y su centro más cercano, y conocer el patrón con mayor distancia fue necesario crear de forma temporal una matriz que contuviera la distancia del patrón *i-esímo* a cada uno de los centros existentes y determinar cuál de ellos era más cercano al patrón, así se asignó de forma provisional el patrón al grupo representado por dicho centro. De esta forma fue posible determinar cuál era el patrón más alejado de cada uno de los grupos, y después de obtenerlos compararlos y seleccionar aquel cuya distancia fuera mayor, siendo este el patrón con la mayor distancia.

Para obtener un agrupamiento más fiable, este es realizado con diferentes valores de θ para cada subconjunto. Por tanto, una vez concluido el agrupamiento de los datos para cada subconjunto y cada valor de θ se aplican dos métodos de validación de *cluster*, Davies-Boulin y Calinski-Harabasz, para determinar que tan buenos fueron los agrupamientos generados.

5.5 Regla de Clasificación

Para integrar el sistema de clasificación se utiliza la regla *k*-NN, donde el parámetro *k* se establece en 2, 5, 7 y 9. La aplicación de la regla de clasificación parte de los resultados obtenidos de la división inicial del conjunto de acuerdo al tamaño disponible de memoria. Cada bloque de datos obtenidos de acuerdo al tamaño de memoria es utilizado como CE para entrenar al clasificador.

5.6 Evaluación de *clustering* y rendimiento del clasificador

Debido a que no existe un criterio definido para seleccionar una medida de similitud, y a que en base a diversos estudios que [Arbelaitz, 2010] menciona muchas de ellas son equivalentes, los resultados de la tarea de agrupamiento serán anali-

zados de acuerdo a los criterios internos y externos, siendo los criterios externos utilizados para comparar la partición generada por el algoritmo de agrupamiento con la partición correcta o real, en el caso de la experimentación con bases de datos que cuentan con la información sobre la agrupación correcta, en este caso D31. Para esto se usaron los métodos de *recuento de pares* con la implementación del *coeficiente Jaccard*, *estadística Rand,* el índice Fowlkes y Millos y la *medida F* (ver sección 2.3).

Teniendo en cuenta, de acuerdo a [Arbelaitz, 2010], que no existen comparaciones extensas sobre los índices propuestos en el criterio interno en consecuencia tampoco existe un criterio definido para la selección del índice de validación, por lo que en esta investigación se utilizó el índice *Calinski-Harabaz*, el índice *Davies-Boulin*, para conocer la calidad de la estructura de los grupos, en específico.

En tanto, para la evaluación del rendimiento del clasificador se utilizó la matriz de confusión por su cualidad de mostrar la información no sólo de los aciertos o errores generales de la clasificación, sino también porque permite observar de forma cuantitativa donde se encuentran las principales fallas de clasificación, así como las asignaciones reales de los patrones. También a partir de esta es posible aplicar estadísticas que permitan conocer más información sobre la clasificación. Otras técnicas utilizadas para evaluar el resultado del sistema fueron: la precisión general, para conocer el porcentaje total de aciertos y errores en la clasificación, la desviación estándar, para conocer qué tan lejanos o cercanos se encuentran los resultados de su valor promedio (ver sección 2.3).

CAPÍTULO 6 RESULTADOS

En este capítulo se presentaran los resultados experimentales obtenidos con la metodología propuesta. Se consideran el algoritmo Batchelor y Wilkins para el agrupamiento y la regla k-NN para la clasificación. Los experimentos se realizaron con las bases de datos D31, User location Joensuu, User location MOPI, Birch 1, Birch 2 y Birch 3 con un procesador Intel Celeron E3300, 3 GB en RAM y una unidad de disco duro de 320 GB.

Las bases de datos fueron procesadas realizando una simulación. Donde el tamaño de la base de datos era más grande que la memoria disponible, se estableció un punto de comparación en el tamaño de todas las bases de datos para que fuesen procesadas en tres bloques continuos, así mismo se estableció un tamaño de forma que fuera posible procesarlas de forma completa, situación que no fue posible con las bases Birch2 y Birch3, ya que la memoria no fue suficiente para realizar este procesamiento, por lo que sólo se muestran los resultados obtenidos a través del uso de la metodología propuesta.

En esta sección se muestran los resultados más relevantes, conseguidos con el valor de K = 3, $0.1 <= \theta <= 0.2$. Se inicia con los resultados del CD D31 (que cuenta con los datos del agrupamiento real), para continuar con las bases de datos reales Joensuu y MOPI, concluyendo con los resultados de las bases de datos Birch de las cuales se tiene conocimiento del número de grupos que contiene.

Cabe destacar que en los apéndices se incluyen todos los resultados que se obtuvieron el los experimentos.

6.1 Clustering

El parámetro utilizado para el algoritmo de *clustering* θ influye en el cálculo del umbral para determinar cuando existen nuevos grupos, por lo tanto, es importante elegir un valor que sea lo más óptimo posible para obtener un agrupamiento de calidad. Para la validación del *cluster* de la base de datos artificial D31 se implementó el recuento de pares mediante la matriz de contingencia para calcular los índices externos mencionados en la sección de la metodología (ver sección 2.3). De igual forma se muestran los resultados obtenidos de los índices C-H y D-B.

En adelante se describen los resultados para las bases de datos Joensuu, MOPI y los conjuntos Birch. Para su validación se implementaron los índices Calinski-Harabasz y Davies-Boulin, a partir de cuyos resultados se determina cual es el valor del parámetro θ más conveniente para el procesamiento de los datos. Es im-

portante notar, que en todos los resultados se distingue una tendencia de crecimiento con respecto al valor de θ en ambos índices, esto se toma como referencia principal en el resultado del índice D-B, cuyo valor entre más bajo sea, el agrupamiento resultante será mejor. Como en la mayoría de los resultados observados en el experimento de D-B, se puede ver que en los últimos tres valores de θ (0.5, 0.7 y 0.9) incrementan significativamente con respecto a los tres primeros (θ = 0.1, 0.2 y 0.3), cabe destacar que estos valores se descartan. En la mayoría de los gráficos incluidos, se muestra sólo el área que es más relevante para el análisis, pero que a su vez muestra el comportamiento del agrupamiento a través del valor de los índices. Es así como, se determinan los valores más óptimos de θ, a partir de los cuales se selecciona el mejor en base al resultado de C-H y el número de grupos generado con el valor correspondiente de θ.

6.1.1 CD D31

Para esta base de datos, de acuerdo a la interpretación de los índices de validación, los resultados de Calinski-Harabasz son ascendentes (ver Figura 19), lo que indica que cualquier valor seleccionado es tan bueno como los demás. Este comportamiento resulta casi igual al que se detectó en el resto de las bases de datos consideradas, a excepción de lo encontrado en D31 cuyos valores de θ = 0.5, 0.7 o 0.9 del índice D-B es mayor que C-H, mientras que el valor esperado es alto de C-H y un valor bajo de D-B, con respecto al resto de los valores resultantes de θ.

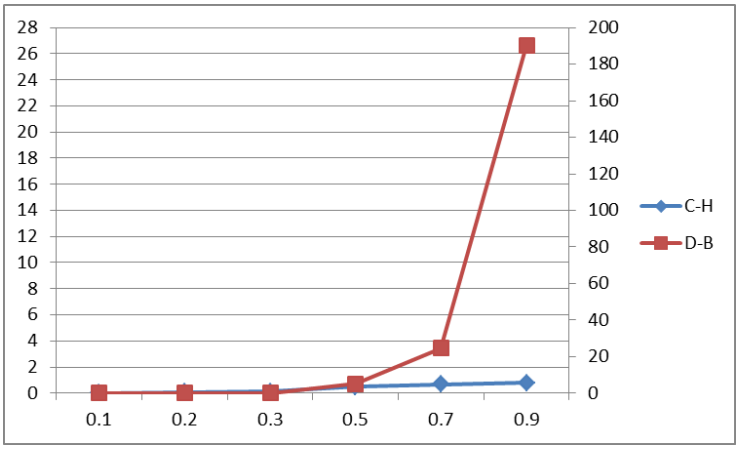


Figura 19. Índices de validación de cluster C-H, D-B para D31 con K=3.

En el caso de la base de datos de D31, los datos que se muestran a continuación toman como referencia el valor de θ = 0.2, ya que se toma en cuenta el número de grupos resultantes con cada valor de θ, 0.2 es uno de los valores que arrojan un agrupamiento cercano en número al original. Otra razón para seleccionarlo es que tiene como resultado en la tarea de clasificación una precisión alta, sin alejarse significativamente del agrupamiento original.

En la Tabla 9 se muestran los valores obtenidos en el recuento de pares (ver sección 2.3.1), de forma *on-line* y de forma *off-line*. Cabe mencionar que en todos los experimentos el primer centro se toma de forma aleatoria.

Tabla 9. Valores del recuento de pares del procesamiento *on-line* y *off-line*

	On-line	Off-line
SS	7965.1	75519
SD	7611.9	68882
DS	7849.5	77931
DD	469424.4	4581118

De forma intuitiva los valores altos de SS y DD indican que existe similitud entre las particiones obtenidas con el algoritmo de *clustering* y las del conjunto original, siempre y cuando los valores de SD y DS sean bajos. Por tanto, de forma rápida puede decirse que la partición realizada por el algoritmo de *clustering* es bastante similar a la original, aunque algunos objetos fueron integrados en grupos diferentes a los que estaban asignados en la partición, esto es debido a que el número de grupos determinado por el algoritmo es diferente al de la partición original, como se explica más adelante.

Es posible observar que los grupos formados, tanto de forma *on-line* como *off-line* son bastante definidos pues el índice DD es alto, aunque en el caso del procesamiento *on-line* el índice SS es más bajo que DS, lo que significa que dentro de los grupos se asignaron patrones que no pertenecían a estos. Lo anterior puede deberse a dos factores, el principal es la influencia del parámetro θ sobre el umbral para formar los grupos, pues a mayor valor de θ mayor radio para el umbral y como consecuencia de esto el número de grupos resultante puede ser diferente al real, como en este caso, lo que provoca que patrones de otros grupos sean asignados a uno que no les corresponde de acuerdo a los datos originales; el segundo factor se debe a la similitud entre los patrones y los centros de grupo, ya que un patrón podría estar en el límite de dos grupos y ser asignado a aquel cuyo centro sea más cercano, que por la ya mencionada influencia del parámetro θ en el umbral, podría no ser asignado al grupo que le corresponde, como se observa en la Figura 20, donde el patrón señalado en la Figura 20b pertenece al grupo señalado en la Figura 20a que corresponde al agrupamiento real. Esta situación se observa en mucho menor medida, pues ajustando el parámetro θ es posible acercarse de forma muy precisa al resultado original, esto puede verse con el valor de $\theta = 0.2$ donde el número de grupos y su asignación es casi igual a la base original que consta de 31 grupos y con el algoritmo se obtiene el número más cercano que es de 37.

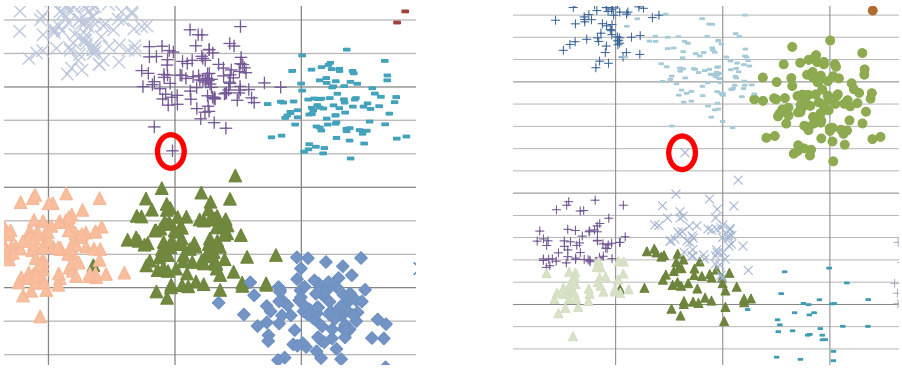


Figura 20. Asignación incorrecta de un patrón. Extracto de la gráfica del agrupamiento real (a) y del agrupamiento *off-line* (b) con el algoritmo de *clustering*.

A partir de los resultados del recuento de pares, se calcularon los índices que se agrupan en la Tabla 10, donde se muestra el resultado de los índices utilizados para valorar la calidad del agrupamiento. De forma general puede observarse en la Tabla 10 que los resultados del procesamiento *off-line* y el *on-line* son muy parecidos, la diferencia entre ellos es de centésimas, además presentan el mismo comportamiento. Esto indica que el procesamiento *on-line* del conjunto de acuerdo a la RAM disponible genera un agrupamiento muy similar al que se obtendría procesando el conjunto *off-line*.

Tabla 10. Resultado de los índices de validación procesando el conjunto D31 de forma *off-line* y *on-line* con $\theta = 0.2$

Índice	Off-line	On-line
Estadística Rand	0.969435926	0.9685451
ARI	0.491336496	0.4895717
Índice Jaccard	0.3396677	0.3390697
Fowlkes y Millows	0.50732666	0.5060091
Medida F	0.88135201	0.896825

El calculó de la estadística *Rand* indica la fracción de aquellos pares que están agrupados de igual manera tanto en el resultado de *clustering* como en la partición original. Mientras más se acerque a la unidad el valor de este índice las particiones serán más similares entre sí. La estadística Rand se enfoca básicamente en calcular la coincidencia entre las particiones comparadas, de tal forma que, como se muestra en la Tabla 10, la concordancia que arroja en el caso de la BD D31 es muy alta.

Por otro lado el ajuste del índice anterior (ARI), de igual forma cuantifica la coincidencia entre las particiones, pero toma en cuenta el índice obtenido y el esperado, de tal forma que si el valor es cero las particiones son independientes y si tiende a uno son iguales, por lo que el valor obtenido indica que le agrupamiento no es cercano al original, pero hay que tomar en cuenta que de acuerdo a [Wagner, 2007]

la significancia de la medida puede verse afectada por la suposición que se hace sobre la distribución.

El coeficiente Jaccard no toma en cuenta los pares de datos que no coinciden (DD), su resultado refleja la proporción de datos que han sido asignados, por lo tanto, de acuerdo a este índice la similitud entre las particiones es muy baja. Mientras que el cálculo del índice Fowlkes y Millows indica la probabilidad de que coincidan los elementos asignados, se encuentren de la misma forma en el agrupamiento y en los datos reales, el valor obtenido indica que la probabilidad es del 50%.

La medida F calcula de forma más precisa que tanto se asemejan los *clusters* resultantes al agrupamiento original, obteniendo una media balanceada de los patrones asignados correctamente respecto al total de los mismos y de los que realmente debían ser asignados, el resultado obtenido en la Tabla 10 indica que la precisión del agrupamiento es buena, siendo los patrones asignados de forma correcta en la mayoría de los casos.

Para ilustrar el resultado de forma gráfica en la Figura 21 se muestra el agrupamiento original, en la Figura 22 se observa el agrupamiento obtenido al procesar el conjunto de forma *off-line* con el algoritmo de *clustering*. Por último la Figura 23 muestra el agrupamiento resultante de procesar el conjunto de forma *on-line* de acuerdo a la capacidad de memoria de la máquina.

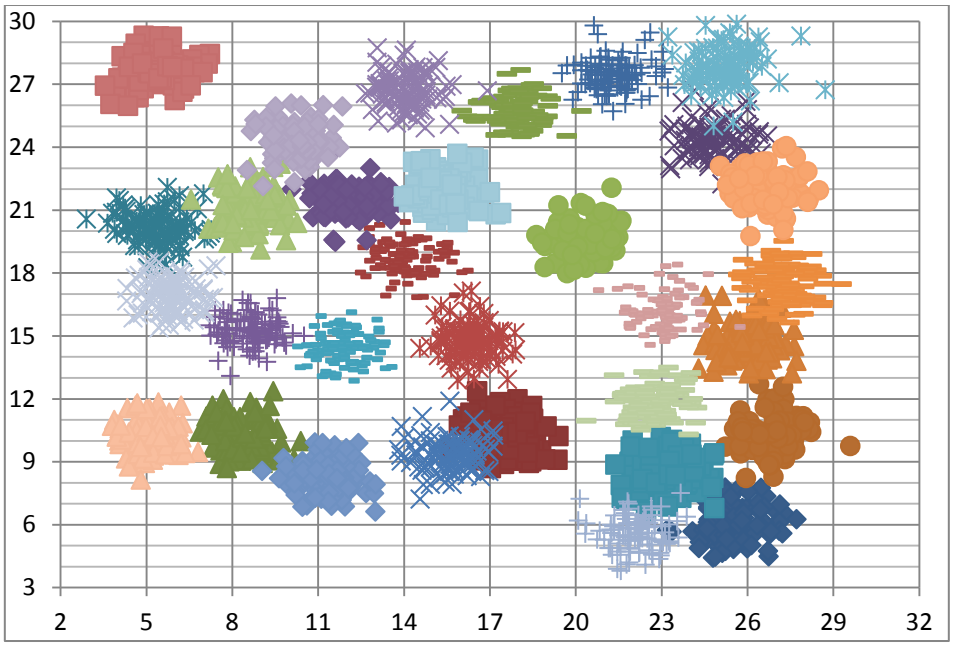


Figura 21. Datos originales de la BD D31.

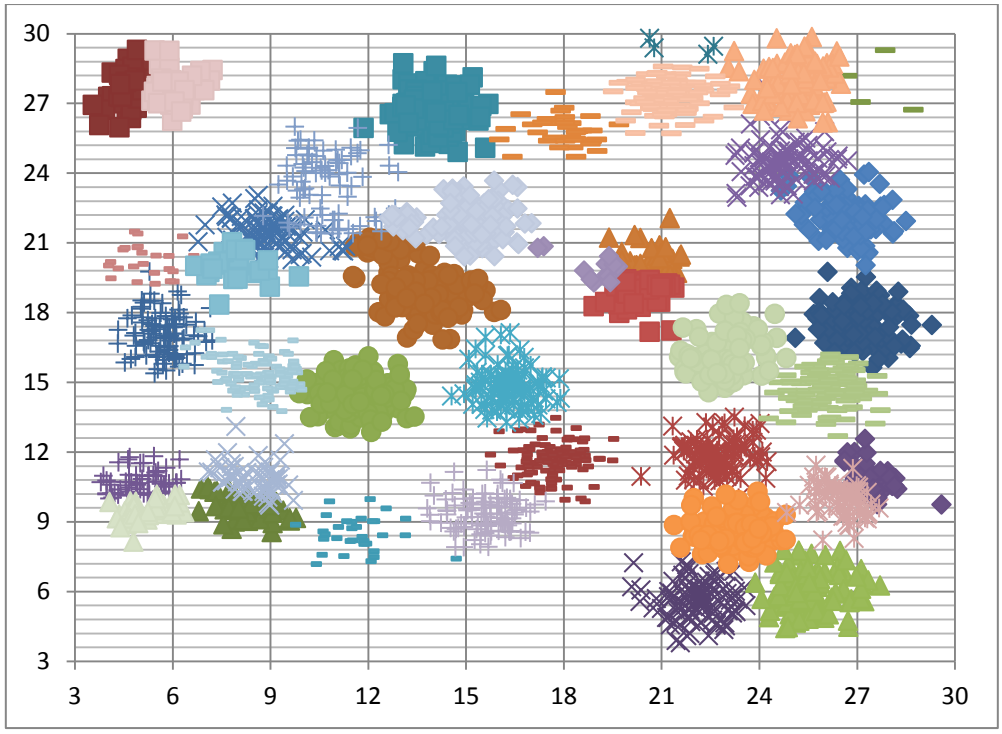


Figura 22. Resultado del algoritmo de *clustering* al agrupar el conjunto de forma *off-line*.

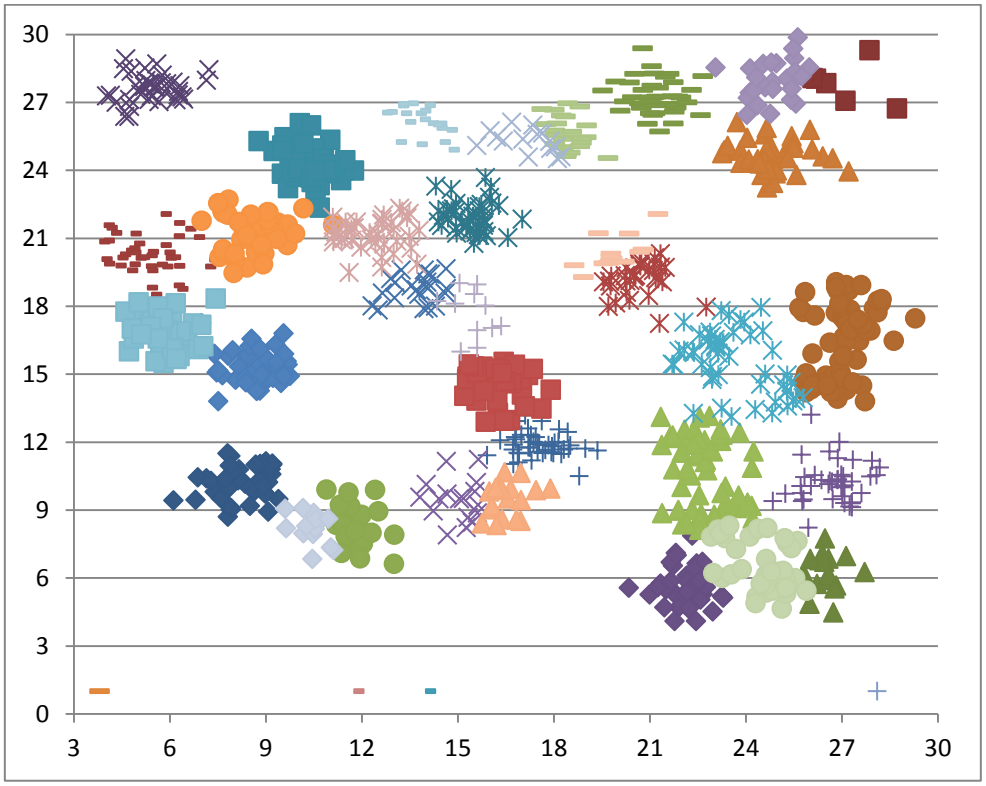
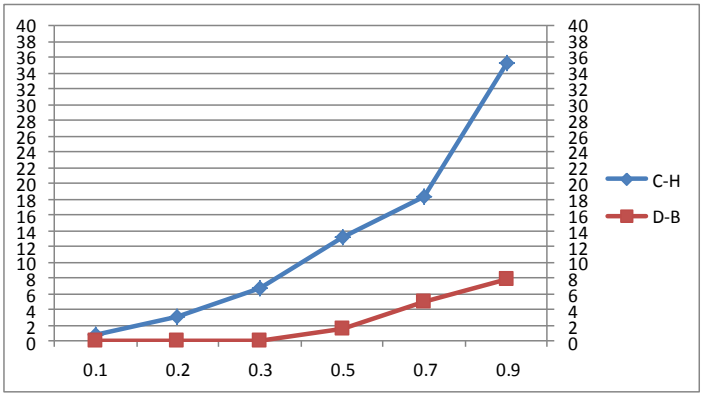


Figura 23. Resultado del algoritmo de *clustering* al agrupar el conjunto de acuerdo al tamaño de memoria.

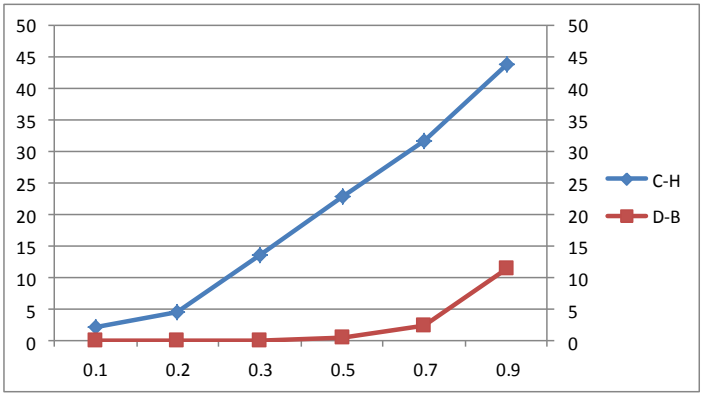
6.1.2 CD Joensuu

En el caso de los CD Joensuu y MOPI, al no contar con ninguna información sobre su agrupamiento se comparan los resultados *on-line* con los del de su procesamiento de forma completa. En la Figura 24 se muestran las gráficas de los agrupamientos resultantes del procesamiento del CD Joensuu, de forma *off-line* (a) y *on-line* (b). De acuerdo a las gráficas se observa que en general el comportamiento de ambos procesamientos lleva la misma tendencia, aunque con $\theta = 0.2$, el valor de D-B sigue siendo pequeño, el C-H es significativamente mayor que con 0.1 y existe mayor similitud en el número de grupos generados, entonces el agrupamiento más balanceado es el obtenido con $\theta = 0.2$.



(a)

C-H	D-B	Grupos
0.782539156	0.000292808	75
3.145888624	0.006643861	29
6.639860423	0.055440428	15
13.12607999	1.599240769	6
18.37597624	4.948387189	4
35.38254563	7.762014636	3



(b)

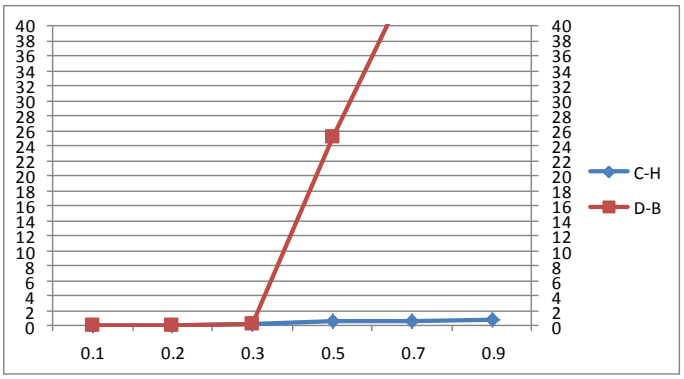
C-H	D-B	Grupos
2.25388551	0.00058929	57
4.45963377	0.00352498	25
13.5032178	0.03621329	12
22.7958235	0.47361588	5
31.7780149	2.49684401	3
43.939682	11.5414901	2

Figura 24. Índices de validación de cluster C-H, D-B para Joensuu con K=3. (a) *off-line*, (b) *on-line*.

6.1.3 CD MOPI

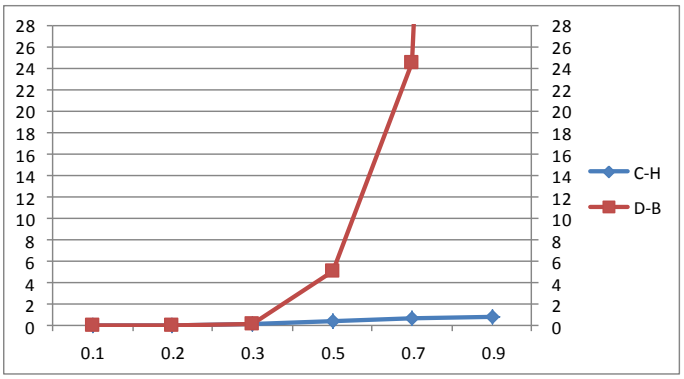
Para el CD MOPI los resultados son similares a los de Joensuu, ya que de igual forma se descartan los últimos tres valores de θ para los cuales el resultado de D-B es mayor a 1. El valor de $\theta = 0.3$ se descarta pues es el valor previo a aquel que se incrementa considerablemente, quedando por lo tanto, los primeros dos valores de θ de los cuales podría seleccionarse cualquiera ya que cubren con lo requerido de acuerdo al resultado de los índices de validación, pero tomando en cuenta el

número de grupos generados en θ = 0.2 obtiene un número de grupos cercano al del procesamiento del conjunto de datos *off-line* (Figura 25).



(a)

C-H	D-B	Grupos
0.019295739	0.000593965	19
0.059647962	0.021311501	11
0.137855536	0.244321076	5
0.538611023	25.09372648	3
0.666525355	46.93172956	2
0.867211814	1068.606773	2



(b)

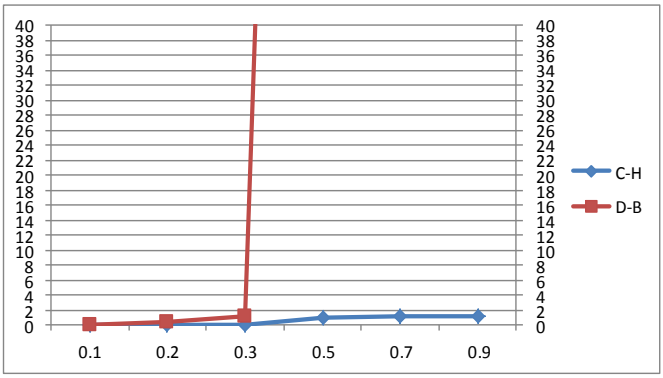
C-H	D-B	Grupos
0.01913797	0.00023392	14
0.07432147	0.00805482	9
0.12976198	0.0851258	4
0.47157364	5.10235998	3
0.66398435	24.5350898	2
0.79557315	190.210248	2

Figura 25. Índices de validación de cluster C-H, D-B para MOPI con K=3. (a) *off-line*, (b) *on-line*.

6.1.4 CD Birch

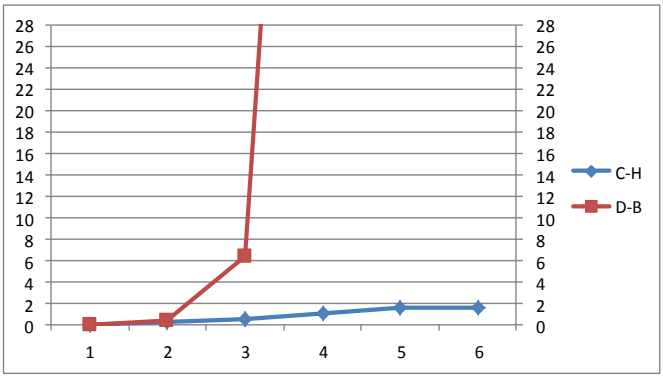
Con lo que respecta a los CD Birch se conoce el número de grupos totales a obtener (100) para cada una. Birch1 es comparada con su resultado al procesarla de forma completa, mientras que los resultados de Birch2 y Birch3 sólo se analizan con los índices C-H y D-B, el número de grupos reales y los datos hasta el momento observados en el comportamiento de θ ya que no fue posible realizar el procesamiento de estos conjuntos de forma completa.

En el conjunto Birch1 los valores más bajos de D-B son θ = 0.1 y 0.2, de estos el valor más alto de C-H corresponde a 0.2, mientras que el número de grupos que se acerca más al que debe obtenerse es de igual forma 0.2, aunque los resultados generados de forma *on-line* sugieren que, de acuerdo al número de grupos y a los dos índices de validación, 0.1 es el valor que agrupa los datos de forma más parecida (Figura 26).



(a)

C-H	D-B	Grupos
0.020293756	0.003401435	227
0.096724693	0.344605275	49
0.093646323	1.074553906	38
0.918546992	298.8428274	4
1.121085977	2942.812182	6
1.131501695	2914.084	3



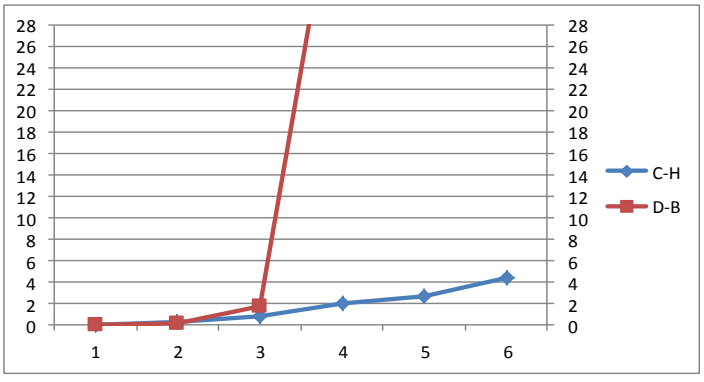
(b)

C-H	D-B	Grupos
0.03746597	0.00493215	140
0.21875289	0.42898651	45
0.52537594	6.43149061	19
1.05781314	113.650933	6
1.65535052	714.917482	2
1.60677907	1061.40863	2

Grupos reales: 100

Figura 26. Índices de validación de cluster C-H, D-B para Birch1 con K=3. (a) *off-line*, (b) *on-line*.

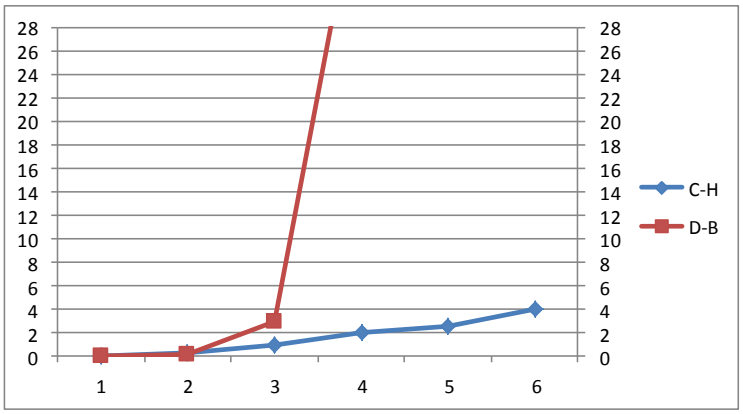
Por último los CD Birch2 (Figura 27) y Birch3 (Figura 28) tienen un comportamiento muy similar, siendo los dos primeros valores de θ los que tienen un índice D-B más bajo y C-H mayor que éste, por tanto, de acuerdo al número de grupos generado $\theta = 0.1$ es el que da mejores agrupamientos.



C-H	D-B	Grupos
0.06158156	0.00355702	106
0.32511817	0.19807755	29
0.83864552	1.76974503	16
2.05864916	46.2465852	4
2.64070931	100.805432	4
4.46470099	1158.88199	2

Grupos reales: 100

Figura 27. Índices de validación de cluster C-H, D-B para Birch2 con K=3. *on-line*



(b)

C-H	D-B	Grupos
0.06332971	0.00351444	113
0.26569742	0.15415016	34
0.95635325	2.94755468	12
2.00847454	42.1127628	5
2.60657187	104.058687	4
4.01601407	481.822349	3

Grupos reales: 100

Figura 28. Índices de validación de cluster C-H, D-B para Birch3 con K=3, *on-line.*

6.2 Clasificación

En el caso de la clasificación, se presentan los resultados de las bases de datos agrupados de tal modo que sea más clara su visualización. Los resultados obtenidos al usar como clasificador la regla del vecino más cercano en su modalidad k-NN [Dasarathy, 1991] al agrupamiento obtenido del algoritmo Batchelor y Wilkins, se muestran en las gráficas correspondientes a las Figura 29 a 37.

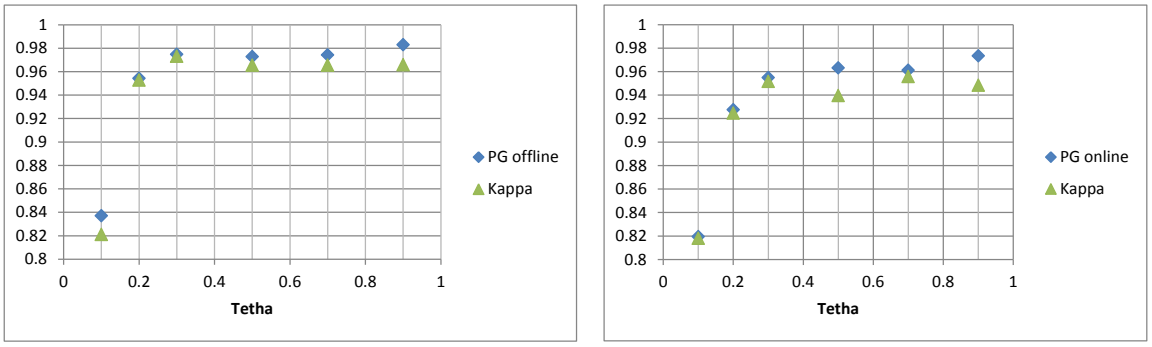
El análisis de los resultados para todos los conjuntos esta ejemplificado con k = 3 de forma *on-line* (a) y completa (b). Con fines de validación, a cada parte del conjunto obtenido se aplicó el método de validación cruzada [Theodoridis, 2006] con dos repeticiones, utilizando un 80% de los patrones para entrenamiento y el 20% restante para el conjunto de prueba, en tanto que para analizar los resultados se utilizó la matriz de confusión, de la que se obtuvieron la precisión general y el coeficiente Kappa (ver sección 2.3.5).

Después se muestra el desempeño obtenido por el clasificador en porcentaje de precisión por conjunto de datos, para cada valor asignado a θ. El eje X representa el valor del parámetro θ, mientras que el eje principal Y el valor de la precisión general, finalmente, la serie indica el conjunto de datos procesado.

En las figuras se observan dos gráficas, donde la imagen (a) corresponde a la precisión general y coeficiente Kappa obtenidos para los conjuntos de datos *off-line*, y en la figura (b) el resultado del procesamiento *on-line*. En todos los resultados siguientes es posible ver varios aspectos, principalmente las precisiones obtenidas tanto con la metodología de procesamiento *on-line*, como con las obtenidas procesando el conjunto de forma *off-line*, sin importar el valor que tome θ son muy parecidas.

6.2.1 CD D31

En la Figura 29 se grafican los resultados obtenidos del procesamiento del conjunto D31 de forma completa (a), con una precisión general de 94.93% y un coeficiente Kappa que indica una concordancia muy buena con todos los valores de θ, ya que los valores del coeficiente Kappa son mayores a .80. Por otro lado, al realizar el proceso *on-line* la PG es de 93.32%, mientras que el valor del coeficiente Kappa al ser mayor a 0.90 indica que la clasificación es muy buena con los valores de θ = 0.3, 0.5, 0.7 y 0.9 y teniendo con θ = 0.1 una concordancia moderada.



(a) (b)

Figura 29. Precisión general y coeficiente Kappa del conjunto D31 con k = 3. (a) *off-line*, (b) *on-line*

6.2.2 CD Joensuu

Para la BD Joensuu la precisión que se obtiene con el conjunto de forma *off-line* es en promedio del 98.06%, y de forma *on-line* es del 99.21%. En tanto que el coeficiente Kappa muestra una concordancia alta en ambos casos, ya que en todos los casos es mayor a .95. Esto indica que el clasificador ha tenido un buen desempeño identificando nuevas instancias (Figura 30).

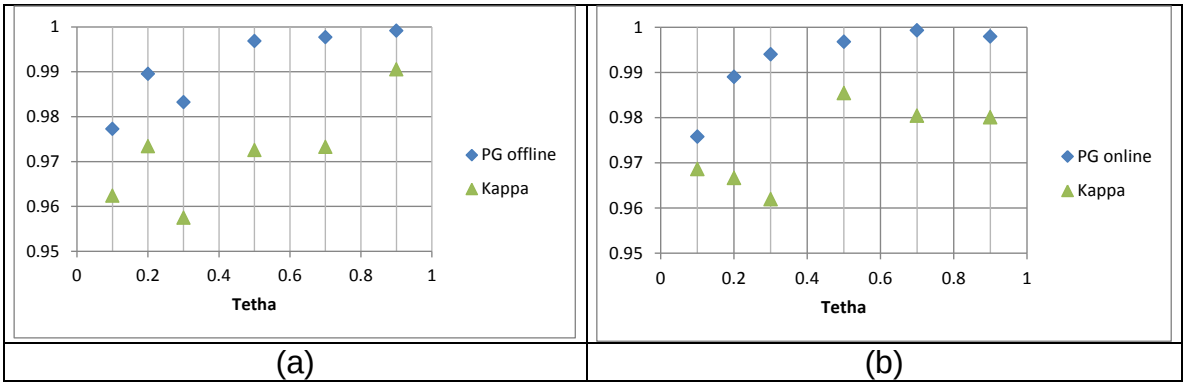


Figura 30. Precisión general y coeficiente Kappa del conjunto Joensuu k = 3. (a) *off-line*, (b) *on-line*.

6.2.3 CD MOPI

El procesamiento del conjunto de datos MOPI se puede ver en las gráficas de la Figura 31. De forma *on-line* supera con 98.98% la precisión obtenida con el procesamiento del conjunto de forma *off-line*, que es de 98.51%. En tanto que el coeficiente Kappa arroja resultados de muy buena concordancia ya que en el caso de los valores para θ = 0.5, 0.7 y 0.9 el valor obtenido es la unidad en los tres casos.

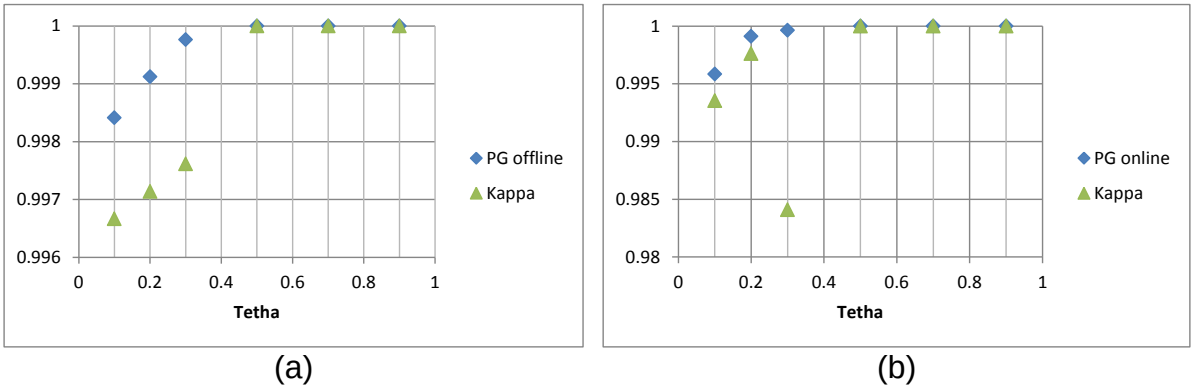
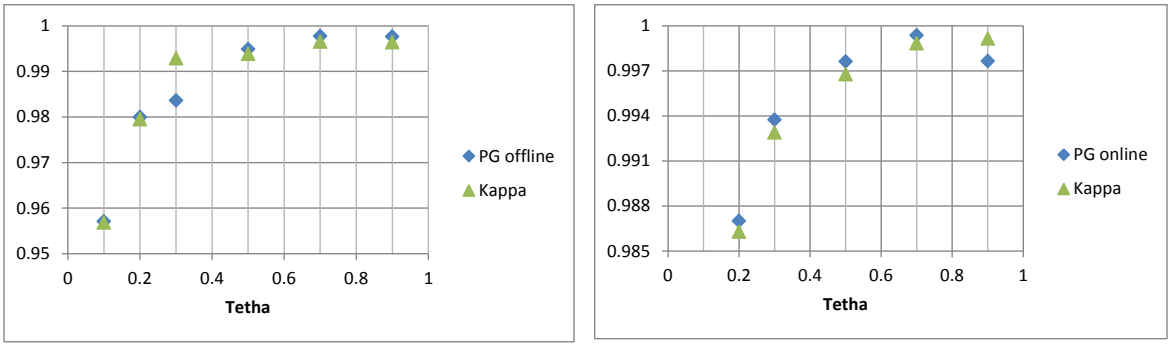


Figura 31. Precisión general y coeficiente Kappa del conjunto MOPI con k = 3. (a) *off-line*, (b) *on-line*

6.2.4 CD BIRCH

Con respecto a las bases de datos Birch se encontró lo siguiente. Para el CD Birch1 se ven los resultados en la Figura 34, donde se observa que el coeficiente Kappa es superior a 0.95 en ambos casos de procesamiento, *off-line* y *on-line*, por lo que se considera una clasificación aceptable.



(a) (b)

Figura 32. Precisión general y coeficiente Kappa del conjunto Birch1 con k = 3. (a) *off-line*, (b) *on-line*

En la Figura 33 se presenta de forma más clara la comparación de la precisión general entre el conjunto de datos *off-line* y de forma *on-line*, notándose que la precisión de forma *on-line* supera por un mínimo porcentaje, 98.98%, al tratamiento del conjunto de forma completa, siendo ésta de 98.51%.

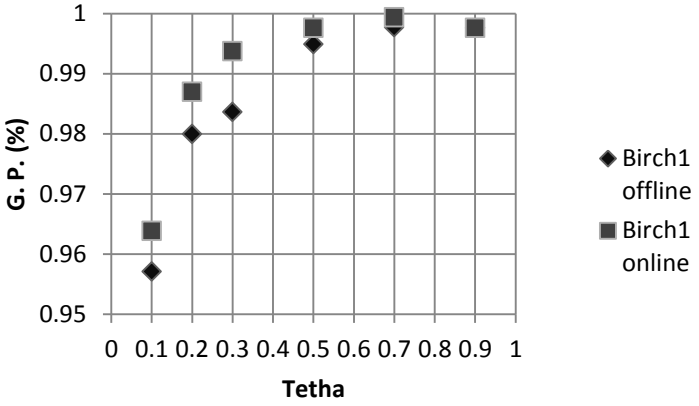


Figura 33. Precisión general del conjunto Birch1 procesado *off-line* y *on-line*.

En adelante podrá observarse claramente una tendencia creciente de la precisión general conforme incrementa θ. En la gráfica (Figura 34) se observa este fenómeno, que es debido a que un valor más grande de θ provoca la creación de grupos de mayor tamaño, teniendo así un espacio de grupos menor pero una muestra mayor de cada uno de los grupos haciendo más fácil para el clasificador realizar la tarea de asignar a un grupo conocido un nuevo elemento. El valor de Kappa refuerza que el desempeño del clasificador ha sido muy bueno, ya que el valor de Kappa es casi de uno en todos los casos.

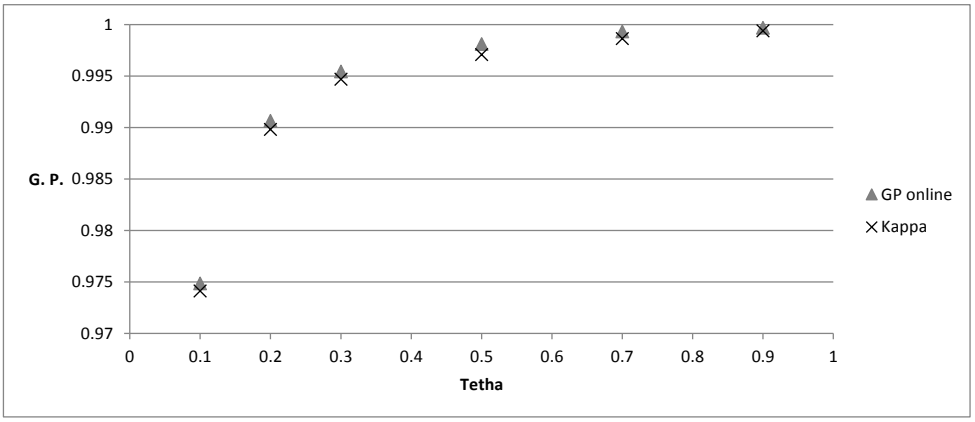


Figura 34. Resultados de Precisión general y coeficiente Kappa para el conjunto Birch2 con k=3.

Por otro lado los resultados de la base Birch3 se muestran en la Figura 35, en la cual se ve nuevamente la tendencia de incrementar la PG y el valor del coeficiente Kappa conforme incrementa θ, aunque esto no significa que el agrupamiento obtenido con el mayor valor de θ sea el mejor, por los motivos mencionados en el análisis de *cluster* (ver sección 6.1).

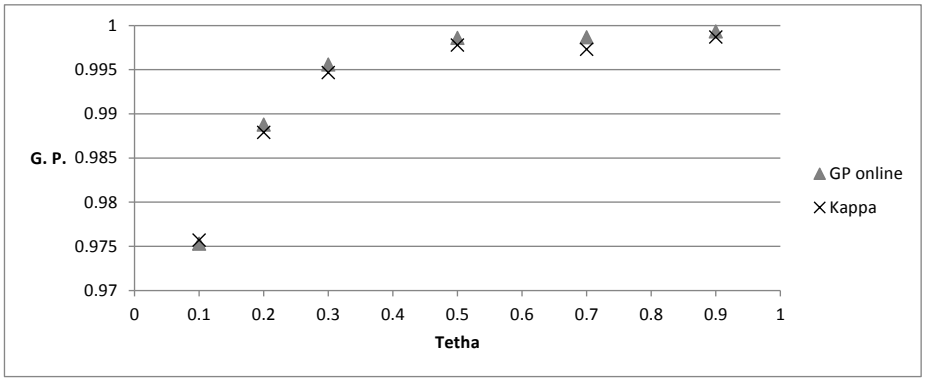


Figura 35. Resultados de Precisión general y coeficiente Kappa para el conjunto Birch3 con k=3.

Por otro lado, el rendimiento del algoritmo de clasificación respecto al resultado de *clustering*, arroja que los valores de θ que mejor rendimiento ofrecen al procesar el CD de forma *off-line* y *on-line* son en su mayoría de 0.3 a 0.9, aunque de acuerdo a los resultados de validación de *cluster*, como ya se había mencionado, el mejor agrupamiento se obtiene con $0.1 <= \theta <= 0.2$, no resultando realmente significativa la diferencia de la precisión obtenida con uno u otro con respecto al resto de valores más grande, pues esta varia por centésimas y en otros casos por milésimas.

Es interesante observar que en general con cualquier valor de θ, procesando el CD de forma *on-line* u *off-line*, el clasificador ofrece una precisión entre 92% y 100% en casi todas las bases de datos probadas, excepto en D31, donde un valor pequeño de θ (0.1) decrece la precisión con respecto a los demás valores a menos del 87%, utilizando cualquiera de los valores asignados a k y llegando a obte-

ner con k = 9 un 68.40% de precisión, por esta razón en la gráfica no se observa este punto.

Lo anterior se debe a que el valor de θ = 0.1 genera un mayor número de grupos pues el umbral determinado es muy pequeño, por lo tanto el radio del grupo también lo es. De esta forma se asignan objetos nuevos en grupos a los que no pertenecen, por su cercanía con alguno de los centros que realmente debería ser parte de un grupo mayor.

6.3 Tiempo de procesamiento

El tiempo requerido en el procesamiento de los CD es un factor importante a analizar, ya que la velocidad en procesamiento es uno de los objetivos que, adicional a la precisión, son perseguidos en minería de datos. En este sentido, el análisis se orienta a validar la propuesta de realizar la escalabilidad del algoritmo de *clustering* al realizar aprendizaje *on-line*, respecto a la utilización del CD en su totalidad cargado en memoria.

La comparativa de los resultados de tiempo del tratamiento de los conjuntos se observa en la Figura 36. Esta figura muestra dos graficas en las que el eje X tiene los valores de θ, en tanto que el eje Y tiene el tiempo requerido para el procesamiento de cada uno de los CD utilizados. Este tiempo es medido en segundos.

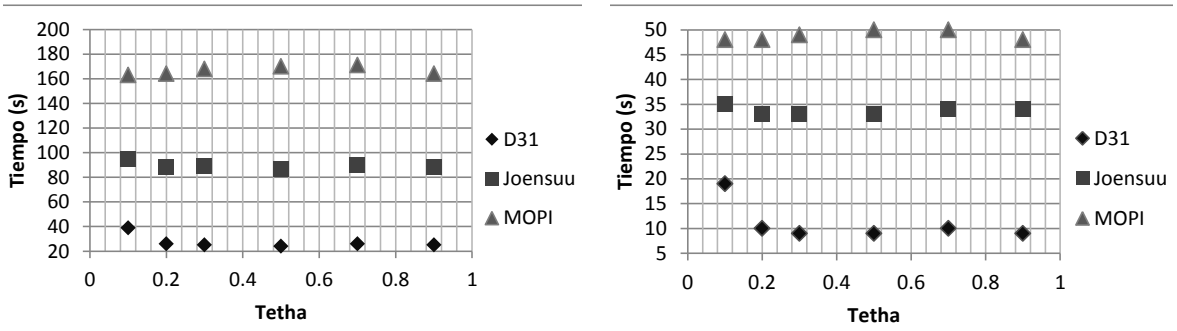


Figura 36. Tiempo de procesamiento obtenido con cada uno de los CD. (a) Tiempo para el conjunto *off-line*, (b) Tiempo para el proceso *on-line*.

En estos resultados es interesante notar que al realizar el procesamiento de los CD con aprendizaje *on-line*, el tiempo es significativamente menor respecto a su contraparte, procesando el CD completo en memoria. Esta situación es sumamente ventajosa tal que al realizar la administración de la memoria se tiene un ahorro hasta del 71% con un rendimiento similar al obtenido cuando el CD completo reside en memoria y se procesa.

Respecto a cada uno de los CD, los resultados arrojan que el procesamiento del CD D31 es 61% más rápido de forma *on-line* que completa. El conjunto Joensuu se procesó en promedio un 62% más rápido de forma *on-line* y el conjunto MOPSI procesado de forma *on-line* tuvo una disminución promedio del 71% de tiempo en contraposición a su tratamiento de forma *off-line*.

De igual forma el conjunto Birch1 (Figura 37) muestra una muy considerable disminución del tiempo en su procesamiento de acuerdo a la memoria RAM disponible, siendo éste de un 96% menos que su tratamiento de forma *off-line*.

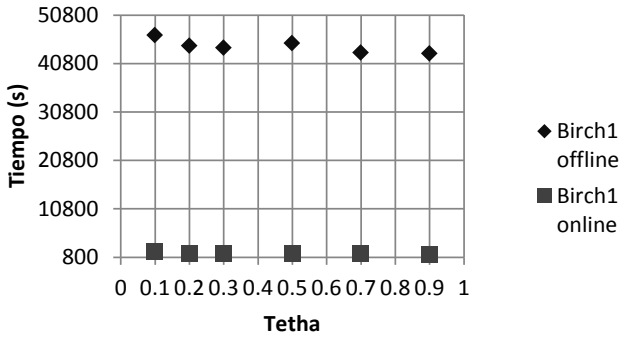


Figura 37. Tiempo de procesamiento obtenido para el CD Birch 1 *off-line* y *on-line*.

Los conjuntos Birch2 y Birch3 se grafican en la Figura 38 y Figura 35Figura 39 respectivamente, en ellas solo se presentan los resultados de tiempo de forma *on-line*. Debido a que no fue posible realizar el procesamiento de estos de forma completa por la limitante de memoria, por lo cual no se cuenta con un punto de comparación, pero es posible hacer notar que el tiempo decrece con los valores mayores a $\theta = 0.2$, aunque también se nota un incremento hacia el mayor de los valores de θ.

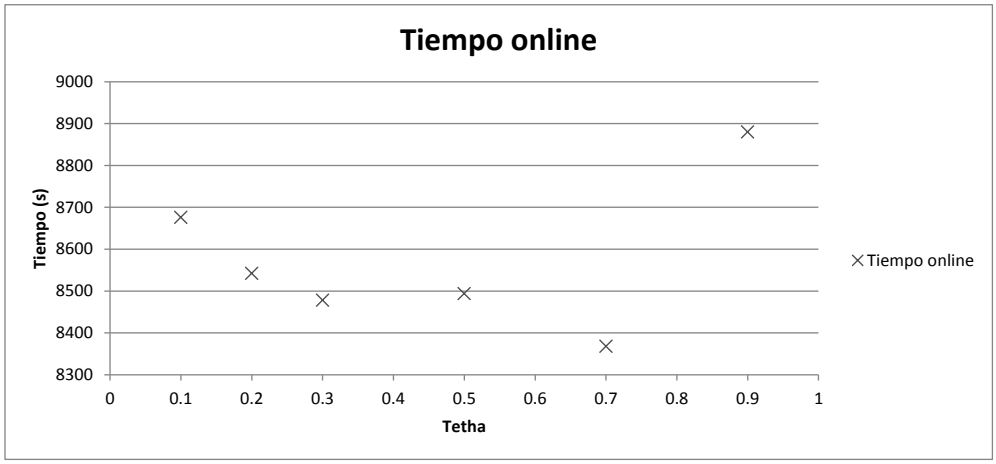


Figura 38. Resultados de tiempo del CD Birch2 con k = 3

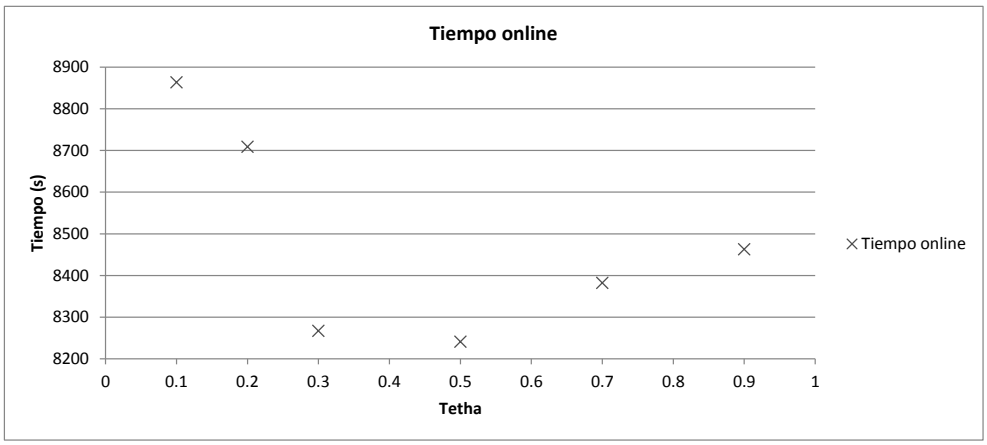


Figura 39. Resultados de tiempo del CD Birch3 con k = 3

En general, para los conjuntos de datos más pequeños se aprecia que la diferencia entre los tiempos del valor de θ más óptimo con respecto al mejor, oscila entre 1 y 15 segundos, siendo en promedio 3 segundos para el procesamiento del conjunto *off-line* y 1 segundo para el *on-line*. Para los conjuntos de datos más grandes (Birch) esta diferencia oscila entre 174 y 468 segundos, siendo la media de 242.44 segundos, hay un aumento significativo si se analiza sólo desde el punto de vista de la metodología, pero debe tomarse en cuenta que de antemano el tiempo de procesamiento con la metodología puede disminuir a más de la mitad con respecto al procesamiento *off-line* lo que hace esta diferencia de tiempo menos significativa.

6.4 Resultado general

Como ya se ha visto los valores de θ y k influyen sobre los resultados finales de la metodología, por lo que su selección resulta un aspecto importante para obtener un mejor balance entre la precisión y el tiempo de procesamiento, ya que el resultado general de procesamiento está estrechamente ligado a los parámetros libres

del algoritmo de *clustering*. En este sentido, es posible notar que, no siempre el valor de θ que arroja la mejor precisión procesa también el conjunto en menor tiempo.

Cuando se procesa el CD de forma *off-line* (ver Tabla 11) la mejor precisión y el menor tiempo son obtenidos con diferente valor de θ (0.9 y 0.5 respectivamente, en dos de los casos). Con respecto al CD MOPI se puede observar un comportamiento un poco diferente, ya que la precisión más alta se obtiene con los valores de θ = 0.5, 0.7 y 0.9 mientras que el menor tiempo de procesamiento se obtiene con θ = 0.1. Para el conjunto de datos Birch1 se muestra un comportamiento más estable entre la precisión y el tiempo, siendo la precisión de 0.9 una diezmilésima menor que la de 0.7 (ver apéndice II). En cuanto a los conjuntos Birch2 y Birch3, como ya se había mencionado, no pudieron ser procesados de forma *off-line* por lo que no se consideran en la tabla.

Tabla 11. Resultados de PG y tiempo de forma *off-line*

CD	Mejor θ en PG	Mejor θ en Tiempo
D31	0.9	0.5
Joensuu	0.9	0.5
MOPI	0.5, 0.7 y 0.9	0.1
Birch1	0.7	0.9

Podemos ver una situación parecida a la descrita al realizar el procesamiento *on-line*, ver Tabla 12. Aquí se obtiene la mejor precisión y tiempo en cuatro de los seis casos. Los CD que proporcionan un mejor rendimiento, de forma *on-line*, con el mismo valor de θ en precisión y tiempo, son D31 y MOPI (θ = 0.9), aunque puede observarse también que ambos CD tienen resultados óptimos con otros valores de θ. No obstante, las diferencias entre los resultados de precisión con diferentes valores de θ y de acuerdo al modo de procesamiento son poco significativas ya que la precisión obtenida en su equivalente con el CD procesado de forma *off-line* es muy similar en todos los caso.

Tabla 12. Resultados de PG y tiempo de la metodología

CD	Mejor θ en PG	Mejor θ en Tiempo
D31	0.9	0.3, 0.5 y 0.9
Joensuu	0.7	0.2, 0.3 y 0.5
MOPI	0.5, 0.7 y 0.9	0.1, 0.2 y 0.9
Birch1	0.7	0.9
Birch2	0.9	0.5
Birch3	0.9	0.5

Con lo que respecta a los conjuntos Birch1 y Birch2 puede notarse un comportamiento muy similar, esto puede deberse a que ambos conjuntos comparten la alta

densidad de los grupos, su tamaño regular y su cercanía, mientras que los grupos de Birch1 también son densos, sin embargo la cercanía de estos y su tamaño es distinta para cada grupo.

Por tanto, seleccionando el valor más óptimo para el agrupamiento de cada conjunto ($0.1 <= \theta <= 0.2$) y tomando en cuenta que el valor de k = 3 ha sido el que genera mejores resultados, el tiempo y la precisión se presentarían de la Tabla 13.

Tabla 13. Resultados de la metodología comparando el valor más óptimo con los mejores valores de θ

CD	θ	PG	Mejor PG	Kappa	Mejor Kappa	Tiempo (s)	Mejor tiempo
D31	0.2	0.92747469	0.97331112	0.924761372	0.955742032	3	3
Joensuu	0.2	0.9889748	0.9992904	0.96664916	0.98543361	33	33
MOPI	0.2	0.9991205	1	0.99761623	1	48	48
Birch1	0.1	0.9869991	0.999342	0.96333783	0.99881998	1594	1419
Birch2	0.1	0.9906483	0.999696	0.97412617	0.99939169	8542	8368
Birch3	0.1	0.9887714	0.9993271	0.9756951	0.9986938	8709	8241

Como se ha mencionado hasta el momento la selección del valor de θ más óptimo, de acuerdo al análisis de los índices de validación (sección 6.1), los resultados de precisión (sección 6.2) y tiempo (sección 6.3) en conjuntos de datos pequeños o de tamaño medio no causan mayor impacto sobre el desempeño de la metodología, mientras que en conjuntos de gran tamaño se ve un incremento de tiempo, pero de forma general el promedio de mejora de tiempo utilizando la metodología *on-line* es mayor al 50%.

PARTE IV
CONCLUSIONES

CAPÍTULO 7 CONCLUSIONES

En esta tesis se describió y desarrollo una metodología para el tratamiento de conjuntos de datos de gran tamaño, para ello se planteó la administración de RAM, el agrupamiento inicial de los datos y su posterior clasificación. Con la administración de RAM se cubrió un aspecto que podría significar una limitante para el tratamiento de estos conjuntos de datos, ya que gracias a la asignación directa de memoria de acuerdo al tamaño de conjuntos de datos fue posible simular el procesamiento de éstos de forma *on-line* utilizando la información que puede brindar todo el conjunto de datos, ya que ninguna instancia es descartada de la muestra, como en otros métodos.

Esta manipulación de memoria permitió no solo procesar la totalidad de los patrones, como se tuvo oportunidad de experimentar con las bases Birch2 y Birch3 que no pudieron procesarse de forma completa debido a que la cantidad de RAM no era suficiente, por lo que se procesó con la metodología de forma satisfactoria. También agilizó el procesamiento de estos conjuntos de datos y permitió designar la mayor cantidad de RAM disponible a esta tarea.

Para conocer el desempeño de ésta se analizaron dos aspectos fundamentales, el rendimiento y el tiempo de procesamiento. Los resultados obtenidos muestran que, tanto en la simulación del aprendizaje *on-line* como en la utilización del CD residente en su totalidad en memoria, la precisión es muy similar.

En general la precisión del clasificador arrojada para todas las bases de datos en su colectividad es mayor al 94% con casi cualquier valor de θ, lo que indica que se ha realizado un buen agrupamiento de los datos, claro y definido, es decir, cada grupo contiene objetos muy parecidos entre sí pero lo suficientemente diferentes a los contenidos en el resto de los grupos de tal forma que son diferenciados por el clasificador de forma correcta.

Lo anterior también indica que se cuenta con una buena muestra para que el clasificador sea capaz de reconocer y clasificar correctamente los objetos nuevos. Aunque la alta precisión de todos los valores de θ en los conjuntos de datos no significa que todos los agrupamientos sean los mejores, ya que al contar con CD con los que se puede comprobar si el número de grupos obtenidos es correcto, se observó que asignando valores grandes a θ (0.5, 0.7 y 0.9) el número de grupos obtenido es significativamente menor que el esperado, por lo que podría considerarse que θ = 0.1 y 0.2, e incluso θ = 0.3 sería un valor aceptable para asignar al parámetro de *cluster*.

Por lo anterior asignar un valor grande a θ significa que no se espera que los grupos sean muy densos o que deliberadamente sea asignado para generar agrupamientos de mayor tamaño que contengan instancias similares de otros grupos. Otro dato importante que debe mencionarse es que aparte de incrementarse la precisión conforme lo hace θ, entre mayor sea el número de datos a procesar la precisión también aumenta. No obstante la reducción en el tiempo requerido por el aprendizaje *on-line* es significativamente inferior.

De igual forma se pudo observar que entre más grande es el conjunto de datos la disminución de tiempo en su tratamiento de forma *on-line* aumenta de forma muy relevante llegando a tener en los experimentos realizados hasta un 96% de ahorro en el tiempo de procesamiento, lo que resulta muy conveniente cuando se cuenta con grandes cantidades de datos que deben ser procesados.

Los resultados obtenidos en la tesis fueron publicados en los tres artículos que se listan, los cuales se adjuntan en el Apendice IV:

1. Santibáñez S., M., Valdovinos R., R. M., Rendón, E., Alejo, R., Marcial-Romero J. R., (2013). Optimización de Recurso para el Tratamiento de Grandes Volúmenes de Datos. *Research in Computing Science Avances en Inteligencia Artificial 62,* 15-24.
2. Santibáñez S., M., Valdovinos R., R. M., Rendón, E., Trueba, A., Alejo E., R., López, E. Applicability of cluster validation indexes for large data sets. IEEE proceedings (aceptado para su publicación).

7.1 Líneas abiertas

De acuerdo al procesamiento del conjunto de datos y a la capacidad de memoria RAM, se obtienen como resultado clasificadores individuales, cada uno de los cuales, podrían ser integrados en un Sistema Múltiple de Clasificación, que proporcionen la herramienta necesaria para consensar las decisiones individuales y obtener la decisión final sobre un patrón nuevo.

Otra de las líneas abiertas se orienta hacia la implementación de algoritmos de opción de rechazo para determinar si aquellos patrones que ingresan al clasificador, de forma dinámica, pueden ser analizados y marcados para ser agregados como parte del conocimiento o para ser descartados por su marcada diferencia con aquellos patrones existentes. De esta forma se intenta disminuir los patrones atípicos o ruidosos que podrían disminuir el rendimiento del clasificador.

De igual modo, como objetivo futuro, se podría realizar la integración de todos estos elementos, para conformar una metodología completa con los Sistemas Múlti-

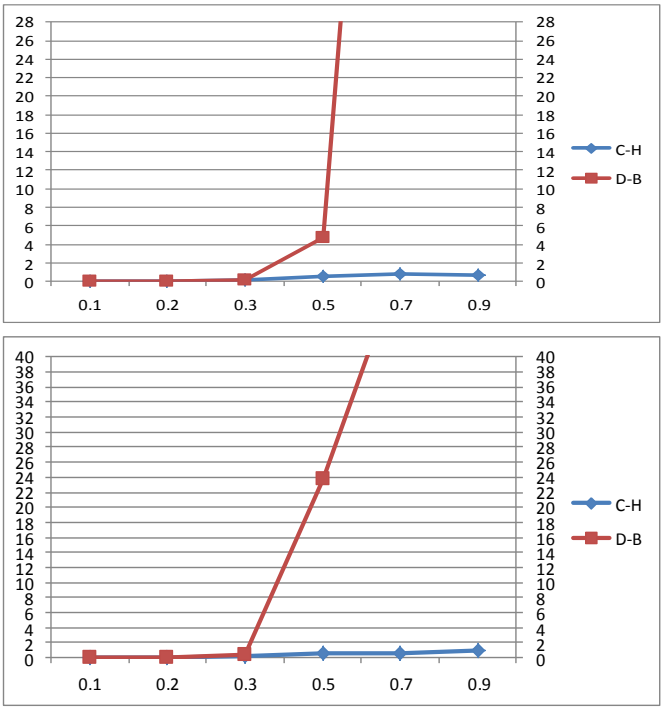
ples de Clasificación que proporcione una vía para el tratamiento de grandes conjuntos de datos de forma *on-line* y brinde resultados en tiempo real fiables.

Por último, la utilización de otras reglas de decisión son opciones que no se descartan en su uso, tales como memorias asociativas, redes neuronales u otra regla que generalice el conocimiento y produzca mayor optimización del uso de memoria temporal del equipo. Es así que esta metodología podría ser probada también con otras tareas relativas a la minería de datos, no sólo con las tareas de *clustering* y clasificación.

APÉNDICES

APÉNDICE I
CLUSTERING

Índices de validación para la BD D31



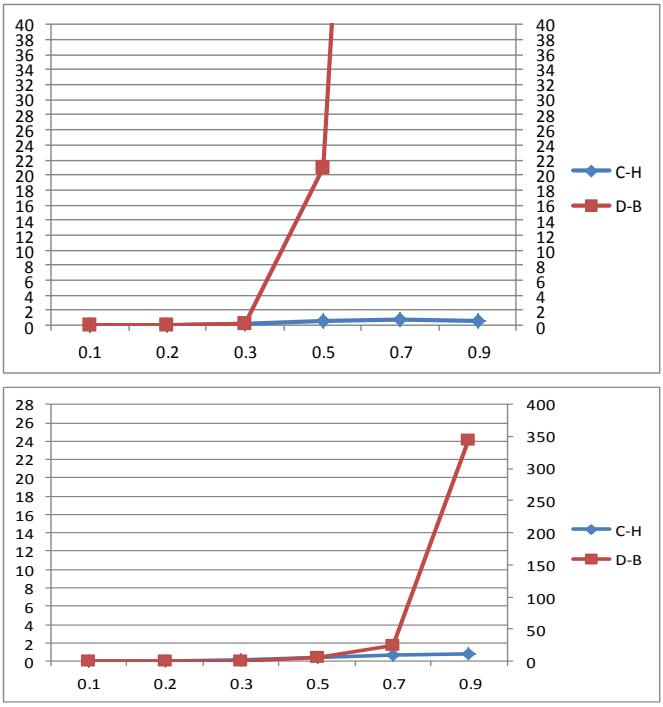
(a)

θ	C-H	D-B	Grupos
0.1	0.01947072	0.00053519	132
0.2	0.04892275	0.01333151	45
0.3	0.16799986	0.43412568	16
0.5	0.55035664	23.7620585	5
0.7	0.60722926	52.3200938	4
0.9	0.86369438	1068.85176	2

(b)

θ	C-H	D-B	Grupos
0.1	0.01863317	0.000206229	134
0.2	0.064557621	0.006740524	44
0.3	0.149812749	0.090672957	16
0.5	0.514115228	4.739736545	6
0.7	0.740082548	107.911945	3
0.9	0.729421456	106.5017051	3

Figura A1.1 D31 con k = 1. Procesamiento (a) *off-line*, (b) *on-line.*



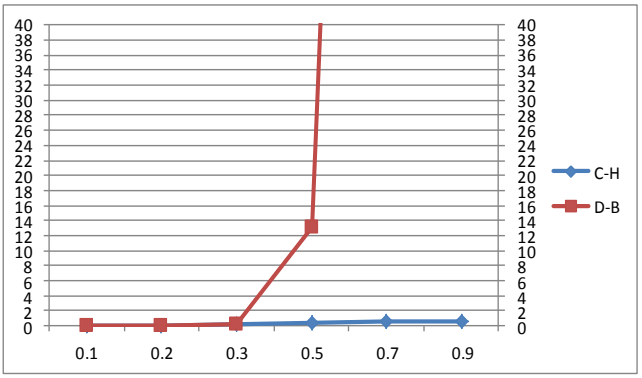
(a)

θ	C-H	D-B	Grupos
0.1	0.014961087	0.000480749	144
0.2	0.054679764	0.014788304	43
0.3	0.131248405	0.255260346	19
0.5	0.610729892	20.89813107	5
0.7	0.687843861	191.2955794	3
0.9	0.585530835	252.0493542	3

(b)

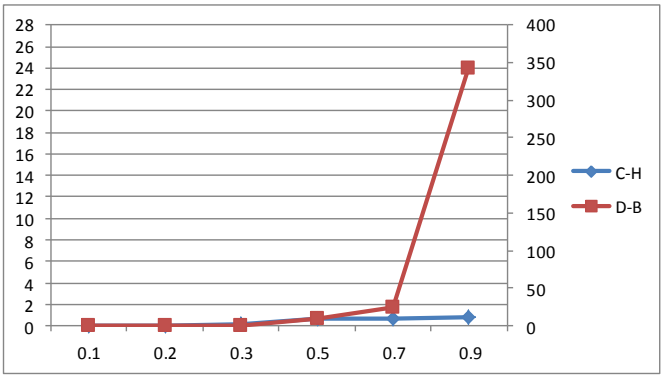
θ	C-H	D-B	Grupos
0.1	0.01930563	0.0002065	133
0.2	0.06729698	0.00816724	40
0.3	0.14234211	0.10738358	21
0.5	0.41755609	6.24703719	6
0.7	0.66093322	24.2465994	4
0.9	0.8526347	345.327651	2

Figura A1.2 D31 con k = 5. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

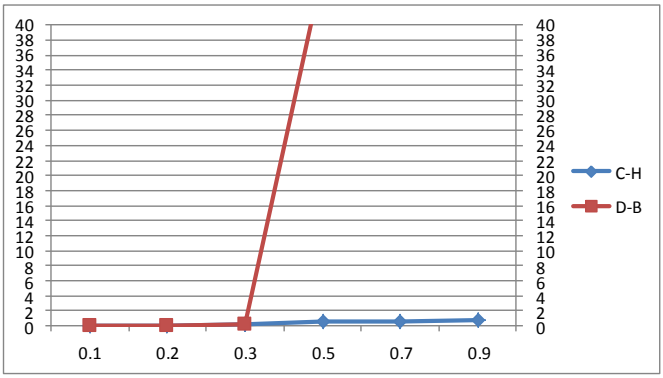
θ	C-H	D-B	Grupos
0.1	0.015671836	0.000478203	142
0.2	0.083577562	0.024908538	34
0.3	0.152664092	0.265516929	18
0.5	0.448587015	13.09282389	6
0.7	0.585897122	251.7385894	3
0.9	0.669287479	184.6092464	3



(b)

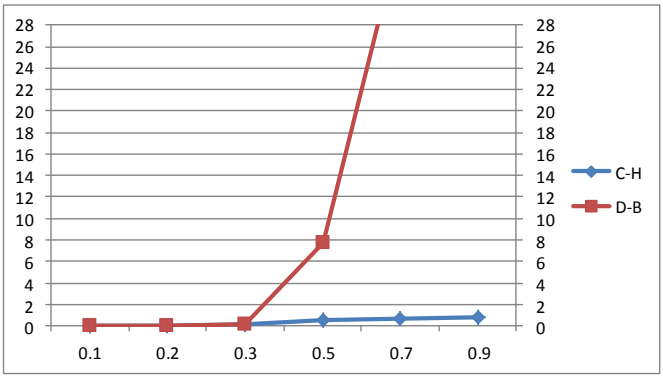
θ	C-H	D-B	Grupos
0.1	0.01865747	0.00020704	129
0.2	0.06833351	0.0073215	37
0.3	0.13853061	0.09458746	18
0.5	0.60478517	9.1646778	4
0.7	0.65628489	25.11171	4
0.9	0.8591545	343.010864	2

Figura A1.3 D31 con k = 7. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

θ	C-H	D-B	Grupos
0.1	0.015727907	0.000486578	141
0.2	0.052890305	0.012401531	45
0.3	0.144099768	0.217319018	19
0.5	0.669133324	47.56798425	4
0.7	0.575766979	250.0863152	3
0.9	0.835120417	1143.135907	2

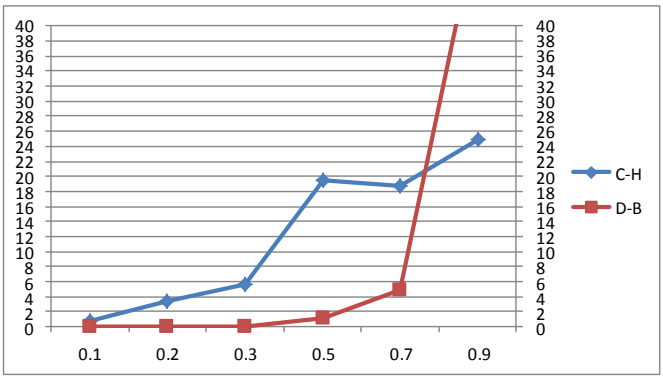


(b)

θ	C-H	D-B	Grupos
0.1	0.01604841	0.00019557	126
0.2	0.0735157	0.0086674	31
0.3	0.14509982	0.08158683	24
0.5	0.53987985	7.68643513	5
0.7	0.67711239	36.3432651	4
0.9	0.76081913	209.517371	2

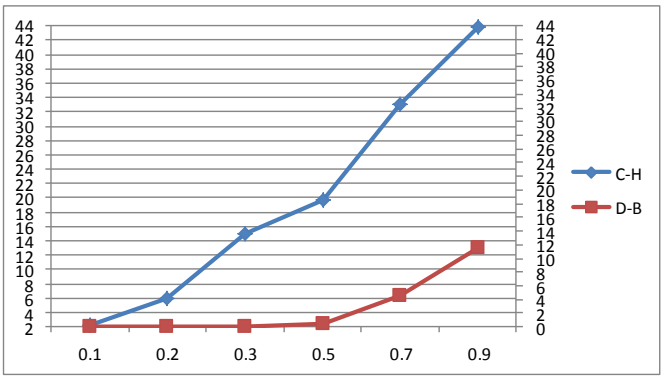
Figura A1.4 D31 con k = 9. Procesamiento (a) *off-line*, (b) *on-line.*

Índices de validación para la BD Joensuu



(a)

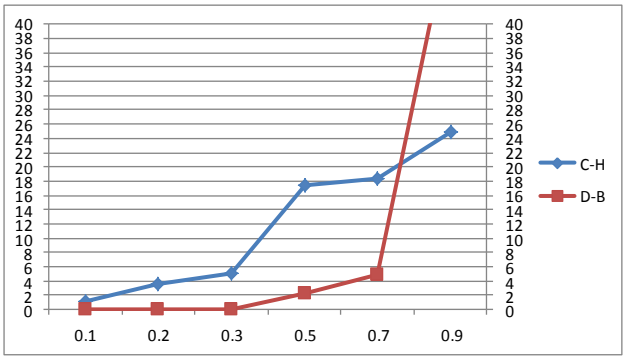
θ	C-H	D-B	Grupos
0.1	0.676676218	0.00025888	80
0.2	3.327652671	0.008234053	28
0.3	5.555151677	0.076236516	14
0.5	19.44970672	1.097334778	6
0.7	18.64649588	4.911203666	4
0.9	24.90420039	54.25988271	2



(b)

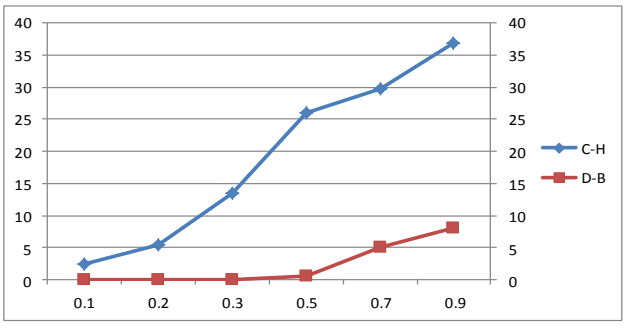
θ	C-H	D-B	Grupos
0.1	2.18340116	0.00060945	58
0.2	5.92866982	0.0045426	22
0.3	14.9830977	0.03850147	11
0.5	19.6177191	0.33210353	5
0.7	33.0528456	4.45792523	2
0.9	43.939682	11.5654971	2

Figura A1.5 Joensuu con k = 1. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

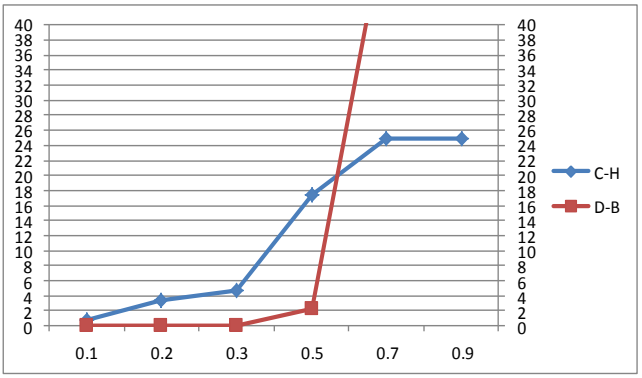
θ	C-H	D-B	Grupos
0.1	1.05080985	0.00046208	67
0.2	3.53663694	0.00880362	27
0.3	5.02123871	0.04614614	16
0.5	17.3656443	2.26833825	5
0.7	18.3759762	4.94653914	4
0.9	24.9042004	54.2606814	2



(b)

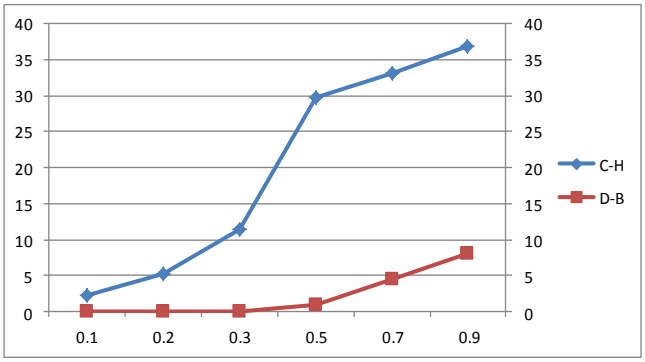
θ	C-H	D-B	Grupos
0.1	2.51908009	0.000701346	58
0.2	5.37578231	0.00495661	25
0.3	13.5032178	0.036210386	12
0.5	26.0976284	0.501600813	5
0.7	29.7625759	4.987667881	2
0.9	36.8470159	8.122022801	2

Figura A1.6 Joensuu con k = 5. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

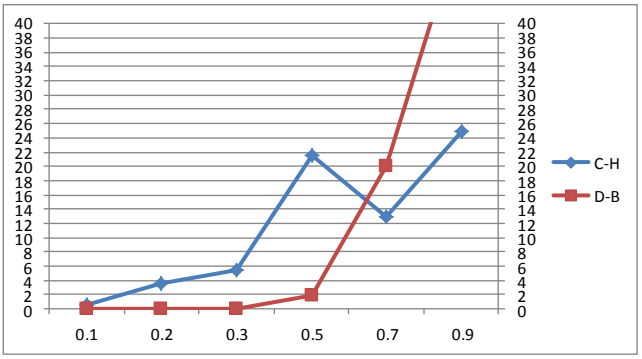
θ	C-H	D-B	Grupos
0.1	0.74859801	3.32E-04	76
0.2	3.32765267	0.00823458	28
0.3	4.68429665	0.04008894	17
0.5	17.3656443	2.26826472	5
0.7	24.9042004	54.3225656	2
0.9	24.9042004	54.2598827	2



(b)

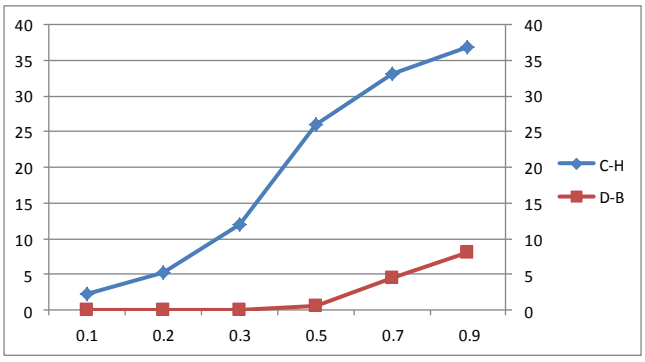
θ	C-H	D-B	Grupos
0.1	2.32476165	0.00064295	60
0.2	5.30286235	0.00475569	25
0.3	11.5240144	0.03177649	12
0.5	29.7489761	0.93172781	3
0.7	33.0528456	4.46208351	2
0.9	36.8470159	8.12334895	2

Figura A1.7 Joensuu con k = 7. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

θ	C-H	D-B	Grupos
0.1	0.65803987	0.00024998	82
0.2	3.50561091	0.00804629	27
0.3	5.41544514	0.05727226	25
0.5	21.6159128	1.86996688	5
0.7	12.8405985	20.0495178	3
0.9	24.9042004	54.2602904	2

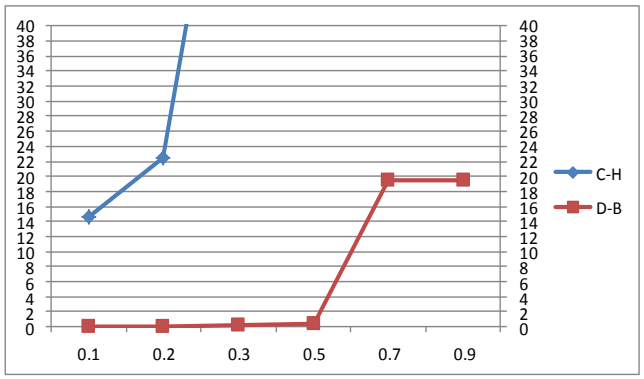


(b)

θ	C-H	D-B	Grupos
0.1	2.20959755	0.00065667	56
0.2	5.29835014	0.00405071	26
0.3	12.0479448	0.02901312	12
0.5	26.0976284	0.50159908	5
0.7	33.0528456	4.45790705	2
0.9	36.8470159	8.12536787	2

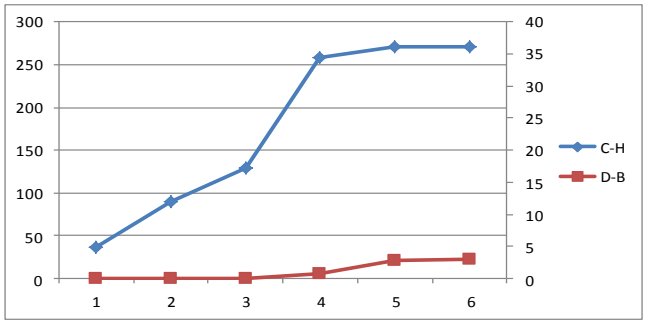
Figura A1.8 Joensuu con k = 9. Procesamiento (a) *off-line*, (b) *on-line.*

Índices de validación para la BD MOPI



(a)

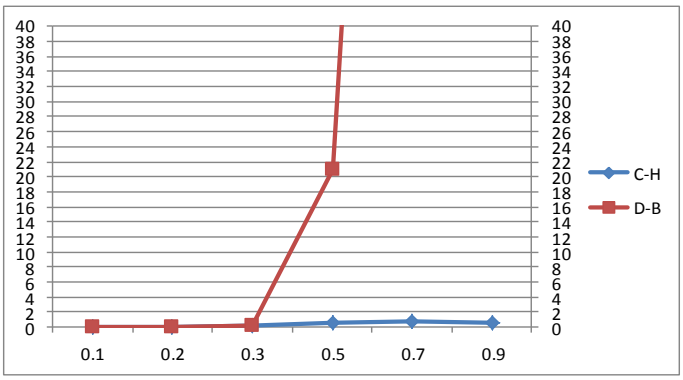
θ	C-H	D-B	Grupos
0.1	14.6902931	0.000629739	19
0.2	22.46897992	0.010311538	11
0.3	79.83362334	0.187973637	5
0.5	101.8802689	0.325017256	4
0.7	142.6076103	19.55209778	2
0.9	142.6076103	19.53996705	2



(b)

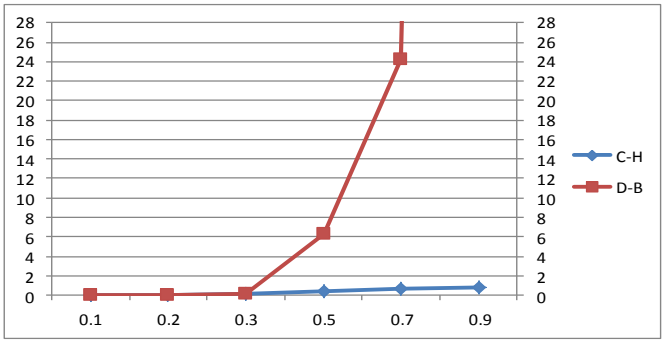
θ	C-H	D-B	Grupos
0.1	37.2809699	0.00154509	14
0.2	90.125306	0.03547567	9
0.3	128.921094	0.06996742	5
0.5	258.160921	0.73429833	3
0.7	271.638436	2.91688168	2
0.9	271.638436	2.92679426	2

Figura A1.9 MOPI con k = 1. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

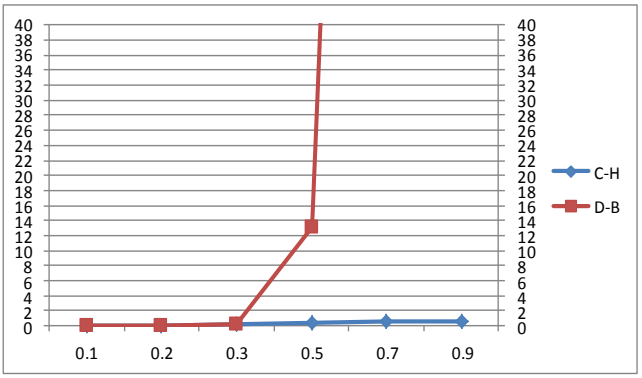
θ	C-H	D-B	Grupos
0.1	0.014961087	0.000480749	19
0.2	0.054679764	0.014788304	11
0.3	0.131248405	0.255260346	7
0.5	0.610729892	20.89813107	4
0.7	0.687843861	191.2955794	2
0.9	0.585530835	252.0493542	2



(b)

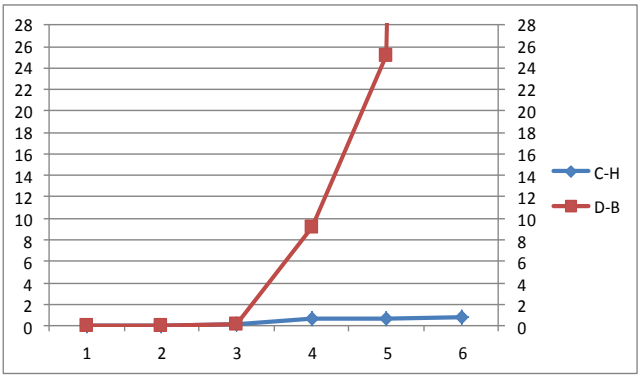
θ	C-H	D-B	Grupos
0.1	0.01930563	0.0002065	14
0.2	0.06729698	0.00816724	9
0.3	0.14234211	0.10738358	5
0.5	0.41755609	6.24703719	3
0.7	0.66093322	24.2465994	2
0.9	0.8526347	345.327651	2

Figura A1.10 MOPI con k = 5. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

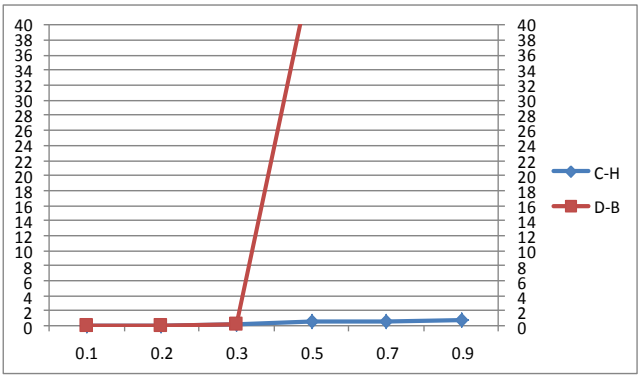
θ	C-H	D-B	Grupos
0.1	0.015671836	0.000478203	19
0.2	0.083577562	0.024908538	11
0.3	0.152664092	0.265516929	7
0.5	0.448587015	13.09282389	4
0.7	0.585897122	251.7385894	2
0.9	0.669287479	184.6092464	2



(b)

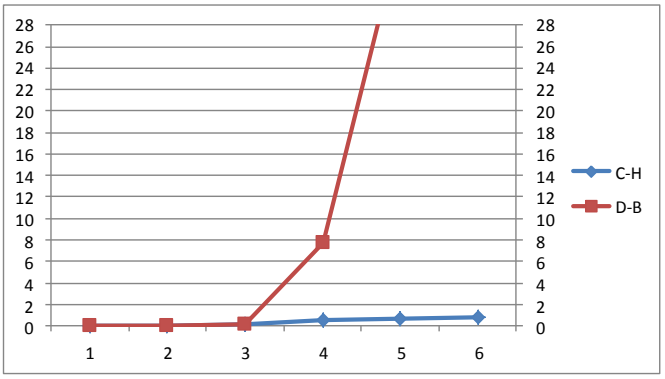
θ	C-H	D-B	Grupos
0.1	0.01865747	0.00020704	14
0.2	0.06833351	0.0073215	9
0.3	0.13853061	0.09458746	5
0.5	0.60478517	9.1646778	3
0.7	0.65628489	25.11171	2
0.9	0.8591545	343.010864	2

Figura A1.11 MOPI con k = 7. Procesamiento (a) *off-line*, (b) *on-line*.



(a)

θ	C-H	D-B	Grupos
0.1	0.015727907	0.000486578	19
0.2	0.052890305	0.012401531	11
0.3	0.144099768	0.217319018	7
0.5	0.669133324	47.56798425	4
0.7	0.575766979	250.0863152	2
0.9	0.835120417	1143.135907	2

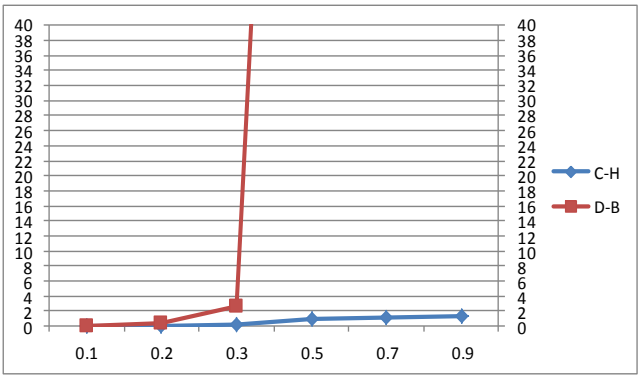


(b)

θ	C-H	D-B	Grupos
0.1	0.01604841	0.00019557	14
0.2	0.0735157	0.0086674	9
0.3	0.14509982	0.08158683	5
0.5	0.53987985	7.68643513	3
0.7	0.67711239	36.3432651	2
0.9	0.76081913	209.517371	2

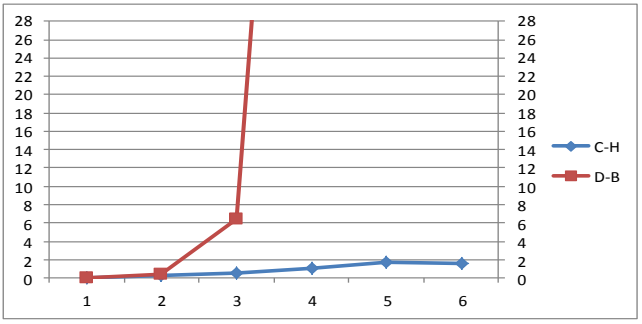
Figura A1.12 MOPI con k = 9. Procesamiento (a) *off-line*, (b) *on-line.*

Índices de validación para la BD Birch1



(a)

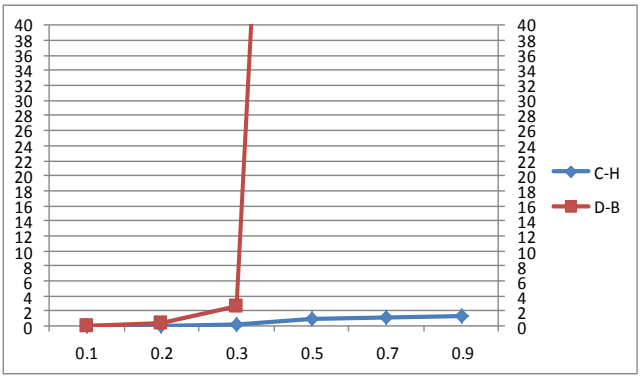
θ	C-H	D-B	Grupos
0.1	0.015319409	0.002294927	272
0.2	0.077705509	0.347674908	53
0.3	0.277297304	5.588374323	19
0.5	0.829685088	315.3727526	6
0.7	1.214569621	3574.397081	3
0.9	1.503804944	8218.841488	2



(b)

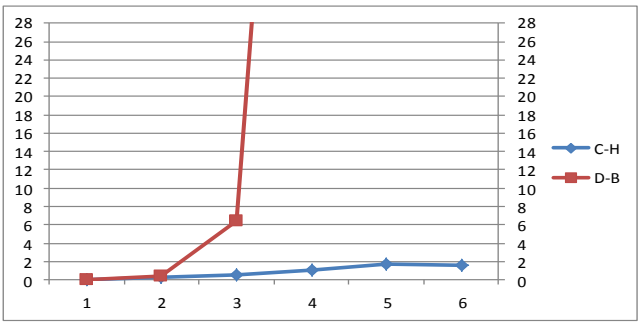
θ	C-H	D-B	Grupos
0.1	0.03746597	0.00493215	140
0.2	0.21875289	0.42898651	45
0.3	0.52537594	6.43149061	19
0.5	1.05781314	113.650933	6
0.7	1.65535052	714.917482	2
0.9	1.60677907	1061.40863	2

Figura A1.13 Birch1 con k = 1. Procesamiento (a) *off-line*, (b) *on-line*.



(a)

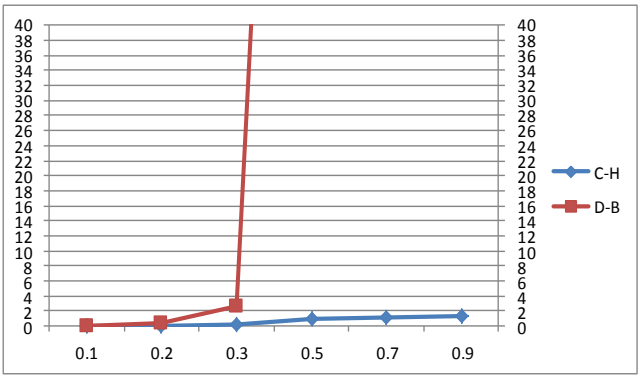
θ	C-H	D-B	Grupos
0.1	0.020485681	0.003649824	224
0.2	0.108757539	0.337874799	48
0.3	0.188559323	2.66294373	25
0.5	0.860862037	196.4323823	6
0.7	1.131577914	2913.816232	3
0.9	1.391615151	26563.51267	2



(b)

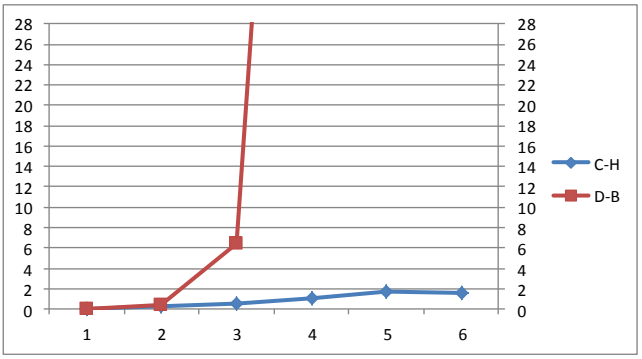
θ	C-H	D-B	Grupos
0.1	0.03746597	0.00493215	140
0.2	0.21875289	0.42898651	45
0.3	0.52537594	6.43149061	19
0.5	1.05781314	113.650933	6
0.7	1.65535052	714.917482	2
0.9	1.60677907	1061.40863	2

Figura A1.14 Birch1 con k = 5. Procesamiento (a) *off-line*, (b) *on-line*.



(a)

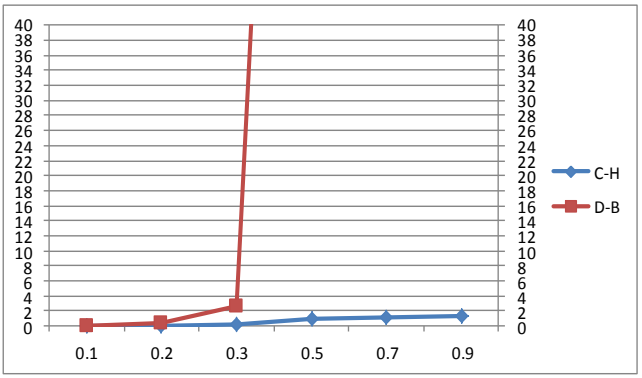
θ	C-H	D-B	Grupos
0.1	0.020485681	0.003649824	224
0.2	0.108757539	0.337874799	48
0.3	0.188559323	2.66294373	25
0.5	0.860862037	196.4323823	6
0.7	1.131577914	2913.816232	3
0.9	1.391615151	26563.51267	2



(b)

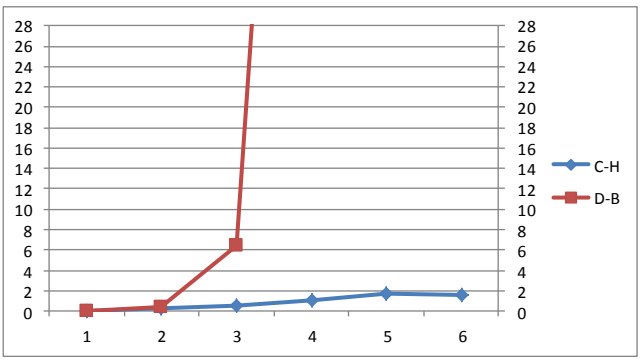
θ	C-H	D-B	Grupos
0.1	0.03746597	0.00493215	140
0.2	0.21875289	0.42898651	45
0.3	0.52537594	6.43149061	19
0.5	1.05781314	113.650933	6
0.7	1.65535052	714.917482	2
0.9	1.60677907	1061.40863	2

Figura A1.15 Birch1 con k = 7. Procesamiento (a) *off-line*, (b) *on-line.*



(a)

θ	C-H	D-B	Grupos
0.1	0.020485681	0.003649824	224
0.2	0.108757539	0.337874799	48
0.3	0.188559323	2.66294373	25
0.5	0.860862037	196.4323823	6
0.7	1.131577914	2913.816232	3
0.9	1.391615151	26563.51267	2

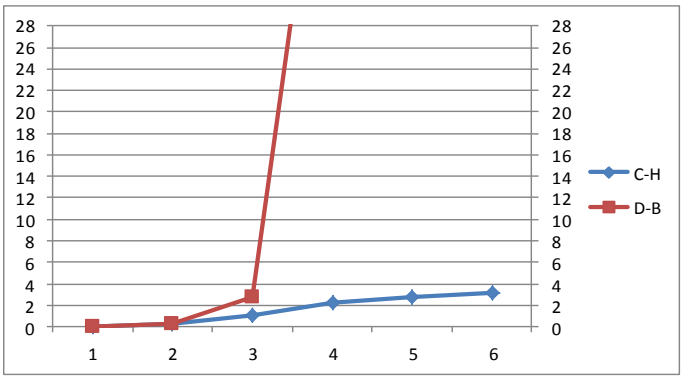


(b)

θ	C-H	D-B	Grupos
0.1	0.03746597	0.00493215	140
0.2	0.21875289	0.42898651	45
0.3	0.52537594	6.43149061	19
0.5	1.05781314	113.650933	6
0.7	1.65535052	714.917482	2
0.9	1.60677907	1061.40863	2

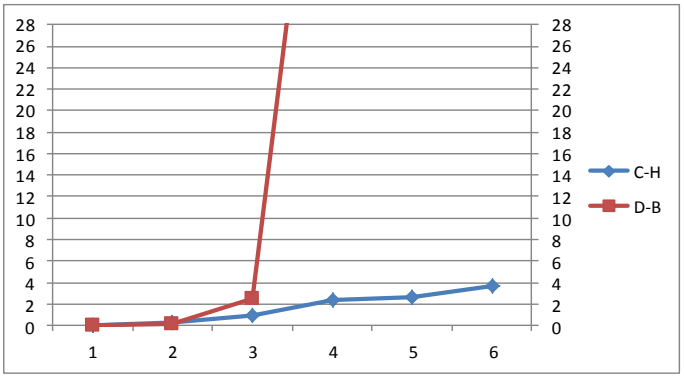
Figura A1.16 Birch1 con k = 9. Procesamiento (a) *off-line*, (b) *on-line.*

Índices de validación para la BD Birch2



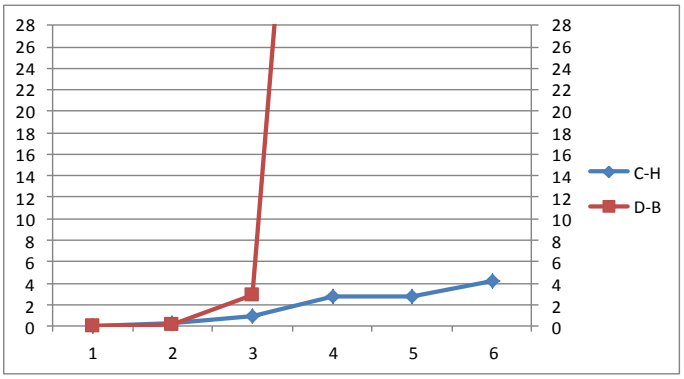
θ	C-H	D-B	Grupos
0.1	0.06332971	0.00351444	113
0.2	0.3379001	0.21743633	30
0.3	1.08776073	2.73636193	12
0.5	2.23982163	56.5801471	6
0.7	2.81489559	221.62204	3
0.9	3.17626918	1899.90329	2

Figura A1.17 Birch2 con k = 1. Procesamiento *on-line.*



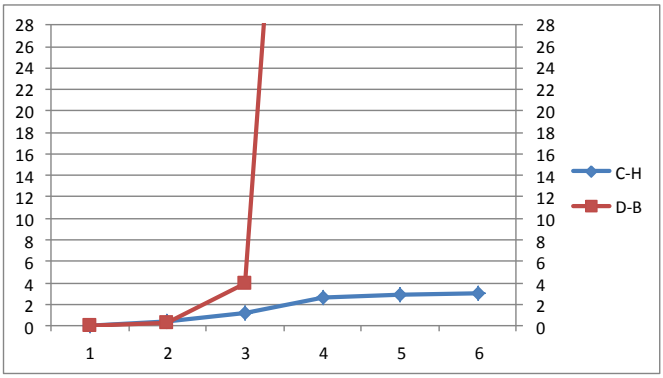
θ	C-H	D-B	Grupos
0.1	0.06426417	0.00324109	124
0.2	0.32261512	0.1566996	28
0.3	0.9491255	2.50177428	11
0.5	2.35611329	62.6393216	6
0.7	2.67583711	193.545548	4
0.9	3.64982659	619.357287	2

Figura A1.18 Birch2 con k = 5. Procesamiento *on-line.*



θ	C-H	D-B	Grupos
0.1	0.05667041	0.00317542	126
0.2	0.33100189	0.17253393	25
0.3	0.99064064	2.9380612	14
0.5	2.70038374	94.9183405	4
0.7	2.77404247	157.381811	4
0.9	4.22627775	2361.1677	2

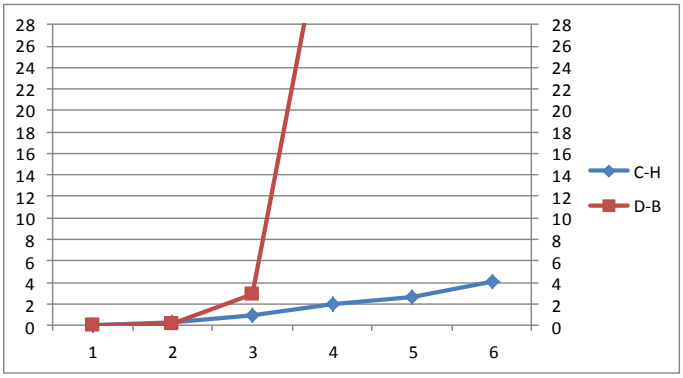
Figura A1.19 Birch2 con k = 7. Procesamiento *on-line.*



θ	C-H	D-B	Grupos
0.1	0.06301899	0.00332181	111
0.2	0.40062785	0.2444924	24
0.3	1.17151338	3.95678002	10
0.5	2.5961381	104.521904	4
0.7	2.86414257	130.496699	4
0.9	3.04604687	1200.17541	3

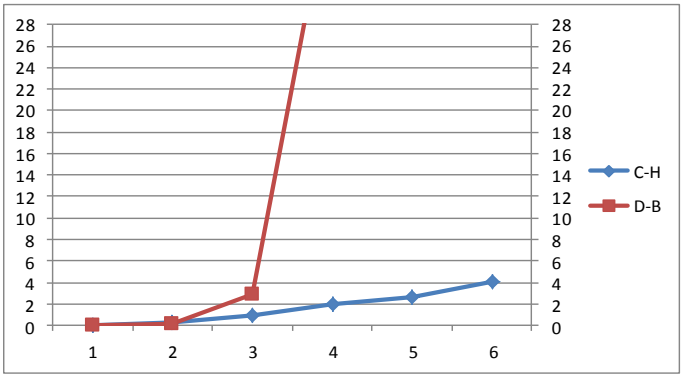
Figura A1.20 Birch2 con k = 9. Procesamiento *on-line.*

Índices de validación para la BD Birch3



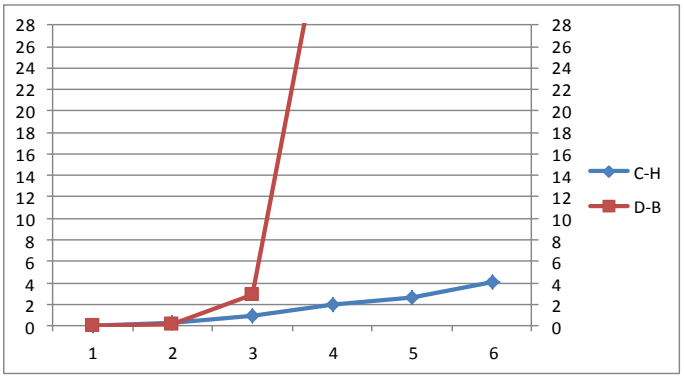
θ	C-H	D-B	Grupos
0.1	0.06332971	0.00351444	113
0.2	0.26569742	0.15415016	34
0.3	0.95635325	2.94755468	12
0.5	2.00847454	42.1127628	5
0.7	2.60657187	104.058687	4
0.9	4.01601407	481.822349	3

Figura A1.21 Birch3 con k = 1. Procesamiento *on-line.*



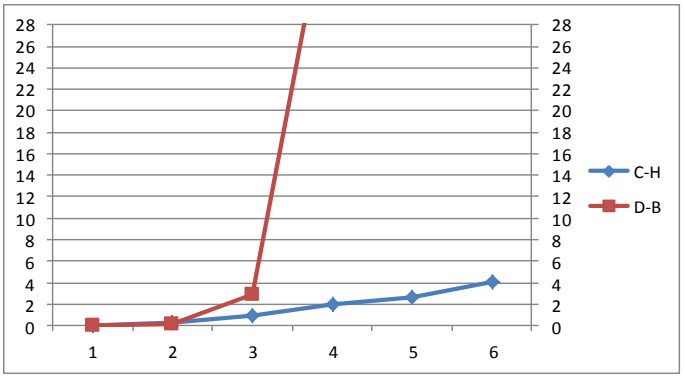
θ	C-H	D-B	Grupos
0.1	0.06332971	0.00351444	113
0.2	0.26569742	0.15415016	34
0.3	0.95635325	2.94755468	12
0.5	2.00847454	42.1127628	5
0.7	2.60657187	104.058687	4
0.9	4.01601407	481.822349	3

Figura A1.22 Birch3 con k = 5. Procesamiento *on-line.*



θ	C-H	D-B	Grupos
0.1	0.06332971	0.00351444	113
0.2	0.26569742	0.15415016	34
0.3	0.95635325	2.94755468	12
0.5	2.00847454	42.1127628	5
0.7	2.60657187	104.058687	4
0.9	4.01601407	481.822349	3

Figura A1.23 Birch3 con k = 7. Procesamiento *on-line.*



θ	C-H	D-B	Grupos
0.1	0.06332971	0.00351444	113
0.2	0.26569742	0.15415016	34
0.3	0.95635325	2.94755468	12
0.5	2.00847454	42.1127628	5
0.7	2.60657187	104.058687	4
0.9	4.01601407	481.822349	3

Figura A1.24 Birch3 con k = 9. Procesamiento *on-line.*

APENDICE II
CLASIFICACIÓN

En este apéndice se muestran la tablas de resultados de la precisión general (PG) obtenida con el clasificador k-NN, utilizando CV para designar la Validación Cruzada, DE para la desviación estándar, *off-line* para el procesamiento del Conjunto de forma *off-line* y *on-line* para las prueba con la metodología.

Precisión BD D31

Tabla A2.1 Precisión general BD D31 k = 1. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.89193548	0.93870968	0.97258065	0.98064516	0.97903226	0.97741935
partición 1	0.86290323	0.96370968	0.97177419	0.98185484	0.97580645	0.97580645
partición 2	0.89420655	0.94710327	0.96473552	0.98236776	0.9697733	0.97229219
partición 3	0.89274448	0.96214511	0.96214511	0.98107256	0.96529968	0.96529968
partición 4	0.83858268	0.94094488	0.96062992	0.96062992	0.98425197	0.98425197
partición 5	0.81773399	0.93103448	0.96059113	0.96059113	0.97044335	0.97536946
Media	0.86583755	0.94719827	0.96539657	0.97447682	0.97408067	0.97505663
DE	0.03247674	0.01317906	0.00546088	0.01079619	0.00692587	0.00622311

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.88571429	0.93061224	0.99591837	0.95102041	0.95918367	0.96326531
partición 1	0.84183673	0.93877551	0.96938776	0.94387755	0.98469388	0.95408163
partición 0	0.81967213	0.96311475	0.95081967	0.96311475	0.97540984	0.97540984
partición 1	0.88205128	0.92820513	0.96410256	0.95384615	0.98974359	0.99487179
partición 0	0.77862595	0.96946565	0.97709924	0.95419847	0.98473282	0.9389313
partición 1	0.86666667	0.99047619	0.98095238	0.94285714	0.95238095	0.98095238
Media	0.8449015	0.95317248	0.97294482	0.95146119	0.97425728	0.96774568
DE	0.01984812	0.01104563	0.00998471	0.01430686	0.01199621	0.02689843

(b)

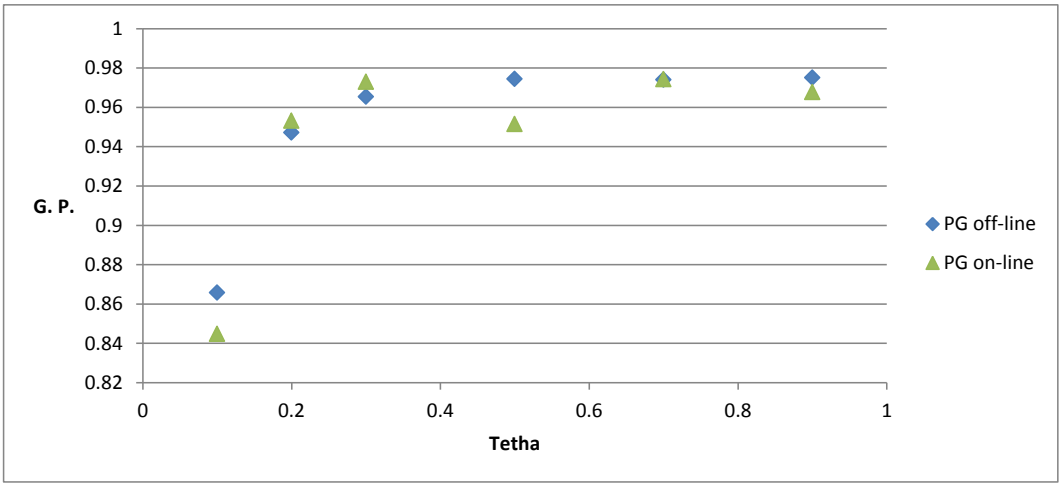


Figura A2.1 Grafico comparativo de la precisión general BD D31 k = 1.

Tabla A2.2 Precisión general BD D31 k = 5. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.877419355	0.95	0.959677419	0.979032258	0.974193548	0.980645161
partición 1	0.856854839	0.955645161	0.963709677	0.975806452	0.971774194	0.973790323
partición 2	0.85138539	0.942065491	0.967254408	0.982367758	0.992443325	0.959697733
partición 3	0.826498423	0.936908517	0.952681388	0.949526814	0.977917981	0.96214511
partición 4	0.724409449	0.937007874	0.948818898	0.948818898	0.976377953	0.976377953
partición 5	0.802955665	0.945812808	0.975369458	0.980295567	0.960591133	0.965517241
Media	0.821661731	0.94454901	0.961210993	0.969200726	0.975504362	0.969665025
DE	0.054858128	0.007427587	0.009714818	0.015742767	0.010310593	0.00843492

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.836734694	0.902040816	0.955102041	0.959183673	0.942857143	0.97959184
partición 1	0.790816327	0.948979592	0.974489796	0.964285714	0.989795918	0.98979592
partición 0	0.848360656	0.93852459	0.942622951	0.954918033	0.987704918	0.96721311
partición 1	0.779487179	0.953846154	0.994871795	0.964102564	0.964102564	0.98974359
partición 0	0.748091603	0.923664122	0.961832061	0.969465649	0.946564885	0.94656489
partición 1	0.704761905	0.942857143	0.952380952	0.980952381	0.961904762	0.96190476
Media	0.783144105	0.934819659	0.963400849	0.965449434	0.965318758	0.97234345
DE	0.073063606	0.005778804	0.00973459	0.002372613	0.008367601	0.01300815

(b)

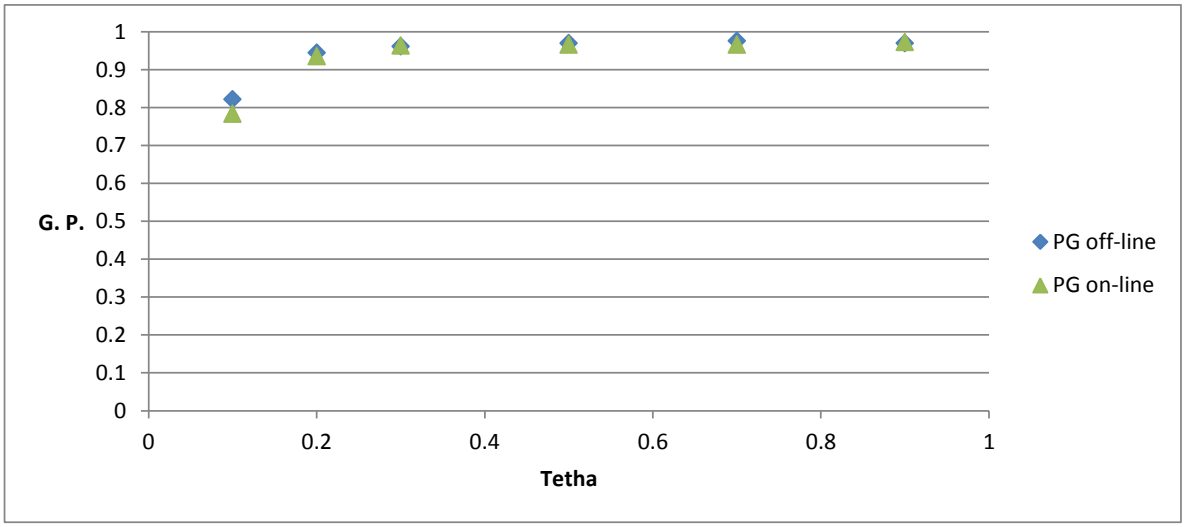


Figura A2.2 Grafico comparativo de la precisión general BD D31 k = 5.

Tabla A2.3 Precisión general BD Birch1 k = 7. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.874193548	0.970967742	0.974193548	0.977419355	0.964516129	0.979032258
partición 1	0.868951613	0.973790323	0.967741935	0.981854839	0.967741935	0.975806452
partición 2	0.828715365	0.947103275	0.987405542	0.972292191	0.972292191	0.974811083
partición 3	0.820189274	0.965299685	0.977917981	0.981072555	0.974763407	0.981072555
partición 4	0.767716535	0.929133858	0.964566929	0.972440945	0.980314961	0.976377953
partición 5	0.748768473	0.950738916	0.960591133	0.975369458	0.985221675	0.980295567
Media	0.81673002	0.956044746	0.972028407	0.976734262	0.974116224	0.977896471
DE	0.051370764	0.01706139	0.009804894	0.004136118	0.007723869	0.002581737

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.787755102	0.930612245	0.959183673	0.955102041	0.955102041	0.99591837
partición 1	0.734693878	0.892857143	0.964285714	0.959183673	0.959183673	0.96428571
partición 0	0.819672131	0.954918033	0.975409836	0.967213115	0.979508197	0.99180328
partición 1	0.764102564	0.933333333	0.979487179	0.984615385	0.979487179	0.98461538
partición 0	0.65648855	0.961832061	0.961832061	0.961832061	0.984732824	0.93129771
partición 1	0.714285714	0.952380952	0.933333333	0.933333333	0.933333333	0.98095238
Media	0.744271573	0.937369015	0.962139927	0.960091729	0.965055438	0.97456247
DE	0.028654397	0.011179758	0.013855294	0.011936848	0.00532639	0.01731905

(b)

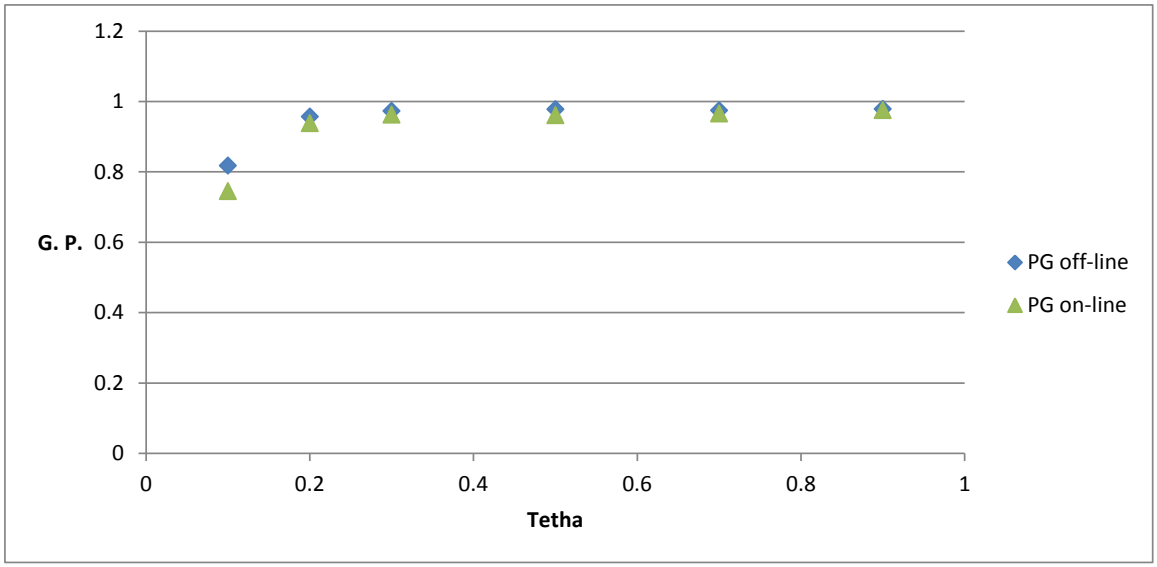


Figura A2.3 Grafico comparativo de la precisión general BD D31 k = 7.

Tabla A2.4 Precisión general BD Birch1 k = 9. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.879032258	0.946774194	0.975806452	0.962903226	0.988709677	0.972580645
partición 1	0.858870968	0.943548387	0.973790323	0.951612903	0.977822581	0.95766129
partición 2	0.841309824	0.944584383	0.98488665	0.974811083	0.964735516	0.977329975
partición 3	0.788643533	0.930599369	0.968454259	0.977917981	0.965299685	0.990536278
partición 4	0.763779528	0.929133858	0.968503937	0.964566929	0.972440945	0.968503937
partición 5	0.694581281	0.935960591	0.960591133	0.985221675	0.965517241	0.965517241
Media	0.80181976	0.938407908	0.971976454	0.969442444	0.972382404	0.971967546
DE	0.069076523	0.007583335	0.008230636	0.012116449	0.009507262	0.01124441

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.804081633	0.955102041	0.971428571	0.963265306	0.987755102	0.95510204
partición 1	0.75	0.954081633	0.974489796	0.933673469	0.964285714	0.93877551
partición 0	0.819672131	0.983606557	0.93442623	0.991803279	0.979508197	0.99180328
partición 1	0.676923077	0.953846154	0.964102564	0.969230769	0.984615385	0.97435897
partición 0	0.595419847	0.938931298	0.961832061	0.938931298	0.961832061	0.96183206
partición 1	0.514285714	0.942857143	0.904761905	0.933333333	0.942857143	0.98095238
Media	0.684060915	0.954631575	0.951513594	0.954796262	0.970016545	0.96697936
DE	0.059857587	0.006334732	0.007120348	0.016258752	0.014263076	0.00858648

(b)

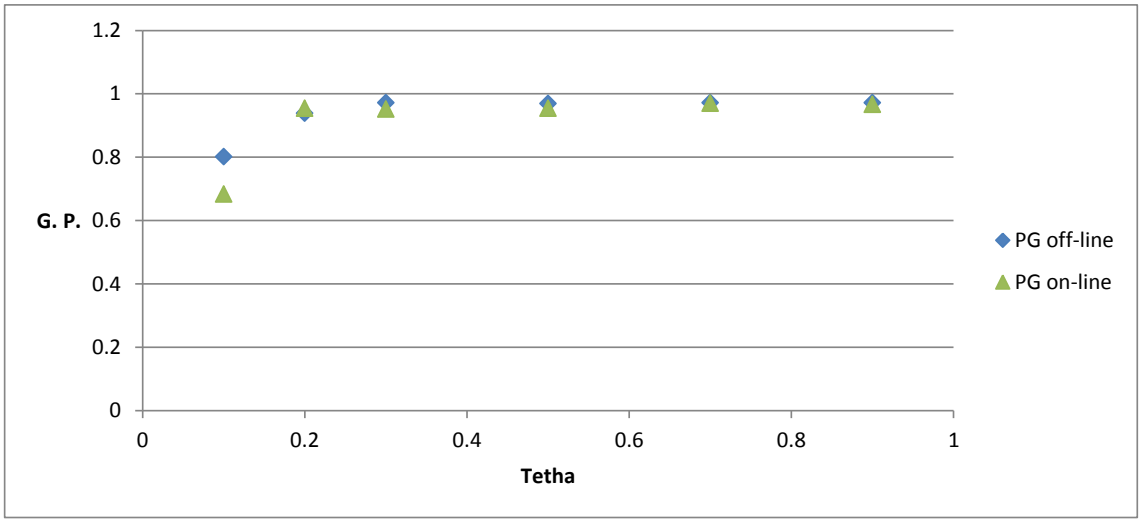


Figura A2.4 Grafico comparativo de la precisión general BD D31 k = 9.

Precisión BD JOENSUU

Tabla A2.5 Precisión general BD Joensuu k = 1. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.9925187	0.99916874	0.99501247	0.99750623	1	1
partición 1	0.98856549	0.99168399	0.99168399	0.997921	1	1
partición 2	0.98311688	0.98701299	0.99090909	0.9961039	0.9987013	1
partición 3	0.98214286	0.99837662	0.98863636	1	0.99675325	1
partición 4	0.96754564	0.98174442	0.99188641	0.9979716	0.9979716	1
partición 5	0.98477157	0.99238579	0.97969543	1	1	0.99746193
Media	0.98307915	0.99171009	0.98962549	0.9982495	0.9989036	0.99957654
DE	0.00853501	0.00666193	0.00528394	0.00151458	0.00135202	0.00103616

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98676749	0.98298677	0.99621928	0.98865784	0.99810964	1
partición 1	0.99054374	0.9787234	0.99763593	0.9929078	1	1
partición 0	0.96204934	0.99240987	0.99240987	0.9943074	0.99810247	0.9943074
partición 1	0.95971564	0.99763033	1	0.99763033	0.98815166	0.99763033
partición 0	0.97959184	1	1	0.97959184	1	1
partición 1	1	1	1	1	1	1
Media	0.97966801	0.99192323	0.99770699	0.99216005	0.997385	0.99865402
DE	0.01025906	0.00546667	0.00525102	0.00428943	0.00273484	0.00656806

(b)

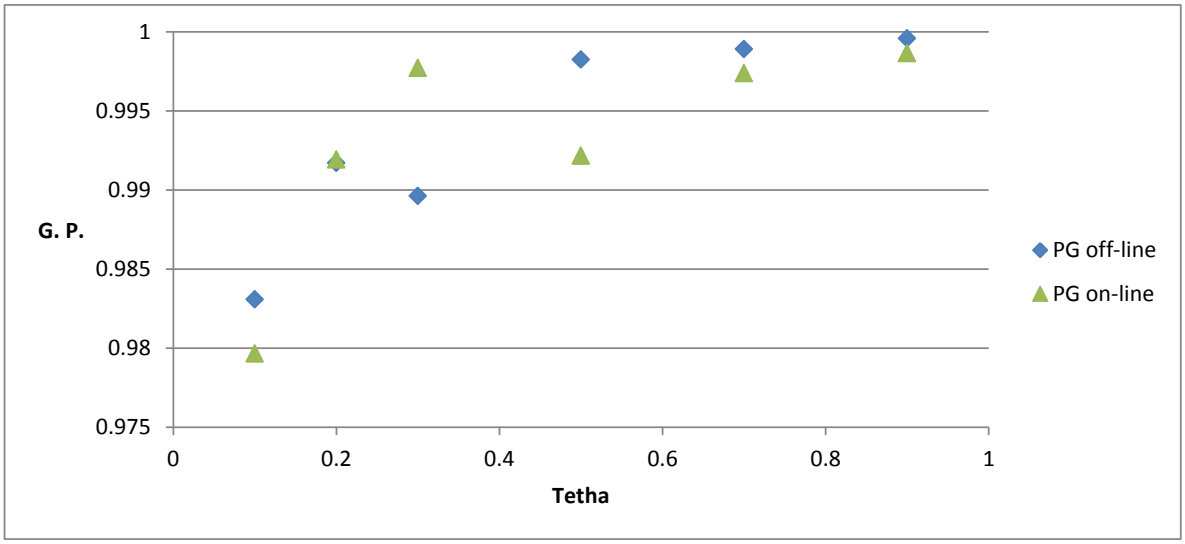


Figura A2.5 Grafico comparativo de la precisión general BD Joensuu k = 1.

Tabla A2.6 Precisión general BD Joensuu k = 5. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.97755611	0.98586866	0.99334996	0.99916874	1	1
partición 1	0.97817048	0.98544699	0.99168399	0.997921	0.997921	1
partición 2	0.97142857	0.98701299	0.9974026	1	0.9987013	1
partición 3	0.96103896	0.98051948	0.98214286	0.99837662	0.99837662	1
partición 4	0.95740365	0.98174442	0.99188641	0.9959432	0.9959432	0.98985801
partición 5	0.95685279	0.98477157	0.99746193	0.99492386	0.99746193	1
Media	0.96703321	0.98422463	0.99230802	0.99772067	0.99806658	0.99830248
DE	0.00986285	0.00253577	0.00561248	0.00193619	0.00135149	0.00414045

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.97353497	0.97731569	0.99621928	0.99621928	0.99432892	0.99810964
partición 1	0.98108747	0.98108747	0.99763593	1	0.9929078	0.99763593
partición 0	0.92979127	0.9772296	0.98671727	0.99240987	0.99240987	0.9943074
partición 1	0.9478673	0.97867299	0.99526066	0.99763033	0.99526066	0.99052133
partición 0	0.97278912	0.97959184	0.99319728	0.99319728	1	1
partición 1	0.98305085	0.99152542	1	0.99152542	0.99152542	1
Media	0.96449022	0.9808915	0.99482957	0.99515907	0.99440155	0.99675664
DE	0.02121675	0.00540231	0.00458693	0.00332493	0.00305143	0.0037019

(b)

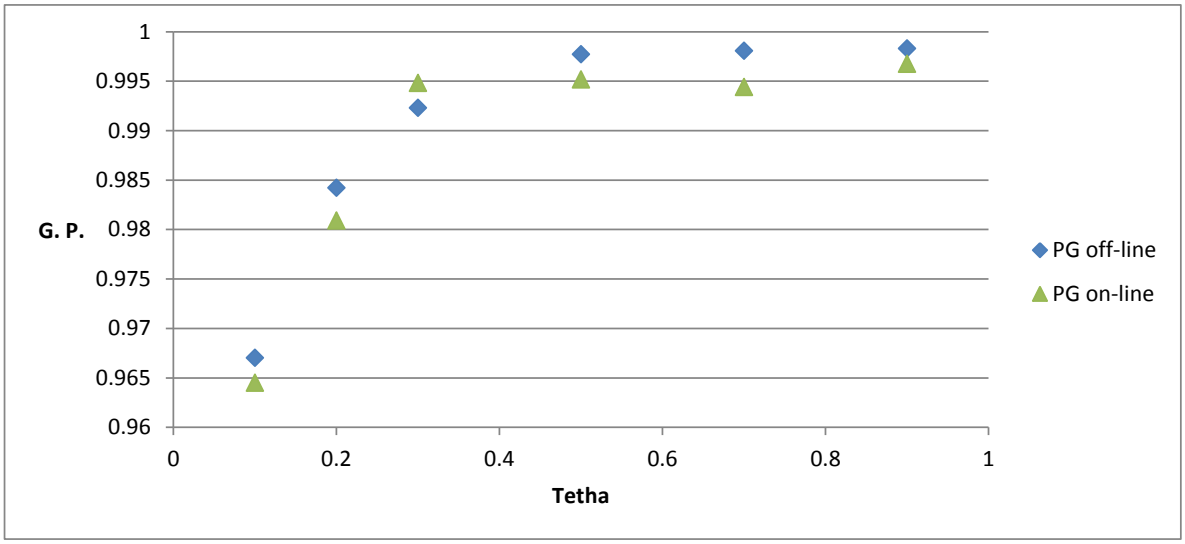


Figura A2.6 Grafico comparativo de la precisión general BD Joensuu k = 5.

Tabla A2.7 Precisión general BD Joensuu k = 7. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.97423109	0.98836243	0.99501247	0.99833749	0.99916874	1
partición 1	0.96777547	0.98960499	0.99376299	0.997921	1	1
partición 2	0.95194805	0.98441558	0.99480519	0.9974026	1	0.9987013
partición 3	0.94967532	0.98538961	0.98376623	0.99837662	1	1
partición 4	0.94726166	0.97160243	0.98580122	0.99188641	1	0.99391481
partición 5	0.91878173	0.99238579	0.98477157	1	0.99492386	0.99746193
Media	0.95144689	0.98527081	0.98964103	0.9973174	0.99901371	0.99834394
DE	0.01935715	0.00730114	0.00539406	0.00280093	0.00203185	0.00239793

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98109641	0.98676749	0.96786389	0.99621928	0.99432892	0.99432892
partición 1	0.9787234	0.98345154	0.9787234	0.99527187	0.9929078	0.99763593
partición 0	0.93358634	0.9829222	0.99051233	0.99240987	0.99240987	0.99810247
partición 1	0.92890995	0.97393365	0.98815166	0.98341232	0.99763033	0.99052133
partición 0	0.96598639	0.93877551	0.98639456	0.99319728	0.99319728	1
partición 1	0.96610169	0.98305085	0.99152542	0.97457627	1	1
Media	0.95884617	0.97467316	0.98382694	0.98915069	0.99507513	0.99675905
DE	0.01471973	0.01372081	0.00893687	0.00495578	0.00691026	0.00275629

(b)

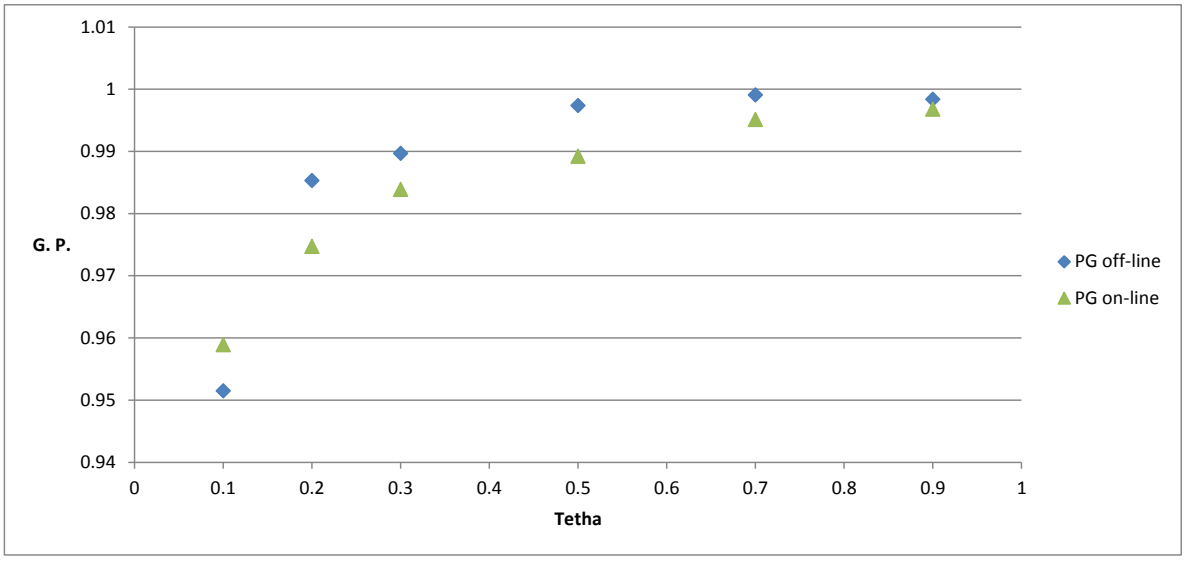


Figura A2.7 Grafico comparativo de la precisión general BD Joensuu k = 7.

Tabla A2.8 Precisión general BD Joensuu k = 9. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96342477	0.99002494	0.98836243	0.99667498	0.99667498	0.99916874
partición 1	0.95530146	0.995842	0.99064449	0.9989605	0.9989605	1
partición 2	0.94415584	0.98181818	0.98441558	0.9974026	0.9974026	0.9974026
partición 3	0.92532468	0.98214286	0.98701299	0.99675325	0.99512987	0.99512987
partición 4	0.92697769	0.96754564	0.96551724	0.9979716	0.99391481	0.9959432
partición 5	0.9213198	0.96192893	0.97969543	0.98477157	0.99746193	0.99746193
Media	0.93928191	0.97981209	0.98257208	0.99541066	0.99658941	0.9975163
DE	0.0175042	0.01296888	0.00917548	0.00528617	0.00180752	0.00184851

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96975425	0.94706994	0.98109641	0.99432892	0.99810964	1
partición 1	0.95508274	0.9787234	0.98817967	0.99054374	0.99054374	0.99763593
partición 0	0.95256167	0.97912713	0.99051233	0.99051233	0.9943074	0.99051233
partición 1	0.91706161	0.98578199	0.98578199	0.98341232	0.99052133	0.99763033
partición 0	0.96598639	0.97278912	0.99319728	0.98639456	1	1
partición 1	0.98305085	0.96610169	0.99152542	0.99152542	0.99152542	0.96610169
Media	0.95702559	0.97151742	0.98837395	0.98944646	0.99416097	0.99190652
DE	0.00885798	0.00659098	0.01303245	0.00373801	0.00534436	0.01017605

(b)

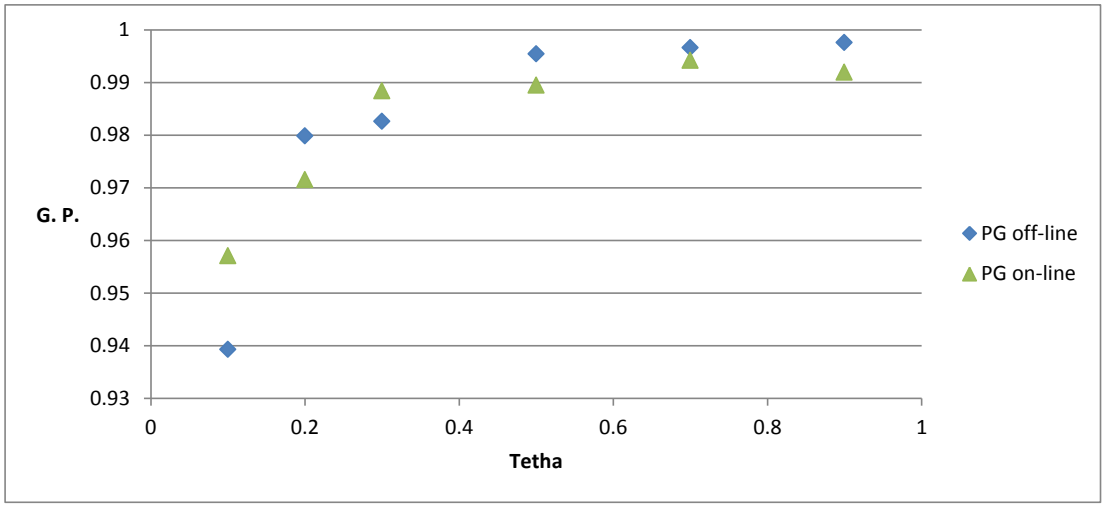


Figura A2.8 Grafico comparativo de la precisión general BD Joensuu k = 9.

Precisión BD MOPI

Tabla A2.9 Precisión general BD MOPI k = 1. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	1	0.99883586	1	1	1	1
partición 1	0.9992722	1	1	1	1	1
partición 2	0.99636033	0.99818016	1	1	1	1
partición 3	0.99659091	1	0.99772727	0.99886364	1	1
partición 4	0.99715909	1	1	1	1	1
partición 5	0.9928952	1	1	1	1	1
Media	0.99704365	0.9995024	0.99962085	0.99981052	1	1
DE	0.00251401	0.00079787	0.00092784	0.00046392	0	0

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	1	1	1	1	1	1
partición 1	1	1	1	1	1	1
partición 0	0.99658703	1	1	1	1	1
partición 1	0.9978678	0.99573561	1	1	1	1
partición 0	1	1	1	1	1	1
partición 1	1	0.99777283	1	1	1	1
Media	0.99907488	0.99891673	1	1	1	1
DE	0.00107083	0.00089154	0	0	0	0

(b)

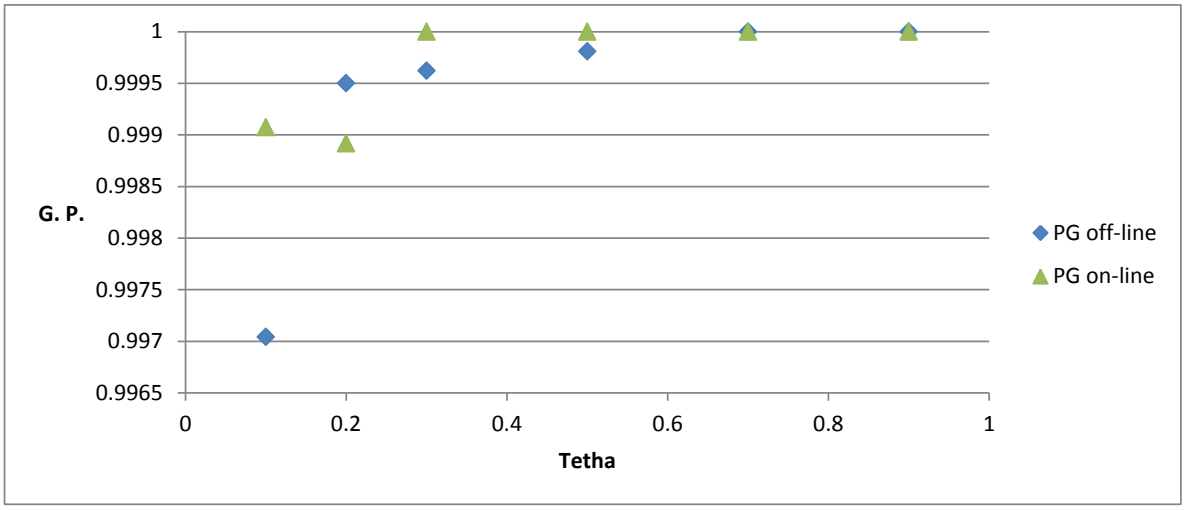


Figura A2.9 Grafico comparativo de la precisión general BD MOPI k = 1.

Tabla A2.10 Precisión general BD MOPI k = 5. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.99883586	1	0.9895227	1	1	1
partición 1	1	1	0.98544396	0.99781659	1	1
partición 2	0.99636033	0.99636033	0.98089172	1	1	1
partición 3	0.99772727	0.99886364	0.97840909	1	1	1
partición 4	1	0.99857955	0.97443182	1	1	1
partición 5	0.9946714	0.98934281	0.98756661	1	1	1
Media	0.9979306	0.99718411	0.98269674	0.99963577	1	1
DE	0.00212197	0.00406917	0.00579321	0.00089137	0	0

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.99299475	1	1	1	1	1
partición 1	0.99781182	1	0.99781182	1	1	1
partición 0	0.99488055	0.99829352	1	1	1	1
partición 1	0.99573561	0.9978678	1	1	1	1
partición 0	0.99821747	0.99465241	0.99465241	1	1	1
partición 1	0.99777283	0.99777283	0.99554566	1	1	1
Media	0.99623371	0.99809615	0.9979992	1	1	1
DE	0.00238886	0.00244336	0.00167103	0	0	0

(b)

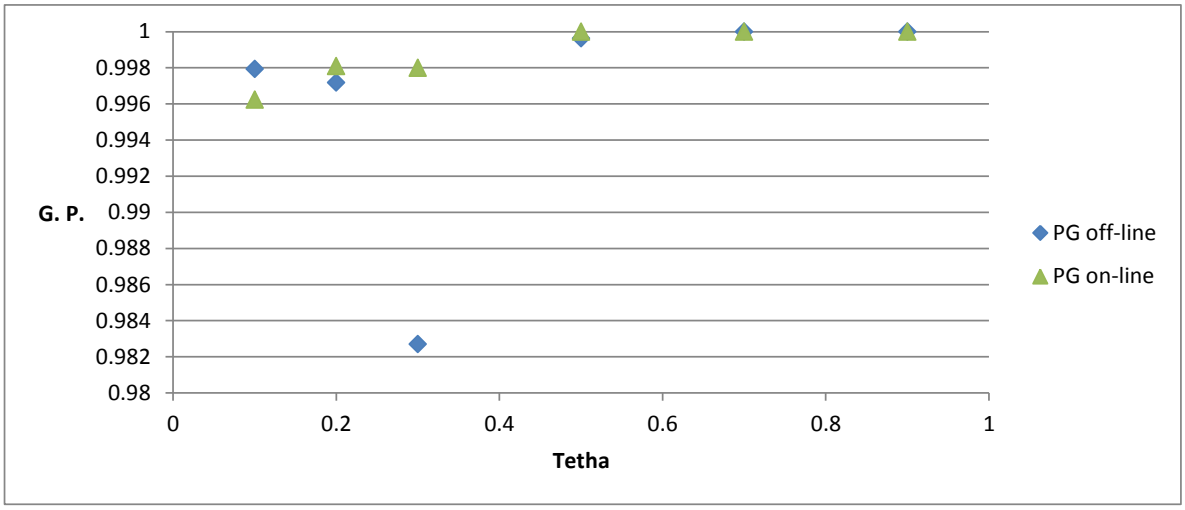


Figura A2.10 Grafico comparativo de la precisión general BD MOPI k = 5.

Tabla A2.11 Precisión general BD MOPI k = 7. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.99767171	0.99767171	0.9854482	0.99941793	1	1
partición 1	0.99563319	0.9992722	0.98544396	0.9992722	1	1
partición 2	0.99545041	1	0.98544131	1	1	1
partición 3	0.99886364	0.99545455	0.9875	0.99545455	1	1
partición 4	0.99005682	0.99715909	0.97443182	1	1	1
partición 5	0.9946714	0.99111901	0.98046181	1	1	1
Media	0.99538731	0.99677513	0.98311115	0.99902279	1	1
DE	0.00304284	0.00320381	0.00485684	0.00177851	0	0

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.99474606	1	1	1	1	1
partición 1	0.99781182	1	1	1	1	1
partición 0	0.99658703	1	1	1	1	1
partición 1	0.98933902	1	1	1	1	1
partición 0	0.99465241	0.99643494	0.99465241	1	1	1
partición 1	0.99331849	0.99554566	1	1	1	1
Media	0.9944055	0.99866161	0.99910674	1	1	1
DE	0.00273467	0.00258636	0.00232126	0	0	0

(b)

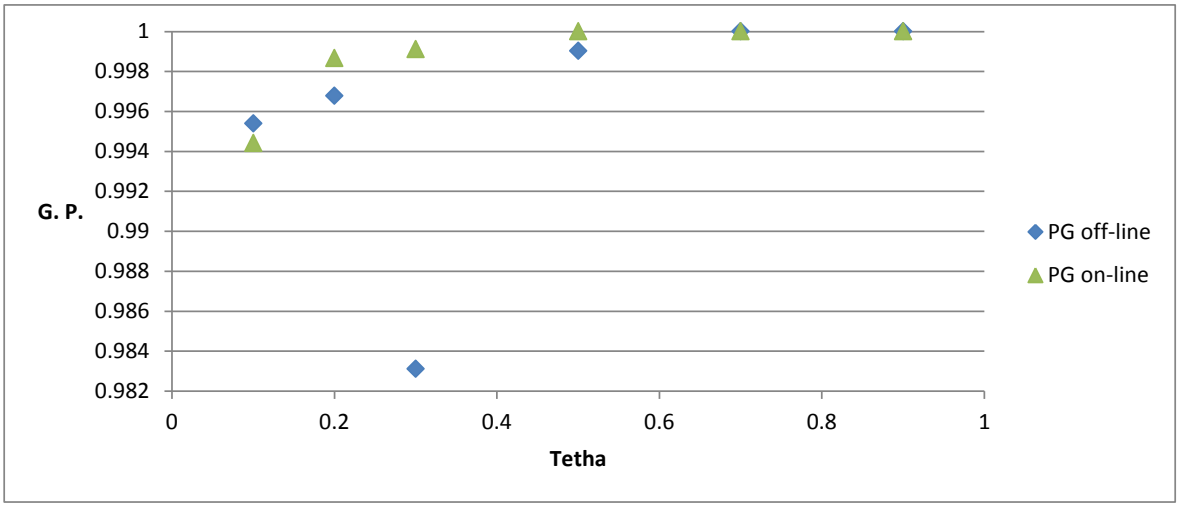


Figura A2.11 Grafico comparativo de la precisión general BD MOPI k = 7.

Tabla A2.12 Precisión general BD MOPI k = 9. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.99592549	0.99883586	0.98603027	0.99941793	1	1
partición 1	0.99126638	0.9985444	0.98617176	1	1	1
partición 2	0.99454049	0.99636033	0.98726115	0.99818016	1	1
partición 3	0.99659091	1	0.97954545	0.99886364	1	1
partición 4	0.99857955	0.99715909	0.98011364	0.99715909	1	1
partición 5	0.99111901	0.9964476	0.97158082	1	1	1
Media	0.99466655	0.99789032	0.98176859	0.99893628	1	1
DE	0.00299205	0.0014649	0.00598646	0.00111628	0	0

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98774081	0.99824869	1	1	1	1
partición 1	0.99562363	0.99781182	1	1	1	1
partición 0	0.99488055	0.99829352	1	1	1	1
partición 1	0.99360341	0.9978678	1	1	1	1
partición 0	0.99643494	0.99821747	1	1	1	1
partición 1	0.99554566	0.99331849	0.98886414	1	1	1
Media	0.99396721	0.99729136	0.99813535	1	1	1
DE	0.00499914	0.00074555	0.00098636	0	0.00315837	0

(b)

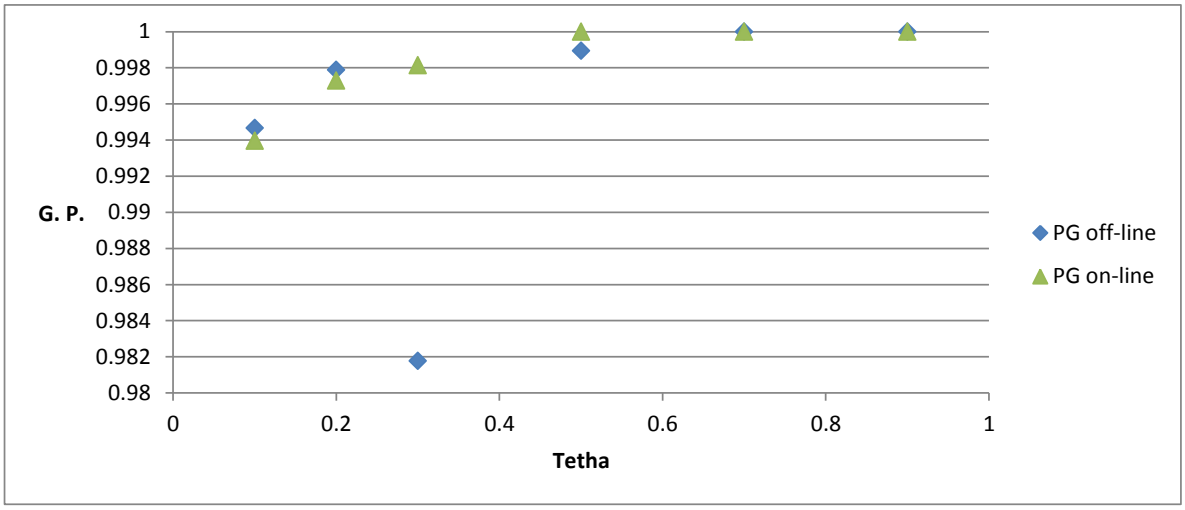


Figura A2.12 Grafico comparativo de la precisión general BD MOPI k = 9.

Precisión BD BIRCH1

Tabla A2.13 Precisión general BD Birch1 k = 1. (a) Conjunto *off-line*, (b) *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.9677	0.9848	0.99105	0.99645	0.9982	0.99915
partición 1	0.9623125	0.9845625	0.9898125	0.995625	0.9975625	0.99925
partición 2	0.95695313	0.98164063	0.98867188	0.99546875	0.99828125	0.99945313
partición 3	0.95322266	0.98037109	0.98847656	0.99609375	0.99833984	0.99941406
partición 4	0.94799805	0.97961426	0.98449707	0.99536133	0.99694824	0.99902344
partición 5	0.93469637	0.97345133	0.98565761	0.99435459	0.99710101	0.99908453
Media	0.95375461	0.98073261	0.98802499	0.99555869	0.99773865	0.99922918
DE	0.01161515	0.00415775	0.00249074	0.00071861	0.00062146	0.00017566

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96324808	0.98385895	0.99180531	0.9957785	0.99913087	0.99950335
partición 1	0.96135341	0.98339283	0.99037715	0.99456775	0.99891355	0.99968959
partición 0	0.9626227	0.9880443	0.99433677	0.99848981	0.99962245	0.99861566
partición 1	0.96303288	0.98599969	0.99229196	0.99842693	0.99795501	0.99905616
partición 0	0.9745	0.99275	0.99775	0.99825	0.99975	0.9995
partición 1	0.971875	0.9934375	0.9971875	0.9990625	0.9990625	0.999375
Media	0.96609192	0.98790592	0.99395433	0.99742788	0.99907223	0.99928989
DE	0.005587	0.00434654	0.00300615	0.00180967	0.00063859	0.00039139

(b)

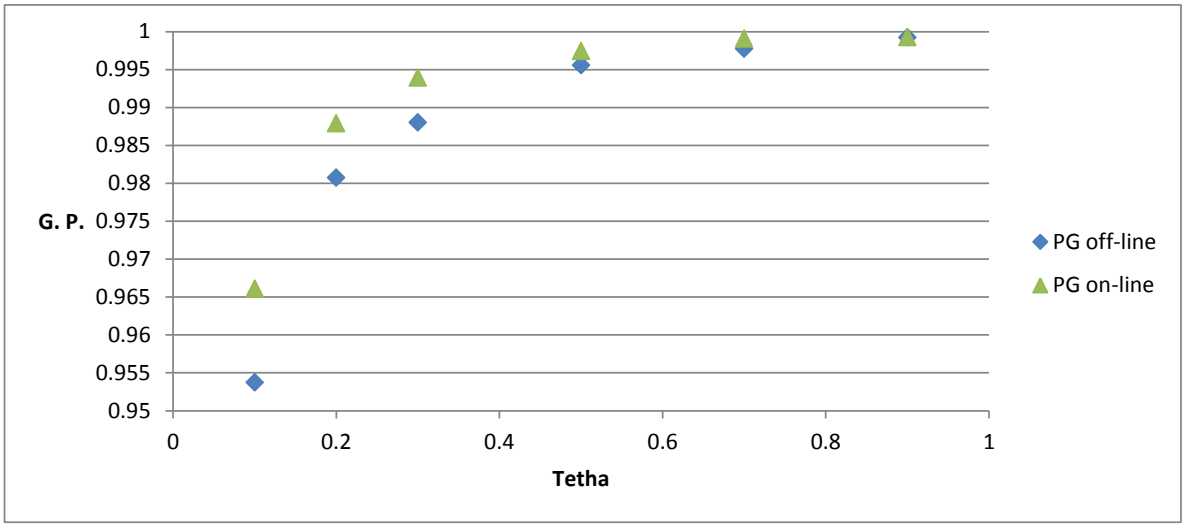


Figura A2.13 Grafico comparativo de la precisión general BD Birch1 k = 1.

Tabla A2.14 Precisión general BD Birch1 k = 5. (a) Conjunto *off-line*, (b) *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96655	0.9878	0.99205	0.99605	0.99825	0.99885
partición 1	0.9623125	0.9836875	0.9881875	0.9961875	0.9973125	0.9979375
partición 2	0.95804688	0.98296875	0.98710938	0.9959375	0.99789063	0.99773438
partición 3	0.95625	0.98076172	0.98886719	0.99550781	0.99775391	0.99785156
partición 4	0.95080566	0.9744873	0.98608398	0.99462891	0.99768066	0.99780273
partición 5	0.94354593	0.97619774	0.98352151	0.99252365	0.99679585	0.99847421
Media	0.95622239	0.9809734	0.98763314	0.99513841	0.99761382	0.99810831
DE	0.00821105	0.00495723	0.00286072	0.00140023	0.00050305	0.00044995

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96262727	0.98187236	0.99143283	0.99627514	0.99975168	0.99888254
partición 1	0.95871488	0.98385845	0.99006674	0.99518858	0.99968959	0.99968959
partición 0	0.96488799	0.98577901	0.99509187	0.99823811	0.99886735	0.9992449
partición 1	0.95784175	0.984112	0.99197735	0.99716848	0.99905616	0.99952808
partición 0	0.97025	0.99175	0.997	0.99975	0.999	1
partición 1	0.96875	0.9946875	0.996875	0.9990625	0.9996875	1
Media	0.96383408	0.98699913	0.99373692	0.99761255	0.99934198	0.99955744
DE	0.00509923	0.00505278	0.00297342	0.00172765	0.00040792	0.00043891

(b)

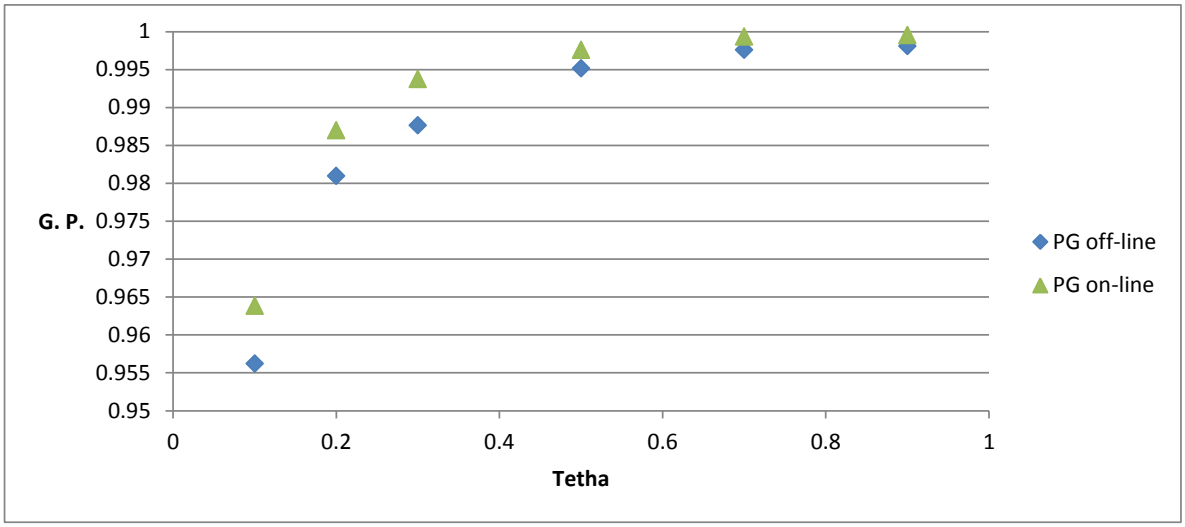


Figura A2.14 Grafico comparativo de la precisión general BD Birch1 k = 5.

Tabla A2.15 Precisión general BD Birch1 k = 7. (a) Conjunto *off-line*, (b) *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.9617	0.9837	0.9909	0.9974	0.9984	0.99695
partición 1	0.9573125	0.9829375	0.9888125	0.997	0.9976875	0.9975
partición 2	0.95515625	0.97929688	0.9884375	0.99679688	0.99679688	0.99710938
partición 3	0.95253906	0.97548828	0.98789063	0.996875	0.99667969	0.99726563
partición 4	0.9387207	0.97558594	0.98706055	0.99523926	0.99719238	0.9954834
partición 5	0.93317058	0.97253586	0.98474214	0.99710101	0.99633811	0.9957278
Media	0.9497107	0.97824888	0.98797213	0.99673511	0.99718219	0.9966724
DE	0.0112563	0.00447541	0.00203792	0.00076251	0.00075524	0.0008498

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96411721	0.98187236	0.99242612	0.99490936	0.99962751	0.99950335
partición 1	0.95995654	0.98184076	0.98851467	0.99441254	0.99906876	0.99906876
partición 0	0.96249685	0.98389127	0.99433677	0.99748301	0.99874151	0.99874151
partición 1	0.95296524	0.98253893	0.99182004	0.9976404	0.99874154	0.99937077
partición 0	0.97025	0.99275	0.996	0.999	0.9995	0.99975
partición 1	0.9696875	0.9896875	0.9978125	0.9990625	0.9990625	0.999375
Media	0.9632275	0.98542103	0.99348045	0.99708297	0.99912358	0.99930151
DE	0.00645575	0.00464702	0.00329774	0.001996	0.00037264	0.0003521

(b)

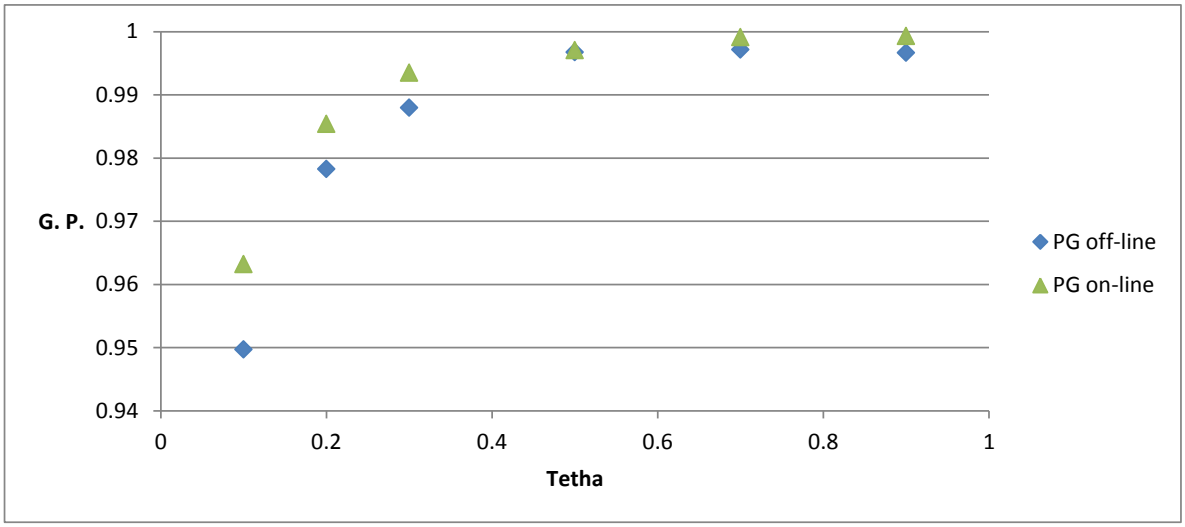


Figura A2.15 Grafico comparativo de la precisión general BD Birch1 k = 7.

Tabla A2.16 Precisión general BD Birch1 k = 9. (a) Conjunto *off-line*, (b) *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.96565	0.98415	0.9924	0.99665	0.99845	0.9981
partición 1	0.9561875	0.982125	0.990375	0.9965625	0.9975625	0.99775
partición 2	0.95648438	0.98125	0.98804688	0.99585938	0.9978125	0.99773438
partición 3	0.94765625	0.98291016	0.98779297	0.99443359	0.99746094	0.99697266
partición 4	0.94274902	0.97583008	0.98901367	0.99438477	0.99853516	0.99658203
partición 5	0.9351541	0.97085749	0.98367409	0.99252365	0.99694843	0.99679585
Media	0.95059414	0.97950925	0.98854681	0.99506792	0.99779477	0.99732233
DE	0.01096553	0.00512569	0.00293151	0.00159408	0.00060984	0.00061724

(a)

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.95927489	0.98087907	0.99031537	0.9957785	0.9990067	0.9990067
partición 1	0.9568524	0.97796058	0.9892907	0.99487816	0.99906876	0.99891355
partición 0	0.95784042	0.98514976	0.99395922	0.99735716	0.99937075	0.9997483
partición 1	0.95296524	0.98379739	0.99386503	0.99811232	0.99858424	0.99889885
partición 0	0.972	0.9875	0.997	0.9985	0.9995	0.99925
partición 1	0.9703125	0.9921875	0.9975	0.999375	0.999375	0.9996875
Media	0.96151491	0.98456849	0.99365033	0.9973323	0.99915086	0.99925076
DE	0.00775482	0.00499702	0.00335589	0.00170776	0.00033736	0.00038351

(b)

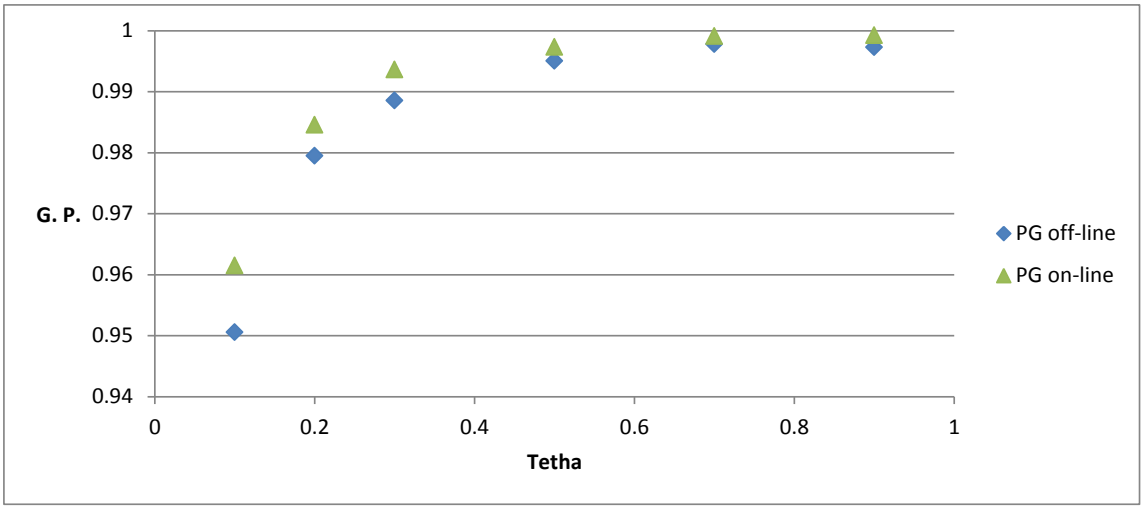


Figura A2.16 Grafico comparativo de la precisión general BD Birch1 k = 9.

Precisión BD BIRCH2

Tabla A2.17 Precisión general BD Birch2 k = 1, *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98252671	0.9947203	0.99459459	0.9982401	0.99924576	1
partición 1	0.97768699	0.99261471	0.99355751	0.9990572	0.9985858	1
partición 0	0.97421708	0.98918375	0.99471765	0.99761036	0.99937115	0.99974846
partición 1	0.97075931	0.99103914	0.99339726	0.99717026	0.99937117	0.99905675
partición 0	0.97484123	0.9890083	0.99706888	0.99926722	0.99975574	1
partición 1	0.97832061	0.99083969	0.99755725	0.99969466	0.99908397	1
Media	0.976385	0.99123235	0.99514755	0.99850622	0.99923553	0.99980081
DE	0.00404706	0.00216385	0.00176525	0.00099531	0.00038798	0.00037817

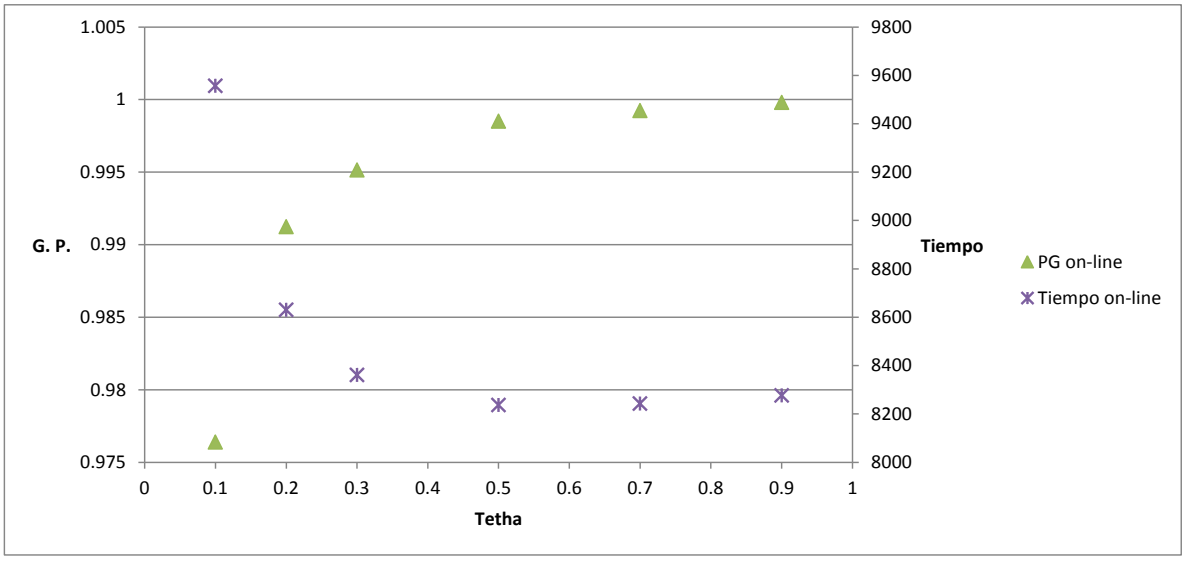


Figura A2.17 Gráfico de la precisión general y tiempo BD Birch2 k = 1.

Tabla A2.18 Precisión general BD Birch2 k = 5, *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.9769956	0.98830924	0.99358894	0.99836581	0.99849151	1
partición 1	0.97548712	0.98711502	0.99607165	0.99874293	0.99874293	1
partición 0	0.97383977	0.9885549	0.99459188	0.99886807	0.99962269	0.99937115
partición 1	0.97406068	0.98899544	0.99308285	0.99827071	0.99952838	0.99874234
partición 0	0.97044455	0.99193942	0.99706888	0.99926722	0.99780166	0.99853444
partición 1	0.97312977	0.98961832	0.99603053	0.99938931	0.99816794	0.99877863
Media	0.97399081	0.98908761	0.99507143	0.99881725	0.99872563	0.99923758
DE	0.00221868	0.00162504	0.00156795	0.00045614	0.00073061	0.0006528

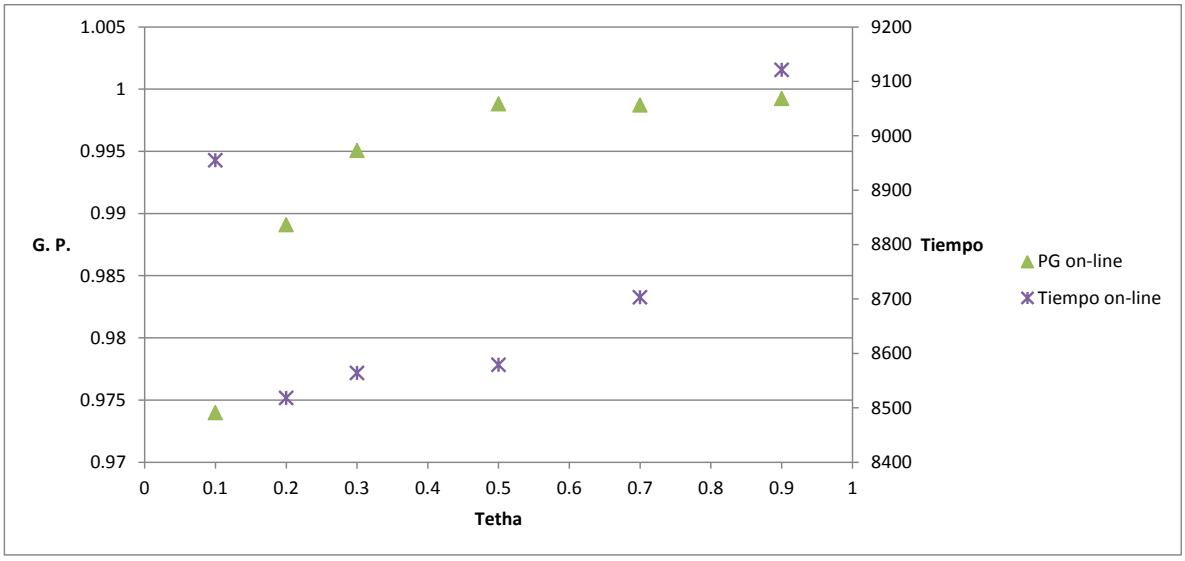


Figura A2.18 Gráfico de la precisión general y tiempo BD Birch2 k = 5.

Tabla A2.19 Precisión general BD Birch2 k = 7, *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.97661848	0.99258328	0.99245757	0.99899434	0.99836581	1
partición 1	0.97297297	0.99261471	0.99481458	0.9985858	0.99842866	1
partición 0	0.97572632	0.98767451	0.99371148	0.99635266	0.99849076	0.99899384
partición 1	0.97217419	0.98490803	0.99229681	0.99732746	0.99921396	0.99858513
partición 0	0.97117733	0.9902296	0.99951148	0.99804592	0.99926722	1
partición 1	0.96641221	0.98717557	0.99908397	0.9978626	0.99877863	1
Media	0.97250788	0.98919349	0.99530831	0.9978611	0.99875744	0.99959632
DE	0.0036472	0.00313092	0.00322216	0.00093848	0.00040033	0.00063833

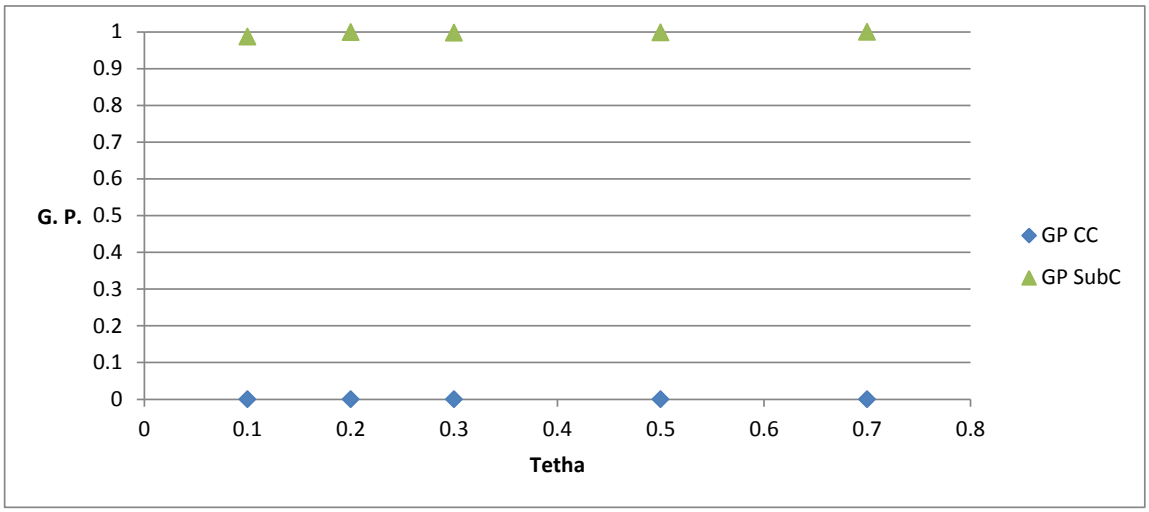


Figura A2.19 Gráfico de la precisión general y tiempo BD Birch2 k = 7.

Tabla A2.20 Precisión general BD Birch2 k = 9, *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.97862979	0.99245757	0.99258328	0.99861722	0.99861722	0.99861722
partición 1	0.97642992	0.99104337	0.99434318	0.99795726	0.99968573	0.99968573
partición 0	0.97610363	0.98993837	0.99459188	0.99886807	0.99911961	0.99911961
partición 1	0.96792957	0.99103914	0.9940261	0.99827071	0.99874234	0.99874234
partición 0	0.96849047	0.99242794	0.9987787	0.99902296	0.99829018	0.99829018
partición 1	0.96458015	0.99175573	0.99847328	1	0.99908397	0.99908397
Media	0.97201316	0.99144329	0.99546336	0.99878916	0.99892308	0.99892308
DE	0.00573329	0.00096788	0.0025466	0.00070957	0.00048434	0.00048434

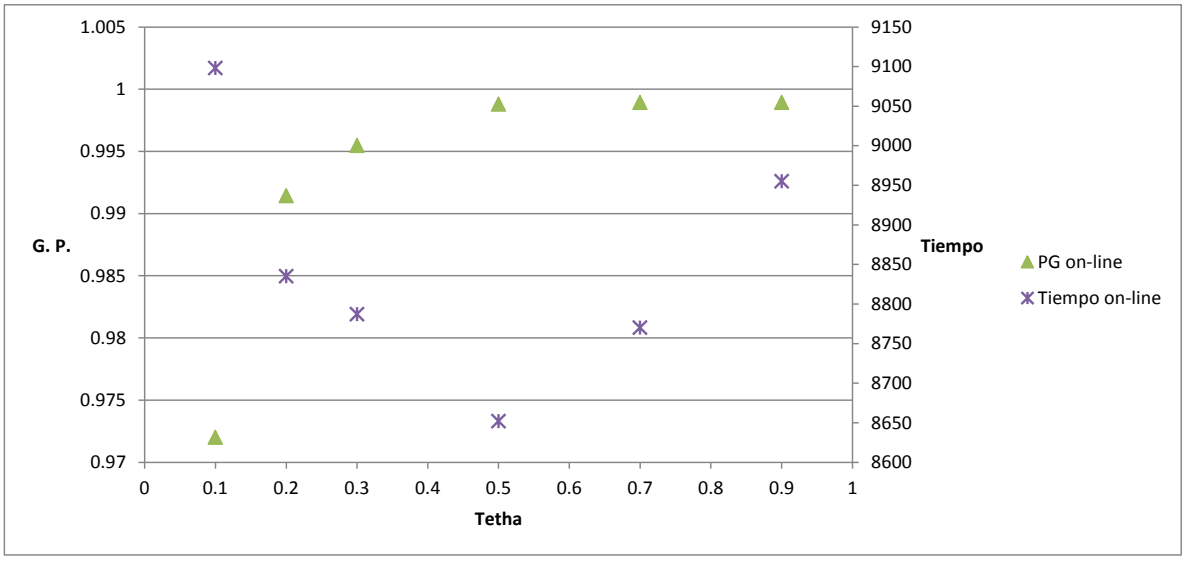


Figura A2.20 Gráfico de la precisión general y tiempo BD Birch2 k = 9.

Precisión BD BIRCH3

Tabla A2.21 Precisión general BD Birch3 k = 1, *on-line*.

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98139535	0.99132621	0.99635449	0.99924576	0.99899434	1
partición 1	0.97941546	0.9907291	0.99575739	0.99764299	0.99921433	1
partición 0	0.97560055	0.98930952	0.9944661	0.99723305	0.99924538	0.99949692
partición 1	0.97406068	0.9882094	0.99685584	0.9979563	0.99842792	0.99889954
partición 0	0.97703957	0.9926722	0.99804592	0.99975574	0.99902296	0.99951148
partición 1	0.97007634	0.99145038	0.9978626	0.99908397	0.99969466	0.99908397
Media	0.97625779	0.99061505	0.9965563	0.99848587	0.99909986	0.99949857
DE	0.00401113	0.00160872	0.00134633	0.00101076	0.00041391	0.0004547

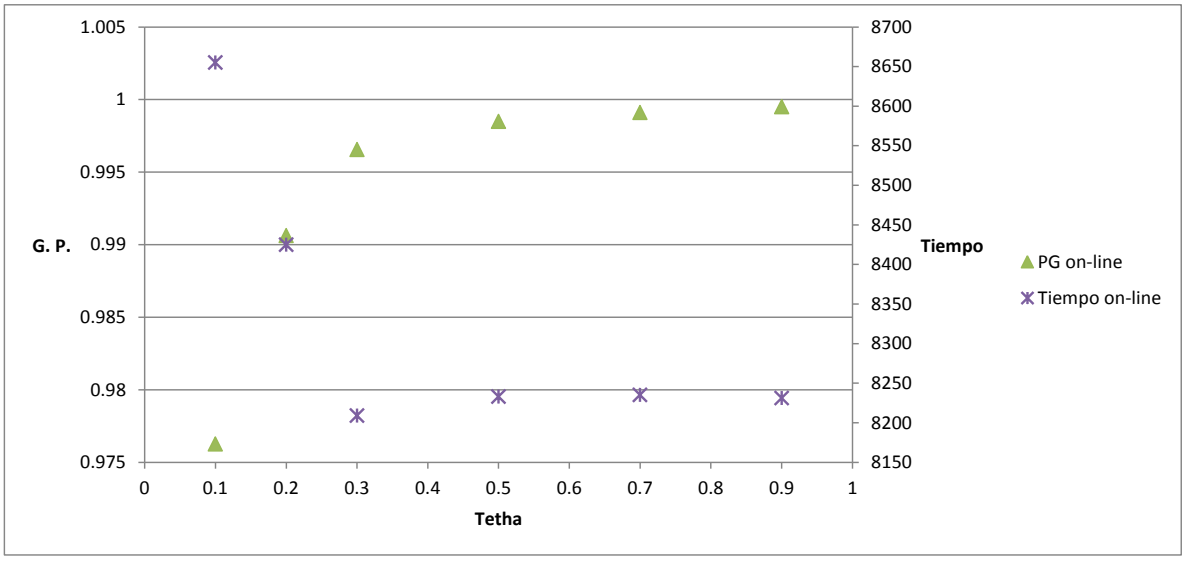


Figura A2.21 Gráfico de la precisión general y tiempo BD Birch3 k = 1.

Tabla A2.22 Precisión general BD Birch3 k = 5, *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98164676	0.99032055	0.99509742	0.99874293	0.99874293	1
partición 1	0.97548712	0.98805783	0.99544312	0.99842866	0.99921433	1
partición 0	0.97622941	0.9885549	0.99459188	0.99811344	0.99911961	0.99811344
partición 1	0.96965886	0.98632291	0.99386889	0.9981135	0.99842792	0.99905675
partición 0	0.96971177	0.9902296	0.99829018	0.99926722	0.9987787	0.99902296
partición 1	0.9719084	0.98748092	0.99694656	0.99908397	0.99877863	0.99969466
Media	0.9740979	0.98849341	0.99570523	0.99862485	0.99884365	0.99931441
DE	0.00463023	0.00156733	0.00162928	0.00048967	0.0002846	0.00073189

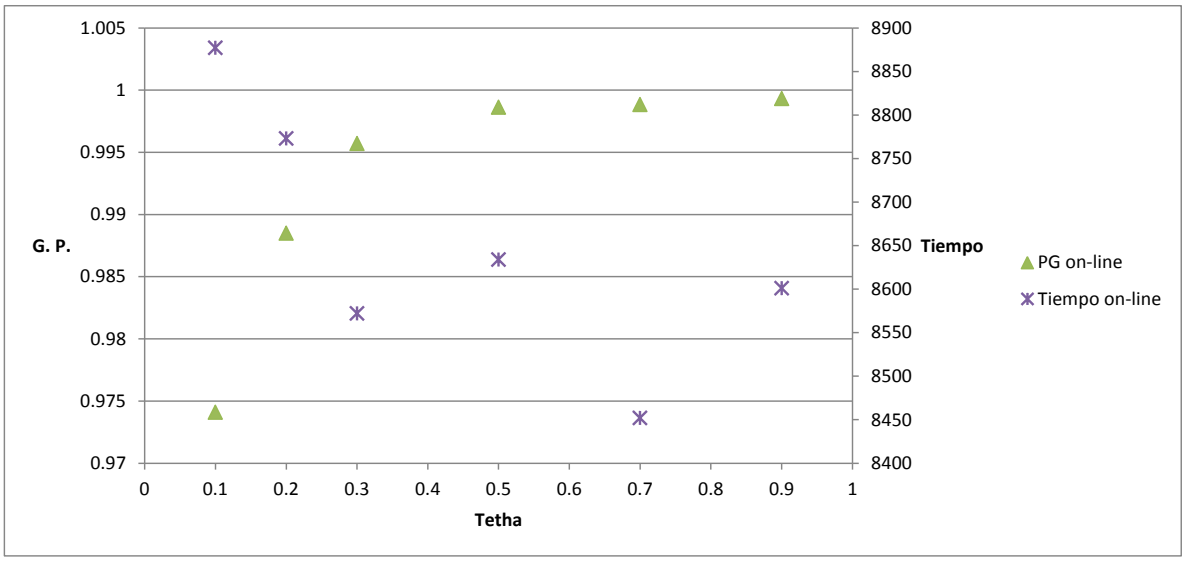


Figura A2.22 Gráfico de la precisión general y tiempo BD Birch3 k = 5.

Tabla A2.23 Precisión general BD Birch3 k = 7, *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98114393	0.9894406	0.9947203	0.99836581	0.99874293	1
partición 1	0.97737272	0.99041483	0.99512885	0.9985858	0.99827153	1
partición 0	0.97245629	0.98805182	0.99421456	0.9978619	0.99823922	0.99886807
partición 1	0.97028769	0.98758057	0.99292564	0.99764188	0.99827071	0.99842792
partición 0	0.97557401	0.98949682	0.99731314	0.99951148	0.99975574	0.99951148
partición 1	0.96824427	0.98778626	0.9951145	1	0.99908397	0.99969466
Media	0.97417008	0.9887946	0.99490197	0.99866079	0.9987272	0.99941685
DE	0.00477467	0.00114699	0.00143679	0.00092587	0.00060656	0.00063959

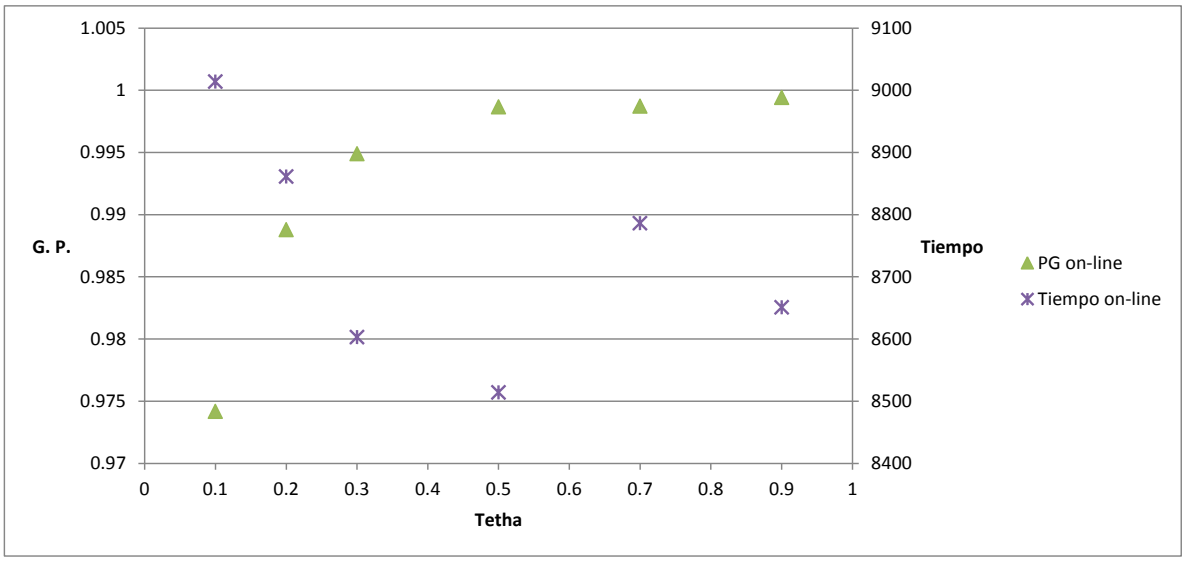


Figura A2.23 Gráfico de la precisión general y tiempo BD Birch3 k = 7.

Tabla A2.24 Precisión general BD Birch3 k = 9, *on-line.*

CV	0.1	0.2	0.3	0.5	0.7	0.9
partición 0	0.98001257	0.9912005	0.99635449	0.99861722	0.99836581	1
partición 1	0.97815839	0.98711502	0.99607165	0.99874293	0.99921433	1
partición 0	0.973714	0.98679411	0.99333417	0.99798767	0.99823922	0.99949692
partición 1	0.96777236	0.98773778	0.99135356	0.99764188	0.9979563	0.99937117
partición 0	0.97459697	0.99169516	0.99462628	0.9987787	0.99902296	0.99951148
partición 1	0.97129771	0.98992366	0.99480916	0.99908397	0.99938931	0.99908397
Media	0.97425015	0.98907578	0.99442344	0.99847527	0.99869784	0.9995772
DE	0.00446242	0.00214174	0.00185721	0.00054528	0.0005867	0.00036167

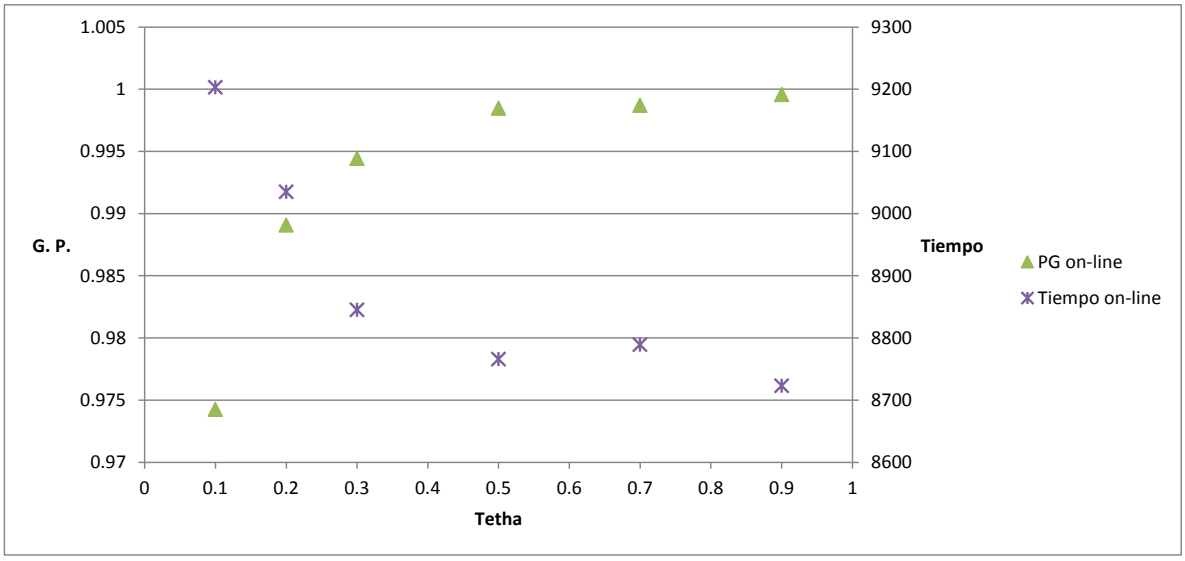


Figura A2.24 Gráfico de la precisión general y tiempo BD Birch3 k = 9.

APENDICE III
RESULTADOS DE TIEMPO

Tiempo de la BD D31

Tabla A3.1 Tiempo de procesamiento de la BD D31.

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	37	39	48	50	55
0.2	23	26	30	32	36
0.3	21	25	30	32	35
0.5	21	24	29	32	34
0.7	21	26	30	32	36
0.9	22	25	29	32	36

(a)

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	20	19	23	26	29
0.2	8	10	11	15	16
0.3	7	9	11	14	18
0.5	8	9	11	14	16
0.7	7	10	12	14	17
0.9	8	9	12	14	16

(b)

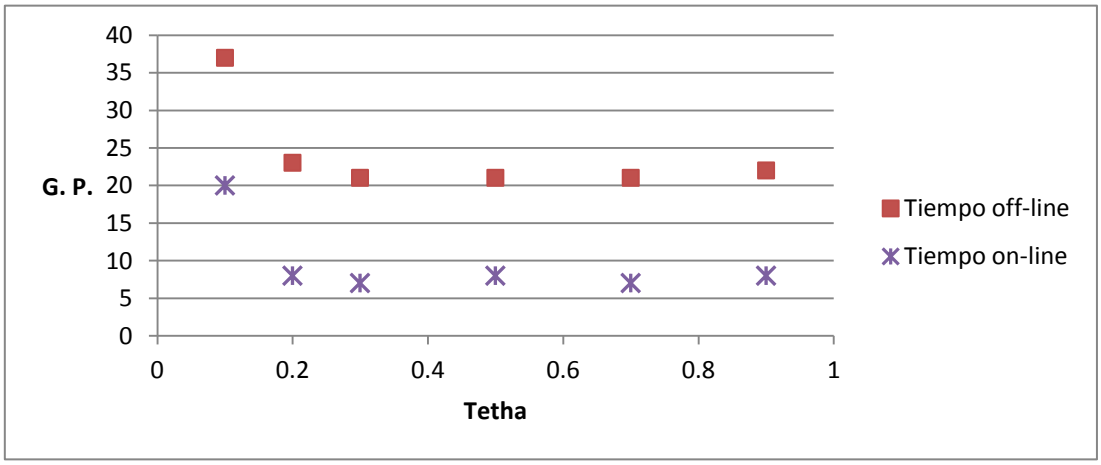


Figura A3.1 Tiempo de procesamiento para D31 k = 1.

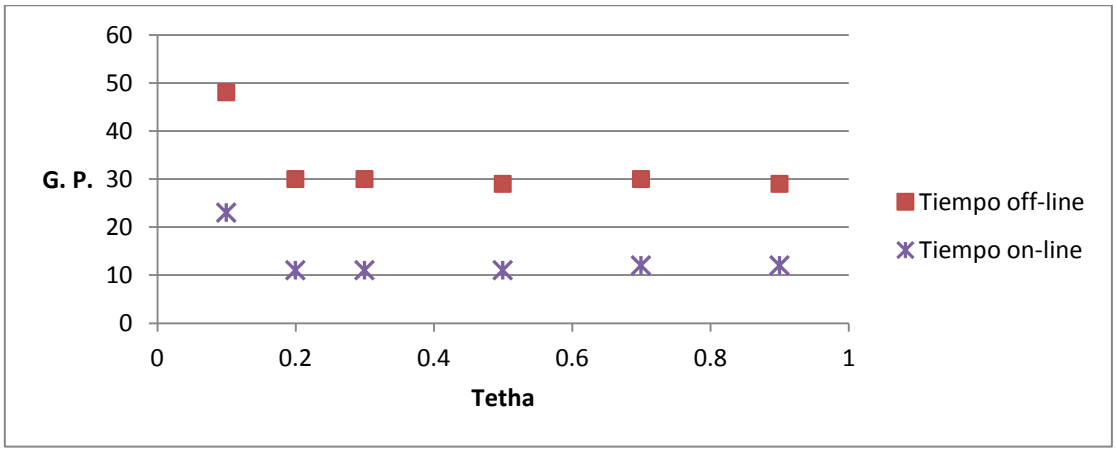


Figura A3.2 Tiempo de procesamiento para D31 k = 5.

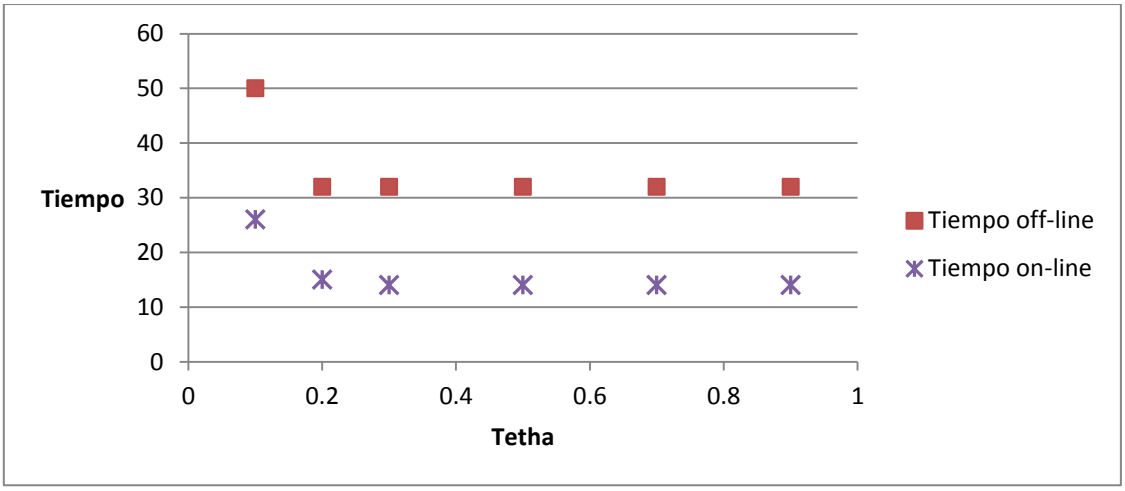


Figura A3.3 Tiempo de procesamiento para D31 k = 7.

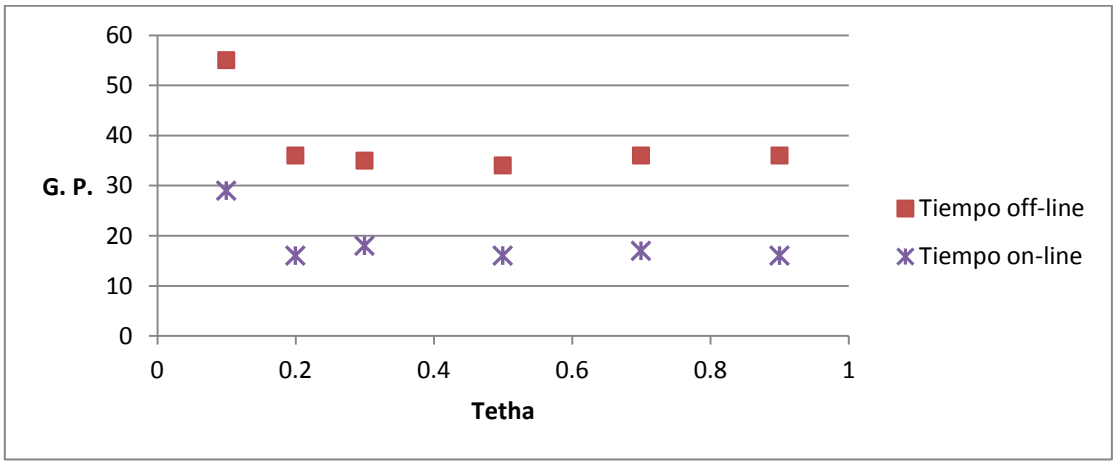


Figura A3.4 Tiempo de procesamiento para D31 k = 9.

Tiempo de la BD JOENSUU

Tabla A3.2 Tiempo de procesamiento de la BD Joensuu.

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	86	95	96	109	114
0.2	81	88	97	105	111
0.3	83	89	95	104	108
0.5	84	87	94	105	110
0.7	82	90	100	105	112
0.9	86	88	100	101	108

(a)

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	30	35	38	40	46
0.2	28	33	37	41	44
0.3	28	33	38	38	45
0.5	28	33	39	38	45
0.7	29	34	37	40	45
0.9	29	34	36	40	44

(b)

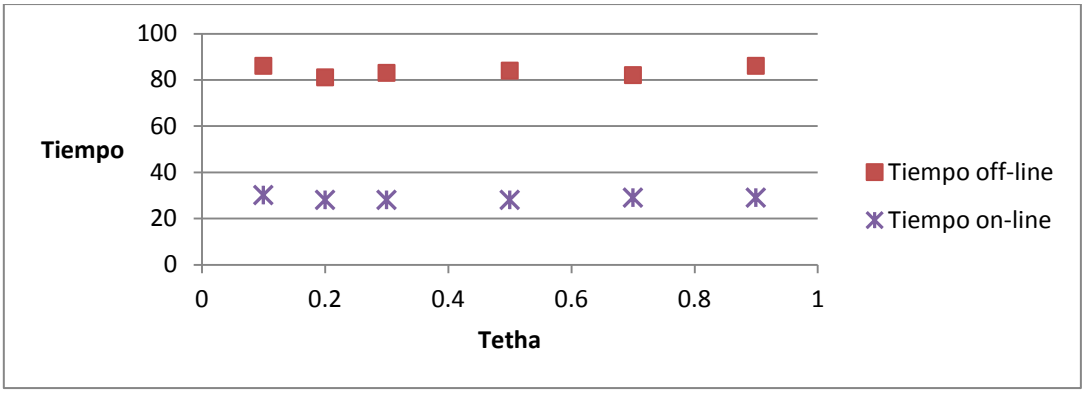


Figura A3.5 Tiempo de procesamiento para Joensuu k = 1.

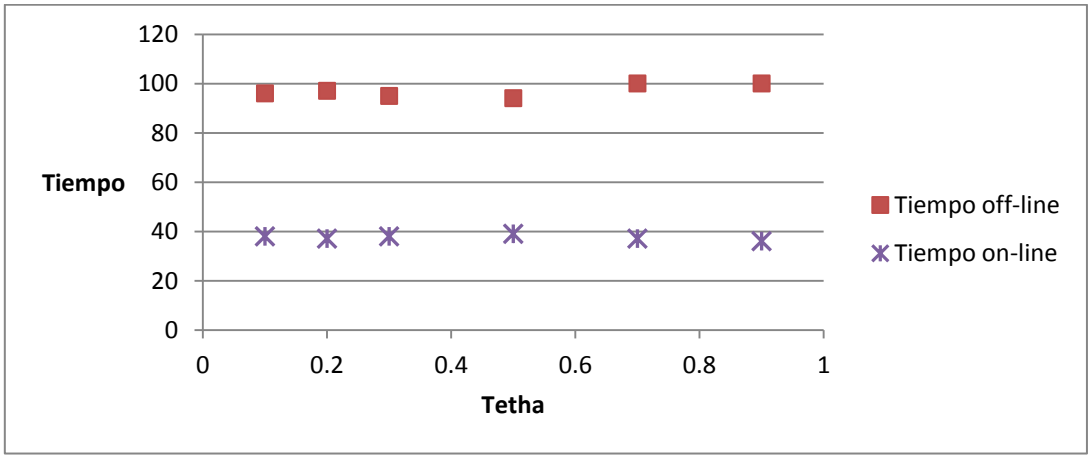


Figura A3.6 Tiempo de procesamiento para Joensuu k = 5.

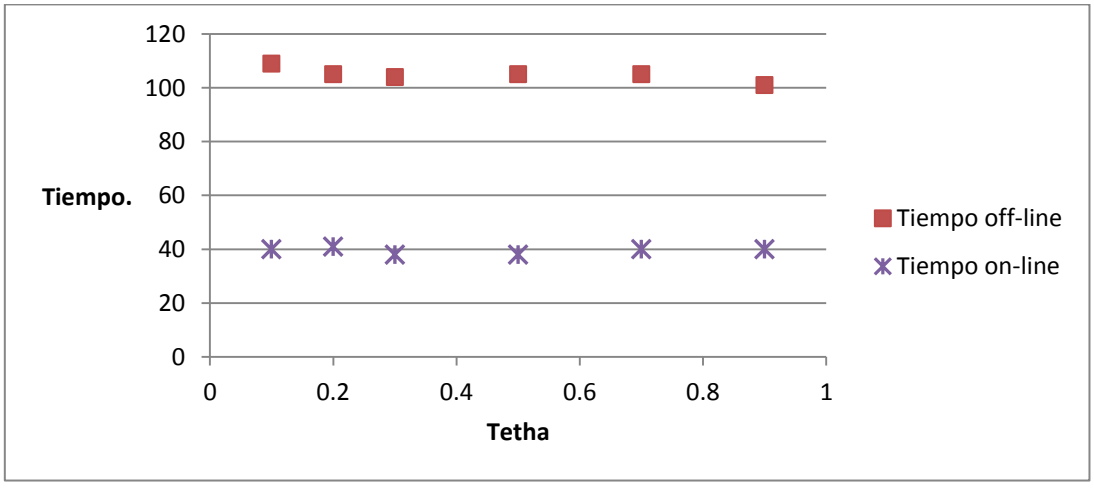


Figura A3.7 Tiempo de procesamiento para Joensuu k = 7.

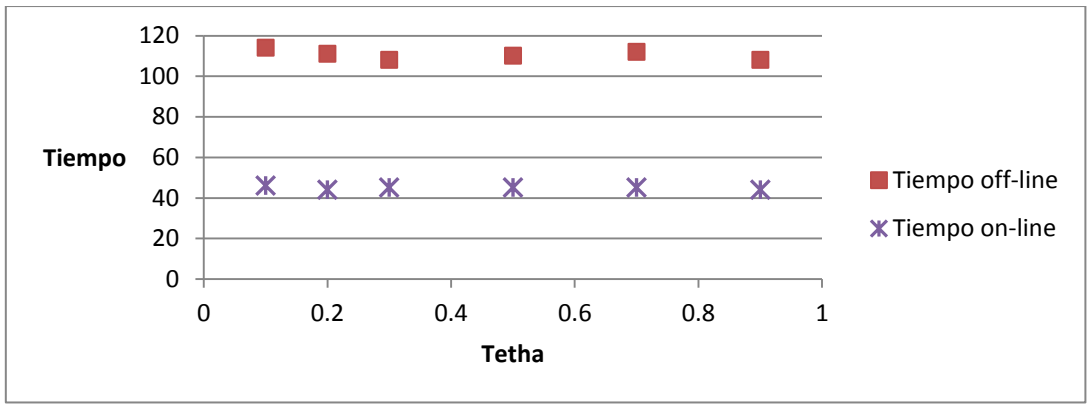


Figura A3.8 Tiempo de procesamiento para Joensuu k = 9.

Tiempo de la BD MOPI

Tabla A3.3 Tiempo de procesamiento de la BD MOPI.

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	158	163	179	188	200
0.2	157	164	182	188	197
0.3	159	168	181	187	198
0.5	160	170	177	190	196
0.7	159	171	178	197	201
0.9	154	164	177	192	201

(a)

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	43	48	52	59	63
0.2	42	48	52	59	63
0.3	44	49	54	60	62
0.5	44	50	56	58	65
0.7	42	50	52	60	64
0.9	44	48	53	59	65

(b)

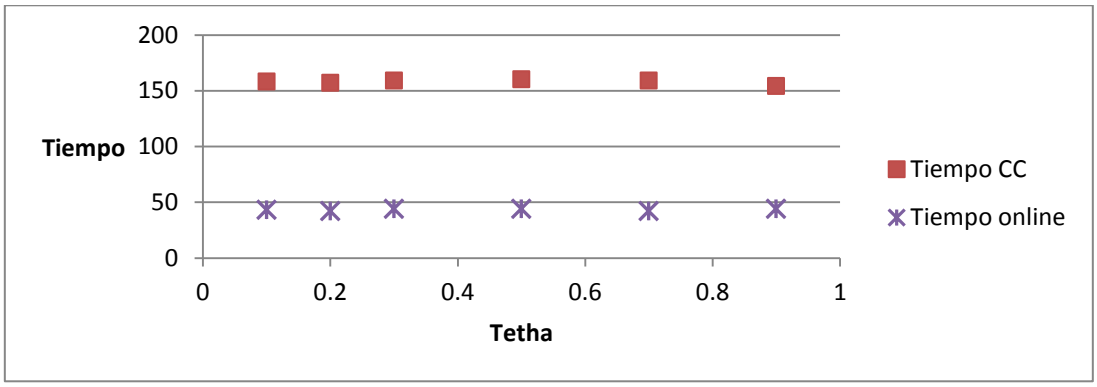


Figura A3.9 Tiempo de procesamiento para MOPI k = 1.

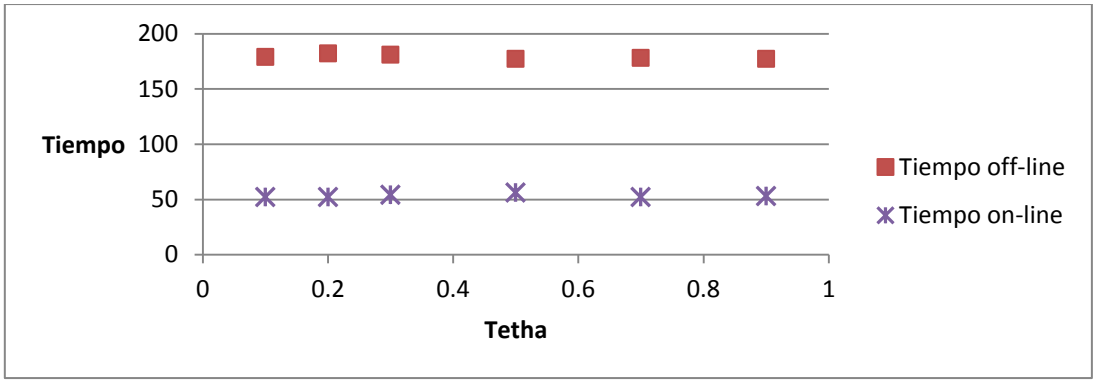


Figura A3.10 Tiempo de procesamiento para MOPI k = 5.

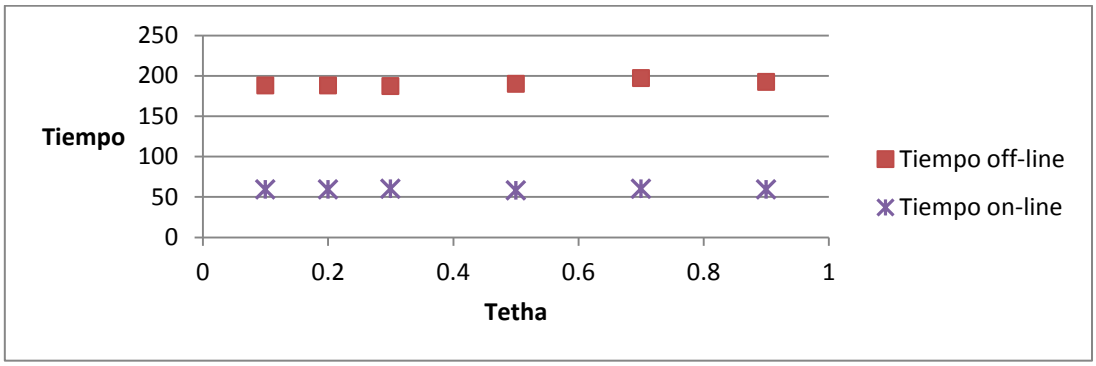


Figura A3.11 Tiempo de procesamiento para MOPI k = 7.

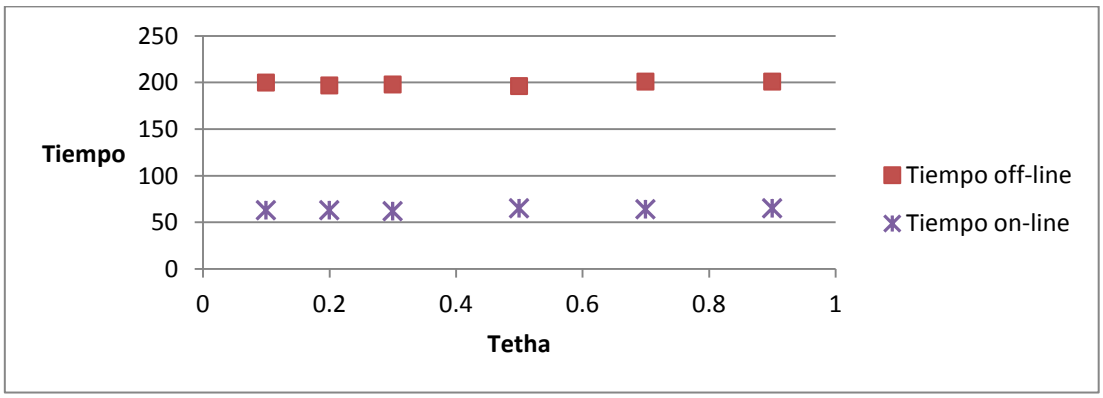


Figura A3.12 Tiempo de procesamiento para MOPI k = 9.

Tiempo de la BD BIRCH1

Tabla A3.4 Tiempo de procesamiento de la BD Birch1.

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	50640	46648	50715	51351	47590
0.2	44541	44447	49520	45625	47687
0.3	43669	44026	46686	44990	43746
0.5	42830	44977	44715	43370	43002
0.7	42464	43029	44710	43277	43122
0.9	42464	42837	44859	43687	3178

(a)

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	1913	1969	2069	2239	2313
0.2	1515	1594	1879	2017	1955
0.3	1397	1435	1676	1876	1842
0.5	1328	1536	1665	1743	1764
0.7	1363	1513	1769	1890	1953
0.9	1357	1419	1620	1963	1906

(b)

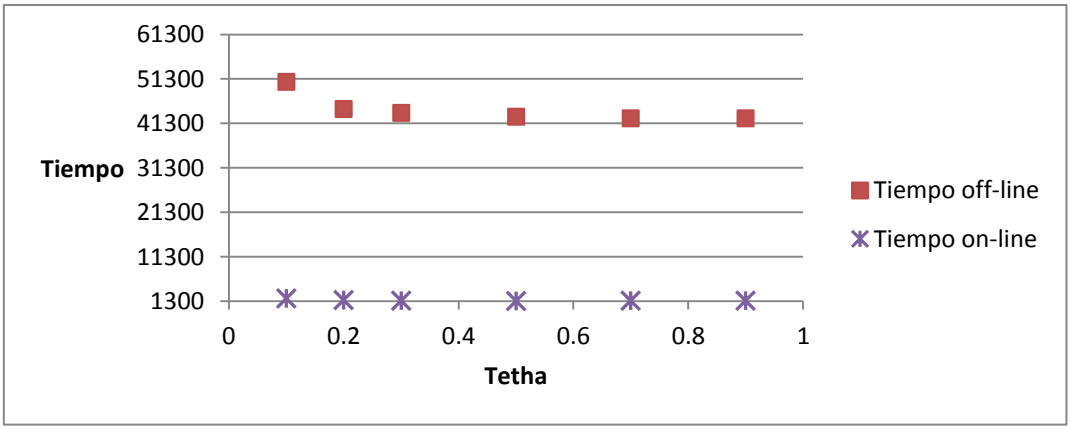


Figura A3.13 Tiempo de procesamiento para Birch1 k = 1.

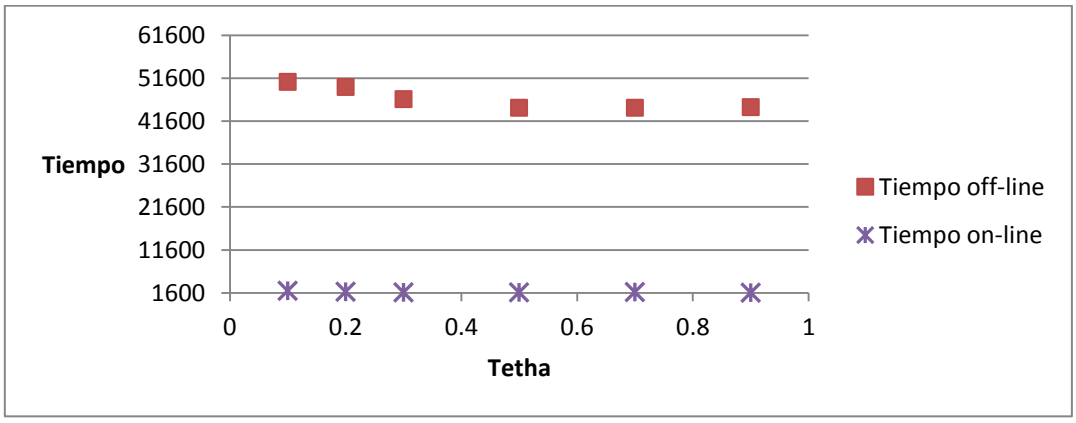


Figura A3.14 Tiempo de procesamiento para Birch1 k = 5.

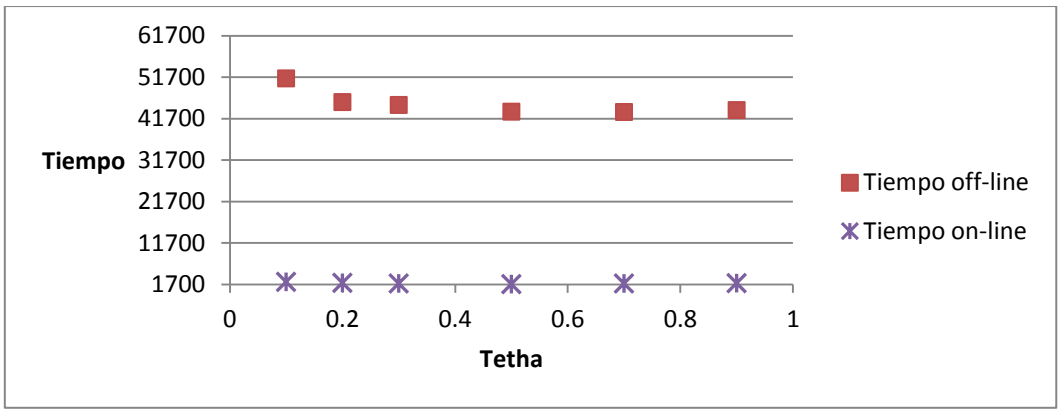


Figura A3.15 Tiempo de procesamiento para Birch1 k = 7.

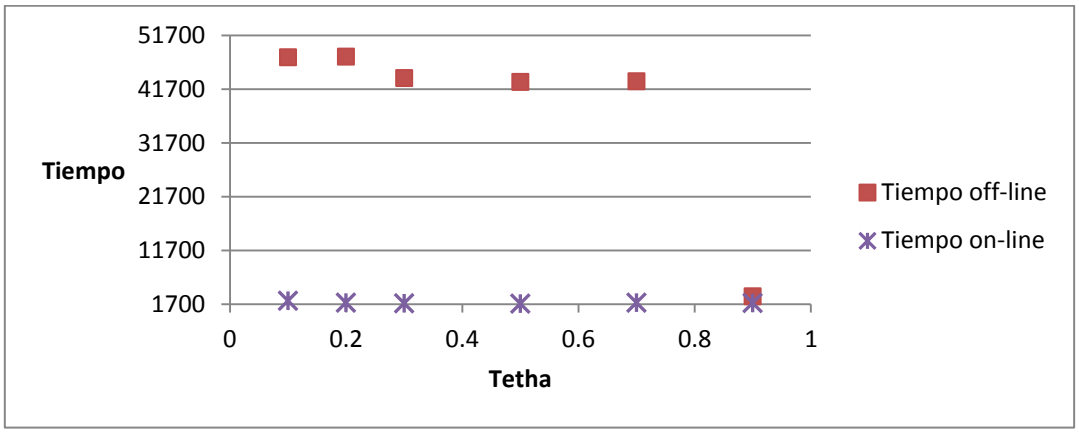


Figura A3.16 Tiempo de procesamiento para Birch1 k = 9.

Tiempo de la BD BIRCH2

Tabla A3.5 Tiempo de procesamiento de la BD Birch2.

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	9557	8676	8955	8725	9098
0.2	8630	8542	8518	8865	8835
0.3	8361	8478	8564	8725	8787
0.5	8237	8494	8579	8652	8652
0.7	8243	8368	8703	8868	8770
0.9	8276	8880	9121	1978	8955

Las gráficas de tiempo se encuentran en el apéndice II.

Tiempo de la BD BIRCH3

Tabla A3.6 Tiempo de procesamiento de la BD Birch3.

Tiempo total (s)					
θ/k	1	3	5	7	9
0.1	8655	8864	8877	9014	9203
0.2	8425	8709	8773	8861	9035
0.3	8209	8267	8572	8603	8845
0.5	8233	8241	8634	8514	8766
0.7	8235	8382	8452	8786	8789
0.9	8231	8463	8601	8651	8723

Las gráficas de tiempo se encuentran en el apéndice II.

ANEXO I
ARTÍCULOS

Optimización de Recurso para el Tratamiento de Grandes Volúmenes de Datos

Mónica Santibáñez[1], Rosa María Valdovinos[2], Erendira Rendón[3], Roberto Alejo[4], J. Raymundo Marcial-Romero[2]

[1]Centro Universitario UAEM Texcoco, Jardín Zumpango s/n Fracc. El Tejocote,
[2]Universidad Autónoma del Estado de México. Facultad de Ingeniería, Cerro de Coatepec s/n Toluca, México,
[3]Instituto Tecnológico de Toluca, Av. Tecnológico s/n Fracc. La Virgen, Metepec, México,
[4]Tecnológico de Estudios Superiores de Jocotitlán, Carretera a Toluca Atlacomulco Km. 44.8 s/n Jocotitlan, México
{monicass_isc@hotmail.com, li_rmvr@hotmail.com, erendon@ittoluca.edu.mx, ralejoll@hotmail.com, jrmarcialr@uaemex.mx}

Resumen. El crecimiento exponencial de los datos en los últimos años ha generado la búsqueda de nuevas soluciones que permitan su tratamiento, con los recursos disponibles de forma rápida y precisa. En minería de datos esta situación es fundamental por utilizar grandes volúmenes de datos en sus procesos. En este artículo se propone una metodología centrada en la administración dinámica de la memoria RAM que permite realizar el aprendizaje *on-line* de patrones. Con la utilización de métodos de agrupamiento y clasificación con la regla de los *k*-vecinos más cercanos, se valida la propuesta sobre 3 conjuntos de datos, los cuales muestran la competitividad de la metodología respecto a una situación donde el conjunto de datos fuera manipulado *off-line*, con una reducción significativa en tiempo de procesamiento.

Palabras clave: Aprendizaje *online*, escalabilidad, minería de datos, *clustering*.

1 Introducción

La última década ha representado, para la tecnología, una de las épocas con mayor avance, con lo que el crecimiento de los datos y el flujo de información ha sido un tema que cada vez cobra mayor interés, ya que el tratamiento de los datos se traduce en la adquisición de conocimiento. En consecuencia, el incremento exponencial de éstos, demanda que el almacenamiento y manejo de los mismos ofrezcan una respuesta en tiempo real de forma dinámica. Esta situación ha propiciado la búsqueda de nuevas soluciones que permitan el tratamiento de grandes volúmenes de datos, considerando los recursos disponibles.

Al respecto, la minería de datos es una de las áreas de la Inteligencia Artificial que se ha preocupado por proporcionar estrategias en el tratamiento de grandes volúmenes de datos para su procesamiento y adquisición de conocimiento [1]. En específico, la

etapa de Reconocimiento de Patrones (RP), centrada en el reconocimiento y clasificación de objetos definidos por sus atributos, ofrece una alternativa en la adquisición de conocimiento a partir de un conjunto de datos (CD). En la aplicación del RP pueden distinguirse dos aproximaciones: aprendizaje *on-line* y aprendizaje *off-line* [2]. Por un lado, el aprendizaje *on-line* contempla el aprendizaje dinámico de patrones, utilizando para ello CD actualizados, en tanto que en el aprendizaje *off-line* se realiza un aprendizaje estático que no facilita la integración de nuevos ejemplos al conjunto de entrenamiento (CE), donde cualquier caso nuevo debe ser integrado, por un experto en el área, en un nuevo CE e iniciar nuevamente el proceso de entrenamiento.

Un aspecto importante a considerar es que muchos de los datos que se generan en tiempo real, carecen de cualquier descripción que indique de forma clara su clasificación. Para esto, las propuestas existentes se orientan a la utilización de algoritmos de agrupamiento o *clustering*, con los cuales es posible generar grupos a partir de datos sin ninguna descripción, cuyos miembros sean lo más parecidos entre sí y lo más diferentes a los de otros grupos, haciendo posible de esta forma encontrar una relación entre los datos [3].

Para que esta propuesta sea viable con grandes volúmenes de datos, es fundamental administrar de forma adecuada los recursos computacionales disponibles. El problema principal en este sentido es que en ocasiones el CD es más grande que el tamaño de memoria disponible para su procesamiento, por lo que es preciso encontrar una forma de tratarlos sin alterar su precisión. Algunas estrategias se orientan a la escalabilidad de algoritmos [4] y otras a la optimización de la memoria [5]. La primera estrategia busca mantener el adecuado funcionamiento del algoritmo de minería de datos al particionar el CD, en tanto que la segunda, busca el procesamiento de los datos atendiendo a los recursos disponibles del equipo en uso.

En el presente artículo se propone una estrategia que combina ambos enfoques, por un lado realizar la administración de la memoria RAM para trabajar con CD de gran tamaño, sobre los cuales se escalan algoritmos de agrupamiento para llevar a cabo el aprendizaje *on-line* con igual o mejor precisión que si se estuviese trabajando con el CD completo.

La estructura del artículo es la siguiente: en la sección 2 se presenta la fundamentación teórica del aprendizaje *on-line* y se describe el algoritmo de agrupamiento utilizado, en tanto que la sección 3 muestra el funcionamiento y administración de la memoria RAM en Java. La estrategia de solución se describe en la sección 4 y el análisis experimental en la sección 5. Por último se incluye una sección para las conclusiones y las líneas abiertas de estudio.

2 Aprendizaje *On-line*

Actualmente, el proceso de minería de datos aplicado a grandes volúmenes de datos para obtener conocimiento requiere de técnicas capaces de soportar el procesamiento y análisis en tiempo real de datos; es el aprendizaje on-line o incremental, quien cubre estas necesidades, pues con su implementación se procesan flujos continuos de datos, con la capacidad de dar una respuesta en todo momento [6]. Esta apro-

ximación resulta ventajosa principalmente cuando se requiere mantener un conocimiento del clasificador actualizado y dinámico, no obstante, la complejidad de implementación es una tarea no trivial de resolver. En términos generales el proceso de aprendizaje on-line consiste en procesar uno a uno los patrones que son ingresados [7]. De acuerdo a [8], el aprendizaje on-line supera notablemente al aprendizaje off-line (*batch*), pues a través del uso de una escala para medir el tiempo de cálculo, demuestra que el rendimiento sufre una variación significativa en varios casos.

El aprendizaje *on-line* ha sido aplicado en diversas áreas y tareas en las que la generación y el análisis de datos continuos es de gran importancia. Un caso particular es la utilización en áreas donde el procesamiento constante de cada dato aporta al sistema la información para mantenerse actualizado y poder brindar respuesta en tiempo real, como el monitoreo de redes, los sistemas de telecomunicación, los flujos generados por los clics de los clientes, los mercados de valores, entre otras tareas [9].

Dado que muchos de los datos generados en tiempo real no cuentan con ninguna descripción, es necesario contar con algoritmos que permitan identificar algún comportamiento en ellos. Los algoritmos que brindan esta posibilidad son los de agrupamiento o *clustering*. La idea general de estos métodos es separar los patrones en grupos con características similares, utilizando para ello medidas de similitud o disimilitud que aseguren la mayor semejanza entre los patrones que forman un grupo y lo más diferente a los contenidos en otros grupos [10].

En su aplicación se pueden distinguir dos enfoques: el *hard clustering*, en el cual se generan grupos sin solapamiento y que a su vez se divide en algoritmos de partición, que generan un determinado número de grupos dependiendo de la función de densidad y los algoritmos jerárquicos, que generan agrupamientos internos a uno existente, y el *soft clustering*, que trabaja con el solapamiento de grupos. Estos algoritmos están basados en los métodos difusos, las redes neuronales artificiales y los algoritmos genéticos.

En esta propuesta se emplea el algoritmo de *clustering* Batchelor y Wilkins, correspondiente al *hard clustering*, también conocido como algoritmo de máxima distancia. En su funcionamiento requiere del parámetro θ, el cual, de acuerdo a la literatura, debe estar en el rango $0 \leq \theta \leq 1$, e indica la distancia media entre los grupos existentes. A partir del parámetro θ se calcula un umbral de distancia que permite decidir cuándo se crean nuevos grupos [11].

El algoritmo consiste en asignar arbitrariamente uno de los patrones como centro del primer grupo y utilizando la distancia, en este caso Euclídea, seleccionar el patrón más alejado y asignarlo como centro del segundo grupo. A partir de los dos grupos creados se crea una tabla que contiene los pares (patrón, centro más cercano), entre ellos se elige el patrón más alejado de los grupos existentes y se calcula el umbral de distancia, este umbral es comparado con la distancia del patrón a su centro más cercano, si el umbral es superado se crea un nuevo grupo, si no concluye este paso y se continua con la agrupación libre, donde los patrones que no fueron seleccionados como centro de grupo de forma iterativa son asignados al centro más cercano, recalculando de esta forma el centroide, hasta que se concluye con todos los patrones [11].

3 Manejo de memoria RAM con Java

El manejo de memoria permite adaptar el recurso de RAM a las necesidades que se tengan, por ejemplo, es posible tener acceso a los datos almacenados en la RAM mediante acceso directo a través de una dirección de memoria y realizar algunas acciones como la lectura y escritura [12]. En el caso de las aplicaciones, la memoria RAM a utilizar es designada de antemano, en el caso de Java (el lenguaje de programación que se utiliza para desarrollar la propuesta), interactúa con la RAM a través de la JVM (JVM por sus siglas en inglés, Java Virtual Machine), siendo esta la responsable de que Java sea multiplataforma. En la división de memoria de la JVM se maneja el espacio que será asignado tanto a los datos permanentes, como a datos que son dinámicos, es decir, aquellos necesarios sólo durante un periodo de la ejecución (Fig. 1) [13].

Fig. 1. Estructura de la memoria JVM.

En la estructura de la memoria de la JVM, la memoria *Heap* se crea al iniciar la máquina virtual de Java, esta se carga inicialmente con espacio de 64 MB, los cuales pueden aumentar o contraerse, de acuerdo a las necesidades de la aplicación. Dentro de esta área de memoria se implementa el administrador automático de almacenamiento (*automatic storage management*) [14] que mantiene un historial de los objetos y al momento de no ser referenciados libera el espacio con el uso del recolector de basura o *Garbage Collector*.

El área de métodos se crea con el inicio de la máquina virtual y puede también ser de tamaño fijo o dinámico, de acuerdo a los requerimientos en tiempo de ejecución, aunque también implementa un límite máximo, en este caso la memoria tampoco necesita ser contigua. Java no provee mayor información sobre la memoria, ya que estos datos se dejan a discreción del implementador, pero dentro de la información que si puede ser consultada, se encuentran las instrucciones para manipular el tamaño de las dos áreas de memoria [15]. Al respecto, una de las propuestas más recientes es la de Choi quien propone un algoritmo escalable de memoria externa, que permite tratar con el problema *MaxRs* de las bases de datos espaciales y los algoritmos *in-memory*, así mismo propone un algoritmo para el problema *MaxCRS* [5].

4 Herramientas y métodos

En esta sección se describen las herramientas y métodos utilizados en la propuesta, donde se busca realizar una recuperación eficiente de patrones en forma on-line y la aplicación de escalabilidad al algoritmo de agrupamiento.

4.1 Administración de memoria

Para identificar la capacidad de memoria RAM del equipo en uso, y a partir de ésta determinar el tamaño que los subconjuntos han de tener para ser procesados de forma consecutiva, se utiliza la librería Libsigar [16], que implementa las funciones necesarias para conocer el tamaño total de memoria en MB, así como el número de procesadores con los que cuenta el equipo, cantidad de espacio libre de memoria, que es manejada en KB, entre otros datos del sistema. Al trabajar con Java, puede modificarse el tamaño por default asignado a la memoria, para aumentar la capacidad y evitar desbordamientos.

Una vez identificado el recurso disponible, se destina un 10% para la ejecución de la aplicación. De acuerdo a este espacio disponible se asigna el 80% a la memoria volátil de la JVM y el 20% a la memoria permanente. Esto se debe a que en la memoria volátil se almacenan las variables dinámicas, que serán utilizadas por un "periodo", mientras que la memoria permanente almacena aquellos datos que estarán presentes durante toda la ejecución del programa, entre ellas se encuentran las librerías, clases, interfaces, entre otras.

Para fines de experimentación con los conjuntos de datos obtenidos, el tamaño fijado en la simulación para la memoria, se realiza a partir del tamaño del conjunto de datos en Bytes. Si el tamaño del conjunto en Bytes no es un número entero entonces este se redondea al siguiente entero, ya que para calcular el número de Bytes cargados en memoria se manejan sólo cantidades enteras. Enseguida la cantidad obtenida como tamaño del conjunto de datos es dividida entre el número de conjuntos con los que se desea realizar la prueba, a esta cantidad se suma el número de patrones existentes en el conjunto, ya que a cada línea hay que agregar el espacio que ocupa el carácter de salto de línea. De esta forma es posible especificar en los experimentos el tamaño total de RAM.

Para el caso particular de los experimentos aquí realizados se utilizó una computadora Dell Insipiron 1545 con 3 GB en RAM, procesador Intel® Core™ 2 Duo T6600 a 2.20 GHz, Disco Duro de 250 GB, con sistema operativo Windows 7 de 64 bits.

4.2 Conjuntos de datos

Los conjuntos de datos fueron tomados del repositorio de la SIPU (por sus siglas en inglés *Speech and Image Processing Unit*) de la Universidad del Este de Finlandia [17]. La Tabla 1 muestra las características de los CD utilizados. La primera columna corresponde a un conjunto artificial de formas arbitrarias, D31, en la segunda y tercera columna se muestra una descripción de los CD reales que corresponden a ubicaciones de sitios de interés generados por usuarios.

Table 1. Conjunto de ubicaciones de usuarios

Característica	D31	Joensuu	MOPSI
No. de vectores	3100	6014	8589
No. de *clusters*	31	-	-
Dimensiones	2	2	2
Tamaño de archivo	49.5KB	110KB	155KB

Pese al tamaño de los CD, no fue posible conseguir uno que de forma real superara los recursos disponibles de RAM. Por esta situación, se introdujo a la aplicación un valor que simula que el tamaño de la memoria es menor al real, para poder trabajar con la división del conjunto de datos de acuerdo a la capacidad de RAM.

Cada uno de los CD se divide de acuerdo al tamaño asignado a la memoria volátil de la JVM. Por lo tanto se obtienen n subconjuntos de un tamaño no mayor al especificado. Cada uno de estos subconjuntos es procesado de forma iterativa hasta concluir con todo el CD. Los subconjuntos obtenidos para todos los conjuntos de datos fueron tres y los tamaños correspondientes son: D31=19.5312KB, Joensuu=48.4131KB y MOPSI=52.2461KB. Con los subconjuntos obtenidos se aplicó el algoritmo heurístico incremental para encontrar los grupos de datos que comparten características similares. Como ya se explicó en la sección 2, el algoritmo requiere del parámetro libre θ. Para fines de esta experimentación, los valores de los parámetros libres son: θ = 0.1, 0.2, 0.3, 0.5, 0.7, 0.9.

5 Resultados y discusión

Para fines de validación, a cada subconjunto obtenido se aplicó el método de validación cruzada [18] con 2 repeticiones, utilizando un 80% de los patrones para entrenamiento y el 20% restante para el conjunto de prueba, en tanto que para analizar los resultados se utilizó la precisión general, expresada por la siguiente ecuación (1).

$$PG = \frac{\sum_e x}{p}. \quad (1)$$

donde e = patrones correctamente clasificados y p = total de patrones presentados para su clasificación.

5.1 Clasificación

La Figura 2 muestra los resultados obtenidos al aplicar el algoritmo Batchelor y Wilkins, seleccionado por ser un algoritmo que no requiere conocer de antemano el número de grupos que deben ser encontrados, además de contar sólo con un parámetro libre que permite calcular de forma dinámica el umbral para saber si debe crear un nuevo grupo o no. Como valor de referencia, se incluyeron los resultados obtenidos con el CD original, es decir ejecutando el algoritmo de agrupamiento sobre el CD cargado en su totalidad en la RAM. El algoritmo de clasificación base utilizado es la

regla del vecino más cercano en su modalidad k-NN [19], siendo k=3 para todos los conjuntos de forma completa y on-line, ya que este es el valor que obtuvo la precisión más alta procesando el conjunto real con mas instancias de forma completa.

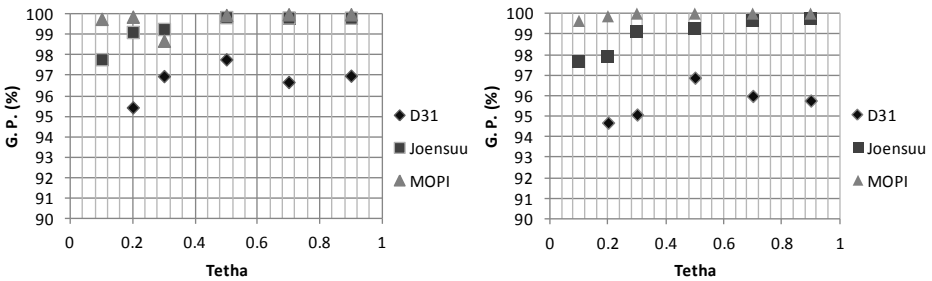


Fig. 2. Precisión general obtenida al procesar el conjunto con el método de *clustering* y la regla k-NN, para cada uno de los conjuntos de datos. (a) precisión para el conjunto completo, (b) la precisión de forma *on-line*.

En ambas gráficas de la Figura 2, se puede observar la precisión general obtenida para cada valor de θ, donde el eje X representa el valor del parámetro θ, y el eje Y el valor de la precisión general, finalmente, la serie indica el conjunto de datos procesado. En la Figura 2a puede observarse la PG obtenida para los conjuntos de datos completos, y en la Figura 2b el resultado del procesamiento por subconjuntos.

En estos resultados es posible observar varios aspectos, primeramente las precisiones obtenidas tanto con la metodología de procesamiento on-line, como las obtenidas utilizando el CD completo, sin importar el valor que tome θ son muy parecidas. En el caso del conjunto D31 procesado de forma completa se tiene una precisión general de 95.09%, mientras que al realizar el proceso on-line es de 93.43%. Para el conjunto Joensuu la precisión que se obtiene con el conjunto completo es en promedio del 99.28, y on-line es del 98.90%. Por último, el procesamiento on-line supera con 99.91% la precisión obtenida con el CD completo, que es de 99.72%.

Por otro lado, respecto al rendimiento del algoritmo de *clustering*, es posible observar que los valores de θ que mejor rendimiento ofrecen al procesar el CD completo son de 0.5 a 0.9, en tanto que para el aprendizaje on-line en su mayoría es de 0.7. Un caso particular son los resultados obtenidos con el CD MOPSI, pues con cualquier valor de θ ofrece una precisión entre 99% y 100%.

Finalmente, cuando se asigna el valor de $\theta = 0.1$, la precisión es menor con respecto a los demás valores. Esta situación puede observarse con claridad en el conjunto D31, cuya precisión general para dicho valor oscila entre 71 y 86%.

5.2 Tiempo de procesamiento

El tiempo requerido en el procesamiento de los CD es un factor importante a analizar, ya que la velocidad en procesamiento es uno de los objetivos que, adicional a la precisión, son perseguidos en minería de datos. En este sentido, el análisis se orienta a

validar la propuesta de realizar la escalabilidad del algoritmo de *clustering* al realizar el aprendizaje on-line, respecto a la utilización del CD en su totalidad cargado en memoria.

La comparativa de los resultados de tiempo del tratamiento de los conjuntos se observan en la Figura 3. Esta figura muestra dos gráficas en las que en el eje de las X se incluyen los valores de θ, en tanto que en el eje de las Y se tiene el tiempo requerido para el procesamiento de cada uno de los CD utilizados. Este tiempo es medido en segundos.

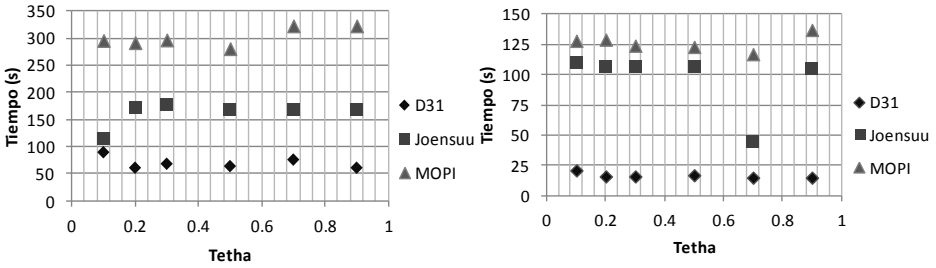


Fig. 3. Tiempo de procesamiento obtenido con cada uno de los CD. (a) Muestra la gráfica para el CD completo, (b) El tiempo por aprendizaje *on-line*.

En estos resultados es interesante observar que al realizar el procesamiento de los CD con aprendizaje on-line, el tiempo es significativamente menor respecto a cuando se procesa el CD completo en memoria. Esta situación es sumamente ventajosa ya que al realizar la administración de la memoria se tiene un ahorro hasta del 50% con un rendimiento similar a si el CD hubiese estado residente en su totalidad en la RAM (Fig. 2).

Respecto a cada uno de los CD, es posible observar que el CD D31 es 22% más rápido de forma on-line que cuando se procesa el CD de forma completa. El conjunto Joensuu se procesa en promedio un 38% más rápido que de forma completa y el conjunto MOPSI procesado de forma on-line tuvo una disminución promedio del 58% de tiempo en contraposición a su tratamiento de forma completa.

El resultado de tiempo de procesamiento está estrechamente ligado a los parámetros libres del algoritmo de *clustering*. En este sentido, es posible notar que no siempre el valor de θ que arroja la mejor precisión procesa también el conjunto en menor tiempo. Cuando se procesa el CD completo, excepto con el conjunto Joensuu la mejor precisión y el menor tiempo son obtenidos con diferente valor de θ (0.5 y 0.2 respectivamente). Esta misma situación se presenta al realizar el procesamiento on-line, el CD que proporciona un mejor rendimiento con el mismo valor de θ en precisión y tiempo, es MOPSI ($\theta = 0.3$). No obstante, estas diferencias son poco significativas ya que la precisión obtenida en su equivalente con el CD completo es muy similar en todos los casos. Por último, para todos los conjuntos de datos la diferencia entre los tiempos con respecto al mejor, oscila entre 5 y 15 segundos.

6 Conclusiones

En este artículo se propone la administración de la memoria para permitir la realización del aprendizaje on-line de patrones, dos aspectos fueron analizados, el rendimiento del clasificador y el tiempo de preprocesamiento. Los resultados obtenidos, muestran que, tanto en la simulación del aprendizaje on-line, como la utilización del CD residente en su totalidad en memoria, la precisión es muy similar. No obstante, la reducción en el tiempo requerido por el aprendizaje on-line, es significativamente inferior, siendo en la mayoría de los casos hasta en un 50%.

Esta situación podría estar ocasionada por la cantidad de patrones a procesar en cada momento, en el aprendizaje *on-line* es menor. Esto tiene que ver no sólo con la memoria, sino también con la cantidad de procesos que realiza el procesador. Por tanto, la administración de memoria y la adaptación de los grandes CD hacen que puedan ser procesados en menor de tiempo, sin perder precisión y aprovechando al máximo el recurso computacional. Respecto a los valores de θ se pudo observar que los mejores resultados se obtienen cuando su valor oscila entre 0.3 y 0.9.

Las líneas abiertas de estudio se orientan a ampliar la experimentación con otros CD, buscando la adquisición de CD que superen de manera real el recurso computacional disponible. Además de analizar la conveniencia de utilizar los agrupamientos realizados de forma *on-line* para construir *ensembles* de clasificadores.

Agradecimientos. Este trabajo fue realizado gracias al apoyo recibido del proyecto de la UAEM Generación y conteo de cubiertas de aristas en graficas asíclicas.

Referencias

1. Hernández O., J., Ramírez Q., M. J., Ferri R., C.: Introducción a la Minería de Datos. Person Educación, Madrid (2004)
2. Alppaydin, E.: *Introduction to Machine Learning* (2ª ed.). The MIT Press, Cambridge, MA (2010)
3. Aggarwal, C.: Framework for Clustering Massive-Domain Data Streams. En: IEEE 25th International Conference on Data Engineering, pp. 102-113. IEEE Press, New York (2009)
4. Khalilian, M., Mustapha, N.: Data Stream Clustering: Challenges and Issues. En: Proceedings of the International MultiConference of Engineers and Computer Scientists 1 (2010)
5. Choi, D.W., Chung, CW., Tao, Y.: A Scalable Algorithm for Maximizing Range Sum in Spatial databases. Proceedings of the VLDB Endowment, 5th edn. (2012)
6. Ferrer T., F. J., Aguilar R., J. S.: Aprendizaje Incremental de Reglas en Data Streams. Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA, pp. 261-270. Thomson, Sevilla (2005)
7. Mathieu, C., Sankur, O.: Online correlation clustering. Symposium on Theoretical Aspects of Computer Science, 573-584 (2010)
8. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online Dictionary Learning for Sparse Coding. Proceedings of the 26th International Conference on Machine Learning. (2009)
9. Beringer, J., Hullermeyer E.: Online clustering of data streams. Technical Report 31, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany (2003)

10. Marques de Sá P., J.: Pattern Recognition, Concepts, Methods and Applications. Springer, Alemania (2001)
11. Murty, M. N., & Devi, V. S.: Pattern recognition: An algorithmic approach. Springer (2012)
12. Cortijo, B. F. J.: Técnicas no supervisadas: Métodos de agrupamiento. Consultado en septiembre de 2012. Apuntes disponibles en la web de la asignatura [http://www-etsi2.ugr.es:8080/depar/ccia/RF0708/material.htm]
13. Lindhom, T., Yellin, F., Bracha, G., Buckley, A.: The Java Virtual Machine Specification. Java SE 7 Edition. Oracle America, Estados Unidos (2011)
14. Menchaca, M. R., García C. F. (2000). Arquitectura de la Máquina Virtual Java. Revista Digital Universitaria 1(2). Consultado en septiembre 2012, http://www.revista.unam.mx/vol.1/num2/art4/
15. Sun, Microsystems (2006). Memory Management in the Java HotSpot Virtual Machine. Sun Microsystems Inc.
16. Hyperic HQ, http://support.hyperic.com/display/SIGAR/Home
17. Speech and Image Processing Unit, Clustering datasets, http://cs.joensuu.fi/sipu/datasets/
18. Theodoridis, S., Koutroumbas K.: Pattern Recognition (4ª ed.). Elseiver Academic Press, Estados Unidos (2006)
19. Dasarathy, B.V.: Nearest Neighbor Norms: NN Pattern Clasification Techniques. IEEE Computer Society Press, Los Alamos, CA, (1991)

Applicability of cluster validation indexes for large data sets

M. Santibáñez, R. M. Valdovinos, A. Trueba
Universidad Autónoma del Estado de México
UAEM
Facultad de Ingeniería, Cerro de Coatepec s/n Toluca, México
monicass_isc@hotmail.com,
li_rmvr@hotmail.com,atruebae@hotmail.com

E. Rendón
Instituto Tecnológico de Toluca
ITT
Av. Tecnológico s/n Fracc. La Virgen, Metepec, México
erendon@ittoluca.edu.mx

R. Alejo, E. López
Tecnológico de Estudios Superiores de Jocotitlan
TESJo
Carretera a Toluca Atlacomulco Km. 44.8 s/n Jocotitlan, México
ralejoll@hotmail.com

***Abstract*—Over time, it has been found there is valuable information within the data sets generated into different areas. These large data sets required to be processed with any data mining technique to get the hidden knowledge inside them. Due to nowadays many of data sets are integrated with a big number of instances and they do not have any information that can describe them, is necessary to use data mining methods such as clustering so it can permit to lump together the data according to its characteristics. Although there are algorithms that have good results with small or medium size data sets, they can provide poor results when they work with large data sets. Due to above mentioned in this paper we propose to use different cluster validation methods to determine clustering quality, as its analysis, so at the same time to determine in an empiric way the more reliable rates for working with large data sets.**

Keywords— data mining, clustering, cluster validation, validation indexes

I. INTRODUCTION

Nowadays, the data, collected from different areas, represents one of the biggest and more interesting information sources, so their processing deliver hidden knowledge to professionals as patterns form, this allows to use it in crucial processes like medicine, financial operations and purchase trends analysis [1].

Data mining and its tools are in charge of processing data for obtaining knowledge that provides a guideline for decision making. These tools are algorithms capable of finding data patterns, either through a training process by which it gets learn to recognize common features among data as from a training sample (offline learning) or by a continuous processing, which data are unknown *a priori* (online learning) [2].

Focusing on the online learning is relevant to try with large data sets generated in real time, it might be possible makes a real time of updating [3], because it depends on the constantly entering patterns to the system and this, according to [4], reduce significantly the processing time.

There are different tasks in data mining that can be performed; one of them is clustering, it refers to data processing that does not have any type of information. For this task is implemented algorithms that can group data according to their similitude, either by determining a radius distance or through the density found among the instances [5]. Clustering is a task that requires a way of measuring grouping quality, if it does not have previous information it has to guarantee that the result is optimum. For this, it has been developed different validation techniques that allow to measure from several points of view the quality and structure of the groups obtained as a result of the clustering task [6]. It is important to analyze the behavior of these techniques in an on-line environment and with large size data sets.

This article is divided into five sections: the first one explains the clustering and validation methods. Next, introduce a review of previous work on this subject; after that, in section three the proposal is described, and in section four and five are shown the experiments and the results, respectively. Finally, conclusions and study open lines are exposed in section six.

II. RELATED WORKS

In literature there are a lot of proposals and works about diversity and usage of indexes of cluster's validation. Some of them are focused in analyzing the origin or constitution of some of these indexes, like the case of [7] who exposes mathematically the origin of the F-Measure. Others analyze one of the most diffused indexes as Rand Index and it's adjustment (Adjusted Rand Index) as it is the case of [8] who propose the ARI as a metric for evaluating the supervised classification and features' selection. The same way [9] exhibits ARI and exemplifies its implementation. Meanwhile [10] proposes a fuzzy extension of Rand Index and from this formulation drifts into other indexes as ARI, Jaccard, Fowlkes and Mallos, among others.

Other authors exhibit validation indexes based on an existing one for evaluating an improvement or a new clustering algorithm or index proposal, it's the case exhibit by [11] who proposes to resume clustering division and refractionation algorithms based on models and analyzes the clusters agreement obtained and original groups comparing results from Fowlkes-Mallows validation index and based, also, in the number of entries in blank from confusion matrix.

On the other hand Xuan [12] analyzes some of the most diffused measurements on information theory and proposes an adjustment based on the forecast proposed by Hubert and Arabie from mutual information and Information variation distance (VI) from which its normalization takes place based on normalization index. This adjustments help to maintain the index close to zero.

In its proposal Vendramin [13] exhibits a methodology for comparing, on an efficient way, 24 existing validation indexes in literature and lot aided, proposes a different analysis from Milligan and Cooper that allows identifying the agreement to the reference group (original group) and, as concordance among results of indexes, cluster's quality. On its side Wanger [6] exhibits a succession of indexes based on three ideas, pair remembering, the sum of overlapping clusters and mutual information. From this division and on, it is explained each of the indexes for, after this, propose a succession of axioms that allows to define if a certain validation index is okay taking into account that it is doing a dynamic clustering.

III. CLUSTERING

The clustering is a data mining task designed for identifying similar features between the data and put similar patterns together in the same group, while among the groups exist a difference between the features of their patterns. This task starts from an unlabeled data set, that is to said, there is no specific information indicating the classification of the patterns and it is expected that real data does not have prior information [14].

There are two principal approaches of clustering and different kinds of algorithms for doing this task. Murty [15] expose a categorization of these types, the first one corresponds to soft clustering which handles the cluster overlapping through fuzzy and genetic techniques and hard clustering where is not permitted the group overlapping. This last approach is divided into hierarchical and partition algorithms.

The hierarchical algorithms are divided into divisive and agglomerative algorithms. The first one use a top-down strategy to divide the data, and the second does the opposite, taking each point as a group and putting together the similar ones to make a big group; both using a tree known as dendrogram. The partition algorithms are based in the square error and mainly in prototypes; this prototypes serve as a reference on how a pattern should look like to belong to a cluster. In this final classification is where the clustering algorithm used here belongs.

A. Clustering algorithm

As seen, there are different kinds of clustering algorithms, but the one analyzed and implemented here is the Batchelor and Wilkins algorithm [16]. For applying the clustering algorithm the number of groups to obtain does not need to be previously known and it just needs one free parameter to calculate dynamically the radius of the groups.

This algorithm takes as a first center any of the objects and takes as a second center the furthest from the first, from these two centers are calculated the groups threshold and select again the furthest object from the two centers, its distance is compared with the nearest center. If the threshold is exceeded a new center is created, if not, then the object is assigned to the nearest center, this is done until there are no more centers. Finally, the assignation of objects is verified with its nearest centers. The main pseudo-code is shown below [16]:

Inputs:
  $\boldsymbol{X}$ set de M patterns $\{X_1, X_2, \ldots, X_M\}$
  $\theta \in [0,1]$ Fraction of the average distance between clusters
Outputs:
  $A \in N$ Number of found clusters
  $S1, S2, \ldots, S_A$ A pattern sets (clusters)
  $Z_1, Z_2, \ldots, Z_A$ Cluster centers
Auxiliar variables:
  $\boldsymbol{L}$ Unprocessed patterns
  $\boldsymbol{T}$ Pairs set (nearest pattern-center)
Initialization
  L←X
  $Z_1 \leftarrow X_1$ A←1 $S_1 \leftarrow \{X_1\}$ L←L-$\{X_1\}$
  Let $m : \delta(X_m, Z_1) = \max_{X_i \in \boldsymbol{L}} \{\delta(X_i, Z_1)\}$
  $Z_2 \leftarrow X_m$ A←2 $S_2 \leftarrow \{X_m\}$ L←L-$\{X_m\}$
Do cluster?
  While
    $T \leftarrow \emptyset$
    For each $X \in L$
      $m \leftarrow$NearestCluster(X) {save pair in T }
      $T \leftarrow T \cup \{(X, Z_m)\}$ {formed by X y Z_m
    End-for // each $X \in L$
    Let $n : \delta(X_n, Z) = \max_{(X,Z) \in \boldsymbol{T}} \{\delta(X, Z)\}$
    threshold←CalculateThreshold(θ)
    If $\delta(X_n, Z)$>threshold {Do cluster}
      $Z_{A+1} \leftarrow X_n$ $A \leftarrow A+1$ $S_A \leftarrow \{X_n\}$ $L \leftarrow L-\{X_n\}$
      end←FALSE
    If-not // In this case $\delta(X_n, Z) \leq$threshold
      end←TRUE
    End-if // If $\delta(X_n, Z)$>threshold
  Until end =*TRUE*

Free clustering
  For each $X \in L$
    $m \leftarrow$ NearestCluster (X)
    $S_A \leftarrow S_m \cup \{X\}$
    $Z_m \leftarrow$RecalculateCenter(m)
  End − for//each $X \in L$
End of algorithm

B. Cluster validation

One of the clustering features is that normally does not know the number of groups that will be formed from a data set. To evaluate clustering results in a quantity way, there are cluster validation methods that can help to know the assignation of parameters and restrictions ways that have been established. In general, cluster validation can be shown into three big approaches according to [5]; external criteria, internal criteria and relative criteria. The first criteria correspond to the validation through independence structures, it means, it reflects the intuition that it had about grouping structure, also randomness of them.

The Internal criteria evaluates the quality of clustering based on shape and internal distances, it is no necessary any prior knowledge about the right partition that has to be got, thus it is easier to implement in real tasks. It is important to mention that the cases can be used is when the clustering structure is hierarchical or from just one clustering. The test applications based on internal criteria can be reviewed in detail in [5]. The last criteria value the clustering through the comparison of the algorithms and their parameters, for example, applying different parameters in one algorithm.

According to [17] there is no criteria that ensure 100% the clustering quality because of the origin of the task, there is not any information about the result that should obtain, and it makes task to value the optimum number of clusters and their more complex quality, it mentions that external criteria assumes it has the knowledge about the right partition (ground truth o standard gold) that must get like a result, situation that is less probable in real tasks, but it can perform in experimental environments.

This study applies internal and external validation criteria. For their operation it is recommended to apply the contingency Matrix, this matrix compare the result of both partitions (in this case, real partition and the clustering result), where rows represent a cluster $X= \{C_1, \ldots, CK\}$ and columns the other cluster $P=\{C'_1, \ldots, C'_K\}$, in this case the actual clustering, where C represents a particular cluster. Whereby, each cell contains the number of patterns that both have in common (Fig. 1). While the sum of the columns represents the number of instances that should be assigned to the cluster and the sum of the rows contains the number of actual instances assigned to the cluster.

	C_1	C_2	...	C_K	
C'_1	n_{11}	n_{12}	...	n_{1K}	$n_{1.}$
C'_2	n_{21}	n_{22}	...	n_{1K}	$n_{2.}$
⋮	⋮	⋮	⋮	⋮	⋮
$C'_{K'}$	$n_{K'1}$	$n_{K'2}$	...	$n_{K'K}$	$n_{K'.}$
	$n_{.1}$	$n_{.2}$	...	$n_{.K}$	$n_{..}=N$

Figure 1. Contingency Matrix for counting-pairs. Extracted from [6].

Counting-pairs. It belongs to the internal criteria. From contingency matrix, can be calculated various index to understand the structure and quality of the clustering got. For this validation method it takes two vectors from the data set, and it compares if both are in the same cluster C and in the same group P, if it's the value is identified by SS (eq. 1). If both selected vectors are in the same cluster C but different group P then the value corresponds to the SD (eq. 2) value. If both vectors are in different cluster C and different group P then the DD (eq. 4) value must be increased. Finally, if both vectors are in different cluster and same group P then the value that should be increased is DS (eq. 3). Following, the equations are shown for these four values from the contingency matrix, and the validation index is listed.

$$SS = \left(\frac{1}{2}\right)\sum_{i=1}^{k}\sum_{j=1}^{k'} n_{ij}^2 - \left({}^{N}/_{2}\right) \quad (1)$$

$$SD = \left(\frac{1}{2}\right)\sum_{j=1}^{k'} n_j^2 - \left(\frac{1}{2}\right)\sum\nolimits_{i=1}^{k}\sum_{j=1}^{k'} n_{ij}^2 \quad (2)$$

$$SD = \left(\frac{1}{2}\right)\sum_{j=1}^{k'} n_j^2 - \left(\frac{1}{2}\right)\sum\nolimits_{i=1}^{k}\sum_{j=1}^{k'} n_{ij}^2 \quad (3)$$

$$DD = \frac{N(N+1)}{2} - \left(\frac{1}{2}\right)\left[\sum_{i=1}^{k} n_i^2 + \sum_{j=1}^{k'} n_j^2\right] \quad (4)$$

Where, n_{ij} corresponds to the number of instances in common between the clusters obtained with the algorithm and the original partitions. The following index can be obtained:

- Rand statistic. (RI, Rand Index) "Represent the fraction of pairs of cases in the same state in both partitions" [18]. i.e. the proportion of patterns equally classified. This index takes value 1 if partitions are identical.

$$R = \frac{(SS + DD)}{(SS + SD + DS + DD)} \quad (5)$$

- Adjusted Rand Index (ARI). It is the adjustment of the previous index and takes into account the hyper geometric space. According to [9] in one Milligan's previous study, ARI is recommended as a good index to determine the similitude between two clustering with different number of groups. This index was created to solve RI differences, they consist of increasing their value to more than one according to the groups quantity increase or the value is not constant. Therefore the ARI result is always between 0 and 1.

$$ARI = \frac{\binom{n}{2}(SS + DD) - [(SS + SD)(SS + DS) + (\mathrm{DS} + DD)(\mathrm{SD} + DD)]}{\binom{n}{2}^2 - [(SS + SD)(SS + DS) + (DS + DD)(SD + DD)]} \quad (6)$$

  Where, $\binom{n}{2}$ is the number of possible pair combinations.

- Jaccard coefficient. Ignores the cases that have been assigned to different clusters in both partitions. This is convenient when there are large clusters quantities and this value can be very high.

$$J = \frac{SS}{(SS + SD + DS)} \quad (7)$$

- Fowlkes y Mallows index. It also ignores the patterns assigned to different groups in both partitions.

$$FM = \frac{SS}{\sqrt{m_1 m_2}} = \sqrt{\frac{SS}{SS + SD}\frac{SS}{SS + DS}} \quad (8)$$

- F measure. It helps to determine how much the clustering resulting groups resemble to those that could have been achieved through manual sorting, therefore requires information on the actual grouping of the cluster [18]. This measure combines the precision and recall measures. Thus we have a set of clusters $C=\{C_1, \ldots, C_k\}$ and the actual classification $C = \{C'_1,\ldots, C'_k\}$, the precision and recall measures are as follows:

$$F_{i,j} = \frac{2}{\frac{1}{prec(i,j)} + \frac{1}{rec(i,j)}} \quad (9)$$

Where $prec(i, j) = |C_j \cap C'_i|/|C_j|$ y $rec(i, j) = | C_j \cap C'_i|/|C'_i|$

- Calinski-Harabasz (C-H). Also known as Percentage Variation Criterion (VRC). It evaluates the quality of the clusters through the use of variance of the patterns within the cluster and between the clusters [13]. This distance uses the centers [19]. Its performance is obtained comparing the resulting calculation of the index by varying the algorithm parameters and selecting as best grouping that has the highest value, which can be seen as a peak in the resulting graph. Even if the results have a linear trend, up or down then there is no reason to prefer one solution over another.

$$CH(C) = \frac{(N - |P|)inter_{CH}(P)}{(|P| - 1)intra_{CH}(P)} \quad (10)$$

Where $inter_{CH}(P) = \sum_{C \in P} |C|\, d(C, \bar{X})$ e $intra_{CH}(P) = \sum_{C \in P} \sum_{x \in C} d(x, \bar{C})$

- Davies-Boulin (D-B). It is based on the within-group and between-group ratio to evaluate a particular data partition, that is to say, quantifies the proportion of dispersion [19].

$$DB(C) = \frac{1}{|P|} \sum_{C_k \in P} \max_{C_l \in P/C_k} \left\{ \frac{S(C_k) + S(C_l)}{d(\bar{C}_k, \bar{C}_l)} \right\} \quad (11)$$

Where, $S(C) = 1/|C| \sum_{x \in C} d(x, \bar{C})$

IV. SET UP

As seen there are many measures that permit to validate the result of clustering task, although according to [6] there is no specific way to compare the clustering result. According to [12] there is no established measure recognized as the best one. Thus, in this article is exposed the using and way of analysis of the clustering results obtained through the implementation of the exposed methodology in [21], it means that corresponds to the online processing of the dataset according to the RAM availability in accordance with their arrival and comparing these results with those which are obtained offline, namely processing the entire data set at a time.

To perform the cluster validation on data sets that have information on their actual grouping, or reference set, raises the using of the counting-pairs index as well as the F-measure. For the data sets lacking of information or reference set, were used the Davies-Boulin and Kalinsky-Harabasz validation index, also the precedent obtained on the data sets that has actual grouping information.

The proposal is to calculate the result of each validation index and taking into account the suggestions for their analysis, comparing these values with each other and with the number of groups known beforehand to determine if the clustering quality is good as well as knowing the most appropriated value of the clustering algorithm parameter to obtain a grouping with quality close to the original. In the situation of those sets that have their actual grouping. Table 1 shows the datasets used for testing. These sets were taken from the repositories of the SIPU (Speech and Image Processing Unit, http://cs.joensuu.fi/sipu/datasets/).

The parameter used for the clustering algorithm, θ, influences the calculation of the threshold to determine when there are new groups, therefore it is important to choose a value that is as optimal as possible to get a quality clustering, for which θ was assigned values of 0.1, 0.2, 0.3, 0.5, 0.7 y 0.9. For cluster validation of artificial dataset D31 was implemented the counting-pairs through contingency matrix to calculate the index obtained from it, which are: RI, ARI, Jaccard index, Fowlkes & Millows index and the F-measure. Likewise shows the results obtained from index C-H and D-H.

V. EXPERIMENTAL RESULTS

A. D31

For data base D31, according to validation indexes, Calinski-Harabasz results are ascending (Fig. 2), which indicates that any value selected is as good as any other. This behavior is almost the same in the rest of the data base excepting that in a D31 with the values θ = 0.5, 0.7 o 0.9 D-B index is higher than C-H, meanwhile the expected ones are a high value from C-H and a value beneath D-B, compared with the rest of the resultant values of θ.

So, data showed next, takes as a reference the value of θ and 0.2 is one of the values that shows a close group, in number as the original, in the case of D31 data base. Another reason for choosing it is that it also has as a result, in the task of classification, a high precision without getting far from the original group. In Table 2 shows the values obtained in the counting-pairs, online and offline.

TABLE I. DATA DESCRIPTION

Features	D31	Joensuu	MOPSI
No. Of vectors	3100	6014	8589
No. Of clusters	31	-	-
Dimension	2	2	2
File size	49.5KB	110KB	155KB

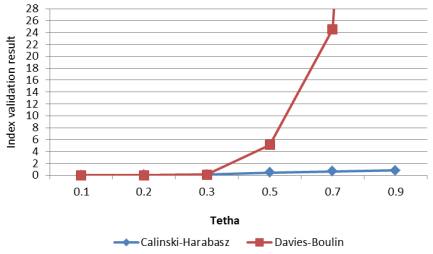


Figure. 1. Cluster validation indexes CH, DB for D31.

TABLE II: VALUE COUNTING-PAIRS OF THE PROCESSING ONLINE AND OFFLINE

	Online	offline
SS	7965.1242	75519
SD	7611.8491	68882
DS	7849.465	77931
DD	469424.39	4581118

In an intuitive way, the *SS* and *DD* high values indicates that exists similarities between divisions obtained with clustering algorithm, which is quite similar to the original, although some objects were integrated in different groups which were assigned in the partition, this is because the number of groups determined by the algorithm is lower than the original division, as it is explained ahead.

It can be observed, in Table 2, that groups formed, both online as offline are well defined because *DD* is higher, although in online processing the SS index is lower than DS, it means that within the groups are assigned patterns from a different one. This may be due to two situations, the first, the influence of θ parameter on the threshold to formed groups, then the greater the value of θ the greater the radius of groups and consequently the number of resulting groups can be different to real number, as in this case, which causes that patterns are assigned to one group that is not their own, according to the real data.

The second situation is that the patterns are so similar, that the algorithm ranks them as part of a different group from the original. This situation is observed in much smaller extent, because setting the θ parameter can be approached, very accurately the original result, this can be seen with the value of θ = 0.2, where the number of groups and their allocation is almost equal to that of the original set, consisting of 31 groups and the closest number the algorithm gets is 37.

From the counting-pairs results were calculated the index in Table 3, which shows the result of the index used to assess the quality of the clustering. To begin, generally can be seen in the Table 3 that results of offline and online processing are quite similar, the difference among them is about one hundredth, besides having the same behavior. This indicates that online processing of the dataset according to RAM available generate a clustering very similar to the offline clustering.

The Rand statistic calculation indicates the fraction from those pairs that are grouped the same way as in the clustering result as in the original partition. While if is closer to one the value of this index, the more will be similar between each other. Rand statistic focuses basically on calculating the coincidence between compared partitions; it is shown in Table 3, that concordance between data set D31 is very high.

TABLE III: RESULTS OF THE VALIDATION INDEXES PROCESSING THE DATA SET D31, OFFLINE AND ONLINE WITH Θ = 0.2

Indexes	Offline	Online
Rand Statistic	0.969435926	0.9685451
ARI	0.491336496	0.4895717
Jaccard index	0.3396677	0.3390697
Fowlkes y Millows	0.50732666	0.5060091
F-measure	0.88135201	0.896825

On the other hand the ARI, which is an adjustment from last index, quantifies the coincidence between compared partitions but takes into account the obtained and expected index, in such a way that if the value is zero, the partitions are independent and approaches to one, are the same. So, the obtained value indicates that the group isn't close to the original, but it has to be taken into account that according to [6] the significance of the measure can be affected by the supposition that is made about the distribution.

Jaccard coefficient doesn't take into account data pair that doesn't match (*DD*), and as a result of this, reflects the proportion of data that has been assigned, so, according to this index, the similarity between partitions is very low. While Fowlkes and Millows index calculus indicates de probability that the elements are assigned the same way in the group and in the real data, thus the obtained value indicates that probability is 50%.

On the other side, F-Measure calculates in a more accurate way how much clusters are similar between each and the original cluster, obtaining a balanced average of the patterns assigned correctly respect to the total of them and those who should have been assigned, thus the result obtained in Table 3 indicates the accuracy of the clustering is good, being the patterns assigned in a correct way in its most.

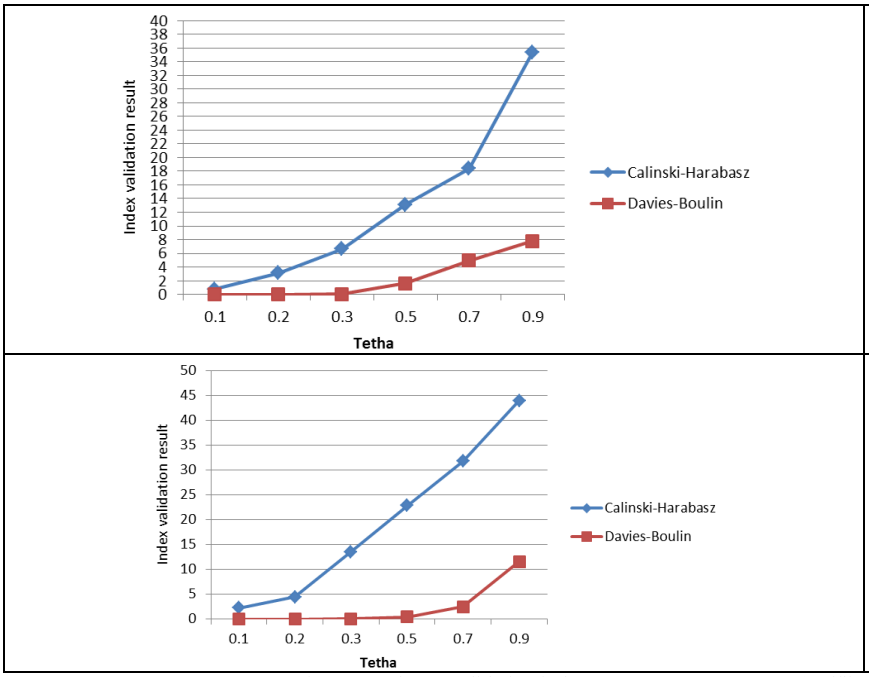
B. Joensuu and MOPI

Following will be described the results for the datasets Joensuu and MOPI. For their validation were implemented the Calinski-Harabasz and Davies-Boulin index, from the results of which is determined what is θ more convenient parameter value for processing the data. It is important to point, as it has been mentioned, that in all the results there is a growing tendency to the θ value, in both index, so that is taken as main reference the D-B index score, for which the lower value the better clustering result.

Therefore, as the majority results of D-B, the last three θ values increase significantly with respect to the first three, these last are discarded, so most of the graphics included here are shown just the most relevant area for the analysis, but at the same time shows the behavior of the clustering through the index value. Consequently the optimal values of θ are determined, from which is selected the best based on the C-H result and the number of generated groups with the corresponding θ value.

In the case of the Joensuu and MOPI datasets, and the lack of information about their clustering, their online results are compared with their offline processing results. In Fig 3 can be seen the graphics from the resulting clustering of Joensuu dataset, offline (a) and online (b). According to the graphs, the general behavior in both processing techniques take to the same tendency, even the value of θ = 0.2 the D-B value is still small, the C-H value is significantly higher than with θ = 0.1 and there is more similarity in the number of generated clusters. So the more balanced result of clustering is generated with θ = 0.2.

For MOPI dataset the results are similar to those of Joensuu, as likewise discards the last three values of θ for which the D-B result is bigger than 1. The value of θ = 0.3 is



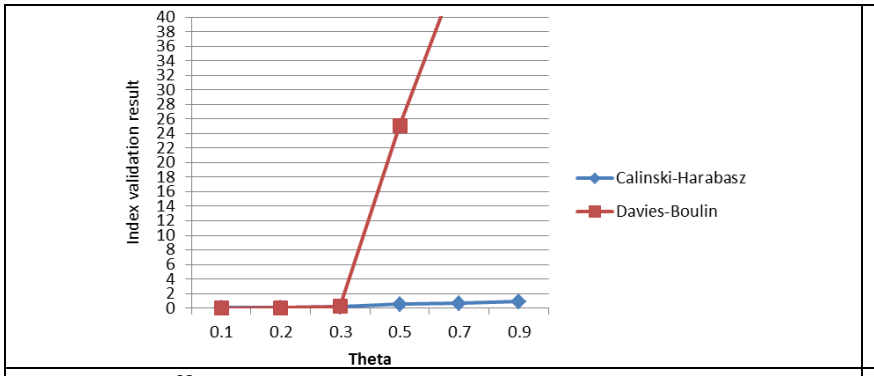
(a)

C-H	D-B	Groups
0.782539156	0.000292808	75
3.145888624	0.006643861	29
6.639860423	0.055440428	15
13.12607999	1.599240769	6
18.37597624	4.948387189	4
35.38254563	7.762014636	3

(b)

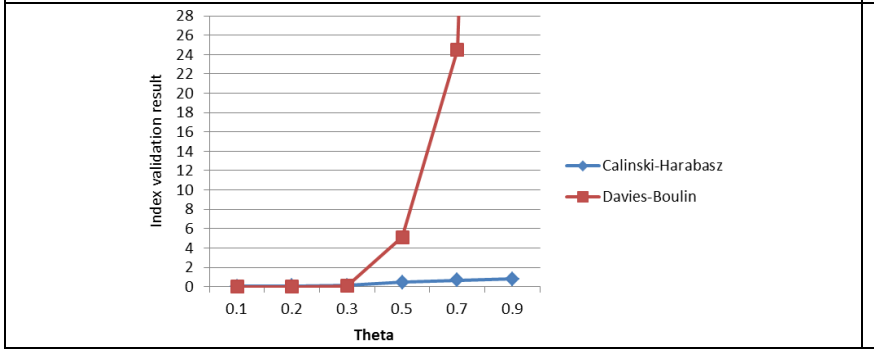
C-H	D-B	Groups
2.25388551	0.00058929	57
4.45963377	0.00352498	25
13.5032178	0.03621329	12
22.7958235	0.47361588	5
31.7780149	2.49684401	3
43.939682	11.5414901	2

Figure. 2. Cluster validation indexes CH, DB to Joensuu. (a) offline, (b) online.



(a)

C-H	D-B	Groups
0.019295739	0.000593965	19
0.059647962	0.021311501	11
0.137855536	0.244321076	5
0.538611023	25.09372648	3
0.666525355	46.93172956	2
0.867211814	1068.606773	2



(b)

C-H	D-B	Groups
0.01913797	0.00023392	14
0.07432147	0.00805482	9
0.12976198	0.0851258	4
0.47157364	5.10235998	3
0.66398435	24.5350898	2
0.79557315	190.210248	2

Figure. 3. Cluster validation indexes CH, DB to MOPI. (a) offline, (b) online.

discarded because the prior value is increased considerably, thus the first two θ values could be selected randomly, because, according to the result of index validation, these takes its expected value, but θ = 0.2 generates a number of groups near to offline processing data sets (Fig.4).

Based on the previous analysis of the results is possible to see that no all the validation index quantify the clustering quality in the same way, and this quantity sometimes does not completely describe the clustering quality. Due to mention before, taking into account that the results for D31 dataset are known, could be determined that in the most of the experiments, with all the θ values for this dataset, the F-measure is closer to the right description of the cluster quality, as it was mentioned it considers the number of assigned patterns correctly to respect all these and the ones that really should be assigned.

Furthermore, in the datasets which only knows the number of groups, the Calinski-Harabazs and Davies-Boulin index with the real number of groups can be used to determine which the largest quality clustering, thus is finding the corresponding θ value for that clustering. From both index have a growing trend, the clustering analysis based on both allows finding more reliable clustering of better quality. As for the datasets that lack of any information describing the actual grouping, both indexes are used again to define the quality.

VI. CONCLUSIONS

In this article we checked some of the most used cluster validation index to determine the clustering quality. This quality was analyzed based on the result of all the indexes, to know its quality and finally which of them quantifies the actual, being the F-measure the one that offers the most reliable results. The results presented correspond to those obtained from the clustering step of the methodology presented in [21] so detailed presentation and analysis.

The clustering result is quite close to the expected and through this analysis can be seen that for these datasets an optimal θ value can be specified as follows: $0.1 < \theta <= 0.2$. Consequently, for determining the quality of clustering in datasets with information about the real grouping, the F-measure gives the clearest value about the quality of the clustering result. As seen in the case of datasets with lack of information no index is the best, and then both are used.

The Future lines for this issue are oriented to the proposal to implement different validation index and analyze their behavior, as well as work with other datasets to parse through the index an optimal θ value, and compare if is the same as in this paper.

ACKNOWLEDGMENT

We would like to thank to the Instituto Tecnológico de Toluca and also acknowledge the PROMEP/103.5/12/4783 project for their financial support.

REFERENCES

[1] E. Alppaydin, Introduction to Machin Learning 2nd ed. Cambridge, MA: The MIT Press, 2010.

[2] A. Schwaighofer, J. Q. Candela, T. Borchert, T. Graepel, and R. Herbrich, "Scalable clustering and keyword suggestion for online advertisements," in Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising, ACM, 2009, pp. 27-36.

[3] J. Beringer, E. Hullermeyer, Online clustering of data streams. Technical Report 31, Department of Mathematics and Computer Science. Philipps-University Marburg: Germany, 2003.

[4] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in Proceedings of the 26th International Conference on Machine Learning, 2009.

[5] S. Theodoridis, K. Koutroumbas, Pattern Recognition 4th ed. Estados Unidos: Elseiver Academic Press, 2006.

[6] S. Wagner, and D. Wagner, Comparing clusterings: an overview, Universität Karlsruhe, Fakultät für Informatik, 2007.

[7] Y. Sasaki, Y. "The truth of the F-measure" in Teach Tutor mater, 2007, pp. 1-5.

[8] J. M. Santos, and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," ICANN, Artificial Neural Networks, ICANN, 2009, pp. 175-184.

[9] K. Y. Yeung, W. L. Ruzzo, "Details of the adjusted rand index and clustering algorithms," in Bioinformatics, vol. XVII, 2001, pp. 763–774.

[10] R. J. Campello, "A fuzzy extension of the rand index and other related indexes for clustering and classification assessment" in Pattern Recognition Letters, vol. XXVIII no. 7, 2007, pp. 833-841.

[11] J. Tantrum, A. Murua, and W. Stuetzle, "Hierarchical model-based *clustering* of large datasets through fractionation and refractionation," in Information Systems, vol. XXIX no. 4, 2004, pp. 315-326.

[12] V. N. Xuan, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?," in Proceedings of the 26th International Conference on Machine Learning, Montreal, Canada, 2009.

[13] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka, On the comparison of relative clustering validity criteria, Sparks, 2009.

[14] J. Hernández, M. J. Ramírez, C. Ferri, Introducción a la Minería de Datos. Madrid: Person Educación, 2004.

[15] M. N. Murty, and V. S. Devi, Pattern recognition: An algorithmic approach. Springer, 2012.

[16] B. F. Cortijo, Técnicas no supervisadas: Métodos de agrupamiento [*online*]. 2001. Disponible en: http://www-etsi2.ugr.es:8080/depar/ccia/RF0708/material.htm.

[17] G. O. Arbelaitz, R. J. Muquerza, "Aportaciones a la clasificación no supervisada y a su validación. Aplicación a la seguridad informática,". Tesis doctoral, Universidad del País Vasco, Facultad de Informática, 2010.

[18] D. A. Ingaramo, M. L. Errecalde, and P. Rosso, "Medidas internas y externas en el agrupamiento de resúmenes científicos de dominios reducidos," in Procesamiento de Lenguaje Natural vol. XXXIX , 2007, pp. 55-62.

[19] J. Lewis, M. Ackerman, and V. De Sa, "Human cluster evaluation and formal quality measures," in Proc. 34th Annual Conference of the Cognitive Science Society, 2012.

[20] A. B. Garay, R. P. Escarcina, and Y. T. Valdivia, "Validación de clusters usando IEKA y SL-SOM".

[21] M. Santibáñez, R. M. Valdovinos E. Rendón, R. Alejo, and J. R. Marcial-Romero, "Optimización de Recurso para el Tratamiento de Grandes Volúmenes de Datos," in Research in Computing Science Avances en Inteligencia Artificial vol. *62*, 2013, pp. 15-24.

REFERENCIAS

[Abe, 2010] Abe, S. (2010). Support vector machines for pattern classification. Springer.

[Alber, 2006] Alber, S. (2006). Online algorithms. En: D. Goldin, A. S. Smolka y P. Wegner (eds.), Interactive Computacion: The new paradigm, Berlin-Heidelberg: Springer, pp. 143-164.

[Alejo, 2008] Alejo E., R. (2008). Análisis del error en redes neuronales: Corrección de los datos y distribuciones no balanceadas. Tesis doctoral. Universitat Jaume I, Departament de Llenguatges I Sistemes Informatics.

[Alpaydin, 2010] Alppaydin, E. (2010). Introduction to Machin Learning (2ª ed.). Cambridge, MA: The MIT Press.

[Al-Sultan, 1993] Al-Sultan K.S., Selim S, Z. (1993). A global algorithm for the fuzzy clustering problem. Pattern Recognition 26(9): 1357-1361.

[Anagnostopoulos, 2010] Anagnostopoulos, T., Anagnostopoulos, C., Hadjiefthymiades, S. (2010). An Online Adaptive Model for Location Prediction. In: Autonomic Computing and Communications Systems. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 23, 64-78.

[Arbelaitz, 2010] Arbelaitz G., O., Muquerza R., J. (2010). Aportaciones a la clasificación no supervisada y a su validación. Aplicación a la seguridad informática. Tesis doctoral. Universidad del País Vasco, Facultad de Informática.

[Arenas, 2011] Arenas, S., Haeger, J. F., Jordano, D. (2011). Aplicación de técnicas de teledetección y GIS sobre imágenes Quickbird para identificar y mapear individuos de peral silvestre (Pyrus bourgeana) en bosque esclerófilo mediterráneo. Revista de Teledetección 35, 55-71.

[Arizmendi, 2010] Arizmendi A., A. (2010). Determinar el número óptimo de grupos utilizando índices de validación interinos y externos. Tesis de licenciatura en Ingeniería en Sistemas Compu-

tacionales. Instituto Tecnológico de Toluca.

[Barbakh, 2008] Barbakh, W., y Fyfe, C. (2008). Online clustering algorithms. International Journal of Neural Systems, 18(03), 185-194.

[Beringer, 2003] Beringer, J., Hullermeyer E. (2003). Online clustering of data streams. Technical Report 31, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany.

[Berral, 2010] Berral, M. I. (2010). Operaciones auxiliares de montaje de componentes informáticos. España: Editorial Paraninfo.

[Betancourt, 2005] Betancourt, G. A. (2005). Las máquinas de soporte vectorial (svms). Scientia et Technica, 1(27)

[Bishop, 2006] Bishop M., C. (2006). Pattern recognition and machine learning. Estados Unidos: Springer.

[Borroto, 2008] Borroto G., M., Villuendas R., Y., Medina P., M. A., Martínez L., J., Ruíz S., J. (2008). Selección y construcción de objetos para el mejoramiento de un clasificador supervisado: un análisis crítico. Reporte Técnico de Reconocimiento de Patrones. Habana: Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV).

[Braidot, 2008] Braidot, N. (2008). Neuromanagement: Cómo utilizar a pleno el cerebro en la conducción exitosa de las organizaciones. Argentina: Ediciones Granica S. A.

[Bunke, 2002] Bunke, Horst, Kandel, A. (2002) Hybrid Methods in Pattern Recognition. Singapore: World Scientific.

[Campbell, 2011] Campbell, C., y Ying, Y. (2011). Learning with support vector machines. Synthesis Lectures on Arti_cial Intelligence and Machine Learning, 5(1), 1-95.

[Campello, 2007] Campello, R. J. (2007). A fuzzy extension of the rand index and other related indexes for clustering and classification assessment. En Pattern Recognition Letters, vol. XXVIII no. 7, pp. 833-841.

[Cao, 2006] Cao, F., Tung, A. K., y Zhou, A. (2006). Scalable clustering using graphics processors. In Advances in Web-Age Information Management. Springer Berlin Heidelberg, pp. 372-384.

[Casillas, 2003] Casillas, A., González de L., M. T., Martínez, R. (2003). Algoritmo de Clustering On-Line Utilizando Metaheurísticas y Técnicas de Muestreo. Procesamiento de Lenguaje Natural 31, 57-65.

[Cerda, 2008] Cerda L., J., Villarroel del P., L. (2008). Evaluación de la concordancia inter-observador en investigación pediátrica: Coeficiente de Kappa. Revista chilena de pediatría 79(1): 54-58.

[Chaitanya, 2011] Chaitanya S., S., Rao, S. (2011). Comparison of purity and entropy of k-means clustering and fuzzy c means clustering. Indian Journal of Computer Science and Engineering (IJCSE) 2(3): 343-346.

[Chapelle, 2006] Chapelle O., Scholkopf B., Zien A. (2006). Semi-Supervised Learning. Estados Unidos: The MIT Press.

[Chiang, 2003] Chiang J., Hao E. (2003). Anew kernel-based fuzzy clustering approach: Support vector clustering with cell growing. IEEE Transactions on Fuzzy Systems, Vol. 11 (4): 518-527.

[Clavijo, 2006] Clavijo R., D., Bernal V., M. & Silva. J. F. (2006). Sistema Inteligente de Reconocimiento de Enfermedad Coronaria (Isquemia). Archivos de Medicina 6(12): 56-64.

[Constela, 2010] Constenla, G. R. (2010). Reconocimiento Óptico de Dígitos con Redes Neuronales. Tesis de licenciatura en Ingeniería en Informática. Universidad de Belgrano: Buenos Aires.

[Cortijo, 2001] Cortijo, B. F. J. (2001). Técnicas no supervisadas: Métodos de agrupamiento. Consultado en septiembre de 2012. Apuntes disponibles en la web de la asignatura [http://www-etsi2.ugr.es:8080/depar/ccia/RF0708/material.htm]

[Cristianni, 2000] Cristianini, N., y Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge university press.

[Dasarathy, 1991] Dasarathy, B.V.: Nearest Neighbor Norms: NN Pattern Clasification Techniques. IEEE Computer Society Press, Los Alamos, CA, (1991)

[Diehl, 2003] Diehl, C. P., y Cauwenberghs, G. (2003) SVM incremental learning, adaptation and optimization. In Neural Networks, 2003. Proceedings of the International Joint

Conference 4, pp. 2685-2690. IEEE.

[Díez, 2004] Díez, J. L., Navarro, J. L., Sala, A. (2004). Algoritmos de agrupamiento en la identificación de modelos borrosos. Revista Iberoamericana de Automática e Informática Industrial 1(2): 32-41.

[Dipti, 2004] Dipti S., Lakhmi C. J. (2010). Innovation in Multi-Agent System and Applications. Studies in Computacional Intelligence, 310(1). Estados Unidos: Springer.

[Dunne, 2007] Dunne, R.A. (2007). A statistical approach to neural networks for pattern recognition. New Jersey: John Wiley and Sons.

[Durán, 2007] Durán, R. L. (2007). El Gran Libro del PC interno: programación de sistemas hardware a fondo. Alfaomega – Marcombo

[Escolano, 2003] Escolano, F. (2003). Inteligencia artificial: modelos, técnicas y áreas de aplicación. Editorial Paraninfo.

[Espinosa, 2001] Espinosa, D. V. (2001). Evaluación de Sistemas de Reconocimiento Biométrico. Departamento de Electrónica y Automática. Escuela Universitaria Politécnica de Mataró.

[Ethem, 2004] Ethem, A. (2004). Introduction To Machine Learning. Estados Unidos: MIT Press.

[Everitt, 2001] Everitt B., Landau S., Leese M. (2001). Cluster Analysis. Arnold.

[Fernández, 1995] Fernández G., M. A., Ramos S., I. (1995). Vida Artificial. Colección Ciencia y Técnica 10. Universidad de Castilla- La Mancha.

[Fernández, 2001] Fernández L., M., Macilio N., A., Miró P., G., Domínguez G., R. (2001). Aplicación de la transformada Wavelet en la interpretación de cortes sísmicos de reflexión. Minería y Geología Revista de Ciencias de la Tierra 18(1): 61-67.

[Ferrer, 2005a] Ferrer T., F. J., Aguilar R., J. S. (2005a). Minería de Data Streams: Conceptos y Principales Técnicas. Jornadas Técnicas de Minería de Datos. Universidad de Castilla-La Mancha.

[Ferrer, 2005b] Ferrer T., F. J., Aguilar R., J. S. (2005b). Aprendizaje Incremental de Reglas en Data Streams. Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA. Sevilla:

Thomson. 261-270.

[Flóres, 2008] Flórez L., R., Fernández F., J. M. (2008). Las Redes Neuronales Artificiales: Fundamentos Teóricos y Aplicaciones Prácticas. Netbiblo.

[Flórez, 2008] Flórez L., R., Fernández F., J. M. (2008). Las Redes Neuronales Artificiales: Fundamentos Teóricos y Aplicaciones Prácticas. Netbiblo.

[Forestiero, 2013] Forestiero, A., Pizzuti, C., y Spezzano, G. (2013). A single pass algorithm for clustering evolving data streams based on swarm intelligence. Data Mining and Knowledge Discovery, 26(1), 1-26.

[Frank, 2010] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Giraud, 2000] Giraud-Carrier, C. (2000). A note on the utility of incremental learning. AI Communications, 13(4), 215-223.

[Guan, 2011] Guan, T., Yu, Y., y Xue, T. (2011). An online multiscale clustering algorithm for irregular data sets. In Future Computer Sciences and Application (ICFCSA), 2011. International Conference. IEEE, pp. 209-211.

[Guarnizo, 2008] Guarnizo L., C. (2008). Análisis de reducción de ruido en señales EEG orientado al reconocimiento de patrones. Revista Tecnológicas 21: 67-80.

[Hernández, 2004] Hernández O., J., Ramírez Q., M. J., Ferri R., C. (2004). Introducción a la Minería de Datos. Madrid: Person Educación.

[Hoi, 2013] Hoi, S. C., Jin, R., Zhao, P., y Yang, T. (2013). Online multiple kernel classication. Machine Learning, 90(2), 289-316.

[Hsu, 2003] Hsu, C. W., Chang, C. C., y Lin, C. J. (2003). A practical guide to support vector classification.

[Hubert, 1985] Hubert L., Arabie P. (1985). Comparing Partitions. Journal Classification 2, 193 -218.

[Ingaramo, 2007] Ingaramo, D. A., Errecalde, M. L., Rosso, P. (2007). Medidas internas y externas en el agrupamiento de resúmenes científicos de dominios reducidos. Procesamiento de Lenguaje Natural 39: 55-62.

[Irigoyen, 2011] Irigoyen M., M. (2011). Reconocimiento de patrones meteorológicos mediante técnicas neuronales y estadísticas. Tesis de licenciatura en Ingeniería en Informática. Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación.

[Jain, 2006] Jain, S., Lange, S., y Zilles, S. (2006). Towards a better understanding of incremental learning. In Algorithmic Learning Theory. Springer Berlin Heidelberg, pp. 169-183.

[Jiang, 2010] Jiang, P., y Singh, M. (2010). SPICi: a fast clustering algorithm for large biological networks. Bioinformatics, 26(8), 1105-1111.

[Juárez, 2003] Juárez Z., M. (2003). Incorporación de aprendizaje continuo a la Regla del Vecino más cercano. Tesis de Maestría en Ciencias de la Computación, Instituto Tecnológico de Toluca: México.

[Käll, 2007] Käll L., Canterbury J., D., Weston J., Noble W. S., MacCoss M. J. (2007). Semi-supervised learning for peptide identification from shotgun proteomics datasets. Estados Unidos: Nature Publishing Group.

[Khaleghi, 2012] Khaleghi, A., Ryabko, D., Mary, J., y Preux, P. (2012). Online clustering of processes. In International Conference on Arti_cial Intelligence and Statistics, pp. 601-609.

[Kidera, 2006] Kidera, T., Ozawa, S., y Abe, S. (2006). An incremental learning algorithm of ensemble classi_er systems. In Neural Networks, 2006. IJCNN'06. International Joint Conference. IEEE, pp. 3421-3427.

[Kifer, 2004] Kifer, D., Ben-David, S., y Gehrke, J. (2004). Detecting change in data streams. In Proceedings of the Thirtieth international conference on Very large data bases 30, pp. 180-191. VLDB Endowment.

[Krumke, 2006] Krumke, S. O. (2006). Online optimization: Competitive analysis and beyond. ZIB

[Kumar, 2007] Kumar, N., Joshi, R. S. (2007). Data Clustering Using Artificial Neural Networks. Proceedings of National Conference on Challenge & Opportunities in Information Technology (COIT).

[Kuncheva, 2004] Kuncheva L., I. (2004). Combining pattern clasifiers: Methos and Algorithms. Estados Unidos: Wiley-Interscience.

[Lewis, 2012] Lewis, J., Ackerman, M., and De Sa, V. (2012). Human Cluster Evaluation and Formal Quality Measures. Proc. 34th Annual Conference of the Cognitive Science Society.

[Lindhom, 2011] Lindhom, T., Yellin, F., Bracha, G., Buckley, A. (2011). The Java Virtual Machine Specification Java SE 7 Edition. Estados Unidos: Oracle America.

[Mairal, 2009] Mairal, J., Bach, F., Ponce, J., y Sapiro, G. (2009). Online dictionary learning for sparse coding. In Proceedings of the 26th Annual International Conference on Machine Learning. ACM, pp. 689-696.

[Maloof, 2003] Maloof, M. A. (2003). Incremental rule learning with partial instance memory for changing concepts. In Neural Networks, 2003. Proceedings of the International Joint Conference on (Vol. 4, pp. 2764-2769). IEEE.

[Marin, 2008] Marin M. R. (2008). Inteligencia Artificial: Métodos, técnicas y aplicaciones. España: Mc-Graw Hill.

[Marques de Sá, 2001] Marques de Sá P., Joaquim (2001). Pattern Recognition, Concepts, Methods and Applications. Alemania: Springer.

[Mashayekhi, 2012] Mashayekhi, H., Habibi, J., Voulgaris, S., y van Steen, M. (2012). GoSCAN: Decentralized scalable data clustering. Computing, 1-26.

[Maulik, 2000] Maulik U., Bandyopadhyay. (2000). Genetic algorithm-based clustering technique. Pattern Recognition 33: 1455-1465.

[Menchaca, 2000] Menchaca, M. R., García C. F. (2000). Arquitectura de la Máquina Virtual Java. Revista Digital Universitaria 1(2). Consultado en septiembre 2012. Disponible en: http://www.revista.unam.mx/vol.1/num2/art4/

[Mendaza, 2010] Mendaza, O. A. (2010). Estudio de un sistema de reconocimiento biométrico mediante firma manuscrita online basado en SVM usando Análisis Formal de Conceptos. IV Jornada de Reconocimiento Biométrico de Personas, S3: Reconocimiento de Firma y Escritura. España: Universidad de Zaragoza. Instituto de Investigación en Ingeniería de Aragón.

[Méndez, 2006] Méndez del R., Luis (2006). Más allá del Business Intelligence: 16 experiencias de éxito. España: Gestión 2000.

[Micheli, 2000] Micheli-Tzanakou, E. (2000). Supervised and Unsupervised Pattern Recognition: Feature extraction and computational

intelligence. Florida: CRC Press.

[Microsoft, 2012] Microsoft msdn (2012). Conjuntos de datos de entrenamiento y prueba (Analysis Services - Minería de datos). SQL Server, disponible en [http://msdn.microsoft.com/es-es/library/bb895173.aspx]

[Morris, 1994] Morris M., M. (1994). Arquitectura de computadoras 3ª edición. México: Pearson Educación.

[Murty, 2012] Murty, M. N., & Devi, V. S. (2012). Pattern recognition: An algorithmic approach. Springer.

[Niño, 2011] Niño, C. J. (2011). Sistemas operativos monopuesto. Editorial Editex.

[Ortega, 2010] Ortega L., Galiano I., Hurtado L., F., Sanchis E., Segarra E. (2010). Un método de aprendizaje semi-supervisado para la modelización semántica en comprensión del habla. Procesamiento de Lenguaje Natural, 45, 199-205.

[Pal, 2001] Pal. A., Pal, S. K. (2001). Pattern Recognition: From Classical to Modern Approaches. Singapore: World Scientific.

[Paraechivescu, 2011] Paraschivescu, M., Stefan, S., Bogdan, M. (2011). Verification of an algorithm (DWSR 2500C) for hail detection. Atmósfera 24(4): 417-433.

[Pavoine, 2009] Pavoine S., Vallet J., Dufour A. B., Gachet S., Daniel H. (2009). On the challenge of training various types of variables: application for improving the measurement of functional diversity. Oikos, 118, 391-402.

[Pérez, 2005] Pérez, A., Larrañaga P., Inza, I. (2005). Estimar, descomponer y comparar el error de mala clasificación. Evaluando y analizando el comportamiento de algoritmos de inducción de clasificadores. Universidad del Pais Vasco-Euskal Herriko Univertsitatea, Intelligent System Group.

[Ramírez, 2011] Ramírez Q., J. y Chacon M., M. I. (2011). Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década. Revista de ingeniería eléctrica y computación 9(1), 7-16.

[Ripley, 1996] Ripley, B. D. (1996). Pattern recognition and neural networks. Cambridge Uni. Press, Cambridge.

[Ripley, 2008] Ripley, B., D. (2008) Pattern Recognition and Neural Networks. Inglaterra: Cambridge University Press.

[Rullán, 2011] Rullán S., Gama C., LM, Galindo A., A., Olthoff A. (2011). Clasificación no supervisada de la cobertura de suelo de la región sierra de Tabasco mediante imágenes LANDSAT ETM+. Universidad y Ciencia, Tropico Húmedo 27(1): 33-41.

[Salcedo, 2007] Salcedo C., F. J. (2007) Modelos Ocultos de Markov: Del Reconocimiento de Voz a la Música. España: Lulu.com

[Sánchez, 2001] Sánchez A., Gema (2001). Un modelo sintáctico para la representación, segmentación y reconocimiento de símbolos texturados en documentos gráficos. Tesis Doctoral, Universitat Autónoma de Barcelona y la Université Henry Poincaré.

[Sanjurjo, 2009] Sanjurjo F., J. (2009). Sistema de ayuda a la decisión de inversión en bolsa. Tesis de licenciatura en Ingeniería en Informática (ICAI). Universidad Pontificia Comillas.

[Santibáñez, 2013] Santibáñez, M., Valdovinos, R. M., Rendon, E., Alejo, R., y Marcial-Romero, J. R. (2013). Optimizacion de Recurso para el Tratamiento de Grandes Volúmenes de Datos. En Research in Computing Science Avances en Inteligencia Articial 62, pp. 15-24.

[Santos, 2009] Santos, J. M., Embrechts, M. (2009), On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification, ICANN, Artificial Neural Networks, ICANN, 175-184.

[Schmitt, 2008] Schmitt J., Hollick M., Roos C., Steinmetz R. (2008). Adapting the User Context in Realtime: Tailoring Online Machine Learning Algorithms to Ambient Computing. Mobile Networks and Applications, 13 (6), 583-598.

[Schwaighofer, 2009] Schwaighofer, A., Candela, J. Q ., Borchert, T., Graepel, T. y Herbrich, R. (2009). Scalable clustering and keyword suggestion for online advertisements. En Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising, ACM, 2009, pp. 27-36.

[SIPU, 2012] Speech and Image Processing Unit (2012). Clustering datasets. [http://cs.joensuu.fi/sipu/datasets/] Finlandia: University of Eastern Finland, School of Computing.

[Sleator, 1985] Sleator DD, Tarjan RE. (1985). Amortized eficiency of list update and paging rules. Communications of the ACM 28,

202208.

[Smola, 2008] Smola, A. J., Vishwanathan, S. V. N. (2008). Introduction to Machine Learning. Reino Unido: Cambridge University Press.

[Speech, 2012] Speech and Image Processing Unit (2012). Clustering datasets. [http://cs.joensuu.fi/sipu/datasets/] Finlandia: University of Eastern Finland, School of Computing.

[Steinward, 2008] Steinwart, I., Christmann, A. (2008). Support Vector Machines. Springer.

[Sun, 2006] Sun, Microsystems (2006). Memory Management in the Java HotSpot Virtual Machine. Sun Microsystems Inc.

[Sung, 2000] Sung C.S., Jin H.W. (2000). A tabu-search-based heuristic for clustering. Pattern Recognition, 33, pp. 849-858

[Taher, 2008] Taher, N., Bahman B. F., Majid, N. (2008). An Efficient Hybrid Evolutionary Algorithm for Cluster Analysis. World Applied Science Journal 4(2): 300-307.

[Tantrum, 2004] Tantrum, J., Murua, A., y Stuetzle, W. (2004). Hierarchical model-based clustering of large datasets through fractionation and refractionation. En Information Systems, vol. XXIX no. 4, 2004, pp. 315-326.

[Theodoridis, 2006] Theodoridis, S., Koutroumbas K. (2006). Pattern Recognition (4ª ed.). Estados Unidos: Elseiver Academic Press.

[Tomek, 1976] Tomek, I. (1976). An experiment with the Edited Nearest-Neighbor rule. IEEE Transactions on Systems, Man and Cybernetics SMC 6, 448-452.

[Tripathy, 2011] Tripathy, B. K., Ghosh, A. (2011). SSDR: An Algorithm for Clustering Categorical Data Using Rough Set Theory. Advances in Applied Science Research 2(3): 314-326.

[Tu, 2009] Tu, L., y Chen, Y. (2009). Stream data clustering based on grid density and attraction. ACM Transactions on Knowledge Discovery from Data (TKDD), 3(3), 12.

[UCI, 2010] Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[Vadrevu, 2011] Vadrevu, S., Teo, C. H., Rajan, S., Punera, K., Dom, B., Smola, A. J. y Zheng,

Z. (2011). Scalable clustering of news search results. In Proceedings of the fourth ACM international conference on Web search and data mining. ACM, pp. 675-684.

[Valdovinos, 2006] Valdovinos R., R. M. (2006). Técnicas de Submuestreo, Toma de Decisiones y Análisis de Diversidad en Aprendizaje Supervisado con Sistemas Múltiples de Clasificación. Tesis doctoral. Universitat Jaume I, Departamento de Lenguajes y Sistemas Informáticos.

[Vázquez, 2008] Vázquez M., F. D. (2008). Algoritmos de Aprendizaje Continuo Mediante Selección de Prototipos para Clasificadores Basados en Distancias. Tesis doctoral. Universitat Jaume I, Departamento de Lenguajes y Sistemas Informáticos.

[Vélez, 2001] Vélez T., A. C. (2001). Un modelo genético para minería de datos. Tesis de maestría en Ingeniería de Sistemas. Universidad de Colombia, Facultad de Minas.

[Vendramin, 2009] Vendramin, L., Campello, R.J.G.B., Hruschka, E.R. (2009) On the comparison of relative clustering validity criteria. Sparks.

[Wagner, 2007] Wagner, S. y Wagner, D. (2007). Comparing clusterings: an overview, Universität Karlsruhe, Fakultät für Informatik, 2007.

[Xuan, 2009] Xuan V., N., Epps, J., Bailey, J. (2009). Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary? Proceedings of the 26th International Conference on Machine Learning. Montreal, Canada.

[Yang, 2002] Yang, Y., Webb G. (2002) A comparative study of discretization methods for naive-Bayes classifiers. In Proceedings of the Pacific Rim Knowledge Acquisition Workshop, pages 159–173.

[Yeung, 2001] Yeung, K. Y., Ruzzo, W. L. (2001), Details of the Adjusted Rand Index and Clustering Algorithms, Bioinformatics 17, 763–774.

[Yolis, 2003] Yolis, E. (2003). Algoritmos genéticos aplicados a la categorización automática de documentos. Tesis de licenciatura en Ingeniería Informática. Universidad de Buenos Aires, Facultad de Ingeniería.

[Younis, 2005] Younis, O., y Fahmy, S. (2005). Flowmate: Scalable *on-line* ow clustering. IEEE/ACM Transactions on Networking (TON), 13(2), 288-301.

[YourKit, 2003] YourKit, Blog (2003). JVM Memory Structure. Consultado en septiembre de 2012, disponible en [http://www.yourkit.com/docs/kb/sizes.jsp]

[Zubiaga, 2008] Zubiaga M., A. Fresno V. (2008). Aproximaciones a SVM semisupervisado multiclase para clasificación de páginas web. Memoria de Tesis de Master en Máster en Tecnologías del Lenguaje en la Web. Universidad Nacional de Educación a Distancia: España.

[Zubiaga, 2009] Zubiaga M., A., Fresno F., V., Martínez U., R. (2009). Comparativa de Aproximaciones a SVM Semisupervisado Multiclase para Clasificación de Páginas Web. Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural, SEPLN 42: 63-70.