COMPARACIÓN DE ALGORITMOS PARA CLUSTERING DE "STREAMS" DE SERIES DE TIEMPO

Por

Ana María Aparicio Carrasco

Tesis sometida en cumplimiento parcial de los requerimientos para el grado de

MAESTRÍA EN CIENCIAS

en

COMPUTACIÓN CIENTÍFICA

UNIVERSIDAD DE PUERTO RICO
RECINTO UNIVERSITARIO DE MAYAGÜEZ
DEPARTAMENTO DE CIENCIAS MATEMÁTICAS

Mayo, 2012

Aprobada por:

Alexander Urintsev, Ph.D Miembro, Comité Graduado	Fecha
Ana Carmen González, MS Miembro, Comité Graduado	Fecha
Edgar Acuña Fernández, Ph.D Presidente, Comité Graduado	Fecha
Ricky Valentín Rullán, Ph.D Representante de Estudios Graduados	Fecha
Omar Colón Reyes, Ph.D Director del Departmento	Fecha

Abstract of Thesis Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

COMPARISON OF ALGORITHMS FOR CLUSTERING TIME SERIES DATA STREAMS

By

Ana María Aparicio Carrasco

May 2012

Chair: Edgar Acuña Fernández
Major Department: Department of Mathematical Sciences

In recent years, technological advances have resulted in a huge increment in data production as in the evolution of methods that facilitated its collection. The data that arrive continuously and massively with infinite tendency are known as data streams. The source of these data is, for instance, sensors, bank personal transactions and automated measuring tools among others. The algorithms for processing this kind of data must provide rapid and real time responses, which implies that they must maintain a decision model all the time. The clustering of data streams by variables finds groups of variables (data streams) with similar behavior over time. In this work we compare two different approaches of algorithms for clustering of data streams by variables: ODAC, a divisive hierarchical algorithm and CORREL that operates over the Sliding Windows model and performs clustering by partitioning. Based on the experimental it is study concluded that ODAC outperforms CORREL because of its performance and independence from the distribution of data streams. However, it required a big amount of data points ("examples") to discover the inherent clustering structure.

Resumen de Tesis Presentado a la Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

COMPARACIÓN DE ALGORITMOS PARA CLUSTERING DE "STREAMS" DE SERIES DE TIEMPO

Por

Ana María Aparicio Carrasco

Mayo 2012

Consejero: Edgar Acuña Fernández
Departamento: Ciencias Matemáticas

En años recientes, los avances tecnológicos han dado lugar a un enorme incremento en la producción de datos y han facilitado su recolección. Los datos que llegan continuamente, de forma masiva y con tendencia infinita se conocen como *data streams*. Sus fuentes son por ejemplo sensores, transacciones bancarias y mediciónes automatizadas. Los algoritmos destinados a procesar estos datos deben brindar respuestas rápidas y en tiempo real, lo que implica mantener un modelo de decisión en todo momento. El clustering de *data streams* por variables busca encontrar grupos de *data streams* (variables) con un comportamiento similar a lo largo del tiempo. En el presente trabajo se comparan dos enfoques de algoritmos de clustering de *data streams* por variables: ODAC, un algoritmo jerárquico divisivo y CORREL que utiliza el modelo *Sliding Windows* y realiza el clustering por particionamiento. Basado en el estudio experimental se concluye que ODAC supera a CORREL, debido a su rendimiento e independencia de la distribución de las variables. Sin embargo, este requiere que el conjunto de datos posea una una gran cantidad de observaciones (ejemplos) para descubrir la estructura de clusters subyacente.

A mis padres María y Edgar.

AGRADECIMIENTOS

Agradezco primero a Dios que está por encima de todo y cuya presencia y protección han sido fundamentales en esta etapa y siempre.

Mi más sincero agradecimiento al Dr. Edgar Acuña por sus importantes aportes que muchas veces fueron la clave para continuar después de verme atascada en algún asunto, por su participación activa y todo el apoyo brindado junto a su paciencia e interés tanto en la culminación de este trabajo como durante mis estudios de maestría y aún mis proyectos personales.

Al profesor Ismael Pagán, no sólo por la confianza brindada en el trabajo que durante mucho tiempo ha sido mi principal sustento durante esta etapa académica sino también por su apoyo y flexibilidad para poder cumplir al mismo tiempo con mis proyectos académicos. Le agradezco además por su amistad y consideración.

A todos los profesores de la UPRM con quienes tomé clases y a los que de alguna manera aportaron en mi formación académica.

Al departamento de Ciencias Matemáticas por brindarme las facilidades y los recursos necesarios para poder realizar mis actividades durante el desarrollo de la tesis.

A mis padres María y Edgar quienes me impulsaron a tomar este paso, a ellos y a mis hermanas Patricia, Karen e Ivette les agradezco por la confianza puesta en mí y porque siempre me han apoyado de muchas maneras. A Karen quien ha estado más cerca y cuyo apoyo y consejos al iniciar esta maestría y luego durante el proceso han sido fundamentales. A Sofia por su cariño y a Ollantay por sus consejos y apoyo.

A Oscar Yupanqui por compartir conmigo sus conocimientos, su tiempo, por su invaluable apoyo y compañía durante esta etapa de mi vida académica y personal. Muchas gracias por confiar siempre en mí y por enriquecer mi vida.

A mis amigos, Jorge Carpio que a pesar de la distancia y las ocupaciones siempre se dió tiempo para darme un consejo y motivarme y hasta darme su acertada opinión a cerca de algunos aspectos de esta tesis, la cual aprecio y tomé en cuenta. Yaquelin Quispe, quien siempre se mantuvo cerca y en especial a Davis Chacón por sus consejos, motivación, críticas e inspiración, por impulsarme a buscar objetivos mayores y ver las cosas de una forma no convencional. Gracias por inspirarme con tu dedicación y esfuerzo, por tu cercanía, por tu importante apoyo y preciada amistad.

TABLA DE CONTENIDO

pagina

LISTA DE TABLAS

Tabla página

LISTA DE FIGURAS

Figura pagina

LISTA DE ABREVIATURAS

ODAC	Online Divisive Agglomerative Clustering (Clustering divisivo aglomerativo en línea).
CORREL	Clustering Multiple Data Streams Based on Correlation Analysis

LISTA DE SIMBOLOS

L	Tamaño de la ventana en CORREL
l	Tamaño de la ventana básica en CORREL
D	Índice de Dunn.
S	Índice Silueta.
Γ_D	Correlación de Hubert con matriz de distancia.

CAPÍTULO 1
Introducción

En años recientes, los avances tecnológicos han dado lugar a un enorme incremento en la producción de datos y a la evolución de métodos que han facilitado su recolección. Estos datos que vienen de sensores, transacciones personales, herramientas de medición automatizadas y otras fuentes, se conocen como *data streams*. A diferencia de los conjuntos de datos convencionales, en los conjuntos de *data streams* los datos llegan continuamente, de forma masiva y con tendencia infinita. En ese contexto la minería de datos se enfrenta a nuevos requerimientos. La demanda de respuestas rápidas y en cualquier momento implica que los algoritmos realicen un procesamiento de los datos en tiempo real, para lo cual deben mantener un modelo de decisión en todo momento. Asimismo, la capacidad de memoria se ve limitada debido a la llegada masiva de los datos por lo que estos algoritmos deben establecer mecanismos para procesar los datos sin la necesidad de almacenarlos o almacenando sólo una cantidad de información necesaria para obtener los resultados deseados. *Clustering* es una técnica ampliamente utilizada en la minería de datos ya sea para describir los datos o como una etapa previa de otras. Su objetivo no es generar un modelo para categorizar los datos futuros sino mas bien, modelar la estructura existente en el conjunto de datos.

En el *clustering* de *data streams* es necesario hacer la distinción entre el *clustering* de ejemplos y el de variables. El objetivo del clustering de ejemplos es agrupar las observaciones (data points) de las variables o series de tiempo. En este caso observaciones de las misma variable pueden pertenecer a diferentes clusters. Por

otro lado el objetivo del clustering de variables es agrupar los *data streams* en sí. Podemos visualizar el conjunto de *data streams* como una tabla en la que los *data streams* o variables son las columnas y sus observaciones correspondientes las filas de la tabla. El clustering de ejemplos descubrirá los grupos de filas mientras que el clustering de variables los grupos de columnas. En este trabajo nos enfocamos en el clustering de variables o series de tiempo [25]. Este problema viene siendo abordado en años recientes. A la fecha podemos encontrar en la literatura algunos algoritmos propuestos, sin embargo este aún es un problema emergente.

1.1 Motivación

Existe una gran cantidad de investigación en el tema de clustering de *data steams* de ejemplos [8, 28, 30]. Se han propuesto muchos algoritmos y desarrollado diferentes modelos para abordar el problema, asimismo podemos encontrar documentación y software como por ejemplo MOA (Massive Online Analysis)[5], un sistema de software libre para minería de *data streams* que entre otras funcionalidades provee métodos de clustering siendo todos de ejemplos. Sin embargo como se mencionó anteriormente el clustering de *data streams* por variables es un tema emergente. Aunque se han propuesto varios algoritmos para abordar este problema, muchos de ellos se desarrollaron de forma aislada comparando sus resultados sólo con algoritmos para *batch* clustering. Existen pocos trabajos que realizan una comparación con otros algoritmos para *data streams*.

La mayoría de los algoritmos propuestos [6, 14, 17, 31] siguen un modelo de dos fases, una fase en línea (*on-line*) que tiene el objetivo de obtener una sinópsis compuesta por ciertas estadísticas suficientes para calcular la medida de proximidad entre los *streams* y una fase fuera de línea en la que se realiza el clustering. Mayormente el algoritmo empleado para realizar el clustering es el K-Means [22] el cual realiza clustering por particionamiento. Los algoritmos utilizan el modelo *Sliding*

Window, es decir que tanto la fase en línea como la de fuera de línea se realizan sobre la ventana de tiempo más reciente.

Otro enfoque muy diferente es el propuesto por Pereira et al. en [26], quienes proponen un algoritmo jerárquico en línea para el clustering de series de tiempo cuyos valores llegan continuamente en el tiempo. Hasta el momento este es el único algoritmo jerárquico que conocemos para esta tarea.

Elegimos los algoritmos CORREL propuesto por Tu et al. en [17], el cual es el más reciente de los algoritmos que siguen el modelo de dos fases, para realizar un estudio comparativo con ODAC.

1.2 Objetivos

1.2.1 Objetivo general

Realizar un estudio comparativo de dos enfoques de algoritmos para el clustering de *data streams* por variables: el modelo jerárquico y el modelo de dos fases con *Sliding Window*.

1.2.2 Objetivos específicos

- Estudiar de forma detallada los algoritmos CORREL y ODAC.
- Realizar una implementación propia de los algoritmos.
- Realizar un estudio comparativo de los algoritmos CORREL y ODAC en términos de eficacia, eficiencia(tiempo de ejecución) y calidad de los resultados.

1.3 Organización de la tesis

En el capítulo 2 se presenta el concepto de Minería de *data streams*, asimismo se realiza una revisión del Clustering clásico, los paradigmas jerárquico y de particionamiento, medidas de proximidad y clustering de *data streams*. También se presentan los requerimientos de los algoritmos de *data streams* por variables, los modelos de datos en el entorno de *data streams*, una breve descripción de los algoritmos de clustering por variables encontrados en la literatura y por último las medidas de validación destinadas a la evaluación de la calidad de los clusters. En

el capítulo 3 se presentan de forma detallada los algoritmos ODAC y CORREL. El capítulo 4 presenta la evaluación experimental comenzando por mencionar los detalles de la implementación de los algoritmos, seguido de la metodología, la evaluación de los resultados obtenidos sobre conjuntos de datos artificiales y reales y por último la discusión de los resultados. En el capítulo 5 se presentan las conclusiones y los trabajos futuros.

CAPÍTULO 2
Revisión de Literatura

2.1 Introducción

Este capítulo contiene los conceptos fundamentales para familiarizarnos con el tema de *data streams* en general y en particular con el clustering de *data streams*. Comenzamos por presentar la Minería de Datos destinada a tratar el problema de *data streams* y caracterizar los *data streams* en sí. Luego, presentamos el clustering de *data streams*. Comenzamos hablando del clustering clásico. Después se presentan las características específicas del clustering de *data streams* debido a las cuales los algoritmos de clustering destinados a trabajar con este tipo de datos deben cumplir con los requerimientos que se presentan en esta parte. Puesto que este trabajo está enfocado en tratar el clustering de *streams* de series de tiempo, es decir, series de observaciones numéricas y continuas, se presentan medidas de proximidad para variables continuas que pueden ser utilizadas en esta tarea. Finalmente, se presentan los paradigmas de clustering que son fundamentales en este estudio, el clustering Jerárquico y el de Particionamiento.

2.2 Minería de "Data Streams"

Tradicionalmente las técnicas de minería de datos han estado enfocadas en el tratamiento de conjuntos de datos de tamaño fijo lo que se denomina en inglés como *batch learning*. En este esquema el procesamiento comienza por almacenar los datos en una base de datos estable y que no se actualice con frecuencia, los algoritmos procesan los datos realizando por lo menos un pasada sobre el conjunto de datos completo y en muchos casos lo hacen más de una vez. Por ejemplo, el ampliamente

conocido algoritmo de clustering K-Means realiza varias pasadas sobre el conjunto de datos hasta obtener convergencia. La razón de esta práctica es que se asume que los datos obedecen a alguna distribución de probabilidad estacionaria. Asimismo, el tiempo de ejecución de los algoritmos está dado en función de la dimensión del conjunto de datos, mientras más grande es, mayor será el tiempo que tomará obtener una respuesta. En muchos casos la respuesta está en el orden de horas o días.

Actualmente, el proceso de recolección, transmisión y almacenamiento de los datos ha cambiado radicalmente. Hoy en día contamos con fuentes de datos generando datos continuamente por ejemplo satélites, web logs, transacciones con tarjeta de crédito y todo tipo de sensores como sensores de temperatura del ambiente, de niveles de agua en ríos, movimientos de la tierra, etc. Estas fuentes de datos vienen generando y almacenando o transmitiendo los datos a través de redes de varios tipos. Los datos generados por estas fuentes se denominan *data streams*.

A diferencia de los conjuntos de datos convencionales, en los conjuntos de *data streams* los datos llegan continuamente en el tiempo por lo que sólo pueden ser accesados de forma secuencial, no es posible el acceso aleatorio. Mas aún, un *data stream* es potencialmente infinito por lo que la capacidad de memoria es limitada con relación al tamaño del *data stream*. Una vez que un elemento es procesado debe ser descartado o archivado. Aún cuando los datos sean archivados, un nuevo acceso sería muy costoso debido a la gran cantidad de datos acumulada. Por lo que los algoritmos deben almacenar resúmenes de los datos pasados dejando espacio en memoria, suficiente para el procesamiento de nuevos datos[12]. Adicionalmente, existe la necesidad de procesar los datos en tiempo real, no sólo por la disponibilidad de una cantidad de memoria limitada sino también por que se desean resultados oportunos a las peticiones del usuario. De esta manera la minería de *data streams* proporciona resultados rápidos pero aproximados a diferencia de la tradicional que arroja resultados precisos pero con un alto costo en tiempo y trabajo computacional.

2.3 Clustering clásico

El análisis de clusters, *clustering* es el estudio formal de los algoritmos y métodos de agrupamiento o clasificación de objetos. Sin embargo, la clasificación que estudia no está basada en ningún modelo de categorias o etiquetado previo. Por este motivo también se le conoce como *clasificación no supervisada*. El objetivo del *clustering* es encontrar una organización conveniente y válida de los datos [3]. Es decir que su objetivo no es generar un modelo para categorizar los datos futuros sino mas bien, modelar la estructura existente en el conjunto de datos. El proceso básico de los algoritmos de clustering consiste en dividir un grupo de objetos en subgrupos denominados *clusters* en base a alguna medida de similaridad. De tal manera que la similaridad entre objetos que pertenezcan a un mismo *cluster* sea mayor que la similaridad entre objetos que pertenezcan a distintos *clusters*. De esta manera un *cluster* está definido en términos de homogeneidad interna y separación externa[1]. En este trabajo nos enfocamos en el clustering determinístico (hard clustering) en el que cada objeto estará exclusivamente asociado con un *cluster*, por lo tanto la intersección entre dos *clusters* será el conjunto vacío mientras que la unión de todos los *clusters* será el conjunto de objetos completo. Existe también la posibilidad de que un objeto pertenezca a todos los *clusters* para lo cual se define un grado de membresía del objeto a cada uno de los *clusters*. Esto se conoce como clustering difuso (*fuzzy* en inglés).

2.3.1 Paradigmas de Clustering

Clustering Jerárquico

El clustering jerárquico puede ser tanto aglomerativo como divisivo, dependiendo de si se parte de considerar a cada uno de los objetos del conjunto de datos como un cluster o "singleton" o a todos los objetos como un solo cluster respectivamente. Ambos métodos organizan los datos en una estructura jerárquica basada en medidas de proximidad, usualmente esta estructura se puede representar gráficamente

como un dendograma, como se muestra en la figura 2–1. En un dendograma, la raíz representa el conjunto total de objetos y cada nodo hoja representa a cada uno de los objetos, los nodos intermedios representan la proximidad entre los objetos. El resultado del clustering puede ser obtenido cortando el dendograma en diferentes niveles. Esta estructura es una representación muy informativa que muestra visualmente la estructura de los clusters potenciales especialmente cuando los datos poseen una estructura jerárquica real como es el caso de las especies de plantas u otros organismos biológicos o en otros campos como en medicina o arqueología. A continuación se describirá brevemente cada uno de los enfoques

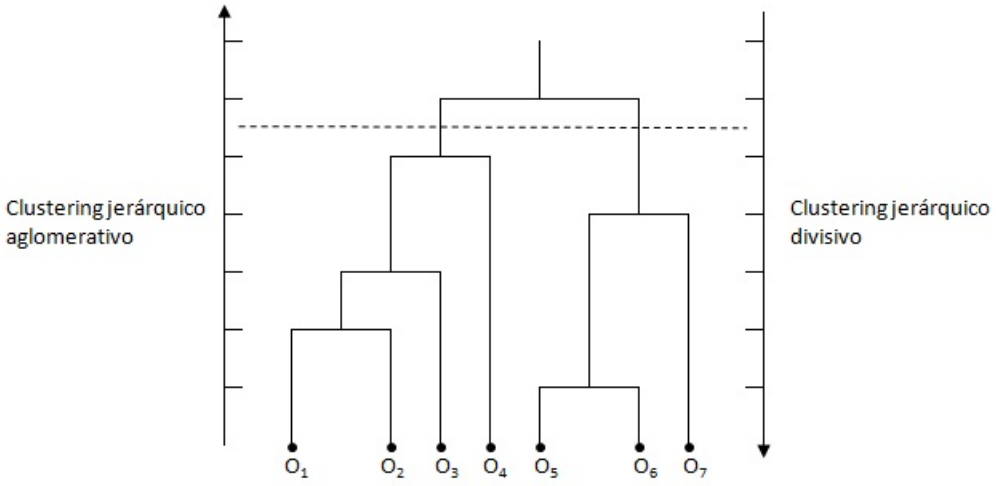

Figura 2–1: Ejemplo de dendograma del clustering jerárquico. El clustering jerárquico divisivo y el aglomerativo operan en direcciones opuestas. La línea punteada muestra un corte con el cual obtendríamos dos clusters. Fuente: R. Xi y D. Wunsch, 2009.

Clustering jerárquico aglomerativo

En el clustering jerárquico aglomerativo se parte de considerar a cada uno de los objetos del conjunto de datos como un cluster *singleton* (de un solo elemento), es decir que se comienza con N clusters donde N es el número de objetos total del conjunto de datos. El proceso en adelante consiste en agrupar sucesivamente los objetos en clusters de acuerdo a una matriz de proximidad. Eventualmente

todos los objetos serán agrupados en un sólo cluster.El proceso general del clustering aglomerativo es resumido a continuación [1]:

1. Comenzar con N clusters de un elemento (singleton clusters). Calcular la matriz de proximidad (usualmente basada en una función de distancia) para los N clusters.
2. En la matriz de proximidad, buscar la distancia mínima $D(C_i, C_j) = min_{1 \leq m \neq l \leq N} D(C_m, C_l)$, donde D es la función de distancia. Combinar los clusters C_i y C_j para formar un nuevo cluster C_{ij}.
3. Actualizar la matriz de proximidad calculando las distancias entre el cluster C_{ij} y los otros clusters.
4. Repetir los pasos 2 y 3 hasta que sólo quede un cluster.

Existen diferentes criterios de distancia para agrupar un par de clusters. Entre los principales están:

Single linkage, este método indica que la distancia entre un par de clusters está determinada por los dos objetos más cercanos pertenecientes a cada uno de ellos.

Complete linkage, contrariamente al anterior, este el método indica que la distancia entre dos clusters estará determinada por la mayor distancia entre dos objetos, uno de cada cluster. Este es efectivo en encontrar cluster pequeños y compactos.

Centroid linkage, también conocido como UPGMC (unweighted pair group method centroid), dos clusters serán agrupados en base a la distancia entre sus centros.

Clustering jerárquico divisivo

Comienza asignando todos los objetos del conjunto de datos a un solo cluster, el procedimiento continúa realizando divisiones sucesivas hasta obtener N clusters, es decir, hasta que cada uno de los objetos sea un cluster *singleton*. Al principio, para un conjunto de datos con N objetos, habrán $2^{N-1} - 1$ posibilidades de división que el algoritmo podría considerar, lo cual sería bastante caro incluso para conjuntos de

datos pequeños. Por este motivo los algoritmos utilizan otros criterios de división. Por ejemplo, el algoritmo DIANA ampliamente conocido, se basa en criterios de diámetro para realizar la división. El cluster a dividir será el de mayor diámetro en cada paso. Elige el objeto más alejado de todos los demás, es decir aquel con la mayor dismilaridad promedio, para formar un nuevo cluster. Posteriormente los demás objetos son asignados a este nuevo cluster si están más cercanos de este que de su cluster original. Recientemente se han propuesto nuevos algoritmos que buscan superar las deseventajas del enfoque jerárquico como son el gran costo computacional y sensitividad frente a los valores anómalos. Este es el caso de BIRCH [30], CURE [28], DBSCAN [21] entre otros, los cuales utilizan técnicas de preprocesamiento de los datos con estructuras de datos más sofisticadas y técnicas especializadas para lidiar con valores anómalos.

Clustering por particionamiento

A diferencia del clustering jerárquico que forma varios niveles de clusters, el clustering por particionamiento divide los objetos del conjunto de datos en k clusters, donde k es un parámetro dado de antemano por el usuario. Normalmente este proceso está sujeto a la optimización de una función objetivo. Dos clases de algoritmos pertenecen a este paradigma, estos son los modelos *generativo* y *reconstructivo* [27]. El modelo *generativo* o basado en probabilidad asume que los objetos del conjunto de datos pertenecen a k distribuciones de probabilidad no conocidas. Los algoritmos Expectation - Maximization [4] y C-Means Fuzzy [7] son ejemplos de este modelo. Por otro lado, el modelo *reconstructivo* tiene el objetivo de minimizar una función objetivo. Muchos algoritmos ampliamente conocidos siguen este modelo, por ejemplo K-Means, K-Medoids, PAM. La objetivo del clustering es obtener particiones compactas y bien separadas. En el clustering por particionamiento utilizando el modelo reconstructivo, la compactibilidad y la separación son evaluadas

por la función objetivo. Una de las más utilizadas es la suma de los errores al cuadrado. La cual está dada por la siguiente expresión:

$$EC = \sum_{i=1}^{k} \sum_{x_j \in C_i} |x_j^i - m_i|^2 \tag{2.1}$$

Donde x_j^i y m_i son el j-ésimo elemento y el centroide del cluster C_i respectivamente.

La partición que minimice EC será considerada la óptima. Este criterio es apropiado para clusters compactos y bien separados, sin embargo es sensible a la existencia de valores anómalos (*outliers*).

Uno de los algoritmos más ampliamente conocidos es el algoritmo *K-Means* [22], es una solución heurística al problema de clustering basado en la presunción de que los ejemplos son producidos por k distribuciones Gaussianas esféricas [27]. El clásico *K-Means* minimiza el criterio de la suma de los errores al cuadrado dado en la ecuación 2.1, definiendo el centroide del cluster como el vector de medias de los objetos que pertenecen al cluster. El algoritmo sigue el paradigma *hill climbing* o ascenso a colina. El procedimiento básico sigue los pasos que se mencionan a continuación:

1. Elegir k objetos aleatoriamente como centroides iniciales de los clusters. Clásicamente K-Means define el centroide de un cluster como el vector de medias de los elementos que lo conforman.
2. Asignar los N objetos al cluster con el centroide más cercano.
3. Determinar el nuevo centroide para cada cluster.
4. Repetir los pasos 2 y 3 hasta que no hayan más cambios en la estructura de clusters.

La complejidad del algoritmo es $O(TkN)$, donde T es el número de iteraciones, como T y k usualmente son mucho menores que N, la complejidad de K-Means es aproximadamente lineal. Adicionalmente su implememtación es sencilla y trabaja

bien en problemas prácticos, en conjuntos de datos que tienen una estructura de clusters compactos y de forma hiperesférica. Sin embargo sufre de algunas debilidades, la elección de diferentes centroides iniciales puede producir diferentes estructuras de clusters ya que su convergencia no es hacia el óptimo global sino hacia un óptimo local, asimismo K-Means es sensible a ruido y *outliers* ya que el cálculo de la media considera a todos los objetos incluso a los *outliers*.

El K-Means está limitado a variables numéricas debido al cálculo de la media, una variación del K-Means, es el K-Medoids que toma como centroide al objeto denominado *medoide*. El medoide está definido como el objeto con el menor promedio de disimilaridad hacia todos los demás objetos en el cluster.

2.4 Medidas de proximidad

Dado que el propósito del clustering es formar grupos que contengan objetos con alta similaridad entre sí y al mismo tiempo objetos en distintos grupos tengan baja similaridad, es necesario definir cómo se realizará la determinación de la similaridad o disimilaridad entre los objetos.

El término proximidad es una generalización para referirnos tanto a la similaridad como a la disimilaridad. Sean $\mathbf{x}_i$ y $\mathbf{x}_j$ dos objetos de dimensión d. Una disimilaridad o función de distancia $D(\mathbf{x}_i, \mathbf{x}_j)$ definida sobre ellos satisface las siguientes condiciones[1]:

1. Simetría

$$D(\mathbf{x}_i, \mathbf{x}_j) = D(\mathbf{x}_j, \mathbf{x}_i) \tag{2.2}$$

2. Positividad

$$D(\mathbf{x}_i\ \mathbf{x}_j) \geq 0 \quad \text{para todo } \mathbf{x}_i\ \mathbf{x}_j \tag{2.3}$$

La disimilaridad D será una métrica si además cumple con la siguientes condiciones:

3. Desigualdad triangular

$$D(\mathbf{x}_i, \mathbf{x}_j) \leq D(\mathbf{x}_i, \mathbf{x}_k) + D(\mathbf{x}_k, \mathbf{x}_j) \quad \text{para todo } \mathbf{x}_i, \mathbf{x}_j \text{ y } \mathbf{x}_k \tag{2.4}$$

4. Reflexibidad

$$D(\mathbf{x}_i, \mathbf{x}_j) = 0 \quad \text{si y solo si } \mathbf{x}_i = \mathbf{x}j \tag{2.5}$$

Si sólo cumple con la primera se denominará semi-métrica.

Igualmente, una similaridad $S(\mathbf{x}_i, \mathbf{x}_j)$ es una función que satisface las siguientes condiciones:

1. Positividad

$$0 \leq S(\mathbf{x}_i, \mathbf{x}_j) \leq S(\mathbf{x}_i, \mathbf{x}_i) = 1, \quad \text{para todo } \mathbf{x}_i, \mathbf{x}_j \tag{2.6}$$

2. Simetría,

$$S(\mathbf{x}_i, \mathbf{x}_j) = S(\mathbf{x}_j, \mathbf{x}_i), \quad \text{para todo } \mathbf{x}_i, \mathbf{x}_j \tag{2.7}$$

Una métrica de similaridad es una similaridad que satisface las siguientes condiciones:

3.

$$S(\mathbf{x}_i, \mathbf{x}_j) S(\mathbf{x}_j, \mathbf{x}_k) \leq [S(\mathbf{x}_i, \mathbf{x}_j) + S(\mathbf{x}_j, \mathbf{x}_k)] S(\mathbf{x}_i, \mathbf{x}_k), \quad \text{para todo } \mathbf{x}_i, \mathbf{x}_j \text{ y } \mathbf{x}_k \tag{2.8}$$

4.

$$S(\mathbf{x}_i, \mathbf{x}_j) = 1 \quad \text{si y solo si } \mathbf{x}_i = \mathbf{x}_j \tag{2.9}$$

Nuestro enfoque son los *streams* de series de tiempo, las cuales son variables continuas. Esto significa que los datos individuales son números que provienen de

algún tipo de medición. Por este motivo presentamos a continuación medidas de proximidad aplicables a este tipo de variables.

En clustering, la distancia Euclidiana o norma $\mathbf{L}_2$ es la más comunmente utilizada, está representada por la siguiente fórmula:

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\sum_{l=1}^{d} \left|x_{il} - x_{jl}\right|^{\frac{1}{2}}\right)^2, \tag{2.10}$$

donde $\mathbf{x}_i$ y $\mathbf{x}_j$ son dos objetos de dimensión d.

No es difícil observar que la distancia Euclidiana es una métrica ya que cumple con las ecuaciones 2.2- 2.5. De todas las medidas de proximidad, siempre se preferirán las *métricas* [27]. Investigaciones adicionales muestran que la distancia Euclidiana tiende a formar clusters de forma hiperesférica [1]. Asimismo, los clusters formados con la distancia Euclidiana son invariantes a las traslaciones y rotaciones en el espacio de las variables. Sin embargo, si las variables son medidas con unidades de medida diferentes, las variables medidas con valores altos tendrán una contribución mucho mayor que las variables medidas con valores bajos. Una forma de lidiar con el problema mencionado es normalizar los datos para hacer que cada variable tenga una contribución igual utilizando métodos de normalización tales como el Z-Score o Min-Max.

La distancia Euclidiana puede ser generalizada como un caso especial de la métrica de Minkowski o norma $\mathbf{L}_p$, definida como:

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\sum_{l=1}^{d} \left|x_{il} - x_{jl}\right|^{\frac{1}{p}}\right)^p \tag{2.11}$$

Haciendo $p = 1$ y $p \to \infty$, obtenemos dos casos especiales:

1. Distancia Manhattan o city-block

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sum_{l=1}^{d} \left|x_{il} - x_{jl}\right| \tag{2.12}$$

2. Distancia o norma $\mathbf{L}_\infty$

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \max_{1 \leq l \leq d} \left| x_{il} - x_{jl} \right| \tag{2.13}$$

Otra distancia que también se usa en análisis Cluster es la distancia de Mahalanobis, la cual también es una métrica y está definida de la siguiente manera:

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(\mathbf{x}_i - \mathbf{x}_j\right)^T \mathbf{S}^{-1} \left(\mathbf{x}_i - \mathbf{x}_j\right) \tag{2.14}$$

donde $\mathbf{S}$ es la matriz de covarianza definida como:

$$\mathbf{S} = E\left[\left(\mathbf{x} - \mu\right)\left(\mathbf{x} - \mu\right)^T\right] \tag{2.15}$$

donde μ es el vector de medias y $E\left[.\right]$ calcula el valor esperado de la variable aleatoria. La distancia de Mahalanobis tiende a formar clusters de forma hiperelipsoidal, los cuales son invariantes a cualquier transformación lineal no singular. Sin embargo, el cálculo de la inversa de $\mathbf{S}$ puede ser muy costoso para conjuntos de datos de grandes dimensiones [1]. Esta distancia se diferencia de la distancia Euclidiana en que toma en cuenta la correlación entre las variables, sin embargo cuando no existe correlación entre las variables, la matriz de covarianza es la identidad y por lo tanto es equivalente a la distancia Euclidiana.

La distancia también se puede derivar de una medida de similaridad como el coeficiente de correlación. En clustering de *data streams* el coeficiente de correlación de Pearson se ha venido utilizando ampliamente debido a la facilidad de cómputo de forma incremental. Está definido por la siguiente ecuación:

$$corr\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{\sum_{l=1}^{n} x_{il} x_{jl} - n\overline{\mathbf{x}_i}\ \overline{\mathbf{x}_j}}{\sqrt{\sum_{l=1}^{n} x_{il}^2 - n\overline{\mathbf{x}_i}^2}\sqrt{\sum_{i=1}^{n} x_{jl}^2 - n\overline{\mathbf{x}_j}^2}} \tag{2.16}$$

donde $\overline{\mathbf{x}}_i = \frac{1}{n}\sum_{l=1}^{n} x_i$, n es la longitud de un segmento del *stream* en el momento actual o el tamaño de la ventana de tiempo. El coeficiente de correlación está en el

rango de $[-1, 1]$, donde 1 indica una alta correlación positiva y -1 una alta correlación negativa.

Existen varias posibilidades para definir una medida de disimilaridad o función de distancia a partir del coeficiente de correlación de Pearson.

Algunas posibles funciones que caen en el rango de $[0, 1]$ son las siguientes:

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(1 - corr\left(\mathbf{x}_i, \mathbf{x}_j\right)\right) \tag{2.17}$$

Esta medida es una función de distancia ya que cumple con las propiedades de simetría y positividad sin embargo no es una métrica. La siguiente es una versión modificada que si satisface las propiedades para ser una métrica.

$$D\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sqrt{\frac{1 - corr\left(\mathbf{x}_i, \mathbf{x}_j\right)}{2}} \tag{2.18}$$

denominada RNOMC por sus siglas en inglés (*rooted normalized one-minus-correlation*)

Cuando el coeficiente de correlación se utiliza para medir similaridad, es necesario utilizar valor absoluto ya que en este caso los valores de la correlación con diferentes signos pero con el mismo valor absoluto tienen igual significado [3].

2.5 Clustering de "Data Streams"

En el clustering de *data streams*, es necesario hacer distinción entre dos tipos de clustering dados por la forma en cómo se realiza el agrupamiento de los datos. El clustering de ejemplos y el de variables. En el clustering clásico esta distinción no es relevante ya que la matriz de datos puede ser fácilmente transpuesta y utilizar los mismos algoritmos para ambos propósitos. En el caso de los *data streams* trasponer la matriz no es posible ya que en ningún momento se tiene acceso al conjunto de datos completo. Por lo tanto no podremos utilizar los mismos algoritmos para clustering de ejemplos y para clustering de variables.

Clustering de "Data Streams" por ejemplos

El clustering de ejemplos es la tarea más común en aprendizaje no supervisado, consiste en realizar una partición del conjunto de ejemplos correspondientes a m variables o fuentes de datos (*data streams*). El agrupamiento está basado en alguna medida de proximidad entre los ejemplos y es independiente de las fuentes de datos a las cuales estos pertenezcan. Existe una gran cantidad de trabajos de investigación que proponen algoritmos de clustering de data streams de ejemplos. Algunos de los más citados son CluStream [8], HPStream [9], E-Stream [16], COBWEB [11], DenStream [10].

Clustering de "Data Streams" por variables

En este caso, cada *data stream* es tomado como una unidad, por lo tanto todos los puntos que pertenezcan al mismo *data stream* pertenecerán al mismo cluster. El objetivo del *clustering* de data streams por variables es encontrar grupos de variables (*data streams*) con un comportamiento similar a lo largo del tiempo. Formalmente, sea $X = \langle x_1, x_2, \ldots, x_n \rangle$ el conjunto completo de streams y $X^t = \langle x_1^t, x_2^t, \ldots, x_n^t \rangle$ el ejemplo conteniendo las observaciones de todos los streams x_i en el tiempo t [12]. El objetivo de un sistema de clustering de *data streams* es mantener una partición P de los streams y hacerla disponible en cualquier instante de tiempo t.

2.6 Requerimientos de los algoritmos de Clustering de "Data Streams" por variables

Algunos requerimientos que deben satisfacer los sistemas de clustering de *data streams* son los mencionados en [25].

2.6.1 Tiempo y memoria constantes

Dada la tendencia infinita de los *data streams* es necesario que el sistema realice un procesamiento de cada registro en tiempo constante ya que de otro modo sería imposible hacer frente a la continua llegada de datos. Por la misma razón, la cantidad de memoria que el sistema utilice no puede estar dada en función del número de

ejemplos. Con el objetivo de satisfacer estas restricciones todos los cálculos que realice el sistema deben ser concebidos de forma incremental, de tal manera que la información sea actualizada continuamente utilizando una memoria de tamaño constante y en un tiempo mínimo.

2.6.2 Representación compacta en todo momento

Ya que es imposible almacenar en memoria todos los datos que han ido llegado a lo largo del tiempo, surge la necesidad de definir una representación compacta de los datos. Esta representación debe contener las estadísticas suficientes para realizar el cálculo de la medida de proximidad entre los *data streams* y por suspuesto, debe poder ser actualizada de manera incremental a la llegada de nuevos datos. De esta manera se necesitará realizar una sola pasada sobre los datos. Esto es también un requerimiento para estos sistemas ya que como se mencionó anteriormente, se dispone de un tiempo limitado haciendo imposible volver a visitar los ejemplos pasados aún cuando estos hayan sido archivado en dispositivos de almacenamiento secundarios.

2.6.3 Detección de cambios en la estructura (drift detection)

Es de esperarse que la distribución de los ejemplos correspondientes a los *data streams* cambie a lo largo del tiempo, dado el dinamismo inherente a este tipo de datos. Un cambio estructural ocurre en el instante en que la estructura de clustering obtenida con los datos previos ya no representa las actuales relaciones de proximidad entre los *data streams*. Los algoritmos destinados a realizar clustering de *data streams* deberán contemplar métodos para detectar y adaptar el clustering a estos cambios con el propósito de mantener el modelo actualizado en todo momento.

2.6.4 Incorporación de nuevos streams relevantes

Hasta el momento todos lo sistemas o algoritmos diseñados para operar sobre *data streams* consideran conjuntos de datos tienen de ancho fijo, es decir que el número de *data streams* o variables es fijo y que solo el número de ejemplos se incrementa con el tiempo. Sin embargo, en aplicaciones reales existe la posibilidad

de que el número de variables también se incremente con el tiempo. Por ejemplo estaciones de en el caso de un sistema de sensores localizados en... (Ejemplo). Por lo tanto el sistema debería se capaz de incorporar nuevas variables en cualquier momento del proceso de clustering.

2.6.5 Disponibilidad de resultados en todo momento

Ya que el procesamiento de los *data streams* es continuo, no existe realmente un momento en el que se pueda decir que el proceso terminó y que es hora de dar un resultados. Por el contrario, el resultado del clustering será consultado en cualquier momento y el sistema deberá ser capaz de dar una respuesta actualizada al momento en que se haga la consulta.

2.7 Modelos de Datos en el entorno de "Data Streams"

2.7.1 El modelo "Landmark"

En este modelo las estadísticas son calculadas sobre los datos entre un punto específico en el tiempo llamado *landmark* y el presente. Considera a todos los datos con igual importancia por lo que no elimina los datos antiguos. Su ventana de tiempo es de tamaño variable y se mantiene creciendo a medida que el tiempo avanza.

2.7.2 El modelo "Sliding Window"

El modelo *Sliding Window* se basa en la suposición de que los datos recientes son más importantes que los datos pasados y que por lo tanto podemos concentrarnos en una ventana de tiempo que contenga un segmento final válido del *data stream*. Los límites del segmento de datos contenido en la ventana podrán estar dados por un intervalo de tiempo, el inicio de un evento, un número fijo de elementos. Se considera que la ventana se mueve sobre el stream a medida que los datos van llegando de tal manera que sólo los datos contenidos en la ventana son relevantes. La forma más simple es una ventana de tiempo de tamaño fijo w, su funcionamiento sigue un modelo "*first in - first out*", cuando un nuevo elemento j llega, si ya existen w elementos en la ventana, entonces el nuevo elemento se inserta

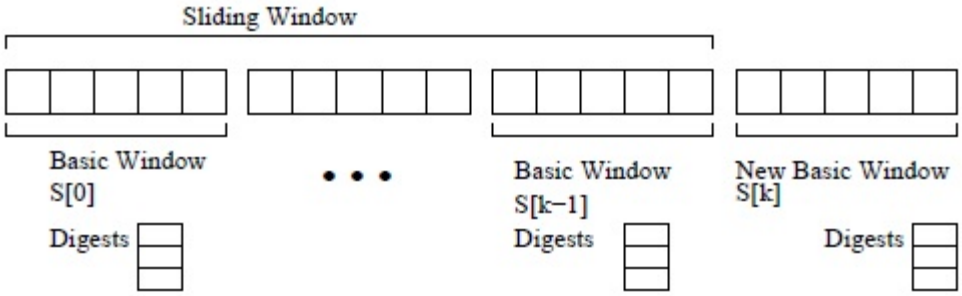

Figura 2–2: Sliding Windows con Ventanas Básicas. Fuente: Zhu y Shasha en [2]

y se descarta el más antiguo $j-w$. Los algoritmos que utilizan *Sliding Windows* para procesar los streams, realizan el cálculo de de las estadísticas sobre la ventana, por lo que los elementos contenidos en la ventana siempre están almacenados en memoria. Con el objetivo de facilitar una eliminación eficiente de los datos pasados y la incorporación de los nuevos, Zhu y Shasha, proponen en [2] la técnica de las "Ventanas Básicas". La cual consiste en dividir la ventana en ventanas más pequeñas de tamaño v, llamadas"Ventanas Básicas". La actualización de los elementos en la ventana se realizará utilizando las "Ventanas Básicas" de tal manera que cuando lleguen v elementos estos sean incorporados a la ventana mientras que v elementos son descartados. La figura 2–2 esquematiza esta técnica.

2.7.3 El modelo "Damped Window"

En el análisis de streams de datos, frecuentemente se asume que los datos más recientes tienen una relevancia y pertinencia mayor [20]. En el modelo *Damped Window* se establece una mayor importancia a los datos más recientes para esto se asigna un peso a los datos el cual decrece exponencialmente en el tiempo mediante una función de atenuación [10].

2.8 Algoritmos para clustering de "Data Streams" por Variables

Yang en [32](2003) presenta sin entrar en detalles un método incremental de clustering de streams que emplea una medida de distancia ponderada. Un valor actual de un stream es más importante que uno anterior por lo tanto este valor actual

deberá tener mayor peso en la función de distancia. Utiliza un factor de desvanecimiento para controlar la influencia de los valores de los streams en el futuro. El tiempo de vida de un cluster es modelado a través de una serie de transiciones entre tres estados: semilla potencial (potencial seed), cluster embrión (embryo cluster) y cluster natural (nature cluster) dependiendo de su grado y madurez.

Beringer et al. en [14] (2006) desarrollan una versión en línea del algoritmo K-Means. Los autores resumen los streams mediante la Transformada Discreta de Fourier conocida como DFT (Discrete Fourier Transform) por sus siglas en inglés, y adoptan la distancia euclidiana ponderada entre dos streams como medida de similaridad.

El algoritmo utiliza el modelo "Sliding Window" con una ventana de tamaño fijo w y asume que los datos llegan de manera síncrona. Realiza pre-procesamiento de los streams sobre la ventana dividida en m bloques de tamaño v. El objetivo del pre-procesamiento es obtener los coeficientes de la DFT sobre los streams normalizados y ponderados. El clustering se lleva a cabo mediante una versión incremental del algoritmo K-Means basado en distancia euclidiana aplicada sobre los v primeros coeficientes de la DFT. Esta es una versión incremental del algoritmo, cada vez que se completa un nuevo bloque, se realiza el pre-procesamiento y se obtienen las distancias entre los streams por pares. Luego se continúa con el algoritmo K-Means estándar. De esta manera la estructura del clustering actual sirve de inicialización para la nueva.

Para modelar la evolución de los clusters, el algoritmo adapta incrementalmente el número óptimo de clusters, denominado K^*. Esta adaptación se restringe a una variación de una unidad ya sea que disminuya o aumente. La justificación es que usualmente la estructura de clustering no cambia abruptamente. Utiliza una medida de calidad $Q(k)$ para obtener K^* y realiza la elección bajo el siguiente criterio.

$$K^* < -argmaxQ(K-1), Q(K), Q(K+1) \tag{2.19}$$

Como se mencionó anteriormente, el algoritmo aplica el modelo "Sliding Window". Cuando un nuevo bloque de datos de tamaño v llega, este entra a la ventana y el bloque más antiguo es descartado. Además se utiliza un coeficiente de atenuación sobre los bloques. Debido a que los datos antiguos vienen a ser cada vez menos relevantes en el tiempo. La complejidad de procesar cada bloque es cuadrática en K.

Dai et al. en [6](2006) proponen un sistema para realizar clustering sobre streams de datos dinámicamente. Los autores lo denominan COD del inglés "Clustering On Demand". El sistema tiene un esquema de dos fases: fase de mantenimiento en línea y fase de clustering fuera de línea. En la fase de mantenimiento en línea el objetivo es realizar la recolección de estadísticas en una sola pasada sobre los datos que van llegando. Las estadísticas se guardan en una estructura jerárquica que se actualiza incrementalmente. Esta estructura tiene el objetivo de proveer resúmenes de los datos en diferentes resoluciones de tiempo. Se genera una estructura jerárquica por cada stream, así se tendrán un total de N estructuras para N streams. En la fase de clustering fuera de línea proponen un algoritmo adaptativo que obtiene las aproximaciones de los streams a partir de las jerarquías de resúmenes de la manera más precisa posible para dar como resultado los clusters para el intervalo de tiempo de interés. Proponen dos técnicas para obtener los resúmenes: wavelets y líneas de regresión. Realizan una comparación entre estas dos y finalmente proponen una versión adaptativa del sistema en el que combinan las dos técnicas para tomar ventaja de los puntos fuertes de ambas con respecto al espacio de almacenamiento requerido para los resúmenes y la precisión de las aproximaciones.

Fan, Watanabe y Asakura en [31](2008) proponen un algoritmo que se enfoca en la detección de similaridad en subsecuencias de data streams cuyo tamaño es especificado por el usuario. La idea es reducir la dimensionalidad de los data streams para reducir el costo del cálculo de la similaridad entre ellos, para lo cual mantiene un resúmen de estadísticas. El sistema realiza la aproximación de los ejemplos de forma incremental en segmentos lineales bajo un nuevo criterio de segmentación basado en la Máxima Distancia Vertical (MDV) entre los puntos del data stream y la línea de aproximación. En este proceso de aproximación se garantiza que el error de aproximación para cada ejemplo es menor que una tolerancia máxima δ dada por el usuario. El sistema es de dos fases similarmente a los dos anteriores. La fase en línea es la fase de mantenimiento de las estadísticas de la forma en como se explicó anteriormente. La fase fuera de línea es en la que se realiza el clustering. El clustering fuera de línea se realiza sobre las estadísticas tomadas en la fase en línea, se emplea un algoritmo de abstracción para obtener las subsecuencias de interés del usuario lo más precisamente posible. El algoritmo finalmente realiza el clustering sobre Sliding Windows utilizando el algoritmo K-Means.

Pereira et al. introducen el algoritmo ODAC en [26] (2008). El sistema construye un árbol binario de clusters siguiendo un enfoque divisivo basado en la correlación entre las series de tiempo. Las hojas del árbol contienen los clusters resultantes. El sistema realiza procedimientos de expansión y agregación sobre el árbol usando un nivel de significancia dado por la cota de Hoeffding para cada hoja. En el procedimiento de expansión una hoja se divide en dos hojas hijas creando una especificación del cluster. Mientras que en el procedimiento de agregación una hoja se agrupa con su subárbol hermano en el nodo padre creando una generalización de los clusters.

Tu et al. en [17](2009) propusieron un algoritmo basado en correlación. Los autores describen un esquema de compresión de los datos para múltiples streams.

El esquema genera una sinopsis la cual permite reconstruir incrementalmente los coeficientes de correlación sin la necesidad de acceder a los datos originales. Utiliza una modificación del algoritmo k-means para generar los clusters, el algoritmo también ajusta dinámicamente el número de clusters en tiempo real con el objetivo de detectar los cambios en la evolución de los streams de datos. Se puede observar que el algoritmo también es de dos fases como en casos anteriores.

2.9 Validación de clusters

La validación de clusters se refiere al proceso de evaluación de los resultados del clustering en una forma objetiva y cuantitativa [3]. Puesto que el clustering es un proceso no supervisado ya que no se cuenta con una clasificación previa y más aún no se conoce la distribución previa de los *data streams*, no hay forma de evaluar la validez de sus resultados por comparación. De esta forma las medidas de validación solo nos darán una indicación de la calidad de los clusters mas no una medida exacta de su efectividad.

2.9.1 Medidas de validación externa

De acuerdo con Wunsch y Xu [1] si $\mathbf{P}$ es una partición preespecificada del conjunto de datos $\mathbf{X}$ con N elementos, sea $\mathbf{P}$ independiente de la estructura de clustering $\mathbf{C}$ resultante de un algoritmo de clustering, entonces la evaluación de $\mathbf{C}$ mediante un criterio externo se logrará mediante la comparación de $\mathbf{C}$ y $\mathbf{P}$. Considerando un par de elementos x_i y x_j de $\mathbf{X}$, hay cuatro casos diferentes basados en cómo x_i y x_j están puestos en $\mathbf{C}$ y $\mathbf{P}$.

- Caso 1: x_i y x_j pertenecen al mismo cluster en $\mathbf{C}$ y a la misma categoría en $\mathbf{P}$
- Caso 2: x_i y x_j pertenecen al mismo cluster en $\mathbf{C}$ pero diferentes categorías en $\mathbf{P}$
- Caso 3: x_i y x_j pertenecen a diferentes clusters en $\mathbf{C}$ pero a la misma categoría en $\mathbf{P}$

- Caso 4: x_i y x_j pertenecen a diferentes clusters en **C** y a diferentes categorías en **P**

Denotaremos por a,b,c y d cada caso respectivamente. Ya que el número total de pares de elementos es $N\,(N-1)\backslash 2$, denotado como M, tenemos $a+b+c+d=M$.

Algunos indices de validación que siguen este criterio son:

1. **Indice Rand** mide el porcentaje de pares asignados correctamente sobre el número total de pares. Su resultado estarán en el rango $[0,1]$, los valores cercanos a 1 indican una alta similaridad entre las estructuras **P** y **C**

$$R = (a+d) \setminus (a+b+c+d) \tag{2.20}$$

2. **Coeficiente Jaccard** similar al índice "Rand" , también evalua la proporción entre los pares correctamente asignados y el total de pares, sin embargo ovbia el término d pues este puede ser calculado a partir de los otros tres. Sus resultados estarán en el rango $[0,1]$, los valores cercanos a 1 indican una alta similaridad entre las estructuras **P** y **C**

$$J = a \setminus (a+b+c) \tag{2.21}$$

3. **Índice Fowlkes and Mallow** sus resultados estarán en el rango $[0,\sqrt{2}]$, los valores cercanos a $\sqrt{2}$ indican una alta similaridad entre las estructuras **P** y **C**

$$FM = \sqrt{\frac{a}{a+b}\frac{a}{a+c}} \tag{2.22}$$

4. **Estadístico de Hubert** Γ sus resultados estarán en el rango $[-1,1]$. Similarmente a los anteriores, valores altos indican una alta similaridad entre las estructuras **P** y **C**

$$\Gamma = \frac{Ma - m_1 m_2}{\sqrt{m_1 m_2 (M-m_1)(M-M_2)}} \tag{2.23}$$

donde $M = N(N-1)/2$, $m_1 = a + b$ y $m_2 = a + c$.

2.9.2 Medidas de validación interna

Las medidas de validación interna evalúan la calidad del clustering basandose solo en información relacionada al conjunto de datos en sí, por ejemplo: la matriz de proximidad) y del clustering a evaluar. En muchas aplicaciones reales por lo general no contaremos con información externa, como una clasificación previa. En estos casos las medidas de validación internas serán la única opción para evaluar el resultado del clustering. Existen dos criterios en los que se basan estas medidas: compactibilidad y separación.

La compactibilidad es una medida de la cercanía de los objetos dentro de un cluster. Comunmente es evaluada en términos de varianza, una varianza baja indica una buena compactibilidad o en términos de distancia, ya sea distancia entre pares o relativa al centro.

La separación mide cuan distinto o bien separado está un cluster de los otros. Muchos índices evalúan la separación en términos de distancia ya sea entre los centros de los clusters o entre pares de objetos que pertenezcan a distintos clusters, otros índices la evalúan en términos de densidad.

A continuación se presentan algunos índices de validación interna comunmente utilizados.

1. **Índice de Dunn** (Dunn, 1974). Está dado por la siguiente ecuación:

$$D = min_{i=1\ldots n_c}\left\{min_{j=i+1\ldots n_c}\left(\frac{d\left(c_i, c_j\right)}{max_{k=1\ldots n_c}(diam(c_k))}\right)\right\}, \text{ donde} \tag{2.24}$$

$$d(c_i, c_j) = min_{x\in c_i, y\in c_j}\left\{d(x,y)\right\} \quad \text{y} \quad diam(c_i) = max_{x,y\in c_i}\left\{d(x,y)\right\}$$

Distancias grandes entre los clusters nos indicarán que estos están bien separados, asimismo diámetros pequenños nos indicarán compactibilidad en los clusters, por lo que los valores altos de este índice nos indican un buen resultado clustering. Sus valores están limitados al intervalo $[0, \infty]$.

2. **Índice Silueta (Silhouette)** (Rousseeuw, 1987). Este índice es el promedio del ancho de la silueta de todos los clusters. El ancho de la silueta de un objeto $\mathbf{x}$ que pertenece a un cluster C_k de tamaño n_k esta definido por:

$$S(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{max\left[b(\mathbf{x}), a(\mathbf{x})\right]} \tag{2.25}$$

donde, $a(\mathbf{x})$ es la disimilaridad promedio del objeto $\mathbf{x}$ con respecto a todos los demás objetos en el mismo cluster,

$$a(\mathbf{x}) = \frac{1}{n_k - 1} \sum_{\mathbf{x}_i \in C_k, i \neq j} d(\mathbf{x}_i, \mathbf{x}_j) \tag{2.26}$$

y $b(\mathbf{x})$ es el mínimo de los promedios de las disimilaridades entre el objeto $\mathbf{x}$ y todos los objetos que pertenecen a otros clusters.

$$b(\mathbf{x}) = min_{h=1,\ldots,K, h \neq k} \left[\frac{1}{n_h} \sum_{y \in C_h} d(x, y)\right], \quad \text{donde} \quad K \quad \text{es el número de clusters} \tag{2.27}$$

Finalmente, el índice Silueta global esta dado por:

$$S = \frac{1}{K} \sum_{k=1}^{K} \left[\frac{1}{n_k} \sum_{x \in C_k} S(\mathbf{x})\right] \tag{2.28}$$

Los valores del índice están en el rango $[-1, 1]$. Para el objeto $\mathbf{x}$, un valor cercano a -1 significa que el objeto está mas cercano, en promedio, a otro cluster que al que pertenece. Mientras que un valor cercano a 1 significa que la distancia promedio a su propio cluster es significativamente más pequeña que la distancia hacia otros clusters[19]. Por lo tanto, valores altos de este índice

indicarán una buena calidad del clustering en términos de compactibilidad y separación. Adicionalmente, para determinar el número óptimo de clusters se debe maximizar el valor del índice.

3. **Correlación de Hubert con matriz de distancia**. Sea D la matriz de similaridad entre los n objetos del conjunto de datos, tal que $D(\mathbf{x}_i, \mathbf{x}_j)$ es la similaridad entre $\mathbf{x}_i$ y $\mathbf{x}_j$. Sea P la matriz de similaridad que representa la pertenencia o no de los objetos al mismo cluster. Donde $P(\mathbf{x}_i, \mathbf{x}_j) = 1$ si $\mathbf{x}_i$ y $\mathbf{x}_j$ pertenecen al mismo cluster y $P(\mathbf{x}_i, \mathbf{x}_j) = 0$ en caso contrario. El índice Γ_D, está dado por la correlación entre las matrices D y P:

$$\Gamma_D = \frac{n(n-1)}{2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P(\mathbf{x}_i, \mathbf{x}_j) D(\mathbf{x}_i, \mathbf{x}_j) \tag{2.29}$$

Valores cercanos a 1 indican una buena calidad del clustering.

2.9.3 Medidas de validación relativa

Los índices de validación relativa evalúan la calidad del clustering en base a la comparación de los resultados de de diferentes esquemas del mismo algoritmo obtenidos bajo diferentes configuraciones de sus parámetros. El objetivo es encontrar el grupo de parámetros que conlleve al mejor clustering, es decir aquel que represente mejor la estructura del conjunto de datos. Existen dos casos en este problema:

I) Cuando el número de clusters no es uno de los parámetros: en este caso se realizan varias corridas del algoritmo utilizando diferentes valores para sus parámetros en cada corrida. Se elige el rango de valores más amplio posible para el cual el número de clusters permanezca constante. Finalmente los valores apropiados para los parámetros del algoritmo serán los que correspondan a la mediau del rango elegido. Este proceso también identifica el número de clusters subyacente en el conjunto de datos.

II) Cuando el número de clusters es uno de los parámetros: en este caso la evaluación de la calidad del clustering estará sujeta a un índice de validación v. El

procedimiento consiste también en realizar varias corridas del algoritmo en forma anidada. Sea n_c el número de clusters. Se corre el algoritmo para todos los valores de n_c tomados de un rango definido previamente por el usuario. Para cada uno de los valores de n_c el algoritmo se corre r veces utilizando diferentes valores para los demás parámetos del algoritmo y se calcula el valor del índice de validación v en cada corrida. Finalmente realizamos una gráfica de los mejores valores de v obtenidos para cada uno de los n_c en función de n_c. Para identificar el mejor esquema del algoritmo existen dos criterios dependiendo del comportamiento de v con respecto de n_c. Si el índice de validación no presenta una tendencia de incremento o decremento a medida que el número de clusters n_c se incrementa (decrementa), buscamos el máximo o el mínimo de la gráfica. Por otro lado, si el índice de validación crece (o decrece) cuando el número de clusters se incrementa, buscamos un valor de n_c para el cual ocurra un cambio local significativo en el valor del índice v. Esto indica el número de clusters subyacente en el conjunto de datos. Mas aún, si no existe el cambio mencionado, esto indica que el conjunto de datos no posee una estructura de clusters.

Un índice utilizado en este tipo de validacíon es el índice modificado de Hubert Γ, el cual está dado por:

$$\Gamma = \frac{(n)(n-1)}{2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} P_{ij} Q_{ij} \tag{2.30}$$

donde P es la matriz de proximidad y Q es una matriz de $n \times n$ donde $Q(\mathbf{x}_i, \mathbf{x}_j)$ es la distancia entre los objetos representativos de los clusters a los cuales pertenecen $\mathbf{x}_i$ y $\mathbf{x}_j$. Un valor alto en el índice indica una estructura compacta en los clusters.

En esta tesis, se utilizan los índices de validación interna índice de Dunn, la Silueta y la correlación de Hubert con matriz de distancia.

CAPÍTULO 3
Algoritmos para clustering de "Data Streams" por Variables

3.1 Introducción

La mayoría de los algoritmos mencionados en la sección anterior tienen un esquema de dos fases. Una fase en línea que es en la que se obtienen las estadísticas necesarias para el cálculo de la proximidad entre los *streams* y una fase fuera de línea en la que se realiza el clustering. Estos algoritmos tienen diferentes estratégias para obtener las estadísticas como un resumen o sinopsis compacta. Por ejemplo la (DFT) Transformada Discreta de Fourier en [14], segmentación lineal en [31], wavelets y líneas de regresión en [6]. Estos algoritmos operan sobre una ventana de tiempo bajo el argumento de que las observaciones recientes son más importantes que los datos pasados y realizan el clustering utilizando un algoritmo de particionamiento para clustering *batch* como el K-means. De entre todos estos algoritmos el más reciente es CORREL, propuesto por Tu et al.[17] en el 2009. Por otro lado ODAC [26] es el único que utiliza un enfoque jerárquico tanto para almacenar las estadísticas como para realizar el clustering. Ya que nos interesa estudiar el desempeño de ambos esquemas estudiaremos los algoritmos CORREL y ODAC. En este capítulo se detallan ambos algoritmos.

3.2 Clustering Aglomerativo-Divisivo en línea (ODAC)

Este es un algoritmo de clustering jerárquico divisivo-aglomerativo para series de tiempo introducido por Pereira et al. [26] en el 2008. El sistema construye un

árbol binario de clusters siguiendo un enfoque divisivo basado en la correlación entre las series de tiempo a las que a partir de ahora les llamaremos las variables.

Las hojas del árbol contienen los clusters resultantes, es decir que agrupan un subconjunto de las variables del conjunto de datos de forma determinística, es decir que la unión de todas las hojas es el conjunto completo de variables y la intersección es el conjunto vacío. El sistema realiza procedimientos de expansión y agregación sobre el árbol usando un nivel de significancia dado por la cota de Hoeffding para cada hoja.

En el procedimiento de expansión una hoja se divide en dos hojas hijas creando una especificación del cluster. Mientras que en el procedimiento de agregación una hoja se agrupa con su subárbol hermano en el nodo padre creando una generalización de los clusters. En el algoritmo 1 se muestra el proceso global de ODAC

3.2.1 Medida incremental de disimilaridad

El sistema utiliza el coeficiente de correlación de Pearson dado en la siguiente ecuación:

$$corr\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{\sum_{l=1}^{n} x_{il}x_{jl} - n\overline{\mathbf{x}_i}\ \overline{\mathbf{x}_j}}{\sqrt{\sum_{l=1}^{n} x_{il}^2 - n\overline{\mathbf{x}_i}^2}\sqrt{\sum_{i=1}^{n} x_{jl}^2 - n\overline{\mathbf{x}_j}^2}} \tag{3.1}$$

donde $\overline{\mathbf{x}}_i = \frac{1}{n}\sum_{l=1}^{n} x_i$, n es la longitud de un segmento del *stream* en el momento actual o el tamaño de la ventana de tiempo.

Los factores para calcular la correlación pueden ser actualizados incrementalmente. Sea $A = \sum a_i$, $B = \sum b_i$, $A_2 = \sum a_i^2$, $B_2 = \sum b_i^2$, $P = \sum a_i b_i$ Podemos escribir la correlación mediante la siguiente expresión incremental:

$$corr(a, b) = \frac{P - \frac{AB}{n}}{\sqrt{A_2 - \frac{A^2}{n}}\sqrt{B_2 - \frac{B^2}{n}}} \tag{3.2}$$

Algoritmo 1: Algoritmo global ODAC

```
1  begin
2      while TRUE do
3          Leer el próximo ejemplo
4          Actualizar las estadísticas suficientes en todas las hojas
5          foreach n_min Ejemplos do
6              foreach l ∈ Leaves do
7                  Actualizar las disimilaridades(D)
8                  Actualizar la cota de Hoeffding ε_l para esta hoja
9                  d_1 ← d(x, y) ← max(D)
10                 d_2 ← (max(D) \ d_1)
11                 if (d_1 − d_2 > ε_l or ε_l < τ) then
12                     // Prueba para agregación
13                     TestAggregate(l)
14                     if no se debe hacer una agregación then
15                         // Prueba para división
16                         TestSplit(l)
17                     end
18                 end
19             end
20         end
21     end
22 end
```

Las estadísticas A,B,A_2,B_2 y P son las suficientes para realizar el clustering.La medida de disimilaridad utilizada en ODAC está dada por la raíz cuadrada normalizada de uno menos la correlación.

$$rnorm_{(a,b)} = \sqrt{\frac{1 - corr(a,b)}{2}} \tag{3.3}$$

la cual tiene rango [0,1]. Como se mencionó en la sección 2.4 cuando la correlación se utiliza como medida de similaridad se utiliza el valor absoluto. Sin embargo, en ODAC no se utiliza el valor absoluto considerando que una correlación negativa significa una alta disimilaridad.

En cada paso t, si el número de variables en el cluster C es k, entonces su matriz de disimilaridad $M^t \in R^{k \times k}$ está dada por $M^t_{i,j} = d_t(i,j)$. Similarmente a

DIANA, el diámetro de un cluster es la mayor disimilaridad entre dos variables que pertenecen al mismo cluster o la varianza de la variable en el caso de clusters con un sólo elemento.

3.2.2 Construcción del árbol

El proceso principal en ODAC es la construcción de una estructura de árbol que represente la jerarquía de los clusters presentes en el conjunto de datos. El sistema actualiza incrementalmente las estadísticas suficientes en las hojas para calcular la matriz de disimilaridad cada vez que llega un nuevo ejemplo. Después de haber recibido un número mínimo *nmin* de ejemplos se calcula la matriz de disimilaridad para cada hoja y se realizan las pruebas para determinar si se realizará una división o un agrupamiento.

La decisión de división o agregación se apoya en un nivel de significancia dado por la cota de Hoeffding [13] la cual establece que para n observaciones independientes de la variable aleatoria x con rango R y con confianza $1-\delta$, la media verdadera de x es como mínimo $\overline{x}-\epsilon$, donde $\overline{x}$ es la media observada de la muestra y $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$.

La cota de Hoeffding tiene la ventaja de ser independiente de la distribución de probabilidad de las observaciones [24].

A medida que se procesan los ejemplos, cada hoja C_k posee un valor de ϵ diferente, designado por ϵ_k. Dado esto uno puede considerar dos distancias diferentes solo si su diferencia es mayor que ϵ_k. Los criterios de división y agregación están basados en el diámetro del cluster, es decir la distancia entre las dos variables mas alejadas. Sea $D_k = \{d(x_i, y_j) \,|x_i, y_j \in C_k, i < j\}$, $d_1 = d\{(x_1, y_1)\} = argmax_{d(x,y)}(D_k)$ y $d_2 = d(x_2, y_2) = argmax_{d(x,y)}(D_k \setminus \{d(x_1, y_1)\})$. Si la diferencia entre el d_1 y la segunda mayor distancia (d_2) es estadísticamente significativa de acuerdo a la cota de Hoeffding, entonces d_1 es el diámetro real del cluster. Esto es,

$$d_1 - d_2 > \epsilon_k \Rightarrow diam(C_k) = d_1 \tag{3.4}$$

Con esta regla, solo se tomará una decisión de división o agregación sobre un cluster cuando su verdadero diámetro sea conocido con confianza estadística dada por la cota de Hoeffding. Esta regla indica el momento cuando una hoja ha sido alimentada con la cantidad suficiente de ejemplos para soportar la desición.

Criterio de División

En ODAC, los autores analizan las relaciones entre las distancias dentro del cluster. La heurística que utilizan para realizar una división es la siguiente:

Para un cluster C_k dado, una hoja se convierte en un nodo con dos hojas hijas si la siguiente condición se cumple:

$$(d_1 - d_0) \mid \left(d_1 - \overline{d}\right) - \left(\overline{d} - d_0\right) \mid > \epsilon_k \tag{3.5}$$

donde d_0 es la mínima distancia entre las variables dentro del cluster y $\overline{d}$ es el promedio de todas las distancias en el cluster.

El criterio de división de ODAC presenta dos conceptos inherentes: mientras más alejada esté la máxima distancia de la mínima distancia, mayor será la posibilidad de una división. También mientras más alejada este la distancia media del promedio de las distancias máxima y mínima, mayor será la probabilidad de que ocurra una división. La expresión da el posicionamiento global de la media con respecto al rango de distancias existentes. Este criterio ha sido útil también como criterio de parada del crecimiento del árbol.

Manejo de empates

La condición presentada en la ecuación 3.4 nos lleva a pensar en dos problemas. Los casos en los que el cluster tiene muchas variables casi equidistantes y los casos en los que hay dos o más variables con alta disimilaridad. Para distinguir entre estos dos casos se debe hacer un ajuste.

Teniendo en cuenta que el sistema se aplica a streams de datos con una dimensión grande, posiblemente cientos o miles de variables, se adopta un enfoque

heurístico. En base a las técnicas presentadas en [24] y [15]. Se introduce el parámetro τ, el cual determina hasta cuando esperaremos que el sistema verifique si se consiguió el diámetro real antes de forzar las pruebas para la división y agregación. En cualquier momento si $\tau > \epsilon_k$ las pruebas son aplicadas asumiendo que las hojas procesaron la cantidad suficiente de ejemplos, entonces se considera la mayor distancia como el diámetro real.

Expansión del árbol

Cuando un cluster pasa la prueba de división, el proceso consiste en generar dos nuevos clusters a partir de dos pivots los cuales son las variables que definen el diámetro del cluster, es decir x_1 y y_1 donde $d_1 = (x_1, y_1)$. Cada uno de los pivots forma un nuevo cluster, las demás variables del cluster original se asignan al nuevo cluster que tenga el pivot más cercano. Las estadísticas suficientes de cada nuevo cluster son inicializadas. El espacio total requerido por los dos nuevos clusters es siempre menor que el requerido por el cluster previo. El algoritmo 2 muestra el procedimiento de división.

3.2.3 Detección de cambios en la estructura

Las estadísticas sólo se actualizan en las hojas cada vez que llega un nuevo ejemplo. Esto implica que las decisiones de expansión o agregación de la estructura está basada en los datos correspondientes a una ventana de tiempo sobre el stream. Cada nodo tiene su propia ventana de tiempo.

Las hojas son los nodos que corresponden a los datos más recientes. Estas expresan los cambios en las relaciones entre las variables que agrupan. La estrategia que ODAC utiliza para la detección de cambios en la estructura los clusters está basada en el análisis de los diámetros. En datos estacionarios la correlación entre las series de tiempo permanece constante. En ese caso el criterio de división garantizaría que el diámetro de los nuevos clusters sea menor que el del original. Sin embargo, en problemas del mundo real es usual lidiar con streams de datos no estacionarios,

Algoritmo 2: ODAC: Prueba de división

Input: l: Una hoja en la estructura de clusters
$X = \{x_1, \ldots, x_j\}$: Conjunto de variables en l

begin
  $d_1 \leftarrow d_{(x_1,y_1)} = argmax_{d_{(x,y)}} \left(D_k \setminus \left\{d_{(x_1,y_1)}\right\}\right)$
  $d_2 \leftarrow argmax_{d_{(x,y)}} \left(D_k \setminus \{d\,(x_1, y_1)\}\right)$
  $\overline{d} \leftarrow$ el promedio de las distancias en el cluster
  $d_0 \leftarrow$ la mínima distancia entre las variables de l
  if $(d_1 - d_0) \,|\, \left(d_1 - \overline{d}\right) - \left(\overline{d} - d_0\right) | > \epsilon_k$ **then**
    crear dos nuevas hojas:C_x y C_y con x_1 y y_1 como pivots:
    $x_1 \in C_x \wedge y_1 \in C_y$ **foreach** $x_i \in X$ **do**
      asignar las variables al cluster con el pivot más cercano
      if $d\,(x_i, x_1) \leq d\,(x_i, y_1)$ **then**
        Asignar x_i a C_x
      end
      else
        Asignar x_i a C_y
      end
    end
  end
end

donde la correlación entre las series de tiempo no es constante. Para cada hoja c_k, podemos verificar si la decisión de división que la creó aún representa la estructura de los datos. Se asume que la suma de los diámetros de los nodos hijos no es mayor que dos veces el diámetro del padre. Por lo tanto la prueba de agregación se realiza sobre los diámetros de c_k, de su hermano c_s, y de su padre c_j, elegimos agregar en c_j si:

$$(2diam\,(c_j)) - (diam\,(c_k) + diam\,(c_s)) < \epsilon_j \tag{3.6}$$

con ϵ_j dado por la cota de Hoeffding del padre c_j. Apoyado por la confianza dada por los datos procesados por el padre. El sistema reduce el número de clusters debido a que la división previa ya no refleja la mejor estructura de los datos. La

nueva hoja inicializa sus estadísticas y el cambio en la estructura fue detectado. El algoritmo 3 muestra el proceso de agregación.

Algoritmo 3: ODAC: Prueba de agregación

Input: c_k: Una hoja en la estructura de clusters

```
1 begin
2 |  c_s ← sibling(c_k)
3 |  c_j ← parent(c_k)
4 |  Actualizar las disimilaridades de c_k y c_s si es necesario
5 |  if (2diam(c_j) − (diam(c_k) + diam(c_s)) < ε_j then
6 |  |  juntar c_k y c_s en c_j
7 |  |  reinicializar las estadísticas de c_j
8 |  end
9 end
```

3.2.4 Análisis de Complejidad

Uso de Memoria

La complejidad se definirá en términos de espacios de memoria, asumiendo que cada espacio es capaz de almacenar un valor de punto flotante. En ODAC se mantiene la estructura de árbol en memoria. Cada nodo del árbol mantiene las estadísticas suficientes para las variables que contiene, para n variables el sistema necesitaría $2n$ espacios de memoria para los vectores que contengan las sumas y sumas al cuadrado, y $\frac{n(n-1)}{2}$ para el vector que contenga la mitad (triangular superior) de la matriz de multiplicaciones por pares de las variables. Por lo tanto la cantidad de memoria necesaria para almacenar las estadísticas suficientes para un nodo con n variables es:

$$\frac{n(n-1)}{2} + 2n = \frac{n(n+3)}{2} = O(n^2) \tag{3.7}$$

El peor de los casos se dá cuando en cada división de una hoja con n variables una de las hijas resulta conteniendo una única variable y la otra el resto de las variables. En este caso la cantidad adicional de espacios de memoria por cada división crece en 2 por una de las hijas y en $\frac{(n-1)(n+2)}{2}$ para la otra hija ya que

se necesitan $2(n-1)$ espacios de memoria para almacenar las sumas y sumas al acuadrado, y $\frac{(n-1)(n-2)}{2}$ para almacenar las multiplicaciones por pares. Por esta razón el incremento en la cantidad de espacios de memoria en cada división es como máximo:

$$2 + \frac{(n-1)(n+2)}{2} = \frac{n(n+1)}{2} + 1 \tag{3.8}$$

En este caso se realizarán como máximo $n-1$ divisiones. Sea k el número de divisiones realizadas y n_k el número de variables en el nodo más grande después de k divisiones ($n_k = n-k, n_0 = n$). La cantidad de unidades de memoria necesaria con respecto al número de divisiones es:

$$f(k) = f(k-1) + \frac{n_{k-1}(n_{k-1}+1)}{2} + 1 \tag{3.9}$$

con $f(0) = \frac{n(n+3)}{2}$. Por lo tanto la cantidad de unidades de memoria total requerida es:

$$\frac{n(n+3)}{2} + n + \sum_{k=0}^{n-1} \frac{(n-k)(n-k+1)}{2} = O(n^3) \tag{3.10}$$

Por lo tanto la complejidad del sistema en términos de la cantidad de memoria requerida es cúbica con respecto al número de variables pero constante con respecto al número de ejemplos.

Tiempo

El análisis de complejidad con respecto al tiempo está basado en el número de operaciones realizadas durante el procesamiento. En la línea 4 del algoritmo global de ODAC (Algoritmo 1) se realiza la actualización de las estadísticas suficientes para cada una de las hojas. Ya que cada variable aparece sólo una vez en las hojas, la actualización implica la realización de n sumas y n sumas al cuadrado. En el peor de los casos, el árbol consiste en un sólo nodo, un sólo cluster, conteniendo todas las

variables. En este caso en número de multiplicaciones es $n(n-1)/2$, este número de multiplicaciones disminuye cuando se realizan divisiones de los clusters. Por lo tanto en el peor de los casos la actualización de las estadísticas suficientes resulta en $2n + \frac{n(n-1)}{2}$ sumas y $n + \frac{n(n-1)}{2}$ multiplicaciones. Considerando que el cómputo necesario para ejecutar una suma y una multiplicación es el mismo, la cantidad de operaciones a realizar en la actualización es:

$$3n + n(n-1) = O(n^2) \tag{3.11}$$

Por lo tanto la actualización de las hojas tiene una complejidad cuadrática con respecto al número de variables, pero constante para cada ejemplo.

En la línea 7 del Algoritmo 1, se realiza el cáculo de las disimilaridades para cada hoja entre las variables que contenga. En el peor de los casos, donde sólo se tiene una hoja conteniendo a todas la variables el número de disimilaridades a calcular es de $n(n-1)/2$ por lo que también el número de operaciones para este cálculo es es de orden cuadrático con respecto al número de variables.

En [27] los autores indican que la complejidad del procedimiento de agregación es constante porque sólo consideran la realización de la prueba (línea 5 del Algoritmo 3), sin embargo el proceso de agregación incluye la actualización de las estadísticas del hermano y del padre del nodo que se está procesando de ser necesario. El proceso de juntar (línea 6) es constante, sin embargo al finalizar se deben reinicializar las estadísticas del nodo padre (que se está convertiendo en hoja), el proceso de reinicializar las estadísticas es de orden cuadrático, ya que se deben reinicializar los vectores de sumas, sumas al cuadrado y productos por pares. Por lo tanto este proceso es de orden cuadrático con respecto al número de variables.

El proceso de división incluye el cálculo de de la distancia máxima, mínima y la media de las distancias en la matriz de disimilaridad. Este procedimiento es tambíen de orden cuadrático con respecto al número de variables, porque aunque se considera

que la matriz de disimilaridad ya existe, se realizan por lo menos $\frac{n(n-1)}{2} - 1$ sumas para el cálculo de la media y un número igual de comparaciones para determinar la distancia máxima y mínima. Es importante mencionar que a medida que se realizan las divisiones de los nodos (descubrimiento de nuevos clusters) el número de operaciones necesarias para procesar las nuevas hojas disminuye como mínimo en $k - 1$, donde k es el número de variables en el nodo que se divide y en el mejor de los casos en $\frac{k}{2}$.

En conclusión ya que los procesos que más consumen tiempo como son la actualización de las estadísticas, el cálculo de las disimilaridades, los procesos de agregación y división son de orden cuadrático con respecto al número de variables, entonces el tiempo de ejecución del algoritmo ODAC es de orden cuadrático con respecto al número de variables. Ya que el tiempo de ejecución para procesar cada ejemplo es constante, es decir que el algoritmo siempre tomará el mismo tiempo para procesar el ejemplo 1 que el ejemplo 1000 o el ejemplo m donde m es un número arbitrario de ejemplos, el algoritmo es de orden lineal con respecto al número de ejemplos.

3.3 Clustering de Múltiples Data Streams basado en Análisis de Correlación - CORREL

Este sistema fue propuesto por Tu et al. en [17] el 2009, lo denominan CORREL. El sistema mantiene una sinopsis que permite el cálculo de los coeficientes de correlación sin la necesidad de acceder a los datos originales. Utiliza el modelo *Sliding Window* y utiliza una implementación modificada del algoritmo K-means para generar los resultados del clustering. Ajusta dinámicamente el número de clusters en tiempo real, de esta manera se detecta la evolución de los cambios en los streams de datos.

CORREL utiliza un coeficiente de atenuación $\lambda \in [0, 1]$ para los datos similarmente al modelo *Damped Window*. Sin embargo, la actualización de las estadísticas se realizan como en *Sliding Window* con ventanas básicas. El coeficiente

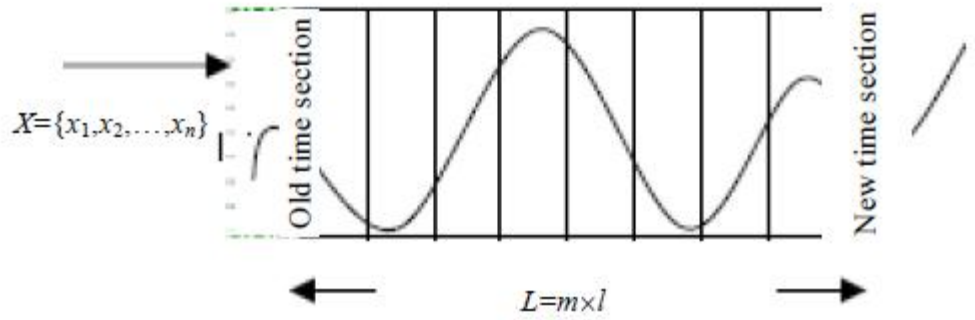

Figura 3–1: Segmento de tiempo L dividido en m segmentos de tamaño l

de atenuación gradualmente disminuye la importancia de cada ejemplo en el tiempo. Supongamos que t es el tiempo actual y que recibimos el punto x_i en el instante i. Luego, se reemplaza el valor original de x_i por:

$$x_i\,(t) = \lambda^{t-i} x_i \tag{3.12}$$

Los resultados del clustering son reportados en tiempo real en el horizonte de tiempo $[t - L + 1, t]$ Con el objetivo de lograr un procesamiento eficiente, los streams de datos de tamaño L, se particionan en m segmentos de tiempo de igual tamaño: $l = L/m$. En el momento en el que se acumula un nuevo segmento de tamaño l, los resultados del clustering se vuelven a calcular. Es decir que utiliza una ventana de tamaño L y ventanas básicas de tamaño l.

3.3.1 Representación Comprimida de Correlación

El algoritmo CORREL realiza clustering continuamente para la ventana de tamaño fijo L. El algoritmo similarmente a ODAC, mantiene un conjunto de estadísticas suficientes para realizar el clustering. El conjunto de estadísticas se denomina *representación comprimida de correlación* **CCR** por sus siglas en inglés (compressed correlation representation).

El algoritmo recibe un ejemplo en cada instante de tiempo t. Cada l ejemplos, se calcula el CCR correspondiente al segmento de tiempo $[t - l + 1, t]$ (ventana básica). Después de los primeros m segmentos de tamaño l se calcula el CCR para el primer

segmento de tiempo de tamaño L (ventana de tamaño L - *sliding window*) el cual se denomina CCRL. Posteriormente el CCRL será actualizado cada vez que llegue un nuevo segmento de tamaño l.

Para realizar el clustering utiliza el algoritmo "k-means" basado en correlación. Para la detección de cambios en la evolución de los streams el sistema utiliza un nuevo algoritmo denominado ***adjust-k*** el cual como su nombre indica ajusta el número de clusters dinámicamente. El algoritmo 4 muestra esquemáticamente el proceso del algoritmo CORREL. En la implementación realizada en este trabajo se utiliza "k-medoids" en lugar de "k-means" esto debido a que entendemos que el centroide debe estar dado por una de las variables.

Algoritmo 4: Correl-cluster

```
1  begin
2      t = 0 while el stream de datos no termina do
3          set t = t + 1
4          leer un nuevo ejemplo x_k (t) , k = 1, ..., n uno de
5          cada uno de los n streams de datos
6          if (t  mod  l = 0) then
7              calcular la CCR del nuevo segmento de tiempo
8              actualizar las CCRs de los segmentos existentes
9              if t = L then
10                 calcular el CCRL inicial
11                 para la ventana de tiempo [1, L]
12             end
13             else
14                 if t > L then
15                     actualizar incrementalmente CCRL
16                     para la nueva ventana de tiempo
17                     llamar a correlation-k-means()
18                     llamar a adjust-k()
19                     devolver el resultado del clustering
20                 end
21             end
22         end
23     end
24 end
```

3.3.2 Representación de correlación comprimida

Similarmente a ODAC en CORREL se utiliza la correlación de Pearson para definir la medida de dissimilaridad entre los streams de datos. La ecuación 3.1 muestra la correlación entre dos streams de datos.

Las estadísticas suficientes para realizar el clustering son básicamente aquellas que nos permitirán calcular el coeficiente de correlación entre las variables. Para dos streams de datos a y b, estas son $\sum a_i$, $\sum b_i$, $\sum a_i^2$, $\sum b_i^2$ y $\sum a_i b_i$. Es decir, las mismas estadísticas suficientes que se guardan en ODAC.

Definición 1. *Dados n streams de datos $X_1, \ldots, X_n$ suponiendo que el tiempo actual es t, el segmento de tiempo es de tamaño l, luego la representación de correlación comprimida de este segmento de tiempo está definida como* $\left(\vec{S}(t,l), \vec{Q}(t,l), C(t,l)\right)$.

Los componentes de los vectores $\vec{S}(t,l)$, $\vec{Q}(t,l)$ y la matriz $C(t,l)$ están definidos como sigue:

$$S_i = \sum_{k=t-l+1}^{t} x_{ik}(t), Q_i = \sum_{k=t-l+1}^{t} x_{ik}^2(t), C_{ij} = \sum_{k=t-l+1}^{t} x_{ik}(t)\, x_{jk}(t), i, j = 1, \ldots, n, i < j \tag{3.13}$$

donde, $x_{ik}(t)$ es el dato k de la variable i en el instante de tiempo t.

La representación de correlación comprimida provee información suficiente para calcular los coeficientes de correlación entre cualesquiera dos variables de las n existentes.

Teorema 1. *Sea X_i y $X_j, i, j = 1, \ldots, n$ segmentos de datos de dos variables, su coeficiente de correlación $\rho(x_i, x_j)$ puede ser calculado mediante la siguiente fórmula.*

$$\rho(x_i, x_j) = \frac{C_{ij} - \frac{1}{n} S_i S_j}{\sqrt{\left(Q_1 - \frac{1}{n} S_i^2\right)\left(Q_j - \frac{1}{n} S_j^2\right)}} \tag{3.14}$$

3.3.3 Actualización de la representación de correlación comprimida

Sea $t - t_c = \Delta t$ y λ el coeficiente de atenuación, en cualquier momento de tiempo $t > t_c$ los valores de los datos $x_{ik}(t_c)$ son reemplazados por $x_{ik}(t)$:

$$x_{ik}(t) = \lambda^{t-i} x_{ik} = \lambda^{t-t_c+t_c-i} x_{ik} = \lambda^{t-t_c} \lambda^{t_c-i} x_{ik} = \lambda^{\Delta t} x_{ik}(t_c) \tag{3.15}$$

Ya que los valores de x_{ik} son actualizados, también la representación comprimida de correlación debe ser correspondientemente actualizada.

Sea $\vec{S}_i(t_c, l_c)$, $\vec{Q}_i(t_c, l_c)$ y $C_{ij}(t_c, l_c)$,las componentes de la CCR en el instante de tiempo t_c, luego las componentes actualizadas son:

$$\vec{S}'_i(t_c, l_c) = \lambda^{\Delta t} \vec{S}_i(t_c, l_c) \tag{3.16}$$

$$\vec{Q}'_i(t_c, l_c) = \lambda^{2\Delta t} \vec{Q}_i(t_c, l_c) \tag{3.17}$$

$$C'_{ij}(t_c, l_c) = \lambda^{2\Delta t} C_{ij}(t_c, l_c) \tag{3.18}$$

Las CCR se actualizan cada l instantes de tiempo. En el momento en el que llega un nuevo segmento se calcula una nueva CCR y todas las demás se actualizan.

3.3.4 Agregación de la representación de correlación comprimida

Con el propósito de obtener el clustering para la ventana de tiempo $[t - L + 1, t]$ necesitamos calcular los coeficientes de correlación sobre los datos de dicha ventana. Ya que nosotros calculamos una representación de correlación comprimida para cada segmento de tamaño l, necesitamos combinarlas para formar la representación para la ventana de tiempo $[t - L + 1, t]$.

Tenemos m segmentos de tiempo de tamanño l y m CCR, los podemos denotar como:

$$\left[\vec{S}(t - vl + 1, l), \vec{Q}(t - vl + 1, l), C(t - vl + 1, l)\right]$$

para $v = 1, \ldots, m$. Denotamos la representación de correlación comprimida para la ventana de tiempo $[t - L + 1, t]$ como:

$$\left[\vec{S}_L, \vec{Q}_L, C_L\right]$$

.

Entonces en el tiempo t tenemos:

$$\begin{aligned} \vec{S}_{Li} &= \sum_{k=t-L+1}^{t} x_{ik}(t) = \sum_{v=1}^{m} \left(\sum_{k=t-vl+1}^{t-(v-1)t} x_{ik}\left(t\right) \right) \\ &= \sum_{v=1}^{m} \vec{S}_i\left(t - vl + 1, l\right) \;\; \forall i = 1, \ldots, n \end{aligned} \tag{3.19}$$

similarmente las otras componentes son:

$$\vec{Q}_{Li} = \sum_{v=1}^{m} \vec{Q}_i\left(t - vl + 1, l\right) \;\; \forall i = 1, \ldots, n \tag{3.20}$$

$$C_{Lij} = \sum_{v=1}^{m} C_{ij}\left(t - vl + 1, l\right) \;\; \forall i = 1, \ldots, n \;\; \wedge j = i, \ldots, n \tag{3.21}$$

Las ecuaciones anteriores son usadas sólo cuando recibimos los primeros m segmentos (líneas 9 a 11 del algoritmo 4), posteriormente sólo necesitaremos actualizar la representación de acuerdo a las siguientes ecuaciones:

$$\vec{S}\left(t - L + 1, L\right) = \vec{S}\left(t - L + 1, L\right) + \vec{S}\left(t + l, l\right) - \vec{S}\left(t - ml + 1, l\right) \tag{3.22}$$

$$\vec{Q}\left(t - L + 1, L\right) = \vec{Q}\left(t - L + 1, L\right) + \vec{Q}\left(t + l, l\right) - \vec{Q}\left(t - ml + 1, l\right) \tag{3.23}$$

$$C\left(t-L+1,L\right)=C\left(t-L+1,L\right)+C\left(t+l,l\right)-C\left(t-ml+1,l\right) \tag{3.24}$$

3.3.5 Algoritmo k-means dinámico (correlation-k-means)

Para realizar el clustering sobre la ventana de tiempo especificada por el usuario, los autores proponen el uso del ampliamente conocido algoritmo K-Means [23] con las modificaciones adecuadas. En el algoritmo la distancia entre dos variables X y Y es medida usando el recíproco del coeficiente de correlación $D\left(X,Y\right)=1/\rho(X,Y)$.

El algoritmo realiza un proceso iterativo que particiona el conjunto de variables en un número k de clusters dado en base a distancias basadas en correlación. Se basa en la idea de que dos resultados consecutivos de clustering no cambian muy rápido dentro de un espacio corto de tiempo l y que se pueden traslapar significativamente. El algoritmo puede tomar muchos menos pasos para converger si se comienza en el resultado de un segmento previo.

Los *centros* son elementos fundamentales en la realización del clustering. Podemos definir un centro como una variable representativa del cluster. En general, los centros inicialmente se eligen de forma aleatoria o bajo alguna heurística dada, en nuestra implementación se eligen de forma aleatoria. Posteriormente se calculan las distancias basadas en correlación entre las variables y los centros iniciales y se realiza la asignación inicial de las variables a los clusters cuyo centro sea el más cercano. Estos elementos son los parámetros de entrada para el algoritmo *correlation-k-means*.

El proceso iterativo del algoritmo tiene básicamente los mismos pasos que el proceso inicial previo. Calcular las distancias entre las variables y los centros, asignar las variables a los clusters cuyo centro sea el más cercano y por último determinar un nuevo centro para cada cluster. Los autores no detallan el proceso para la elección de los nuevos centros. Ya que el problema que se está abordando es el clustering sobre variables, la elección del nuevo centro no puede estar dada como la media de

los valores de las observaciones como en el K-Means clásico ya que, necesitamos que el centro sea una determinada variable del conjunto de datos. En la implementacón realizada en este trabajo el nuevo centro es el *medoide* del cluster, similarmente a como se realiza en el algoritmo PAM propuesto por Kaufman & Rousseeuw en 1987. El medoide es la variable con la menor disimilaridad promedio con respecto a las demás variables de su cluster.

Algoritmo 5: correlation-K-Means

Input: Número de clusters k, conjunto de centros $Center_k$, resultado del clustering actual R_k
Output: resultado actualizado del clustering R_k, función objetivo G_k

```
1  begin
2      while no hayan cambios en el clustering do
3          for i=1 to n do
4              calcular las distancias basadas en correlación entre el stream X_k
               y los centros de los k clusters y asignar X_k al cluster cuyo centro
               esté más cercano.
5          end
6          determinar el nuevo para cada cluster
7          actualizar el conjunto de centros Center_k
8      end
9      calcular la función objetivo G_k
10 end
```

Ajuste del parámetro k

El algoritmo incluye un proceso de ajuste del número de clusters denominado $adjust - k$. Recibe como parámetros de entrada el número inicial de clusters k (este se ajusta en el proceso), el conjunto de centros iniciales en base a los cuales se forman los clusters iniciales.

Similarmente a [14], el sistema ajusta el número óptimo de clusters considerando una variación en uno ya sea que aumente o disminuya. Este ajuste está basado en una medida de calidad del clustering o función objetivo $G = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (1/\rho\,(x_j, c_i))$. Para obtener el número óptimo de clusters se sigue los siguientes pasos.

1. Obtener los resultados de clustering para $k + 1$ y $k - 1$

- Para $k+1$: elegir el elemento que esté más alejado del centro del cluster al que pertenece. Formar un nuevo cluster con este elemento. Asignar los demás elementos del cluster al nuevo si están más cercanos del centro del nuevo cluster que del suyo.
- Para $k-1$: elegir los dos clusters más cercanos. Unirlos y elegir un nuevo centro para este nuevo cluster.

2. Calcular la función objetivo para cada uno de los resultados. Es decir para los resultados de clustering con k, $k+1$ y $k-1$ clusters.
3. El k óptimo será aquel para el cual el resultado de clustering obtenga el mayor valor en la función objetivo.

Algoritmo 6: Adjust-k

Input: Número de clusters k, conjunto de centros $Center_k$, resultado del clustering actual R_k

Output: Número de clusters actualizado k', resultado actualizado del clustering R_k, función objetivo G_k

```
1  begin
2      Cálculo de R_{k+1}
3      De entre todos los clusters, elegir el data stream X el cual es el más
       alejado del centro del cluster al que pertenece. Formar un nuevo cluster
       con el data stream X.
4      Center_{k+1}=Center_k ∪ X
5      correlation-K-Means(k+1,Center_{k+1},R_{k+1})
6      Cálculo de R_{k-1}
7      Elegir los dos clusters más cercanos, sean sus centros C_1 y C_2
       respectivamente.
8      Combinar estos dos clusters en un nuevo cluster
9      Calcular el centro C_3 del nuevo cluster
10     Center_{k-1}=Center_k ∪ C_3 − C_1, C_2
11     correlation-K-Means(k−1,Center_k,R_{k-1})
12     Elegir el que tenga el mayor valor para G, de entre R_{k-1}, R_k, R_{k+1},
13     establecer k' y R_{k'} respectivamente
14 end
```

3.3.6 Análisis de Complejidad

Uso de memoria

Como se hizo anteriormente, la complejidad se definirá en términos de espacios de memoria, asumimos que cada espacio de memoria es capaz de almacenar un valor de punto flotante. CORREL almacena las estadísticas suficientes bajo una representación denominada CCR para cada segmento del conjunto de streams de tamaño l. Por lo tanto, si el conjunto tiene n variables, para un CCR se necesitarán $2n$ espacios para el vector de sumas ($\vec{S}$) y el de sumas al cuadrado ($\vec{Q}$), y $\frac{n(n-1)}{2}$ espacios para los productos por pares de la matriz C, ya que en la implementación no es necesario almacenar la matriz completa si no que basta con su triangular estrictamente superior o inferior. En total para almacenar un CCR se necesitan:

$$2n + \frac{n(n-1)}{2} = \frac{n(n+3)}{2} \tag{3.25}$$

espacios de memoria. En total se mantienen $m+1$ CCR, donde m es el número de segmentos de tamaño l y una adicional para el nuevo segmento, ya que en el momento que este llega aun existe el CCR más antiguo el cual será posteriormente eliminado. Por lo tanto, en total se necesitan $(m+1)\left[\frac{n(n+3)}{2}\right]$ espacios de memoria para almacenar las CCR.

También CORREL mantiene la estructura CCRL para almacenar las estadísticas de la ventana deslizante o segmento de tamaño L. Para esta estructura se necesita la misma cantidad de espacios de memoria que para la CCR pues su tamaño depende del número de variables y no del número de ejemplos que ésta resume, esto es: $\frac{n(n+3)}{2}$ espacios de memoria.

El espacio de memoria total del sistema sería:

$$(m+1)\frac{n(n+3)}{2} + \frac{n(n+3)}{2} = (m+2)\frac{n(n+3)}{2} = O(n^2) \tag{3.26}$$

Para CORREL no podemos definir un peor o mejor caso en términos del espacio de memoria que requiere porque trabaja con un tamaño de ventana fijo. La complejidad del sistema en términos de espacio de memoria siempre será de orden cuadrático con respecto al número de variables y constante con respecto al número de ejemplos.

Tiempo

El análisis de complejidad con respecto al tiempo está basado en el número de operaciones realizadas durante el procesamiento, los pasos que realiza CORREL se muestran en el Algoritmo 4. En la línea 7 se realiza el cálculo de la estructura CCR para un nuevo segmento de tamaño l. Este cálculo requiere $(l-1)(2n)$ sumas para los vectores $\vec{S}$ y $\vec{Q}$, y $(l-1)n$ sumas y $l\left[\frac{n(n-1)}{2}\right]$ multiplicaciones para el vector de productos por pares (de la matriz C). Si consideramos que el tiempo de cómputo de las operaciones de suma y multiplicación es el mismo, se tendría un total de:

$$(l-1)(2n)+(l-1)n+l\left[\frac{n(n-1)}{2}\right]=\frac{ln\,(n+5)-3n}{2}=O(n^2) \tag{3.27}$$

operaciones. En la línea 8 del algoritmo se actualizan las CCR existentes, en este caso se actualizan como máximo $(m-1)$ estructuras CCR, esta actualización consiste en ajustar el peso de los CCR. Se realizan $(m-1)2n$ multiplicaciones para la actualización de los vectores $\vec{S}$ y $\vec{Q}$, y $(m-1)\left[\frac{(n-1)}{2}\right]$ multiplicaciones para la actualización de los productos por pares. En total se realizan:

$$2n(m-1)+(m-1)\left[\frac{(n-1)}{2}\right]=(m-1)\left[\frac{5n-1}{2}\right]=O(n) \tag{3.28}$$

El cálculo del CCRL inicial en la línea 10 consiste en juntar las estadísticas de los m CCR, por tanto se requieren:

$$m\left[2n + n\frac{(n-1)}{2}\right] = m\left[\frac{n(n+3)}{2}\right] = O(n^2) \tag{3.29}$$

sumas. La actualización del CCRL consiste en agregar las estadísticas del nuevo CCR y eliminar las del CCR más antiguo, esto requiere entonces $2n + \frac{n(n-1)}{2}$ sumas para agregar y un número igual de restas lo cual resulta en:

$$4n + n(n-1) = O(n^2) \tag{3.30}$$

operaciones. En la línea 17 se realiza el procedimiento "correlation-k-means()" para realizar el clustering. En general, la complejidad del algoritmo *k-means* es de orden $O(NKnT)$ donde T es el número de iteraciones, N el número de ejemplos, n el número de variables y K el número de clusters, donde K, n y T se consideran por lo general mucho menores que N, así la complejidad se considera lineal con respecto al número de ejemplos [1]. En el caso de este procedimiento como se mencionó en la Sección 3.3.5 en la implementación realizada en este trabajo el centro del cluster es su medoide, para calcular el medoide es necesario calcular la matriz de correlación de los elementos pertenecientes al cluster. Siendo este cálculo de orden cuadrático con respecto al número de variables, el procedimiento "correlation-k-means()" sería de orden cuadrático con respecto al número de variables, en este caso el número de ejemplos N se refiere al tamaño del segmento (ventana de tamaño L) el cual es constante. El procedimiento "Adjust k" básicamente hace uso de "correlation-k-means()" para $k+1$ y para $k-1$ por lo tanto también es de orden $O(n^2)$. Finalmente la complejidad del CORREL en términos de tiempo de cómputo es de orden cuadrático con respecto al número de variables pero constante para el procesamiento de cada ejemplo y por lo tanto sería de orden lineal con respecto al número de ejemplos.

CAPÍTULO 4
Resultados Experimentales

Se realizó un estudio comparativo del desempeño de ambos algoritmos en conjuntos de datos artificiales y conjuntos de datos reales. En conjuntos de datos reales no existe un resultado de clustering "correcto" ya que los datos podrían agruparse de forma válida de distintas maneras de acuerdo a distintos criterios. Por ese motivo en una primera etapa se realizaron evaluaciones sobre conjuntos de datos artificiales con el propósito de realizar experimentos sistemáticos en un ambiente controlado para evaluar aspectos específicos. Posteriormente se realizaron experimentos sobre conjuntos de datos reales.

4.1 Detalles de la implementación

Hasta la fecha de redacción de este trabajo no tenemos conocimiento de la existencia de alguna implementación disponible públicamente, tampoco recibimos referencias al respecto por parte de los autores. Por estas razones se realizó una implementación completamente propia de cada uno de los algoritmos basada en el trabajo de tesis doctoral de Pereira [27], el artículo publicado por Pereira et al. [25] para el algoritmo ODAC y el artículo de Tu et al.[17] para el algoritmo CORREL. La implementación nos permitió entender profundamente el funcionamiento de cada uno de los algoritmos así como poder establecer los parámetros y otros aspectos como la medida de disimilaridad con toda libertad.

El lenguaje de programación utilizado para la implementación fue Java Versión 6 SE. Se hizo uso del software Eclipse en su versión Indigo como entorno de desarrollo.

Todos los conjuntos de datos se encuentran almacenados en una base de datos en Microsoft SQL Server Express 2008. Se asume que llega una observación por variable en cada instante de tiempo. La llegada de los datos se simula obteniendo un registro a la vez de la tabla correspondiente en la base de datos.

Todas las pruebas se corrieron sobre un equipo con sistema operativo Windows 7 Profesional ©2009 con capacidad de memoria de 4GB y procesador Intel(R) Core(TM) 2 Duo.

4.2 Metodología

Se realizaron pruebas experimentales tanto en conjuntos de datos artificiales como en conjuntos de datos reales. Las pruebas en conjuntos de datos artificiales estuvieron dirigidas a evaluar los algoritmos en términos de eficacia, capacidad de detección de cambios en la estructura y eficiencia (tiempo de ejecución). Con los conjuntos de datos reales se evaluó la calidad del clustering mediante tres medidas de validación, el índice de Dunn (ecuación 2.24), el índice Silueta (ecuación 2.25) y Correlación de Hubert con matriz de distancia (ecuación 2.29).

Para realizar una comparación justa se buscó establecer condiciones similares en ambos algoritmos. En todas las pruebas se utilizó la misma medida de disimilaridad para ambos algoritmos, la raíz cuadrada normalizada de uno menos la correlación (ecuación 2.18).

Para el algoritmo CORREL se estableció el coeficiente de atenuación λ en 1, en otras palabras se dejó sin efecto debido a que ODAC no utiliza este tipo de ajuste en los datos. Por otro lado los parámetros para ODAC fueron los mismos que utilizaron los autores en sus pruebas experimentales, R y δ igual a 2 y 0.05 respectivamente. Estos últimos intervienen en el cálculo de la cota de Hoeffding. El parámetro τ, que sirve para el manejo de empates se estableció en 0.02. Estos se muestran en la Tabla 4–1.

ODAC	CORREL
$\delta \leftarrow 2$	$\lambda \leftarrow 1$
$R \leftarrow 2$	
$\tau \leftarrow 0.02$	

Tabla 4–1: Parámetros establecidos para los algoritmos.

Elegimos tres medidas de validación internas para evaluar la calidad de los resultados del clustering. El índice de Dunn (ecuación 2.24), índice Silueta (ecuación 2.25) y correlación de Hubert con matriz de distancia (ecuación 2.29). Como se mencionó en la sección 2.9.2 el índice de Dunn y el índice Silueta nos indican una buena calidad del clustering en términos de compactibilidad interna y separación externa, mientras que el último sólo esta relacionado con la compactibilidad interna. Para los tres índices un valor alto nos indica un buen resultado de clustering.

4.3 Evaluación de la eficacia de los algoritmos

Se crearon 4 conjuntos de datos artificiales con el propósito de evaluar la eficacia de los algoritmos para descubrir la estructura de clusters subyacente.

4.3.1 Conjunto de datos 1

Los valores de las series de tiempo para este conjunto se generaron de acuerdo al método propuesto por Enrico Schumann [29] para generar variables aleatorias correlacionadas. El procedimiento tiene dos pasos. El primer paso consiste en generar n observaciones para T variables. Las observaciones se generan mediante un generador de números pseudoaleatorios normalmente distribuidos. Como producto de este paso inicial, se obtiene una matriz R de dimesión $T \times n$. El segundo paso consiste en generar una matriz M de varianzas y covarianzas para establecer un grado de correlación entre las variables. Posteriormente se induce correlación al conjunto de variables utilizando el factor de *Cholesky* de la matriz M.

En ODAC se estableció NMIN en 100 y en CORREL los parámetros L y l se establecieron en 1000 y 100 respectivamente.

Prueba 1:

Para esta prueba se generaron 17 series de tiempo con 200000 observaciones cada una. Con una estructura subyacente de 4 clusters:

$\{v0, v2, v4, v6, v8, v10, v12\}$, $\{v1, v3, v5, v7, v9\}$,$\{v11, v13, v15\}$,$\{v14, v16\}$

ODAC descubrió la estructura de clusters predefinida después de 137667 ejemplos (69%), posteriormente el resultado del clustering se mantuvo estable. Por su parte, CORREL obtuvo la estructura de clusters predefinida después del ejemplo 50 (0.025%), igualmente el resultado del clustering posteriormente se mantuvo estable.

En pruebas adicionales se observó que al variar NMIN en ODAC, es decir, al tomar diferentes ventanas de tiempo para obtener las estadísticas, los resultados en diferentes momentos de tiempo varían. Sin embargo, la estructura de clusters predefinida siempre se consigue después de procesar en promedio aproximadamente un 69% de los ejemplos. Por otro lado, también se variaron los parámetros L y l en CORREL, se observó que independientemente de los parámetros el algoritmo obtiene los clusters después del 0.025% de ejemplos (50 ejemplos).

Prueba 2:

Para esta prueba se seleccionó un subconjunto del generado para la prueba 1. Esta selección consiste en 12 variables con los 200000 ejemplos cada una y con una estructura subyacente de 3 clusters: $\{v0, v2, v4, v6, v8, v10, v12\}$, $\{v11, v13, v15\}$,$\{v14, v16\}$.

En este caso ODAC consigue la estructura de clusters pre-establecida después de 30000 ejemplos (15%) para los diferentes valores de NMIN evaluados. Es decir que con 12 variables y 3 clusters el algoritmo requirió de un 54% menos de ejemplos (del total de 200000 ejemplos) que en el caso de la prueba 1.

Con CORREL la estructura de clusters preestablecida se consigue como en la prueba 1, en promedio después de 0.025%(50 ejemplos) de ejemplos para todos los tamaos de ventana evaluados.

4.3.2 Conjunto de datos 2

Los valores de las series de tiempo para este conjunto se generaron usando el modelo *Random Walk* como en [2]. Las series de tiempo se generaron de acuerdo a la ecuación 4.1.

$$s_i = 100 + \sum_{j=1}^{i} (u_j - 0.5),\ i = 1, 2, \ldots \tag{4.1}$$

donde u_j es un conjunto de series de tiempo que siguen una distribución uniforme. La correlación para las variables que forman un cluster fue inducida utilizando también el algoritmo de Enrico Schumann [29].

Se generaron 12 series de tiempo con 100000 ejemplos cada una. Se preestableció una estructura subyacente de 3 clusters:

$\{v0, v1, v2, v3, v4\}$, $\{v5, v6, v7\}$,$\{v8, v9, v10, v11\}$

El algoritmo ODAC se corrió con NMIN de 1, 100, 1000 y 2000. En todos los casos descubrió la estructura de clusters predefinida después de 28300 ejemplos (28%). Sin embargo la estructura de clustering final posee subdivisiones de los clusters predefinidos. La Figura 4–1a muestra la estructura final del clustering para NMIN igual a 100.

El algoritmo CORREL nunca obtuvo la estructura de clusters predefinida. La Figura 4–1b muestra la estructura obtenida con una ventana de tiempo de tamaño 1000 (parámetros L=1000, l=100).

De acuerdo a la revisión de literatura las series de tiempo que siguen el modelo de datos “Random Walk” poseen un componente tendencial estocástico lo cual hace que el coeficiente de correlación no necesariamente esté asociado a una relación con sentido [18]. Cuando eso sucede se dice que la correlación es *espúrea* o sin sentido.

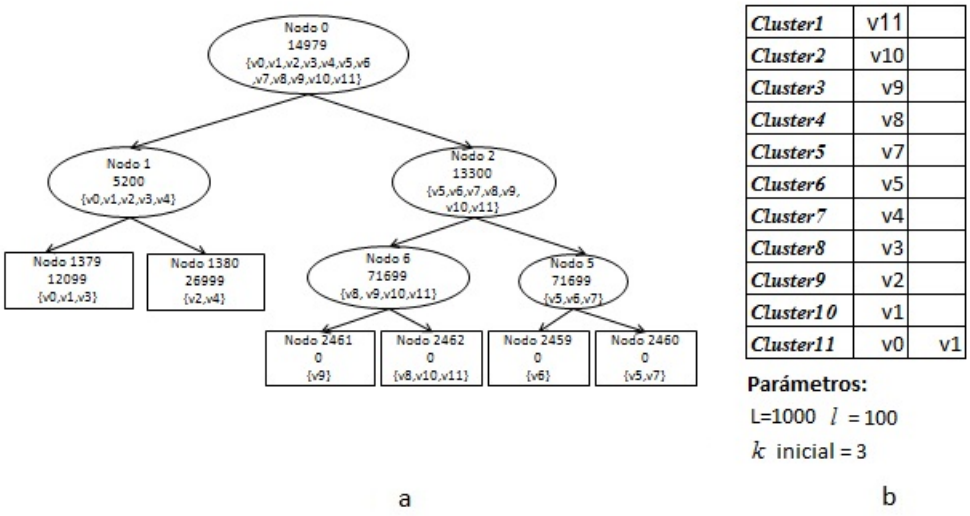

Figura 4–1: Estructura de clustering final con los algoritmos ODAC(a) y CORREL(b) para el segundo conjunto de datos artificial. El parémtro NMIN de ODAC se estableció en 1, los parámetros L y l de CORREL se establecieron en 1000 y 100 respectivamente.

Ya que la medida de similaridad utilizada para ambos algoritmos está basada en correlación es de esperarse que si existen correlaciones espúreas afecten el resultado del clustering. La diferencias de primer orden de las series de tiempo eliminan el componente tendencial. Al correr el algoritmo CORREL sobre las diferencias de primer orden de las series de este conjunto de datos, se pudo observar que el algoritmo descubre la estructura de clusters subyacente después de un promedio de 30 ejemplos. ODAC siempre trabaja sobre las diferencias de primer orden, siendo esto parte de su definición.

4.3.3 Conjunto de datos 3

para este conjunto de datos, las series de tiempo se generaron utilizando la función PCU de la biblioteca *simecol* del lenguaje **R**.

La función $PCU\,(X, rho)$ genera un vector de valores aleatorios distribuidos uniformemente correlacionados con el vector X en base al coeficiente de correlación de Pearson rho. Los vectores en base a los que se crearon cada uno de los clusters se generaron de forma pseudoaleatoria bajo una distribución uniforme.

Se generaron 12 variables con 100000 ejemplos cada una, con una estructura subyacente de 3 clusters.

$\{v0, v1, v2, v3, v4\}$, $\{v5, v6, v7, v8\}$,$\{v9, v10, v11\}$

Ambos algoritmos se corrieron con una ventana de tiempo 100. En ODAC se estableció NMIN en 100 y en CORREL los parámetros L y l se establecieron en 1000 y 100 respectivamente.

ODAC descubrió la estructura de clusters predefinida después de 30000 ejemplos (15%), posteriormente los clusters se subdividen pero se mantienen. Por su parte CORREL obtuvo la estructura de clusters predefinida después de 1000 ejemplos (0.5%), es decir, desde el primer segmento, igualmente después se mantiene la estructura de clusters estable.

La Figura 4–2 muestra el resultado final del clustering obtenido con ODAC.

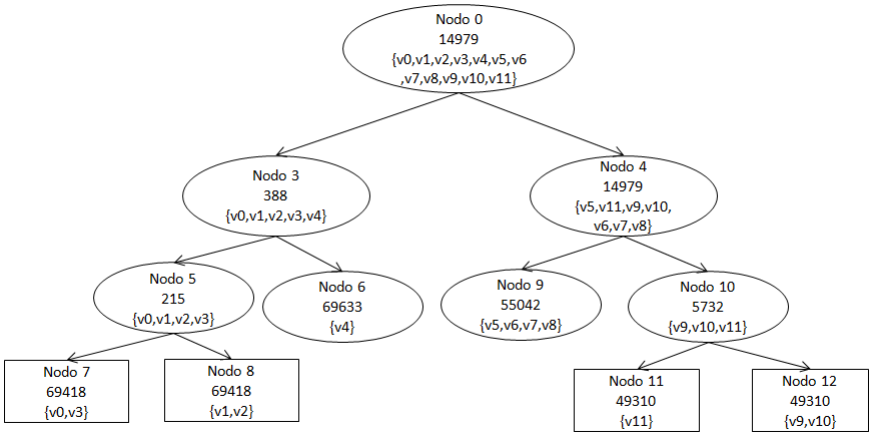

Figura 4–2: Estructura de clustering final con el algoritmo ODAC para el conjunto de datos 3

4.4 Evaluación de la capacidad de detección de cambios

4.4.1 Conjunto de datos 4

Este conjunto de datos se elaboró en base al conjuntos de datos 1 (prueba 2) y al conjunto de datos 2, por lo que posee 12 variables. El objetivo de este conjunto

de datos es evaluar el comportamiento de los algoritmos frente a cambios en la estructura de los clusters en el tiempo.

Para esta prueba, se creó un conjunto de datos de 200000 ejemplos. Los primeros 50000 ejemplos provienen del conjunto de datos 1 y los restantes 150000 ejemplos son las diferencias de primer orden de 150000 ejemplos provenientes del conjunto de datos 2. De esta manera este conjunto de datos posee una estructura de clusters para los primeros 50000 ejemplos y otra diferente para los restantes.

En este conjunto de datos las pruebas se realizaron similarmente a las anteriores con una ventana de tiempo de 100 ejemplos. Para ODAC se estableció NMIN en 100 y para CORREL L en 1000 y l en 100.

CORREL descubre la primera estructura después de un promedio de 30 ejemplos y la segunda en el ejemplo 51000, es decir, inmediatamente que existe el cambio de estructura.

Por su parte ODAC descubre la primera estructura de clusters después de 35400 ejemplos. En el ejemplo 96900 elimina toda estructura formando un slo cluster con todas las variables. Finalmente descubre la estructura 2 en el ejemplo 156800. Como se puede observar ODAC necesita de una gran cantidad de ejemplos para descubrir el cambio a la segunda estructura de clusters ya que en las pruebas realizadas en la sección4.3.2 ODAC descubrió la estructura de clusters para el conjunto de datos 2 después de 28300 ejemplos. Para verificar que esta demora se debe a la detección de cambios se realizó una segunda prueba para la cual se tomó 50000 ejemplos del conjunto de datos 2 para la primera parte y 150000 ejemplos del conjunto de datos 1, es decir, invirtiendo las estructuras de clusters subyacentes en el conjunto.

En este caso ODAC descubre la primera estructura de clusters después de 30000 ejemplos. En el ejemplo 164300 descubre dos clusters, uno es uno de los predefinidos en la segunda estructura, el otro es la unin de los otros dos. Aunque sea una estructura de dos clusters esta es vlida. En el ejemplo 149300 elimina toda estructura

formando un slo cluster con todas las variables. Finalmente descubre la estructura 2 en el ejemplo 194200.

Se puede observar entonces que ODAC necesita menos ejemplos para descubrir la primera estructura de clusters subyacente en el conjunto de datos que para detectar un cambio en la estructura.

4.5 Evaluación de la eficiencia de los algoritmos

4.5.1 Conjunto de datos 5

Este conjunto de datos se generó con el objetivo de evaluar la eficiencia de los algoritmmos, tomando como medida de eficiencia el tiempo de ejecución de los algoritmos en función del número de ejemplos. Las series de tiempo de este conjunto de datos se generaron bajo el modelo Random Walk, consta de 100 series de tiempo cada una con 200000 ejemplos. Se realizaron 20 corridas variando el núnero de ejemplos desde 10000 hasta los 200000 con un incremento de 10000 ejemplos en cada nueva corrida. Ambos algoritmos se corrieron con una ventana de tiempo 100. En ODAC se estableció NMIN en 100 y en CORREL los parámetros L y l se establecieron en 1000 y 100 respectivamente. Como se observa en la Figura 4–3 el tiempo de ejecución de ambos algoritmos se incrementa linealmente en función del número de ejemplos, sin embargo la función lineal correspondiente al tiempo de ejecución del algoritmo ODAC tiene una tasa de crecimiento de 18.62% mientras que la de CORREL de 43.20%. Así el algoritmo CORREL presenta un tiempo de ejecución mayor que ODAC en todo momento, siendo ODAC 35.94% en promedio más rápido que CORREL. Esto se puede concluir de las medidas tomadas las cuales se muestran en la Figura 4–4.

4.6 Evaluación en datos reales

4.6.1 Conjunto de Datos Fisiológicos

Este conjunto de datos se hizo disponible para el Concurso de Modelaje de Datos Fisiológicos *PDMC* por sus siglas en inglés (Physiological Data Modeling Contest)

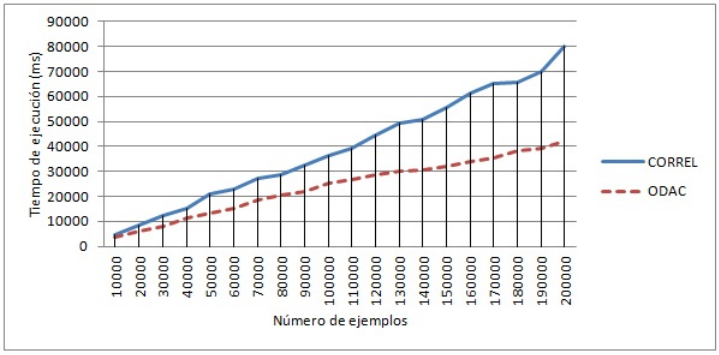

Figura 4–3: Comparación de la escalabilidad de los algoritmos CORREL y ODAC para el Conjunto de Datos 5.

llevado a cabo en la conferencia internacional de "Machine Learning" (ICML) en el año 2004. Los datos pertenecen a la compañía *BodyMedia*. Fueron recolectados de personas que llevaban puestos monitores portátiles de *BodyMedia* mientras hacían sus actividades cotidianas. Los monitores tomaban las medidas de aceleración, flujo de calor, respuesta galvánica de la piel, temperatura de la piel y temperatura corporal. En este trabajo se utilizó el conjunto de datos de entrenamiento. Consta de alrededor de 10000 horas de registro, un registro de datos por cada minuto. Cada registro consta de 16 campos, de los cuales 9 corresponden a datos fisiológicos. Al igual que en [26] nos concentraremos en los sensores del 2 al 9, extrayendo los datos por usuario.

Comparación de los resultados

Se realizó una comparación de los resultados de los algoritmos ODAC y CORREL para los usuarios 6 y 25 (UserID=6, UserID=25). Se estableció NMIN para ODAC en 1 y el tamaño de la ventana para CORREL en 1000 con ventanas básicas de tamaño 150. En este caso los resultados se comparan también con los obtenidos utilizando el algoritmo de *batch* clustering K-means. Este algoritmo se corrió con diferentes valores para el número de clusters (k) obteniéndose el mejor resultado con 3 clusters. Se utilizó la función "kmeans" de Matlab. Los resultados de las medidas de calidad se muestran en las Tablas 4–4 y para K-means y 4–5 para CORREL y

Nro de ejemplos	CORREL		ODAC	
	t (ms)	t(ms)/ ejemplo	t(ms)	t(ms)/ ejemplo
10000	4558	0.4558	3441	0.3441
20000	8579	0.4290	6178	0.3089
30000	12258	0.4086	7774	0.2591
40000	14968	0.3742	11308	0.2827
50000	20767	0.4153	13124	0.2625
60000	23041	0.3840	15122	0.2520
70000	27189	0.3884	18509	0.2644
80000	28709	0.3589	20449	0.2556
90000	32493	0.3610	21824	0.2425
100000	36365	0.3637	25103	0.2510
110000	39004	0.3546	26563	0.2415
120000	44588	0.3716	28569	0.2381
130000	49067	0.3774	29951	0.2304
140000	50650	0.3618	30374	0.2170
150000	55472	0.3698	32184	0.2146
160000	61217	0.3826	34092	0.2131
170000	65225	0.3837	35584	0.2093
180000	65645	0.3647	38122	0.2118
190000	69974	0.3683	39008	0.2053
200000	80010	0.4001	42308	0.2115
Tiempo de ejecución promedio por ejemplo		**0.3837**		**0.2458**

Figura 4–4: Tiempo de ejecución de los algoritmos CORREL y ODAC para el Conjunto de Datos 5.

ODAC. El resultado final para cada uno de los algoritmos se muestra en la 4–5, en la Figura se muestra el resultado de K-means con tres clusters por ser el que arrojó mejores resultados en las medidas de calidad.

Cluster 1	s7		
Cluster 2	s4	s8	s6
Cluster 3	s2		
Cluster 4	s9		
Cluster 5	s3	s5	

Clustering con ODAC

(a)

Cluster 1	s7	s9	
Cluster 2	s4	s8	s6
Cluster 3	s3	s5	
Cluster 4	s2		

Clustering con CORREL

(b)

Cluster 1	s2	s4	s6	s7	s8
Cluster 2	s3	s5			
Cluster 3	s9				

Clustering con K-means

(c)

Figura 4–5: Resultados experimentales con el Conjunto de Datos Fisiológicos para el usuario 6: (a)Clustering obtenido con ODAC, estable a partir de los 16000 ejemplos hasta el final. La misma estructura es obtenida con CORREL hasta los 64000 ejemplos. (b)Clustering obtenido con CORREL a partir de los 64000 ejemplos. (c)Clustering obtenido con K-means

Según los resultados de las medidas de validación para el algoritmo K-means, para los datos correspondientes al usuario 6, el número de clusters óptimo sería de tres. CORREL consigue el mejor resultado según el índice Silhouette, sin embargo

k	S	DVI	Γ_D
2	0.3346	0.9349	0.2123
3	0.3887	1.1029	0.2142
4	0.3595	0.9213	0.1424
5	0.2858	0.4055	0.0804

Tabla 4–2: Resultados de las medidas de calidad para el Conjunto de Datos Fisiológicos para el usuario 6 con K-means.

	S	DVI	Γ_D
ODAC	0.2858	0.7509	0.1328
CORREL	0.3090	0.6809	0.1225

Tabla 4–3: Resultados de las medidas de calidad para el Conjunto de Datos Fisiológicos para el usuario 6 con ODAC y CORREL.

según el índice de Dunn y el índice de Hubert con matriz de distancia, el mejor resultado está dado por k-means, seguido por CORREL.

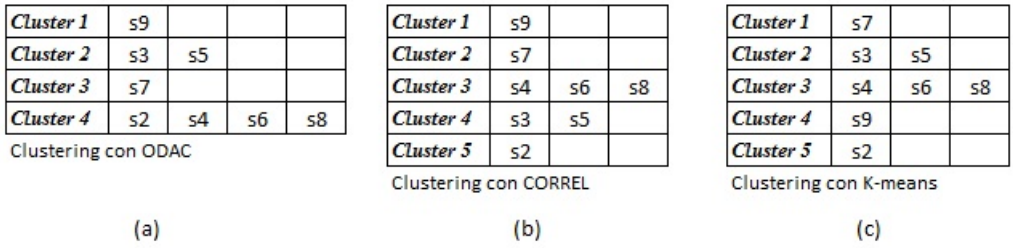

Figura 4–6: Resultados experimentales con el Conjunto de Datos Fisiológicos para el usuario 25: (a)Estructura de clustering obtenida con ODAC, estable a partir de los 16000 ejemplos hasta el final. (b)Estructura de clustering obtenida con CORREL a partir de los 3000 ejemplos. (c)Clustering obtenido con K-means.

Según los resultados de las medidas de validación para el algoritmo K-means, para los datos correspondientes al usuario 25, el número de clusters óptimo sería de cinco.

En este caso K-Means y CORREL obtienen el mismo resultado para el índice de Dunn, siendo este mejor que para ODAC. Sin embargo según el índice Silueta

k	S	DVI	Γ_D
2	0.4321	1.0782	0.2647
3	0.3869	1.0665	0.2518
4	0.2814	0.7006	0.1425
5	0.2300	1.5029	0.1289

Tabla 4–4: Resultados de las medidas de calidad para el Conjunto de Datos Fisiológicos para el usuario 25 con K-means.

	S	DVI	Γ_D
ODAC	0.2909	1.2923	0.2010
CORREL	0.2300	1.5029	0.1289

Tabla 4–5: Resultados de las medidas de calidad para el Conjunto de Datos Fisiológicos para el usuario 25 con ODAC y CORREL.

y el índice de Hubert con matriz de distancia K-Means obtiene mejores resultados, seguido por ODAC y por último CORREL.

Su pudo observar que en este conjunto de datos, la estructura del clustering es más estable, para ODAC. Los cambios en la estructura no llevan a la inicial (un cluster) en ningún momento. En CORREL surgen cambios en la estructura en algunos instantes de tiempo.

4.6.2 Temperaturas en ciudades del mundo

Este conjunto de datos se tomó del Archivo de Temperaturas Promedio Diarias de la Universidad de Dayton. El sitio contiene arhivos de temperaturas promedio de 157 ciudades de Estados Unidos y 167 ciudades internacionales, con datos desde 1995 hasta la fecha. La fuente de datos del sitio es el Centro Nacional de Datos Climáticos de los Estados Unidos (National Climatic Data Center). Se realizó una selección de 19 ciudades de diferentes continentes, las cuales se muestran en la Figura 4–7. Cada variable contiene 6161 observaciones.

Región	Nombre archivo	Ciudad	Estado	País	Missing Values
Africa	EGCAIRO	Cairo		Egypt	19
South/Central America & Caribbean	BANASSAU	Nassau		Bahamas	84
	AGBUENOS	Buenos Aires		Argentina	19
	PRLIMA	Lima		Peru	20
	COBOGOTA	Bogota		Colombia	52
	CUHAVANA	Havana		Cuba	153
	BRBRGTWN	Bridgetown		Barbados	61
North America	ILCHICAG	Chicago	Illinois	EEUU	17
	MABOSTON	Boston	Massachusetts	EEUU	22
	NYNEWYOR	New York City	New York	EEUU	20
	CNQUEBEC	Quebec		Canada	54
Asia	JPTOKYO	Tokyo		Japón	12
	KOSEOUL	Seoul		South Korea	14
Australia/South Pacific	AUMELBRN	Sydney		Australia	17
Europe	OSVIENNA	Vienna		Austria	12
	DLMUNICH	Munich		Alemania	60
	IYROME	Rome		Italia	14
	GRATHENS	Athens		Grecia	34
	URKIEV	Kiev		Ucrania	17

Figura 4–7: 19 ciudades de diferentes continentes seleccionadas para las pruebas. En la última columna se muestran la cantidad de valores perdidos para cada ciudad (variable)

Debido a la presencia de valores perdidos en el conjunto de datos, se implementó un procedimiento en cada uno de los programas para imputar los valores perdidos en el momento de actualizar las estadísticas. La imputación es por la media de los datos que llegaron hasta el momento dentro de la ventana de tiempo actual.

Los resultados obtenidos por CORREL se muestran en la Figura 4–8, los resultados de las medidas de validación se muestran en la Tabla 4–6. Asimismo también se corrió el algoritmo K-means con diferentes valores para el número de clusters. Los resultados se muestran en las Figura 4–9 y en la Tabla 4–6 y en la Tabla 4–7.

Figura 4–8: Estructura final que obtiene el algoritmo CORREL para el conjunto de datos de Temperaturas en Ciudades

Figura 4–9: Estructura final que obtiene el algoritmo K-means para el conjunto de datos de Temperaturas en Ciudades

	S	DVI	Γ_D
CORREL	-0.09752	0.32529	0.24231

Tabla 4–6: Resultados de las medidas de calidad para el Conjunto de Datos de Temperaturas en Ciudades con el algoritmo CORREL.

El algoritmo ODAC mantiene el cluster inicial compuesto por todas las variables.

Según las medidas de validación, CORREL obtiene resultados más altos que K-Means para el índice de Dunn y el índice de Hubert con matriz de distancia. Mientras que para el índice Silueta obtniene un resultado más bajo.

k	S	DVI	Γ_D
2	0.11131	0.30187	0.25312
3	0.05577	0.30187	0.21571
4	0.03116	0.30187	0.20901
5	-0.02993	0.24720	0.10278
6	-0.01302	0.30187	0.17255
7	-0.02025	0.30187	0.17042

Tabla 4–7: Resultados de las medidas de calidad para el Conjunto de Datos Temperaturas en Ciudades con el algoritmo K-means.

4.7 Discusión de resultados

- CORREL obtiene la estructura de clusters desde el primer segmento (después de menos de 1000 ejemplos) en los conjuntos de datos 1 y 3 mostrando un excelente comportamiento. Sin embargo se debe notar que el conjunto de datos 1 se generó bajo la distribución de probabilidad normal y el conjunto de datos 3 bajo la distribución de probabilidad uniforme. CORREL utiliza el algoritmo K-means para realizar el clustering en una ventana de tiempo por lo tanto es de esperarse que este algoritmo se comporte bien para estas distribuciones. Por otro lado para que CORREL descubriera la estructura de clusters para el conjunto de datos 2, para el cual las series de tiempo fueron generadas con el modelo *Random Walk*, fue necesario preprocesar los datos. Aplicando CORREL sobre las diferencias de primer orden, se descubre la estructura de clusters despus de un promedio de 20 ejemplos.
- Por su parte ODAC logró descubrir la estructura de clusters en todos los conjuntos de datos. Aunque se pudo observar que ODAC necesita de muchos ejemplos para descubrir la estructura de clusters subyacente. En los conjuntos de datos de 12 variables y tres clusters ODAC descubre la estructura de clusters después de un promedio de 29433 ejemplos. Mientras que para el conjunto de datos 1-prueba 1 lo hace después de 137667, es decir, que requiere de un 54% más de

ejemplos. De igual forma en el conjunto de datos Fisiológicos ODAC consigue una estructura estable a partir de los 160000 ejemplos.

- Con respecto a la detección de cambios en la estructura ODAC, se pudo observar que ODAC necesita de un gran número de ejemplos para descubrir los cambios. Para descubrir una nueva estructura es necesario que los clusters se vuelvan a unir mediante agregaciones hasta volver posiblemente a una estructura de un cluster conteniendo todas las variables y a partir de entonces descubrir la nueva estructura mediante divisiones. Por lo que es de esperarse que ODAC necesite una mayor cantidad de ejemplos. Esto se debe a que el criterio de división está basado en niveles de confianza dados por la Cota de Hoeffding, por lo que require muchos ejemplos para aceptar una división. Asimismo cada vez que se realiza una divisíon las estadísticas se inicializan en las nuevas hojas por lo que se requerirá una gran cantidad de ejemplos nuevamente para que el algoritmo acepte una nueva división. Por otro lado el diseño de CORREL le permite detectar el cambio casi inmediatamente.
- En los conjuntos de datos Fisiológicos ambos algoritmos tuvieron un buen comportamiento. ODAC obtuvo valores mayores para las medidas de calidad aunque no con una gran diferencia sobre CORREL.
- En el conjunto de datos de Temperaturas en Ciudades CORREL obtuvo una estructura de clusters razonable teniendo en cuenta que los clusters resultantes agrupan ciudades de zonas geográficas similares. Aunque los resultados de las medidas de validación arrojaron valores bajos.
- En cuanto al tiempo de ejecucin ambos algoritmos presentan una tendencia lineal como se observó en el análisis de complejidad, sin embargo ODAC presenta un menor tiempo de ejecucin siendo en promedio 35.94% más rápido.

- Dado que el conjunto de datos de temperaturas en ciudades consta de 19 variables con tan sólo 6161 observaciones, ODAC no pudo descubrir ninguna estructura de clusters. Con CORREL la eleccin de diferentes tamaos de ventana gener resultado muy variados. Se reportó el resultado que obtuvo el mejor resultado.
- ODAC no tiene un tiempo conocido de retraso en la respuesta a diferencia de CORREL y en general de los algoritmos que trabajan con el modelo *Sliding Window* que el tiempo de retraso est dado por el tamao de la ventana (en el caso de los algoritmos que utilizan ventanas bsicas, el tiempo de retraso estara dado por el tamao de la ventana bsica). Sin embargo la ventana de tiempo de CORREL es un valor definido por el usuario a priori lo cual no pude asegurar necesariamente una buena eleccin.

CAPÍTULO 5
Conclusiones

5.1 Introducción

Este trabajo presenta un estudio comparativo de los algoritmos CORREL y ODAC detinados al clustering de *Data Streams* por variables. Ambos siguen estrategias diferentes para abordar el problema. CORREL es un sistema de dos fases que opera sobre Sliding Windows. La fase en línea está destinada a almacenar las estadísticas necesarias para calcular la medida de proximidad y la fase fuera de línea está destinada a realizar el clustering utilizando el algoritmo de particionamiento K-Means que en este trabajo fue reemplazado por K-Medoids. Por otro lado, ODAC es un sistema jerárquico divisivo que construye un árbol binario de clusters siguiendo un enfoque divisivo basado en la correlación entre las series de tiempo.

5.2 Conclusiones

- ODAC muestra un mejor comportamiento en cuanto a eficacia, descubriendo la estructura de clusters en todos los conjuntos de datos sobre los que se corrió, y en tiempo de ejecución mostrando .
- Asimismo ODAC posee la ventaja de no requerir parmetros de entrada crticos como el tamao de la ventana adaptndose dinmicamente. La ventana de tiempo de CORREl es un valor definido por el usuario a priori por lo que una buena eleccin de este parámetro no está asegurada.
- ODAC necesita una gran cantidad de ejemplos para descubrir la estructura de clusters y ms an para detectar los cambios en la estructura de clusters. La definición de criterios de división alternativos, sería un tema a abordar.

- Da la impresión de que resulta conveniente realizar algn pre-procesamiento sobre los datos, sin embargo este preprocesamiento posiblemente dependa de las características específicas del conjunto de datos y no se pueda generalizar o para determinadas series de tiempo se necesite un preprocesamiento más sofisticado.
- Trabajar con las diferencias de primer orden no daa los resultados de los conjuntos de datos artificiales ni los resultados del conjunto de datos fisiolgicos.

5.3 Trabajos futuros

- Realizar una comparación más amplia incluyendo algoritmos de dos fases que realicen preprocesamiento en la recoleccin de estadsticas. Estudiar el efecto del preprocesamiento de los datos en la complejidad, desempeo, y calidad de los resultados de los algoritmos.
- Hacer un estudio más exhaustivo de la influencia del tamaño de la ventana en los algoritmos que utilizan el modelo *Sliding Window* tanto sobre la calidad de los resultados del clustering como sobre el desempeño de los algoritmos, tanto con ventanas de tamaño fijo como variable.
- Estudiar el efecto del coeficiente de atenuación en los algoritmos de dos fases.
- Proponer alternativas al criterio de división del algoritmo ODAC para mejorar su desempeño.

REFERENCIAS BIBLIOGRÁFICAS

[1] D. Wunsch y R. Xu. *Clustering*. Wiley-IEEE Press, 2009.

[2] Y. Zhu y D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *In VLDB*, pages 358–369, 2002.

[3] A. Jain y R. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[4] A. P. Dempster, N. M. Laird, y D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[5] MOA Massive Online Analysis. http://moa.cs.waikato.ac.nz/.

[6] B. Dai, J. Huang, M. Yeh, y M. Chen. Adaptive clustering for multiple evolving streams. *IEEE Trans. on Knowl. and Data Eng.*, 18:1166–1180, Septiembre 2006.

[7] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[8] C. Aggarwal, J. Han, J. Wang y P. Yu. A framework for clustering evolving data streams. In *In VLDB*, pages 81–92, 2003.

[9] C. Aggarwal, J. Han, J. Wang y P. Yu. A framework for projected clustering of high dimensional data streams. *Science*, pages 852–863, 2004.

[10] F. Cao, M. Ester, W. Qian, y A. Zhou. Density-based clustering over an evolving data stream with noise. In *In 2006 SIAM Conference on Data Mining*, pages 328–339, 2006.

[11] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2:139–172, Septiembre 1987.

[12] J. Gama. *Knowledge Discovery from Data Streams.* Chapman and Hall/CRC, 2010.

[13] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar 1963.

[14] J. Beringer y E. Hllermeier. Online clustering of parallel data streams. *Data Knowl. Eng.*, 58:180–204, Agosto 2006.

[15] J. Gama y P.o Medas. Learning decision trees from dynamic data streams. *j-jucs*, 11(8):1353–1366, 2005.

[16] K. Udommanetanakit, T. Rakthanmanon, y K. Waiya- mai. E-stream: Evolution-based technique for stream clustering. In Reda Alhajj, Hong Gao, Xue Li, Jianzhong Li, and Osmar Zaane, editors, *Advanced Data Mining and Applications*, volume 4632 of *Lecture Notes in Computer Science*, pages 605–615. Springer Berlin / Heidelberg, 2007.

[17] L. Tu, L. Chen, y L. Zou. Clustering multiple data streams based on correlation analysis. *Journal of Software*, 20:1756–1767, Julio 2009.

[18] Erick Lahura. El coeficiente de correlación y correlaciones espúreas. 2003.

[19] M. Brun, C. Sima, J. Hua, J. Lowey, B. Carroll, E. Suh, y E. Dougherty. Model-based evaluation of clustering validation measures. *Pattern Recogn.*, 40:807–824, Marzo 2007.

[20] M. Datar y R. Motwani. The sliding - window computation model and results. In *Data Streams Models and Algorithms*, pages 149–167, New York, NY, USA, 2007. Springer.

[21] M. Ester, H. Kriegel, S. Jörg, y X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231, 1996.

[22] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[23] Lloyd Stuart P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[24] P. Domingos y G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.

[25] P. Pereira , J. Gama, y L. Lopes. Requirements for clustering streaming sensors. *Knowledge Discovery from Sensor Data*, pages 35–53, 2008.

[26] P. Pereira, J. Gama y J. Pedroso. Odac: Hierarchical clustering of time series data streams. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):615–627, 2008.

[27] P. Pereira Rodrigues. *Hierarchical Clustering of Time Series Data Streams*. PhD thesis, Universidade do Porto, Portugal, 2005.

[28] S. Guha, R. Rastogi, y K. Shim. Cure: an efficient clustering algorithm for large databases. *SIGMOD Rec.*, 27:73–84, Junio 1998.

[29] E. Schumann. Generating correlated normal variates. http://comisef.wikidot.com/tutorial:correlation. 2011.

[30] T. Zhang, R. Ramakrishnan, y M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25:103–114, Junio 1996.

[31] W. Fan, Toyohide Watanabe, y Koichi Asakura. A framework for flexible clustering of multiple evolving data streams. *Int. J. Adv. Intell. Paradigms*, 1:178–195, Abril 2008.

[32] Jiong Yang. Dynamic clustering of evolving streams with a single pass. *Data Engineering, International Conference on*, 0:695, 2003.

COMPARACIÓN DE ALGORITMOS PARA CLUSTERING DE "STREAMS" DE SERIES DE TIEMPO

Ana María Aparicio Carrasco
(787) 370-9866
Departamento de Ciencias Matemáticas
Consejero: Edgar Acuña Fernández
Grado: Maestría en Ciencias
Fecha de Graduacion: Mayo 2012

En años recientes, los avances tecnológicos han dado lugar a un enorme incremento en la producción de datos y han facilitado su recolección. Los datos que llegan continuamente, de forma masiva y con tendencia infinita se conocen como *data streams*. Sus fuentes son por ejemplo sensores, transacciones bancarias y mediciónes automatizadas. Los algoritmos destinados a procesar estos datos deben brindar respuestas rápidas y en tiempo real, lo que implica mantener un modelo de decisión en todo momento. El clustering de *data streams* por variables busca encontrar grupos de *data streams* (variables) con un comportamiento similar a lo largo del tiempo. En el presente trabajo se comparan dos enfoques de algoritmos de clustering de *data streams* por variables: ODAC, un algoritmo jerárquico divisivo y CORREL que utiliza el modelo *Sliding Windows* y realiza el clustering por particionamiento. Basado en el estudio experimental se concluye que ODAC supera a CORREL, debido a su rendimiento e independencia de la distribución de las variables. Sin embargo, este requiere que el conjunto de datos posea una una gran cantidad de observaciones (ejemplos) para descubrir la estructura de clusters subyacente.