RNPS No. 2143
ISSN 2072-6260
Versión Digital

SERIE GRIS

REPORTE TÉCNICO
Minería de Datos

Minería de flujos de datos a altas velocidades

Ing. Carlos Javiel Moya Cribeiro,
Dr. C. José Hernández Palancar

RT_012 **abril 2010**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2143
ISSN 2072-6260
Versión Digital

SERIE GRIS

REPORTE TÉCNICO
Minería de Datos

Minería de flujos de datos a altas velocidades

Ing. Carlos Javiel Moya Cribeiro,
Dr. C. José Hernández Palancar

RT_012 **abril 2010**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Índice

Minería de flujos de datos a altas velocidades

Ing. Carlos Javiel Moya Cribeiro, Dr. C. José Hernández Palancar

Centro de Aplicaciones de Tecnología de Avanzada, 7a #21812 e/ 218 y 222, Siboney, Playa, Ciudad de La Habana, Cuba
cmoya@cenatav.co.cu

RT_012 CENATAV
Fecha del camera ready: 26 de marzo de 2010

Resumen: En los últimos años, una gran cantidad de flujos de datos potencialmente infinitos son a menudo generados por sistemas en tiempo real, redes de comunicaciones, experimentos científicos, internet, etc. El análisis y procesamiento de estos flujos de datos, así como, la minería y gestión de los mismos, constituyen temas de investigación activos. En este trabajo se ofrece una panorámica general sobre el estado del arte en dos áreas de investigación estrechamente relacionadas: procesamiento y minería de flujos de datos. Se analizan los métodos y algoritmos básicos en el procesamiento de flujos de datos y se plantean los problemas generales a los cuales se enfrenta cualquiera de las técnicas de procesamiento actualmente utilizadas. Además, se realiza un estudio sobre los problemas particulares en materia de minería de flujo de datos, prestándole mayor atención a las técnicas más utilizadas en este contexto: agrupamiento, clasificación, detección de patrones frecuentes y detección de cambios en flujo de datos.

Palabras clave: flujo de datos, procesamiento de flujos de datos, minería de flujos de datos, agrupamiento, clasificación, extracción de patrones frecuentes, detección de cambios

Abstract: In recent years, a huge amount of potentially infinite data streams are often generated by real time systems, communications networks, scientific experiments, internet, etc. The analysis and processing of these data streams as well as their mining and management, constitute active areas of researches. This report presents a general overview on state of the art in two closely related areas: data stream processing and data stream mining. In this report basic methods and algorithms in data stream processing are analyzed. Besides, a study on particular problems about the subject of data stream mining is presented, centering higher attention to most used techniques in this context: clustering, classification, frequent pattern mining and changes detection.

Keywords: Data Stream, Data Stream Processing, Data Stream Mining, Clustering, Classification, Frequent Pattern Mining, Change Detection

1 Introducción

En los últimos años, la gestión y el procesamiento de flujos de datos se ha convertido en un tema de investigación activo en varios campos de las ciencias computacionales como: minería de datos, bases de datos, sistemas distribuidos, etc. Un flujo de datos es considerado como una secuencia ordenada de elementos que llegan de manera continua con el progreso del tiempo. Debido a la naturaleza infinita de los flujos de datos, es imposible mantener todos sus elementos en alguna forma de memoria para luego procesarlos, como consecuencia, cada elemento debe ser procesado en una sola etapa, de forma que no sea necesario mantenerlo en el sistema.

Un sistema de flujos de datos puede producir constantemente, enormes cantidades de datos. Aspectos como el almacenamiento, gestión y el procesamiento de datos que llegan de manera continua en forma de flujos múltiples, rápidos y variables en el tiempo, imponen nuevos retos y problemas de interés para la investigación. Existen diversas aplicaciones en las cuales son

producidos flujos de datos con estas características: monitoreo de redes, sistemas de telecomunicaciones, cualquier tipo de sistema distribuido multi-sensor, etc.

La literatura ha abordado el problema del análisis y procesamiento de los flujos de datos de múltiples maneras. A modo de consenso, para este trabajo dividimos el estudio de los flujos de datos en tres áreas muy relacionadas: procesamiento, gestión y minería de flujos de datos. Como muestra la figura (fig. 1), la gestión y minería serán consideradas formas o tareas dentro del procesamiento de flujos y cada una de estas áreas estará presente en la arquitectura de cualquier Sistema de Gestión de Flujos de Datos (SGFD).

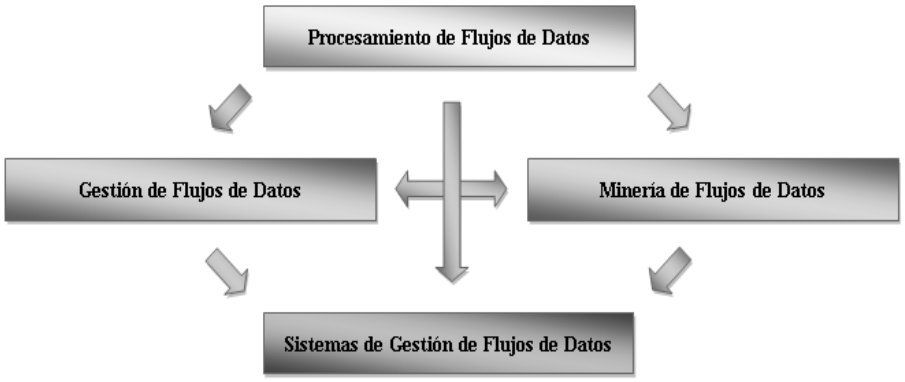

Fig. 1. Áreas de estudio en Flujos de Datos

Este trabajo persigue como objetivo proveer al lector de una visión general sobre el estado del arte en solo dos de las áreas antes mencionadas: procesamiento y minería de flujos de datos. Sin embargo, es difícil separar a cualquiera de estas áreas para un estudio, la relación es estrecha, y en algunas secciones de este trabajo tocaremos aspectos de las restantes. Para llevar a cabo este objetivo, el documento está dividido de la siguiente manera. En el epígrafe 2, se hace un estudio de los métodos y algoritmos básicos en el procesamiento de flujos de datos y se plantean los problemas generales a los cuales se enfrenta cualquiera de las técnicas de procesamiento actualmente utilizadas. El epígrafe 3 está enfocado a los problemas de minería de flujo de datos, prestándole mayor atención a las técnicas más utilizadas en este contexto: agrupamiento, clasificación, detección de patrones frecuentes y detección de cambios en flujo de datos. Las conclusiones de este trabajo son mostradas en el epígrafe 4.

2 Procesamiento de flujos de datos

Para muchas de las más recientes aplicaciones, el concepto de flujo de datos es más apropiado que el de conjunto o base de datos. Naturalmente un conjunto de datos almacenado, es un modelo apropiado cuando porciones significativas de los datos son consultadas una y otra vez, y actualizaciones de los mismos no ocurren o son relativamente infrecuentes. Por el contrario, un flujo de datos es un modelo apropiado cuando grandes volúmenes de datos llegan de forma continua y es imposible o poco práctico almacenar dichos datos en alguna forma de memoria para su posterior procesamiento[1].

El concepto o modelo de flujo de datos difiere del modelo tradicional de conjunto de datos en los siguientes aspectos[2, 3].

- ✓ Los elementos en un flujo de datos llegan de manera inmediata.

- ✓ El sistema no tiene control sobre el orden en que llegan los elementos, por lo cual, el acceso a los mismos para su procesamiento no puede ser de manera aleatoria.
- ✓ Los flujos de datos son potencialmente de tamaño ilimitado.
- ✓ Los elementos que han sido procesados pueden ser descartados, o almacenados en el menor de los casos. La recuperación de elementos ya procesados no es sencilla partiendo de que para lograrlo, estos deben estar de forma persistente en memoria, la cual se considera pequeña en relación al tamaño del flujo de datos.
- ✓ Los elementos de un flujo de datos pueden provenir de varias fuentes de datos en un entorno distribuido.
- ✓ Debido a las restricciones de memoria y tiempo para el procesamiento de los datos, los resultados obtenidos en flujos de datos comúnmente son aproximados.

Un resumen de las diferencias entre el procesamiento de flujos de datos y el procesamiento tradicional, es mostrado en la Tabla 1.

Tabla 1. Diferencias entre el procesamiento de flujos de datos y el tradicional

	Tradicional	Flujo de Datos
Etapas de procesamiento	múltiples	una
Tiempo de procesamiento	ilimitado	restringido
Uso de la memoria	ilimitado	restringido
Tipo de resultado	exacto	aproximado
Distribuido	no	si

En el modelo de flujo de datos los elementos de entrada $a_1, a_2,\ldots\ldots a_j$, llegan de manera secuencial describiendo una función multidimensional A. Los modelos de flujos de datos pueden diferir por la forma en que cada elemento a_i describe a la función A [4]. Basándonos en lo anterior, los modelos de flujos de datos pueden ser:

Modelos estáticos: Cada elemento a_i representa el estado de la función A en el momento i, $A[i]= a_i$. En este modelo, los elementos a_i no pueden ser modificados.

Modelos incrementales: Cada elemento a_i representa un incremento a la función A en el momento j, $A[j]=A[j-1]+ a_i$ donde $A[j]$ representa el estado de la función después de haber arribado el elemento a_j.

Modelos dinámicos: Cada elemento a_i representa un incremento a la función A en el momento j, $A[j]=A[j-1]+ a_i$ donde $A[j]$ representa el estado de la función después de haber arribado el elemento a_j. La diferencia de estos modelos con los incrementales radica en que el incremento a_i puede ser negativo o positivo. Es decir, los elementos a_i pueden ser borrados o modificados.

2.1 Métodos básicos en el procesamiento de flujos de datos

Un flujo de datos es normalmente grande en longitud y dimensión, sin embargo, éstos no son los únicos problemas. El conjunto de valores posibles en una dimensión en específico puede ser también muy grande. Un ejemplo de ello lo constituye el conjunto de todas las direcciones IP en la Internet, cuyo tamaño impide su completo almacenamiento y hace de ello un problema insoluble. La ejecución de consultas que referencien datos históricos también constituye un problema. Es en este tipo de situaciones que los métodos de procesamiento de flujos de datos

son aplicables y se vuelven imprescindibles. Los métodos de procesamiento de flujos de datos proveen respuestas aproximadas usando recursos reducidos.

Es poco práctico almacenar todos los datos provenientes de un flujo para luego ejecutar las consultas pertinentes. Buscar una rápida, aunque aproximada respuesta, como el almacenamiento de resúmenes y sinopsis de la estructura fundamental del flujo de datos, podría ser una alternativa válida. Desde luego, existirá un compromiso entre el tamaño de los resúmenes a almacenar y la precisión de los resultados. Datar *et al.* [5] presentó un conjunto de cuestiones que representan con claridad el problema del procesamiento de flujos de datos:

- ✓ Dado un flujo de datos compuesto por bits (ceros y unos), mantener la cantidad de unos en los últimos *N* elementos presenciados del flujo de datos.
- ✓ Dado un flujo de datos compuesto por números enteros positivos en el rango *[0,……, R]*, mantener en todo momento la suma de los últimos N elementos presenciados en el flujo de datos.
- ✓ Dado un flujo de datos, buscar la cantidad de valores distintos en el dominio de valores *[0,……, N]*.

Todos estos problemas tendrían una solución exacta si se contara con memoria suficiente para almacenar todos los elementos del flujo de datos. ¿Pero cómo podrían ser solucionados si la memoria con la que se cuenta es restringida? Respuestas aproximadas serían de utilidad, si el error asociado a las mismas estuviese en un límite admisible. Se han definido métodos estadísticos de aproximación para la estimación del error como las desigualdades de Markov y Chebyshev y los límites de Chernoff y Hoffding[6].

2.1.1 Medidas estadísticas en flujos de datos

La determinación de medidas estadísticas es uno de los métodos más usados en la extracción de información de la estructura fundamental de un flujo de datos. Sin embargo, debido a la naturaleza infinita de los flujos de datos, no puede realizarse el cálculo estadístico como una tarea independiente y es necesario mezclarlo a otros métodos como el de ventanas de tiempo que reduzcan el espacio de procesamiento (ver sección 2.1.2). La media, la desviación estándar y la correlación, son algunas de las medidas más representativas en este ámbito. La importancia de estas medidas radica en que las mismas permiten mantener estadísticas exactas de una secuencia infinita de elementos sin necesidad de almacenar a cada uno de dichos elementos en la memoria del sistema.

El cálculo de la media en un flujo de datos representado como una secuencia ordenada de elementos $a_1, a_2,\ldots\ldots a_j$ esta dado por la ecuación:

$$\overline{a_i} = \frac{(i-1) \times \overline{a_{i-1}} + a_i}{i}$$

Como se observa, para actualizar el valor de la media con cada elemento que llegue del flujo de datos, sólo es necesario mantener en memoria la cantidad de elementos presenciados y la sumatoria de los mismos. Otra de las medidas estadísticas que pueden brindar información sobre el comportamiento de un flujo de datos, es la desviación estándar, dado por la ecuación:

$$\sigma_i = \sqrt{\left(\sum a_i^2 - \left(\sum a_i\right)^2 / i\right) / (i-1)}$$

En este caso, es necesario mantener la cantidad de elementos presenciados, la sumatoria de cada uno de ellos y la suma de los cuadrados de los mismos. Otra medida de gran utilidad que puede ser determinada de manera recursiva, es el coeficiente de correlación. Dado dos flujos de datos a y b, puede ser de utilidad encontrar la correlación exacta entre ambos flujos de datos. Para ello, es necesario mantener en memoria la suma de los elementos presenciados en cada flujo, la suma de los cuadrados de dichos elementos y la sumatoria del producto cruzado de los elementos de ambos flujos de datos. La correlación exacta de dos flujos de datos está dada por la ecuación:

$$\mathrm{corr}(a,b) = \frac{\sum(a_i \times b_i) - \frac{\sum a_i \times \sum b_i}{n}}{\sqrt{\sum a_i^2 - \frac{(\sum a_i)^2}{n}}\sqrt{\sum b_i^2 - \frac{(\sum b_i)^2}{n}}}$$

2.1.2 Ventanas de tiempo en flujo de datos

Las ventanas de tiempo constituyen otro de los métodos ampliamente usado en el procesamiento de flujos de datos. Con el objetivo de reducir el espacio de procesamiento, las ventanas de tiempo son comúnmente usadas en el problema de la estimación de consultas en flujos de datos en sistemas de gestión; pues en lugar de buscar las respuestas pertinentes en un flujo infinito de datos, las ventanas de tiempo permiten buscar respuestas en subconjuntos finitos de todo el flujo. No es únicamente en la gestión de flujos de datos donde son de utilidad las ventanas de tiempo. Otras aplicaciones vinculadas a la minería de flujo de datos como el agrupamiento, la clasificación y la extracción de patrones frecuentes, requieren de métodos que reduzcan la cantidad de elementos de un flujo a procesar en un momento dado.

Varios métodos de ventanas de tiempo han sido usados en la literatura, siendo los siguientes los más representativos.

Landmark Windows

En el modelo de *landmark* [7], el espacio de búsqueda y procesamiento es un intervalo de tiempo que se extiende desde un tiempo inicial *i* hasta el momento *j*, siendo *j* el tiempo actual en el flujo de datos. Ventanas sucesivas compartirán el mismo tiempo inicial e irán creciendo con el progreso del tiempo. Debido al aumento del espacio de búsqueda en el tiempo y la utilización de datos históricos del flujo de datos; este modelo es poco aplicable en algunos de los actuales SGFD donde la necesidad de procesamiento en una etapa, la velocidad y la naturaleza infinita de los flujos de datos, son problemas conocidos. Para mucho de los actuales SGFD, modelos que hagan uso de datos más recientes en el flujo de datos a procesar, tendrían mayor aplicación.

Fig. 2. Modelo landmark windows

Sliding Windows

En el modelo de ventanas deslizantes [7] solo la parte del flujo de datos acotado por la ventana de dimensiones fijas, será almacenado y procesado. En este modelo, la ventana se desplazará según fluyan los datos en el tiempo. Este tipo de ventanas sigue un comportamiento similar a una estructura de datos tipo FIFO (*first in, first out*); donde una vez observado e insertado en la

ventana el elemento *j*, otro elemento $i = (j - w)$ es descartado; siendo w el ancho de la ventana deslizante.

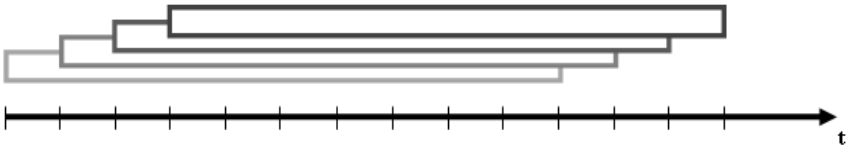

Fig. 3. Modelo sliding windows

Damped Windows

En el modelo *damped windows* [8] el intervalo de tiempo o ventana varía según fluyen los datos, disminuyendo en el tiempo en función de una razón de decrecimiento definida. En este modelo, la disminución de las ventanas en una forma casi exponencial garantiza la relevancia de los datos más recientes en el flujo; convirtiéndolo en un modelo idóneo en aplicaciones donde el efecto de los datos históricos en los resultado del procesamiento disminuya con el progreso del tiempo.

Fig. 4. Modelo damped windows

Tilted Windows

Los modelos previamente analizados le dan un tratamiento a los datos históricos de un flujo de datos que no es el apropiado en determinadas situaciones. Para estos modelos, los datos históricos pueden: estar presentes en la ventana (*landmark windows*) con el inconveniente del aumento del espacio de procesamiento en el tiempo, o simplemente no estarlo (*sliding windows*) y (*damped windows*). Para determinadas aplicaciones como las relacionadas con la minería de flujos de datos, la naturaleza evolutiva de los flujos en el tiempo constituye un problema y los elementos históricos del flujo de datos tienen repercusión en los resultados del procesamiento.

El modelo *tilted windows* utiliza una escala de tiempo comprimida. Los elementos más recientes son almacenados dentro de la ventana con un nivel de granularidad en el tiempo mayor que los elementos menos recientes. La importancia de utilizar este tipo de método de ventanas radica en que todos los elementos históricos no son descartados y se tienen en cuenta en el análisis del flujo. Por otro lado, el nivel de granularidad de los datos más recientes garantiza la relevancia de los mismos en los resultados obtenidos en la etapa de procesamiento.

El modelo *tilted windows* puede ser diseñado de varias maneras. Han y Kamber [9] presentan tres variantes posibles: *natural tilted time windows*, *logarithm tilted windows* y *progressive logarithm tilted windows* Aggarwal *et.al* [10] propone otra variante, *pyramidal time windows*. Ejemplos ilustrativos de las dos primeras variantes son mostrados en las figuras (fig. 5a, 5b) respectivamente. En la variante *natural tilted time windows* los datos son almacenados siguiendo un grado de granularidad acorde a la taxonomía natural del tiempo. En la variante *logarithm tilted windows*, dado un nivel de granularidad máxima de periodo *t*, el nivel de granularidad en el tiempo decrece en la medida que los elementos son menos recientes.

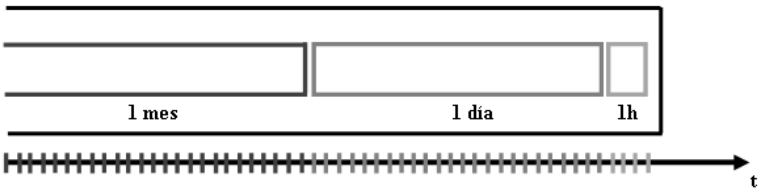

Fig. 5a. Modelo natural tilted time windows

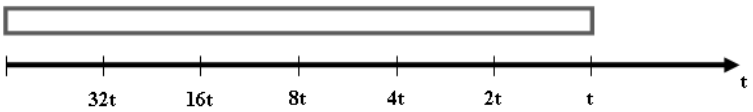

Fig. 5b. Modelo logarithm time windows

Otros modelos de ventanas de tiempo han sido diseñados de acuerdo a necesidades muy particulares de la aplicación donde han sido utilizados. Li y Gopalan [11] proponen dos modelos de ventanas para el agrupamiento de flujos de datos de naturaleza evolutiva: *equal-width time windows* y *elastic time windows*. En el modelo *equal-width time windows*, el ancho de la ventana es el mínimo nivel de granularidad deseado por una aplicación en particular. En el modelo *elastic time window* el ancho de la ventana se ajusta a los cambios que ocurren en el proceso de agrupamiento producto de la evolución de los flujos analizados en el tiempo.

2.2 Algoritmos básicos en el procesamiento de flujos de datos

Los flujos de datos imponen considerables retos a muchas de las actuales aplicaciones de bases de datos y minería de datos por los costos computacionales y de almacenamiento asociados al gran volumen de datos. En ocasiones, el cálculo de medidas estadísticas y la construcción de sinopsis de la estructura fundamental de un flujo de datos pueden ser de utilidad en la búsqueda de soluciones rápidas, aunque aproximadas, como se hacía referencia en apartados anteriores (sección 2.1). Algunos ejemplos de aplicaciones en las que tienen utilidad los algoritmos de construcción de sinopsis son mostrados a continuación:

Estimación de consultas: La estimación de consultas en flujos de datos es probablemente la aplicación en la que mayormente son usadas las técnicas de construcción de sinopsis [12]. El problema de la estimación de consultas es particularmente importante desde el punto de vista de la eficiencia, debido a la necesidad de dar repuestas inmediatas a un conjunto de consultas realizadas sobre un flujo de datos, en un momento dado. Entre los algoritmos más representativos en la construcción de sinopsis para dar solución a este problema tenemos: *sampling*, *wavelets*, *histograms* y *sketches*.

Estimación de asociaciones: La estimación de asociaciones en flujos de datos, es una de las operaciones fundamentales en la búsqueda de relaciones entre flujos de datos diferentes. Esta aplicación tiene gran utilidad en sistemas distribuidos de múltiples sensores, donde los flujos de datos provienen de diferentes fuentes de datos y requieren de algún mecanismo que los relacione. La estimación eficiente del grado de asociación constituye un problema cuando el conjunto de atributos es particularmente grande. Varios algoritmos [13, 14] han sido diseñados recientemente para la estimación eficiente de la asociación entre flujos de datos.

Minería de flujos de datos: Varias aplicaciones de minería en flujos de datos como la detección de cambios no requieren del análisis de datos individuales en un flujo, sino más bien, de una sinopsis temporal que permita una visión general del comportamiento del flujo de datos en el tiempo. Métodos como agrupamiento y *sketches* [15] pueden ser de utilidad en la detección efectiva de cambios en un flujo de datos de naturaleza evolutiva en el tiempo. De forma similar, varios métodos de clasificación [16] pueden ser usados sobre una sinopsis supervisada del flujo de datos.

El diseño y elección de un método para la construcción de sinopsis depende de manera muy particular del problema que queramos resolver. Los algoritmos utilizados en aplicaciones como la estimación de consultas, carecerán de utilidad en otras aplicaciones de minería como detección de cambios y clasificación. Además, la construcción de representaciones compactas de un flujo de datos estará condicionada por otros inconvenientes como la eficiencia y las limitaciones de tiempo y espacio de procesamiento. A continuación hacemos un estudio de los algoritmos de mayor relevancia en la construcción de sinopsis de un flujo de datos.

Sampling

El muestreo (*sampling*) [4, 10, 17] es un método de construcción de sinopsis ampliamente usado en varias aplicaciones de flujo de datos, debido a la capacidad que tiene de brindar una representación original de los elementos del flujo analizado. La técnica de muestreo consiste básicamente en ir tomando elementos de un flujo de datos a determinados intervalos de tiempo, originando una representación resumida del flujo. Debido a esto, el muestreo implica pérdida de información, influyendo así en la aproximación de los resultados.

Una de las ventajas que ofrece la técnica de muestreo, es la eficiencia, facilidad de implementación y adaptación en cualquier operación de flujos de datos. Si la velocidad de un flujo es mayor que la capacidad de procesamiento de un sistema; el muestreo puede ser utilizado como mecanismo de desaceleración virtual de los datos que llegan al sistema de procesamiento. Varios métodos de construcción de sinopsis como *wavelets*, *histograms* y *sketches* son difíciles de adaptar en el caso de flujos de datos multidimensionales; generalmente, la técnica de muestreo aleatorio es el único método a elegir en aplicaciones de alta dimensionalidad.

Wavelets

La transformada *wavelets* [6, 18] es una técnica matemática que permite representar una señal como la suma de forma de ondas más simples a diferentes escalas y posiciones. Es una técnica de fácil implementación ampliamente usada en flujos de datos como método de descomposición jerárquica y construcción de sinopsis de la estructura fundamental de los datos de un flujo. La técnica de *wavelets* consiste básicamente en crear una descomposición de las características fundamentales de una señal (flujo de datos) en un conjunto de coeficientes; donde los coeficientes de mayor orden mostrarán las tendencias globales del flujo de datos y coeficientes de menor orden mostrarán tendencias locales con mayor nivel de detalle.

La descomposición jerárquica de las características de un flujo de datos no implica pérdida de información; pues es posible reconstruir todo el flujo de datos a partir de los coeficientes obtenidos. Sin embargo es posible la eliminación de coeficientes y la introducción, en su lugar, de pequeños errores en el proceso de reconstrucción del flujo de datos.

Histograms

Otro método clave para la reducción y construcción de representaciones compactas de un flujo de datos es el método de histogramas (*histograms*) [10, 19]. Los histogramas son una técnica estadística ampliamente utilizada que consiste básicamente en representar de forma gráfica la propiedad fundamental de un flujo de datos en forma de barras; siendo la superficie de cada barra proporcional a la frecuencia de los atributos representados.

Un histograma está definido por un conjunto de intervalos no solapados y cada intervalo estará definido por un conteo de frecuencia y por los límites que caracterizan a un atributo. Los requerimientos de memoria estarán definidos por la cantidad de atributos del flujo que queramos

representar existiendo un compromiso entre la memoria disponible y la aproximación en la representación del flujo analizado.

Los histogramas han sido utilizados en una gran variedad de aplicaciones tanto de gestión como de minería de flujos de datos. Guha, S. y N. Koudas [20] y Buccafurri, F. y G. Lax [21] hacen uso de histogramas en la estimación de consultas a flujos de datos. Por otro lado Sebastião, R. y J. Gama [19] proponen el uso de histogramas para la deteccion de cambios en flujos de datos de naturaleza evolutiva.

Sketches

La técnica de *sketches* [4, 10] deriva su inspiración en la técnica de *wavelets*. De hecho, los métodos de *sketches* son considerados como la versión aleatoria de las técnicas de *wavelets*. Es el método de construcción de sinopsis más eficiente en cuanto a utilización de la memoria se refiere, motivo por el cual han sido utilizados de manera efectiva en aplicaciones relacionadas a la gestión de flujos de datos como la estimación de asociaciones y la estimación de consultas [12-14].

Los métodos de *sketches* también han tenido aplicación en la minería de flujos de datos. Ejemplo de ello lo constituye la detección de cambios [15, 22] y el problema de encontrar heavy hitters (elemento más frecuente) en un flujo de datos [23-25]. La eficiencia de este método de construcción de sinopsis puede ser aprovechada en el procesamiento de flujos de datos en entornos distribuidos en vista a minimizar los costos de comunicación entre los múltiples sensores y el sistema de procesamiento. Sin embargo, los métodos de *sketches*, carecen de aplicabilidad en el caso de flujo de datos multidimensionales dado a la complejidad de su mecanismo de representación de las características fundamentales de un flujo.

Micro-clúster

Otro de los métodos usados en la construcción de representaciones resumidas de un flujo de datos es la técnica de micro-clúster [10, 26]; la cual se define como la extensión temporal del *cluster feature vector* definido en [27]. Una de las ventajas de esta técnica de resumen lo constituye el hecho de ser aplicable en el contexto de flujos de datos multidimensionales. La naturaleza evolutiva de los flujos y las restricciones en el tiempo de procesamiento, son problemas para los que también es adaptable la utilización de micro-clúster.

Aggarwal, C.C *et al*.[10] hacen uso de la técnica de micro-clúster en aplicaciones de minería como clasificación y agrupamiento de flujos de datos de naturaleza infinita, evolutiva y multidimensionales. La técnica de micro-clúster es propuesta además como método de resumen en aplicaciones de gestión de flujos como la estimación de consultas [28].

2.3 Problemas generales en el procesamiento de flujos de datos

El desarrollo de Sistemas de Gestión de Flujos de Datos es un tema de interés activo en la comunidad de las ciencias informática. Los SGFD permiten el procesamiento y consulta de flujos de datos continuos, rápidos y variables en el tiempo. Los algoritmos utilizados para procesar este tipo de flujo de datos son capaces de brindar soluciones rápidas y aproximadas con recursos de memoria reducidos.

El epígrafe que culmina ha perseguido como objetivo realizar un recorrido por aquellas técnicas, métodos y algoritmos más utilizados en el procesamiento de flujos de datos. Ventajas, desventajas o aplicación de uno u otro de estos métodos en otras áreas de interés como la minería y gestión de flujos, han quedado también plasmadas. Se han mostrado las marcadas diferencias entre el procesamiento tradicional de conjuntos de datos y el procesamiento de flujos. Se han mencionado aquellos problemas a tener en cuenta, ya sea al adaptar cualquiera de los algoritmos tradicionales de conjunto de datos al contexto de flujos de datos, como a la hora de diseñar un nuevo algoritmo.

La literatura ha dividido el análisis de los flujos de datos en: procesamiento, gestión y minería. Ninguno de los actuales SGFD está ajeno a la presencia de técnicas, métodos algoritmos que respondan a las áreas de investigación antes mencionadas. Los problemas que afectan a los algoritmos de procesamiento de flujos básicamente son los mismos que afectan a los de gestión y minería pues el origen de los mismos está en la naturaleza de los flujos de datos. Los problemas generales a tener en cuenta para diseñar, desarrollar y desplegar algoritmos para el procesamiento de flujos de datos son mencionados a continuación.

Naturaleza infinita de los flujos de datos: El problema de la memoria de procesamiento es una motivación importante para la cual varias técnicas como la construcción de sinopsis, ventanas de tiempo, etc. han sido desarrolladas. La memoria disponible en cualquier sistema de procesamiento de flujos de datos por lo general es mucho más pequeña que el tamaño del flujo a procesar. Los algoritmos deben diseñarse teniendo en cuenta la naturaleza infinita de los flujos y las limitaciones que ello implica en la memoria disponible para el procesamiento.

Naturaleza evolutiva de los flujos de datos: Una característica inherente de los flujos de datos es su naturaleza evolutiva. Los algoritmos de procesamiento pueden volverse obsoletos si son marcados los cambios que sufre un flujo con el progreso del tiempo. Módulos capaces de detectar estos cambios y actualizar así a los algoritmos de procesamiento pudieran influir positivamente en la precisión de los resultados obtenidos.

Naturaleza secuencial de los flujos de datos: Los elementos en un flujo de datos llegan de manera inmediata y secuencial, El sistema de procesamiento no tiene control sobre el orden en que llegan los elementos del flujo, por lo cual, el acceso a los mismos para su procesamiento no puede ser de manera aleatoria. Los algoritmos deben diseñarse teniendo en cuenta esta propiedad.

Altas velocidades de los flujos de datos: Otra característica inherente de los flujos de datos es su alta velocidad. Los algoritmos deben ser capaces de adaptarse a esta propiedad de los flujos. El tiempo de procesamiento será limitado y no serán posibles múltiples etapas de procesamiento a los elementos contenidos en el flujo.

Dimensionalidad de los flujos de datos: Muchos de los actuales sistemas de flujos de datos son capaces de generar flujos de datos de más de dos o tres dimensiones. Además, el conjunto de valores posibles en una dimensión en específico puede ser también muy grande. Los algoritmos de procesamiento deben ser diseñados de acuerdo a la aplicación en la que serán utilizados; la dimensionalidad de los flujos de datos puede significar un problema.

Compromiso entre eficiencia y exactitud: El compromiso fundamental a tener en cuenta en el diseño de algoritmos de procesamiento de flujos de datos radica entre la exactitud con la que se quieren los resultados del procesamiento y las restricciones en cuanto a memoria y tiempo de procesamiento. En ocasiones, es necesario utilizar algoritmos de aproximación que garanticen aceptables límites de error y un alto nivel de eficiencia.

Fuentes de datos múltiples en entornos distribuidos: Un gran número de aplicaciones de flujos de datos radican en entornos distribuidos como es el caso de los sensores de red. Si a ello le añadimos las limitaciones de ancho de banda para la transmisión de los datos y los escasos recursos de memoria y procesamiento de este tipo de dispositivo, podremos ver que el análisis y procesamiento en este tipo de entornos constituye realmente un problema a resolver.

Integración entre gestión y minería de flujos de datos: La integración entre el almacenamiento, consulta, minería, visualización, y procesamiento de continuos flujos de datos, garantizaría la robustez de sistemas de flujos de datos que pueden ser de utilidad en las más diversas aplicaciones. Una importante línea de investigación podría

ser la integración de algoritmos de gestión y minería en vista a diseñar SGFD más completos, compactos y eficientes.

3 Minería de flujos de datos

En los últimos años, una gran cantidad de flujos de datos potencialmente infinitos son a menudo generados por sistemas en tiempo real, redes de comunicaciones, sensores remotos de redes, experimentos científicos, internet y otro gran número de entornos de carácter dinámico. Métodos de análisis y algoritmos de procesamiento deben ser diseñados teniendo en cuenta las limitaciones que imponen los flujos de datos. Como se hacía referencia en el epígrafe 1 de este trabajo, el estudio de los flujos de datos es dividido por la literatura en procesamiento, minería y gestión de flujos. De manera análoga, la minería y gestión de flujos de datos pueden ser consideradas como tareas independientes del procesamiento de flujos de datos.

La minería de datos puede verse como un resultado de la evolución natural de las ciencias de la información y es definida por [9] como la extracción implícita, no trivial, previamente desconocida y potencialmente útil de patrones y conocimiento de un gran volumen de datos. En el contexto de los flujos de datos, la utilización de técnicas como el agrupamiento, clasificación, detección de patrones frecuentes y detección de cambios en flujos de datos de naturaleza evolutiva; para la extracción de conocimiento de estas formas de datos, se han convertido en temas de interés activo para la investigación en la última década. El objetivo de este epígrafe es hacer un estudio sobre los avances y problemas detectados en materia de minería de flujos de datos, haciendo énfasis en aquellas técnicas más representativas.

3.1 Agrupamiento en Flujos de Datos

El agrupamiento (*clustering*) es considerado como la técnica de aprendizaje no supervisado de mayor interés en el análisis y procesamiento de datos. La literatura en este campo se extiende a lo largo de una gran variedad de áreas incluyendo la estadística, minería de datos, reconocimiento de patrones y aprendizaje de máquina. En la última década, el agrupamiento ha sido un tema de investigación activo en el contexto de los flujos de datos.

El agrupamiento es usado comúnmente como una herramienta para descubrir estructuras en los datos a analizar. Una definición informal de la técnica de agrupamiento puede ser el proceso de dividir un conjunto de datos en subconjuntos (clúster), donde miembros de un mismo clúster son similares y miembros de distintos clúster son diferentes [1, 26, 29-32]. La similitud entre elementos de un mismo clúster es definida por alguna función o medida de distancia.

Varias han sido las técnicas utilizadas para agrupar conjuntos de datos. La clasificación de los algoritmos de agrupamiento según la técnica utilizada no es ni rígida ni absoluta pues dichas técnicas pueden llegar a solaparse. De acuerdo con la bibliografía, los algoritmos de agrupamiento de flujos de datos se clasifican en: jerárquicos, basados en particiones, basados en modelos, basados en densidad, basados en mallas y basados en grafos. Una colección de los algoritmos de agrupamiento más representativos de la última década y su clasificación es mostrada en la Tabla 2.

Para el procesamiento particular de flujos de datos rápidos y variantes en el tiempo, la técnica de agrupamiento se ha dividido en un componente *online* y otro *offline*. El componente *online* almacenará periódicamente detallados resúmenes estadísticos de la estructura fundamental del flujo. El componente *offline* utilizará dichos resúmenes junto a otras variables de entradas definidas por el usuario, para proveer de una mayor comprensión de los clústeres formados cada vez que se requiera.

Además de los problemas asociados a: la memoria y tiempo de procesamiento, la naturaleza evolutiva y secuencial de los flujos, la velocidad y multidimensionalidad de los elementos de un flujo; los algoritmos de agrupamiento deben ser diseñados teniendo en cuenta otros requerimientos[32]:

Manipulación de *outliers*: En estadística, un *outlier* es un elemento numéricamente distante del resto de los elementos de un grupo de datos. En el contexto de los flujos de datos, factores externos como: la interferencia electromagnética, fallas de energía y otras formas de ruido; pueden generar elementos extraños que difieran del resto de los elementos del flujo. Los algoritmos de agrupamiento de flujos de datos deben ser diseñados con la capacidad de manipulación de este tipo de elementos.

Descubrimiento de clústeres de forma arbitraria: Los algoritmos propuestos en [26, 29] tienen el inconveniente de solo detectar grupos de forma esférica, debido a la función de distancia que utilizan para llevar a cabo el agrupamiento. Que un algoritmo sea capaz de detectar grupos de formas arbitrarias puede influir de manera positiva en su escalabilidad en diversas aplicaciones.

No asumir el número de clústeres: En flujos de datos de naturaleza evolutiva, el número de grupos puede variar con el progreso del tiempo. Fijar el número de grupos a formar por el algoritmo de agrupamiento influirá negativamente en la aproximación de los resultados.

Basado en las consideraciones anteriores, una buena cantidad de algoritmos de agrupamiento de flujos de datos han sido propuestos, pero si bien es cierto que dichos algoritmos han ofrecido efectivas soluciones a los problemas mencionados, también lo es el hecho de que ninguno de ellos ofrece solución a la combinación de estos requisitos. En la Tabla 2 se muestran las fortalezas y debilidades de los algoritmos de agrupamiento más representativos de los últimos años en la solución de los problemas generales antes mencionados.

Tabla 2. Algoritmos de agrupamiento en flujos de datos

Algoritmo	Técnica Utilizada	Flujos de Datos Evolutivos	Flujos de Datos Multidimensionales
(2002) STREAM [1]	particiones		
(2003) CluStream [26]	modelos	✓	
(2005) DUCstream [33]	densidad	✓	
(2005) HPStream [29]	modelos	✓	✓
(2005) GCHDS [30]	mallas	✓	✓
(2006) DenStream [32]	densidad	✓	
(2007) ExCC [34]	densidad	✓	✓
(2008) UMicro [35]	modelos	✓	
(2009) PMicro [36]	modelos	✓	
(2009) RepStream [37]	grafos	✓	
Algoritmos	**Detección de Outlier**	**Clúster de Forma Arbitraria**	**Flujos de Datos Imprecisos**
(2002) STREAM [1]			
(2003) CluStream [26]			
(2005) DUCstream [33]			
(2005) HPStream [29]			
(2005) GCHDS [30]		✓	

(2006) DenStream [32]	✓	✓	
(2007) ExCC [34]	✓	✓	
(2008) UMicro [35]			✓
(2009) PMicro [36]			✓
(2009) RepStream [37]	✓	✓	

Recientemente, dos nuevas direcciones dentro de la minería de flujos de datos han sido presentadas. Los algoritmos DGClust [38] y DSIC [39], se proponen como solución al agrupamiento de flujos de datos cuya fuente lo constituyen sensores de red en un entorno distribuido. Otros dos trabajos, UMicro [35] y PMicro [36], se presentan para el agrupamiento de flujos de datos imprecisos o inexactos. Los flujos de datos imprecisos son aquellos acerca de los cuales se conoce un valor probabilístico de error estimado, asociado a factores externos. El conocimiento previo al procesamiento de dichos errores permite obtener resultados más precisos.

3.2 Clasificación en flujos de datos

La clasificación es otra de las técnicas de minería de datos que ha atraído la atención de investigadores debido al significado de sus aplicaciones. Generalmente, la clasificación es definida como un conjunto de técnicas supervisadas de aprendizaje constituido por dos fases de procesamiento. En una primera fase (aprendizaje o construcción del modelo) se tratará de encontrar un modelo o función capaz de describir y distinguir conceptos o clases de datos en un conjunto de datos de entrenamiento. La segunda fase (clasificación o utilización del modelo) consiste en utilizar el modelo de clasificación obtenido para predecir las clases de datos a las que pertenecen los elementos de nuevos conjuntos de datos [9, 16].

De manera más formal, el problema de la clasificación es definido como [40, 41]: Dado un conjunto de datos de entrenamiento finito, donde cada elemento puede ser representado como un par $e = (v; c)$, siendo v un vector compuesto por a atributos cuyos valores pueden ser numéricos o simbólicos y c una categoría o clase discreta denominada etiqueta. El objetivo de los métodos de clasificación es construir un modelo $c = f(v)$ para clasificar o decidir la categoría o clase a la que pertenecen nuevos elementos no etiquetados.

La clasificación en flujo de datos ha tomado fuerzas en la última década al igual que otras técnicas de minería ya analizadas. Los algoritmos de clasificación no están exentos a los problemas asociados a la naturaleza de los flujos de datos y deben ser diseñados para los requisitos que dichos flujos imponen. Básicamente, los algoritmos de clasificación deben ser capaces de procesar información a altas velocidades, consumir escasos recursos y actualizar los modelos de clasificación mediante la detección de cambios en flujos de datos evolutivos.

Tabla 3. Algoritmos de clasificación en flujos de datos

Algoritmo	Clasificador Utilizado	Escasos Recursos (Memoria)	Altas Velocidades
(2000) VFDT [40]	Árboles de Decisión	✓	✓
(2001) CVFDT [42]	Árboles de Decisión	✓	✓
(2002) OLIN [43]	Redes Difusas		
(2003) Ensemble-based [44]	Combinación		
(2004) SCALLOP [41]	Reglas de Asociación		

(2004) On-Demand [16]	Micro-Clúster	✓	✓
(2005) ANNCAD [45]	Vecino más Cercano		
(2005) UFFT [46]	Árboles de Decisión	✓	✓
(2006) RobustBoosting [47]	Combinación		
(2007) SRMTDS [48]	Combinación	✓	✓
(2008) DSRF [49]	Combinación		
(2008) MSRT [50]	Combinación	✓	✓
(2009) FlexDT [51]	Combinación	✓	✓

Algoritmo	Flujos de Datos Evolutivos	Ruido	Ausencia de Información
(2000) VFDT [40]			
(2001) CVFDT [42]	✓		
(2002) OLIN [43]	✓		
(2003) Ensemble-based [44]	✓		
(2004) SCALLOP [41]	✓		
(2004) On-Demand [16]	✓		
(2005) ANNCAD [45]	✓		
(2005) UFFT [46]	✓		
(2006) RobustBoosting [47]	✓	✓	
(2007) SRMTDS [48]		✓	
(2008) DSRF [49]	✓	✓	
(2008) MSRT [50]	✓	✓	
(2009) FlexDT [51]	✓	✓	✓

Muchos de los algoritmos estudiados en este trabajo atacan el problema de la detección de cambios y actualización de los modelos como la única causa que afecta la calidad de los resultados de la clasificación en flujos de datos. Recientemente, han sido tratados otros dos problemas que afectan la precisión y efectividad de nuestros algoritmos: la capacidad de detección de ruido y la capacidad de procesamiento frente a ausencia de información (*missing values*).

Detección de ruido: En el contexto de los flujos de datos, debido a la influencia de varios factores externos como la interferencia electromagnética, fallas de energía y otras formas de ruido; elementos extraños que difieren del resto de los elementos del flujo, pueden ser generados. Los modelos de clasificación deben contar con la capacidad de detectar este tipo de elementos y así mejorar la aproximación de los resultados.

Ausencia de información: De forma similar, debido a la influencia de factores externos, los elementos de un flujo de datos pueden llegar con la ausencia de determinados atributos (*missing values*). Los modelos de clasificación deben ser robustos ante este problema y brindar la mejor de las soluciones.

Solo uno de los algoritmos estudiados, FlexDT [51], valiéndose de arboles de decisión y lógica difusa, es capaz de afrontar la combinación de todos estos problemas.

Varias técnicas como árboles de decisión, vecino más cercano y reglas de asociación han sido utilizadas para clasificar elementos de un flujo de datos. Sin embargo, en un trabajo realizado por H. Wang et al.[44] se demuestra que la combinación de estas técnicas permite lograr mejores clasificadores y una mayor precisión en los resultados. En otros trabajos de los últimos años [47-51] la tendencia ha sido utilizar combinación de clasificadores, en lugar de utilizar simples técnicas de clasificación.

En la Tabla 3 se muestra una colección de algoritmos de clasificación representativos de los últimos diez años. Además, se presentan las técnicas utilizadas en cada caso, así como, las fortalezas y debilidades de dichos algoritmos en la solución de los problemas fundamentales para clasificar flujos de datos.

3.3 Extracción de patrones frecuentes en flujo de datos

Otro de los problemas básicos en la minería de datos es el hecho de encontrar elementos (*ítems*) o conjuntos de elementos (*itemsets*) frecuentes en un conjunto de datos. El problema original se remonta a principios de los años 90, en el descubrimiento de asociaciones de elementos dentro de transacciones de ventas de mercado almacenadas en una base de datos. Por ejemplo, un itemsets podría ser “leche, pan, soporte = 10%”, lo que significa que el 10% de las transacciones almacenadas en la base de datos contienen a ambos elementos, leche y pan. Como muestra el ejemplo anterior, el soporte es una medida de la frecuencia que tiene un itemsets en un conjunto de datos. Los usuarios pueden definir umbrales de soporte mínimo para descartar aquellos itemsets de menor frecuencia. Si el soporte de un itemsets excede o iguala el umbral de soporte mínimo definido por el usuario se está en presencia de un itemsets frecuente.

La extracción de patrones frecuentes está enfocada al descubrimiento de patrones desde diferentes tipos de conjuntos de datos, incluyendo los conjuntos de datos no estructurados, como textos, semi-estructurados, como XML, y estructurados, como grafos. Por este motivo, los patrones pueden ser un conjunto de términos, secuencias, sub-árboles, sub-grafos, etc. Los patrones frecuentes no solo pueden resumir de manera eficaz la información contenida en un conjunto de datos, sino que también sirven como herramienta básica a otras técnicas de minería, como clasificación, agrupamiento, detección de cambios entre otras.

En el contexto de los flujos de datos, el problema de encontrar patrones frecuentes ha despertado un mayor interés en la última década. Comparada con otras técnicas de minería en flujos de datos, los mayores retos que plantea la extracción de patrones frecuentes están dirigidos fundamentalmente a los requerimientos de memoria, precisión y costo computacional. En determinadas situaciones es de interés encontrar los patrones frecuentes más recientes e ignorar los elementos ya pasados de un flujo de datos. Naturalmente, el uso de ventanas de tiempo, para este propósito permite una disminución de las necesidades de espacio de procesamiento y tiempo de respuesta. Atendiendo al grado de importancia de los elementos y patrones más recientes de un flujo de datos, varios modelos de ventanas han sido utilizados por los algoritmos de extracción de patrones frecuentes: *landmark windows* [52, 53], *damped windows* [54], *sliding windows* [55, 56] y *tilted time windows* [57].

En la Tabla 4 es mostrada una variedad de algoritmos representativos para la extracción de patrones frecuentes en flujos de datos. Como se muestra, la mayoría de estos algoritmos son orientados a falsos positivos, lo que significa que dado un valor de soporte mínimo ($s < 1$) el consumo de memoria es controlado por un parámetro de error ($e < 1$), ($e < s$). Todos los ítems o itemsets con: (1) soporte mayor e igual a s y (2) soporte menor a s y mayor a ($s - e$) son considerados como frecuentes. Esto trae aparejado la existencia de patrones considerados por el algoritmo como frecuentes cuando en realidad no lo son. Aún aplicando métodos heurísticos y procedimientos para reducir la cantidad de falsos positivos, el consumo de memoria y precisión

de estos algoritmos puede estar comprometido partiendo del hecho de que un ítems falso positivo implica varios itemsets falsos positivos.

Tabla 4. Algoritmos para la extracción de patrones frecuentes en flujos de datos

Algoritmo	Orientación	Modelo Utilizado	Escasos Recursos (Memoria)
(2002) Lossy Counting	falsos positivos	landmark windows	
(2003) FP-stream	falsos positivos	tilted time windows	
(2003) estDec	falsos positivos	damped windows	✓
(2004) SWLC	falsos positivos	sliding windows	✓
(2005) FP-DS	falsos positivos	sliding windows	✓
(2006) FDPM	falsos negativos	landmark windows	✓

Una primera aproximación para dar solución a este problema es presentada por Yu J.X. et al [53]. FPDM es un algoritmo orientado a falsos negativos que se vale de un nuevo parámetro para el control del consumo de memoria y de los límites de Chernoff para arrojar mejores resultados en el consumo de recursos y precisión de los resultados, en la extracción de patrones frecuentes en flujos de datos.

Generalmente, la mayoría de los algoritmos propuestos constan de dos etapas. En una primera etapa se calculan las frecuencias de los itemsets al mismo tiempo que se monitorean los nuevos elementos que llegan del flujo de datos. En una segunda etapa son mostrados los itemsets frecuentes teniendo en cuenta requerimientos definidos por el usuario. Debido al gran número de combinaciones de ítems posibles en cada transacción de un flujo de datos, la primera etapa es la más costosa en tiempo y espacio de procesamiento. Por consiguiente, los algoritmos propuestos son insuficientes en el tratamiento y procesamiento de flujos de datos a altas velocidades.

Recientemente, el algoritmo DELAY [58] se propone para dar solución al problema de la extracción de patrones frecuentes en flujos de datos rápidos. En DELAY se presenta una nueva filosofía que delega el conteo de frecuencia para una segunda etapa y la posibilidad de realizar ambas etapas en paralelo, independiente una de la otra. Resultados experimentales que demuestran la superioridad de este algoritmo frente a otros representativos como Lossy Counting y FDPM, hacen que DELAY sea una de las primeras aproximaciones en la extracción de ítems e itemsets frecuentes en flujos de datos a altas velocidades.

Otro asunto de interés para la extracción de patrones frecuentes, sería el hecho de detectar considerables cambios en los patrones de un flujo con el progreso del tiempo. La naturaleza evolutiva de los flujos de datos es un problema al que no escapa ninguno de los algoritmos de minería en flujos de datos, y al que hay que tener en cuanta en el momento de su diseño y de acuerdo a la solución que se quiera brindar. Los patrones que un momento determinado son frecuentes podrían no serlo en un futuro. En un trabajo realizado por Koh J.L. y C.Y. Lin [59], los algoritmos FCDT y FCDB se proponen para monitorear y detectar notables cambios en los itemsets frecuentes en flujos de datos a altas velocidades.

Además de los trabajos presentados en este estudio, otros temas estrechamente relacionados al problema de extraer patrones frecuentes en flujos de datos, se han abordado por los investigadores en los últimos años. A continuación son presentados algunas de las derivaciones así como las soluciones propuestas:

Extracción de patrones frecuentes en flujos de datos distribuidos

La mayoría de los trabajos presentados en los últimos años, para la extracción de patrones frecuentes en flujos de datos, obvian la existencia de aplicaciones y sistemas

generadores de múltiples flujos de datos. Los algoritmos presentados anteriormente no están diseñados para la extracción de patrones frecuentes en múltiples flujos de datos de manera distribuida. Estos algoritmos sólo están preparados para analizar y procesar flujos que vengan en forma de transacciones continuas y ordenadas de una sola fuente de dato. Sin embargo algunos trabajos se han presentado para dar solución a este problema. A. Manjhi et al. [60], R. Fuller y M. Kantardzic [61] proponen algoritmos para la extracción de los k itemsets más frecuentes de la unión de múltiples flujos de datos. El modelo de entorno distribuido utilizado por dichos algoritmos está definido por m flujos de datos y ($m + 1$) nodos denotados por N_1, N_2,……N_m y dedicados a resumir los m flujos. El nodo N_0 es el nodo encargado de la coordinación y sincronismo de los nodos de monitoreo, además de visualizar el conjunto de ítems frecuentes sobre la unión de los m flujos de datos. Luego, otros problemas a resolver bajo estas condiciones, son garantizar la comunicación continua y el sincronismo entre los nodos, sin afectar la aproximación de los resultados.

Extracción de *hierarchy heavy hitters* en flujos de datos

El problema de la extracción de *heavy hitters* en flujos de datos puede verse como una variación del problema original de la extracción de ítems e itemsets frecuentes. De hecho, un *heavy hitters* no es más que un elemento cuya frecuencia en un conjunto de datos no es menor a un umbral de frecuencia definido por el usuario. La diferencia radica, en que en este caso particular se mantiene una jerarquía entre los diferentes elementos del flujo. Dado una jerarquía y un umbral de frecuencia f, el objetivo es encontrar todos los nodos *hierarchical heavy hitters* (HHH) cuyos descendientes en la jerarquía tengan un valor de frecuencia mayor o igual al valor umbral definido por el usuario. G. Cormode et al. [25, 62] ha estudiado el problema de identificar eficientemente (HHH) en flujos de datos.

Extracción de patrones frecuentes en flujos de datos semi-estructurados

Como se hacía referencia al inicio de esta sección, el problema de la extracción de patrones frecuentes está condicionado al tipo de datos que se quiera analizar. Debido al incremento en los avances tecnológico de los últimos años y a la utilización cada vez mayor de la Internet, se ha hecho popular, la utilización de tecnologías de comunicación basadas en datos semi-estructurados como lo son los datos XML. Una de las primeras aproximaciones en la solución del problema de extraer patrones frecuentes en flujos de datos semi-estructurados, es presentada por T. Asai et al. [63].

3.4 Detección de cambios en flujo de datos

Tradicionalmente, los algoritmos de minería de datos se han diseñado para dar solución a problemas bien conocidos como lo son el agrupamiento, la clasificación y la detección de patrones frecuentes en bases de datos y recientemente en flujos de datos. Debido a la naturaleza evolutiva, y la existencia de una componente temporal a tener en cuenta durante el procesamiento; la detección de cambios surge como una nueva estrategia de minería en el contexto de los flujos de datos.

Es un hecho, que en la mayoría de las actuales aplicaciones y sistemas generadores de flujos de datos, la velocidad, dimensionalidad y el cambio continuo de los elementos del flujo son características inherentes. El monitoreo, detección y visualización de estos cambios constituye un problema de marcada importancia en el análisis y procesamiento de los flujos de datos. El problema de la evolución de los datos puede ser de interés desde dos diferentes perspectivas: (1) detección y visualización de los cambios y (2) actualización de otros modelos de minería como

clasificación, agrupamiento y detección de patrones frecuentes anteriormente abordadas en este trabajo.

3.4.1 Detección y visualización de cambios

En muchas de las más recientes aplicaciones es de vital importancia el desarrollo de técnicas y herramientas capaces de proveer una visión general de las características de los datos que se están analizando. La detección y visualización de los cambios que sufren dichos datos en el tiempo, de una forma rápida y amigable para los usuarios es un problema de actualidad dentro de la minería de flujos de datos. El monitoreo de las condiciones ambientales, el tráfico generado por el uso de la Internet, la detección de ataques e intrusiones en nuestras redes, son algunas de las aplicaciones donde se hace útil el uso de herramientas para el diagnóstico de cambios en los datos a procesar.

Varios trabajos han sido propuestos en los últimos años para dar solución a estos problemas. C. C. Aggarwal [64] propone una técnica basada en el concepto de *velocity density estimation* para la comprensión y estimación de tendencias en la evolución de flujos de datos multidimensionales a altas velocidades. La técnica propuesta permite la predicción de tres tipos de cambios en los datos analizados: coagulación, disolución y cambio total de los datos.

Los trabajos presentados por R. Sebastião y J. Gama [19] y D. Kifer et al. [65] basan sus algoritmos en un paradigma de dos ventanas de tiempo para la detección y cuantificación de cambios significativos en la estructura fundamental de los flujos de datos analizados. Una ventana se desliza con el arribo de cada elemento del flujo de datos. Los elementos contenidos en el interior de la ventana deslizante son comparados con los de una ventana de referencia. La ventana de referencia se actualiza cada vez que un cambio es detectado.

El algoritmo STREAM-DETECT [66] se propone para la detección de cambios pequeños, bruscos, frecuentes y menos frecuentes en flujos de datos, mediante la combinación de técnicas de agrupamiento y clasificación. Otro trabajo enfocado a la visualización de cambios valiéndose de gráficos tridimensionales es presentado por B. Gillick et al.[67].

3.4.2 Actualización de los modelos de minería

La actualización de otros modelos de minería como clasificación y agrupamiento, es otra de las perspectivas desde la cual podrían ser de interés la detección de cambios en flujos de datos. Existe una considerable cantidad de trabajos en la literatura enfocados al mantenimiento incremental de los modelos de minería. Sin embargo, en el contexto de flujos de datos a altas velocidades, es de mayor importancia utilizar la evolución de los flujos de datos en vista a obtener una representación cuantitativa de la naturaleza del cambio.

A lo largo de este trabajo un conjunto de algoritmos de minería, que atacan el problema de la naturaleza evolutiva de los flujos de datos han sido analizados. Independientemente de la tarea de minería que queramos realizar, es un hecho que la evolución de los datos contenidos en un flujo puede hacer que nuestros algoritmos se vuelvan obsoletos e imprecisos. Ejemplo de trabajos analizados en este trabajo que cumplen con tal requisito son: [26, 29, 30, 32-37] en agrupamiento de flujos de datos, [16, 41-47, 49-51] en clasificación y [59] en detección de patrones frecuentes.

4 Conclusiones

Como se ha apreciado, en este trabajo se presentó un estudio sobre dos áreas de investigación estrechamente relacionadas: procesamiento y minería de flujos de datos. Se analizaron los métodos y algoritmos básicos en el procesamiento de flujos de datos y se plantearon los problemas generales a los cuales se enfrenta cualquiera de las técnicas de procesamiento actualmente utilizadas. Se hizo un estudio sobre los problemas particulares en materia de minería de flujo de datos, prestándole mayor atención a las técnicas más utilizadas en este contexto: agrupamiento, clasificación, detección de patrones frecuentes y detección de cambios en flujo de datos.

En los últimos años, una gran cantidad de flujos de datos potencialmente infinitos son a menudo generados por sistemas en tiempo real, redes de comunicaciones, sensores remotos de redes, experimentos científicos, internet y otro gran número de entornos de carácter dinámico. Una buena cantidad de algoritmos para el análisis, procesamiento y minería de flujos de datos han sido propuestos. Pero si bien es cierto que dichos algoritmos han ofrecido efectivas soluciones a problemas asociados con: la naturaleza evolutiva y secuencial de los flujos y la velocidad y dimensionalidad de los elementos de un flujo de datos, también lo es el hecho de que muy pocos de ellos ofrece solución a la combinación de todos estos requisitos. Además, debido a las restricciones de memoria y tiempo para el procesamiento de los datos, los resultados obtenidos en flujos de datos comúnmente son aproximados y no exactos como en el procesamiento tradicional de bases de datos. Existirá siempre un compromiso entre la capacidad de nuestros algoritmos de satisfacer los resquicitos antes mencionados y la precisión de los resultados.

Futuras direcciones para la investigación con vistas a lograr mejores soluciones podrían ser: la integración de algoritmos de gestión y minería para diseñar SGFD más completos, compactos y eficientes, el diseño de algoritmos de agrupamiento capaces de detectar clústeres de formas arbitrarias que garanticen la escalabilidad en diversas aplicaciones, la utilización de combinación de clasificadores en lugar de simples técnicas de clasificación para lograr mayor precisión en los resultados, el diseño de algoritmos para la extracción de patrones en flujos rápidos y variables en el tiempo y la reducción de los costos de comunicación y sincronismo para el procesamiento de flujos de datos distribuidos. El análisis y procesamiento de los flujos de datos, así como, la minería y gestión de los mismos, constituyen temas de investigación activos. Aún queda mucho por hacer en el desafiante campo de los flujos de datos.

Referencias bibliográficas

1. O'Callaghan, L., et al., *Streaming-Data Algorithms for High-Quality Clustering*, in *Proceedings of the 18th International Conference on Data Engineering*. 2002, IEEE Computer Society: San Jose, CA, USA. p. 685-694.
2. Beringer, J. and E. Hullermeier, *Online clustering of parallel data streams.* Data & Knowledge Engineering, 2006. **58**(2): p. 180-204.
3. Babcock, B., et al., *Models and Issues in Data Stream Systems*, in *Proceedings of the 21th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2002, ACM: Madison, Wisconsin, USA. p. 1-16.
4. Muthukrishnan, S., *Data Streams: Algorithms and Applications*. 2005: Now Publishers Inc. 126.
5. Datar, M., et al., *Maintaining Stream Statistics over Sliding Windows.* SIAM Journal on Computing, 2002. **31**(6): p. 1794-1813.
6. Gama, J. and M.M. Gaber, *Learning from Data Streams Processing Techniques in Sensor Networks.* 2007: Springer.

7. Gehrke, J., F. Korn, and D. Srivastava, *On Computing Correlated Aggregates over Continual Data Streams.* ACM SIGMOD Record, 2001. **30**(2): p. 13-24.
8. Jiang, N. and L. Gruenwald, *Research Issues in Data Stream Association Rule Mining.* ACM SIGMOD Record, 2006. **35**(1): p. 14-19.
9. Han, J. and M. Kamber, *Data Mining: Concepts and Techniques*. 2 ed. 2006: Morgan Kaufmann Publishers.
10. Aggarwal, C.C., *Data Stream Models and Algorithms*. 2007: Springer.
11. Li, Y. and R.P. Gopalan, *Clustering Transactional Data Streams*, in *Advances in Artificial Intelligence: Proceedings of19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006. LNSC vol. 4304*. 2006, Springer-Verlag Berlin Heidelberg. p. 1069-1073.
12. Chakrabarti, K., et al., *Approximate Query Processing Using Wavelets.* The VLDB Journal, 2001. **10**(2-3): p. 199-223.
13. Dobra, A., et al., *Processing Complex Aggregate Queries over Data Streams*, in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. 2002, ACM: Madison, Wisconsin, USA. p. 61-72.
14. Dobra, A., et al., *Sketch-Based Multi-query Processing over Data Streams*, in *Advances in Database Technology: Proceedings of 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004. LNCS vol. 2992*. 2004, Springer-Verlag Berlin Heidelberg. p. 551-568.
15. Schweller, R., et al., *Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams*, in *Proceedings of the 4th ACM SIGCOMM conference on Internet Measurement*. 2004, ACM: Taormina, Sicily, Italy. p. 207-212.
16. Aggarwal, C.C., et al., *On Demand Classification of Data Streams*, in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004, ACM: Seattle, WA, USA. p. 503-508.
17. Aggarwal, C.C., *On Biased Reservoir Sampling in the presence of Stream Evolution*, in *Proceedings of the 32nd International Conference on Very Large Data Bases*. 2006, VLDB Endowment: Seoul, Korea. p. 607-618.
18. Heinz, C. and B. Seeger, *Wavelet Density Estimators over Data Streams*, in *Proceedings of the 2005 ACM Symposium on Applied Computing*. 2005, ACM: Santa Fe, New Mexico, USA. p. 578-579.
19. Sebastião, R. and J. Gama, *Change Detection in Learning Histograms from Data Streams*, in *Progress in Artificial Intelligence: Proceedings of 13th Portuguese Conference on Aritficial Intelligence, EPIA 2007,Guimarães, Portugal, December 3-7, 2007. LNCS vol. 4874*. 2007, Springer-Verlag Berlin Heidelberg. p. 112-123.
20. Guha, S. and N. Koudas, *Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation*, in *Proceedings of the 18th International Conference on Data Engineering*. 2002, IEEE Computer Society: San Jose, CA, USA. p. 567.
21. Buccafurri, F. and G. Lax, *Fast range query estimation by N-level tree histograms.* Data Knowledge Engineering, 2004. **51**(2): p. 257-275.
22. Krishnamurthy, B., et al., *Sketch-based Change Detection: Methods, Evaluation, and Applications*, in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*. 2003, ACM: Miami Beach, FL, USA. p. 234-247.
23. Charikar, M., K. Chen, and M. Farach-Colton, *Finding Frequent Items in Data Streams*, in *Automata, Languages and Programming: Proceedings of 29th International Colloquium, ICALP 2002 Málaga, Spain, July 8–13, 2002. LNCS vol. 2380*. 2002, Springer-Verlag Berlin Heidelberg. p. 693-703.
24. Cormode, G. and S. Muthukrishnan, *What's Hot and What's Not: Tracking Most Frequent Items Dynamically.* ACM Transactions on Database Systems 2005. **30**(1): p. 249-278.
25. Cormode, G., et al., *Finding Hierarchical Heavy Hitters in Streaming Data.* ACM Transactions on Knowledge Discovery from Data, 2008. **1**(4): p. 1-48.
26. Aggarwal, C.C., et al., *A Framework for Clustering Evolving Data Streams*, in *Proceedings of the 29th International Conference on Very Large Data Bases.* 2003, VLDB Endowment: Berlin, Germany. p. 81-92.

27. Zhang, T., R. Ramakrishnan, and M. Livny, *BIRCH: An Efficient Data Clustering Method for Very Large Databases*, in *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. 1996, ACM: Montreal, Quebec, Canada. p. 103-114.
28. Aggarwal, C.C., *On Futuristic Query Processing in Data Streams*, in *Advances in Database Technology: Proceedings of 10th International Conference on Extending Database Technology, Munich, Germany, 26-31 March, 2006. LNCS vol. 3896*. 2006, Springer-Verlag Berlin Heidelberg. p. 41-58.
29. Aggarwal, C.C., et al., *On High Dimensional Projected Clustering of Data Streams*, in *Data Mining and Knowledge Discovery. vol. 10 (3).* 2005, Springer Netherlands. p. 251-273.
30. Lu, Y., et al., *A Grid-Based Clustering Algorithm for High-Dimensional Data Streams*, in *Advanced Data Mining and Applications: Proceedings of First International Conference, ADMA 2005, Wuhan, China, July 22-24, 2005. LNCS vol. 3584*. 2005, Springer-Verlag Berlin Heidelberg. p. 824-831.
31. Orlowska, M.E., X. Sun, and X. Li, Can Exclusive Clustering on Streaming Data be Achieved? ACM SIGKDD Explorations Newsletter, 2006. 8(2): p. 102-108.
32. Cao, F., et al., *Density-Based Clustering over an Evolving Data Stream with Noise*, in *Proceedings of the 6th SIAM International Conference on Data Mining*. 2006, SIAM: Bethesda, MD, USA. p. 326-337.
33. Gao, J., et al., *An Incremental Data Stream Clustering Algorithm Based on Dense Units Detection*, in *Advances in Knowledge Discovery and Data Mining: Proceedings of 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. LNCS vol. 3518*. 2005, Springer-Verlag Berlin Heidelberg. p. 420-425.
34. Bhatnagar, V. and S. Kaur, *Exclusive and Complete Clustering of Streams*, in *Database and Expert Systems Applications: Proceedings of 18th International Conference, DEXA 2007, Regensburg, Germany, September 3-7, 2007. LNCS vol. 4653*. 2007, Springer-Verlag Berlin Heidelberg. p. 629-638.
35. Aggarwal, C.C. and P.S. Yu, *A Framework for Clustering Uncertain Data Streams*, in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. 2008, IEEE Computer Society: Cancún, México. p. 150-159.
36. Zhang, C., C. Jin, and A. Zhou, *Efficiently Clustering Probabilistic Data Streams*, in *Advances in Data and Web Management: Proceedings of Joint International Conferences, APWeb/WAIM 2009 Suzhou, China, April 2-4, 2009. LNCS vol. 5446*. 2009, Springer-Verlag Berlin Heidelberg. p. 273-284.
37. L\, S., et al., *Incremental clustering of dynamic data streams using connectivity based representative points.* Data Knowl. Eng., 2009. **68**(1): p. 1-27.
38. Yin, J. and M.M. Gaber, *Clustering Distributed Time Series in Sensor Networks*, in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. 2008, IEEE Computer Society: Washington, DC, USA. p. 678-687.
39. Pereira, P., J. Gama, and L. Lopes, *Clustering Distributed Sensor Data Streams*, in *Machine Learning and Knowledge Discovery in Databases: Proceedings of European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008. LNCS vol. 5212*. 2008, Springer-Verlag Berlin Heidelberg. p. 282-297.
40. Domingos, P. and G. Hulten, *Mining High-Speed Data Streams*, in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2000, ACM: Boston, Massachusetts, United States. p. 71-80.
41. Ferrer, F., J.S. Aguilar, and J.C. Riquelme, *Discovering Decision Rules from Numerical Data Streams*, in *Proceedings of the 2004 ACM Symposium on Applied Computing*. 2004, ACM: Nicosia, Chipre. p. 649-653.
42. Hulten, G., L. Spencer, and P. Domingos, *Mining Time-Changing Data Streams*, in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2001, ACM: San Francisco, California, USA. p. 97-106.
43. Last, M., *Online Classification of Nonstationary Data Streams.* Intelligent Data Analysis., 2002. **6**(2): p. 129-147.
44. Wang, H., et al., *Mining Concept-Drifting Data Streams using Ensemble Classifiers*, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003, ACM: Washington, D.C., USA. p. 226-235.

45. Law, Y. and C. Zaniolo, *An Adaptive Nearest Neighbor Classification Algorithm for Data Streams*, in *Knowledge Discovery in Databases: Proceedings of 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005. LNCS vol. 3721*. 2005, Springer-Verlag Berlin Heidelberg. p. 108-120.
46. Gama, J., P. Medas, and P. Rodrigues, *Learning Decision Trees from Dynamic Data Streams*, in *Proceedings of the 2005 ACM Symposium on Applied Computing*. 2005, ACM: Santa Fe, New Mexico, USA. p. 573-577.
47. Wang, Y., Z. Li, and Y. Zhang, *Classifying Noisy Data Streams*, in *Fuzzy Systems and Knowledge Discovery: Proceedings of 3rd International Conference, FSKD 2006, Xian, China, September 24-28, 2006. LNCS vol. 4223*. 2006, Springer-Verlag Berlin Heidelberg. p. 549-558.
48. Hu, X., et al., *A Semi-Random Multiple Decision-Tree Algorithm for Mining Data Streams.* Journal of Computer Science and Technology, 2007. **22**(5): p. 711-724.
49. Abdulsalam, H., D.B. Skillicorn, and P. Martin, *Classifying Evolving Data Streams Using Dynamic Streaming Random Forests*, in *Database and Expert Systems Applications: Proceedings of 19th International Conference, DEXA 2008, Turin, Italy, September 1-5, 2008. LNCS vol. 5181*. 2008, Springer-Verlag Berlin Heidelberg. p. 643-651.
50. Li, P., X. Hu, and X. Wu, *Mining Concept-Drifting Data Streams with Multiple Semi-Random Decision Trees*, in *Advanced Data Mining and Applications: Proceedings of 4th International Conference, ADMA 2008, Chengdu, China, October 8-10, 2008. LNCS vol. 5139*. 2008, Springer-Verlag Berlin Heidelberg. p. 733-740.
51. Hashemi, S. and Y. Yang, *Flexible decision tree for data stream classification in the presence of concept change, noise and missing values.* Data Mining and Knowledge Discovery 2009. **19**(1): p. 95-131.
52. Manku, G.S. and R. Motwani, *Approximate Frequency Counts over Data Streams*, in *Proceedings of the 28th International Conference on Very Large Data Bases*. 2002, VLDB Endowment: Hong Kong, China. p. 346-357.
53. Yu, J.X., et al., *A false negative approach to mining frequent itemsets from high speed transactional data streams.* Information Sciences, 2006. **176**(14): p. 1986-2015.
54. Chang, J.H. and W.S. Lee, *Finding Recent Frequent Itemsets Adaptively over Online Data Streams*, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003, ACM: Washington, D.C., USA. p. 487-492.
55. Chang, J.H. and W.S. Lee, *A Sliding Window Method for Finding Recently Frequent Itemsets over Online Data Streams.* Information Science and Engineering, 2004. **20**(4): p. 753-762.
56. Liu, X., et al., *Dynamically Mining Frequent Patterns over Online Data Streams*, in *Parallel and Distributed Processing and Applications: Proceedings of 3rd International Symposium, ISPA 2005, Nanjing, China, November 2-5, 2005. LNCS vol. 3758*. 2005, Springer-Verlag Berlin Heidelberg. p. 645-654.
57. Giannella, C., et al., *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*, in *Data Mining. Next Generation Challenges and Future Directions*, H. Kargupta, et al., Editors. 2003, MIT Press. p. 105-124.
58. Yang, H., H. Liu, and J. He, *DELAY A Lazy Approach for Mining Frequent Patterns over High Speed Data Streams*, in *Advanced Data Mining and Applications: Proceedings of 3rd International Conference, ADMA 2007 Harbin, China, August 6-8, 2007. LNCS vol. 4632*. 2007, Springer-Verlag Berlin Heidelberg. p. 2-14.
59. Koh, J.L. and C.Y. Lin, *Concept Shift Detection for Frequent Itemsets from Sliding Windows over Data Streams*, in *Database Systems for Advanced Applications: Proceedings of DASFAA 2009 International Workshops: BenchmarX, MCIS, WDPP, PPDA, MBC, PhD, Brisbane, Australia, April 20 - 23, 2009. LNCS vol. 5667*. 2009, Springer-Verlag Berlin Heidelberg. p. 334-348.
60. Manjhi, A., et al., *Finding (Recently) Frequent Items in Distributed Data Streams*, in *Proceedings of the 21st International Conference on Data Engineering*. 2005, IEEE Computer Society: Pittsburgh, PA, USA. p. 767-778.
61. Fuller, R. and M. Kantardzic, *FIDS: Monitoring Frequent Items over Distributed Data Streams*, in *Machine Learning and Data Mining in Pattern Recognition: Proceedings of 5th International*

Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007. LNCS vol. 4571. 2007, Springer-Verlag Berlin Heidelberg. p. 464-478.

62. Cormode, G., et al., *Finding Hierarchical Heavy Hitters in Data Streams*, in *Proceedings of the 29th International Conference on Very Large Data Bases*. 2003, VLDB Endowment: Berlin, Germany. p. 464-475.
63. Asai, T., et al., *Online Algorithms for Mining Semi-structured Data Stream*, in *Proceedings of the 2002 IEEE International Conference on Data Mining*. 2002, IEEE Computer Society: Kyushu Univ., Fukuoka, Japan. p. 27.
64. Aggarwal, C.C., *A Framework for Diagnosing Changes in Evolving Data Streams*, in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. 2003, ACM: San Diego, California, USA. p. 575-586.
65. Kifer, D., S. David, and J. Gehrke, *Detecting Change in Data Streams*, in *Proceedings of the 13th International Conference on Very Large Data Bases*. 2004, VLDB Endowment: Toronto, Canada. p. 180-191.
66. Gaber, M.M. and P.S. Yu, *Detection And Classification Of Changes In Evolving Data Streams.* International Journal of Information Technology and Decision Making, 2006. **5**(4): p. 659-670.
67. Gillick, B., et al., *Visualisation of Cluster Dynamics and Change Detection in Ubiquitous Data Stream Mining*, in *Proceedings of the 3rd International Workshop on Knowledge Discovery from Data Streams*. 2006: Pittsburgh, PA, USA. p. 1-10.

RT_012, abril 2010
Aprobado por el Consejo Científico CENATAV

Editor: Lic. Lucía González Bayona
Diseño de Portada: DCG Matilde Galindo Sánchez
RNPS No. 2143
ISSN 2072-6260
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
Ciudad de La Habana. Cuba. C.P. 12200
Impreso en Cuba