UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Departamento de Ciencias de la Computación e Inteligencia Artificial

Programa Oficial de Postgrado "Tecnologías de la Información y la comunicación"

Tesis Doctoral

CLASIFICACIÓN DE FLUJOS DE DATOS BASADA EN SIMILITUD

Dayrelis Mena Torres

Granada 2014

UNIVERSIDAD DE GRANADA

Editor: Editorial de la Universidad de Granada
Autor: Dayrelis Mena Torres
D.L.: GR 2135-2014
ISBN: 978-84-9083-155-7

Departamento de Ciencias de la Computación e
Inteligencia Artificial

Programa Oficial de Postgrado "Tecnologías de la
Información y la comunicación"

CLASIFICACIÓN DE FLUJOS DE DATOS BASADA EN SIMILITUD

Memoria de Tesis que presenta

Dayrelis Mena Torres

como requisito para optar al grado de
Doctor en Informática

Director de Tesis

Dr. Jesús Salvador Aguilar Ruíz

Granada 2014

El doctorando Dayrelis Mena Torres y el director de la tesis Dr. Jesús Salvador Aguilar Ruíz. Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección del director de la tesis y hasta donde mi conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, 2 de mayo de 2014

Director de la Tesis

Fdo.: Jesús Salvador Aguilar Ruíz

Doctorando

Fdo.: Dayrelis Mena Torres

Gracias a la Junta de Andalucía y la Asociación Universitaria Internacional de Postgrado (AUIP) por hacer posible esta investigación Doctoral.

AUTOR.

A mi familia,
con todo mi cariño.

Agradecimientos

A mi mamá, mi papá y mi hermana, por estar siempre a mi lado, en las buenas y las malas.

A Yanier, por ser mi sostén...la luz de mi vida.

A mi sobrino Diego, por ser la alegría de la familia.

A mi tutor Jesús Salvador Aguilar Ruíz, por su ayuda y compromiso, incondicionales, y por toda su sabiduría.

A mi tutora Yanet Rodríguez Sarabia, por su cariño y ayuda.

A la AUIP, por auspiciar el Programa Doctoral en SoftComputing y por el apoyo financiero que nos dio desde el principio.

A la Universidad "Marta Abreu" de las Villas en Cuba y la Universidad de Granada en España, por fundar el Programa Doctoral en SoftComputing, y darme la oportunidad de formar parte de él.

A todos los profesores cubanos y españoles, que formaron parte del claustro del Programa Doctoral en SoftComputing; especialmente a los profesores: Rafael Bello, Carlos Morell, Raul Perez y al Curro, que me trataron con tanto cariño, y me ayudaron siempre.

A mis compañeros del grupo de SoftComputing, por los buenos momentos que pasamos juntos y por la ayuda recíproca.

A la Universidad "Hermanos Saíz Montes de Oca" de Pinar del Río y a la Universidad Pablo de Olavide en Sevilla, por su apoyo.

A mis compañeros del Departamento de Informática, por ser una segunda familia para mí.

A mis amigos Alfonso y Gualberto, por soportar mis problemas, comprenderme y ayudarme.

A mi amiga Darkys, por su ayuda, su naturalidad y su afinidad conmigo, que agradeceré eternamente.

A todos mis familiares y amigos que han llorado y reído junto a mí, en los últimos 6 años.

A todos, MUCHAS GRACIAS.

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

OBJETIVOS

1.1 Motivación

El arrollador avance de la Informática y las Telecomunicaciones posibilita, en la actualidad, la obtención de enormes volúmenes de información en una gran variedad de actividades. Estos datos heterogéneos, generados secuencialmente y a gran velocidad, son llamados flujos de datos (*data streams*).

Las aplicaciones incluyen las redes de sensores, la medida del consumo de energía eléctrica, el registro de llamadas en redes de telefonía móvil, el reconocimiento de patrones de compra, la organización y rankings de e-mails y el registro en telecomunicaciones o transacciones comerciales y financieras.

La minería de flujos de datos ha atraído la atención de la comunidad científica en los últimos años, con el desarrollo de nuevos algoritmos para el procesamiento y clasificación de datos.

En este campo de investigación es necesario extraer el conocimiento y la estructura representada en cada momento, sin detener la corriente de información, lo que supone un reto en cuanto a almacenamiento, eficiencia, y detección y tratamiento del ruido y los cambios que puedan ocurrir en la función objetivo.

El problema de la clasificación para flujos de datos ha sido abordado desde diferentes perspectivas, algunas ya clásicas. Dentro de las técnicas más conocidas se encuentran: las Redes Neuronales Artificiales, el Aprendizaje Simbólico (Reglas), los Árboles de Decisión, el Aprendizaje Basado en Instancias y, más recientemente, los modelos combinados que utilizan varias de estas técnicas.

1.2 Objetivo

La presente tesis tiene como objetivo fundamental la **obtención de un clasificador para flujo de datos basado en similitud**. Nuestra propuesta se centra en la utilización de las técnicas del Aprendizaje Basado en Instancias (IBL) para obtener un algoritmo de clasificación, capaz de mantener un modelo actualizado en memoria y detectar los cambios de concepto ocurridos en la función objetivo.
Para el logro de este objetivo fundamental se definieron los siguientes objetivos específicos:

- Realizar un estudio de los principales conceptos y características de los entornos de aprendizaje en flujos de datos.
- Realizar un estudio de las técnicas utilizadas en la minería de flujos de datos, haciendo énfasis en la técnica del Aprendizaje Basado en Instancias.
- Realizar un estudio de los algoritmos desarrollados para la minería de flujos de datos, principalmente, modelos de agrupamiento y clasificación.
- Realizar un estudio de los algoritmos que utilizan las técnicas del Aprendizaje Basado en Instancias, incluyendo sus metodologías para la detección y tratamiento de los cambios de concepto.
- Diseñar e implementar un algoritmo de clasificación para flujos de datos utilizando las técnicas del Aprendizaje Basado en Instancias, capaz de resolver los problemas asociados a la velocidad, volumen y cambios de concepto de los flujos de datos.
- Analizar las políticas de actualización de modelos dinámicos, capaces de adaptarse a los cambios de tendencia en la información recibida.
- Validar el algoritmo propuesto con conjuntos de datos sintéticos, bases de datos internacionales y problemas reales de clasificación, utilizando tests estadísticos y medidas de evaluación válidas en este contexto.

1.3 Resumen

La minería de flujos de datos es un campo de estudio que supone nuevos desafíos a nivel mundial. La diseminación de este fenómeno ha necesitado el desarrollo de nuevos algoritmos y aplicaciones, donde muchos son los problemas por solucionar y optimizar aún.
Es importante contar con técnicas de Inteligencia Artificial (IA) en esta área de investigación, con el objetivo de optimizar los procesos y apoyar la toma de decisiones en tiempo real, ampliando así la funcionalidad de sus métodos.
Las técnicas de Aprendizaje Incremental son ampliamente utilizadas en esta área, pues presentan como característica fundamental su capacidad de incorporar nuevas experiencias y evolucionar la base de conocimiento obtenida, desde una estructura sencilla hacia otra más compleja.
En este tema, especialmente los algoritmos para resolver problemas de clasificación, representan una importante tarea, siendo los más tratados por la comunidad de investigadores, dada la necesidad creciente de su aplicación en problemas reales.
La presente investigación profundiza en el estudio de la aplicación de diferentes técnicas de clasificación de flujos de datos y realiza un análisis comparativo con una propuesta original basada en similitud.
Como principal resultado de la investigación se presenta la propuesta de un *Algoritmo de Clasificación para Flujo de Datos Basado en Similitud*, que almacena un conjunto de casos en memoria de forma organizada, manteniendo esta base de casos actualizada a partir de la política de inserción y eliminación que implementa, basada en el uso de estimadores diseñados para apoyar la correcta selección de los casos que deben ser almacenados y adaptándose a los cambios de concepto graduales y abruptos que puedan ocurrir en la función objetivo, a través de una metodología propia. Además se muestran su eficiencia y eficacia sobre conjuntos de datos sintéticos, bases de datos internacionalmente conocidas y ampliamente utilizadas pertenecientes a la UCI Machine Learning Repository (23), y problemas reales de flujos de datos, especialmente problemas desbalanceados.

1.4 Contribuciones del trabajo

En el transcurso de la investigación, algunos resultados han sido publicados, los cuales se presentan a continuación:

- Contribuciones de eventos:

    1. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz, Yanet Rodríguez Sarabia. Técnicas de Clasificación para Data Streams. Taller Internacional de SoftComputing. UCLV. Abril/09.Cuba. CD Evento.
    2. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz, Yanet Rodríguez Sarabia. Incremental Algorithm for Classification Based on Similarity. Memorias del Workshop on Machine Learning and Knowledge Discovery, Cuba-Flanders, Santa Clara, Villa Clara, Cuba 2010. CD Evento
    3. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz, Yanet Rodríguez Sarabia. Modelo de clasificación para data streams basado en similitud. CD Informática 2011. ISBN 978-959-7213-01-7. Cuba.
    4. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz, Yanet Rodríguez Sarabia. Classification Model for Data Streams Based on Similarity. IAE/EIA. Lecture Notes in Computer Science, 2011, Volume 6703/ 2011, 1-9. ISBN 978-3-642-21821-7.EEUU.
    5. Cosme Santiesteban Toca, Dayrelis Mena Torres. Algoritmo de Edición de Datos Biológicos para la Predicción de Estructuras de Proteínas. CD BioVeg 2011. ISBN 978-959-16-1286-1.Cuba.
    6. Mena Torres Dayrelis, Aguilar Ruiz Jesús S, Rodríguez Sarabia Yanet. DMPM: Data Streams classification model based on similarity, for data with concept change. COMPUMAT 2011. ISBN 978-959-250-658-9.Cuba.
    7. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz, Yanet Rodríguez Sarabia: An Instance Based Learning Model for Classification in Data Streams with Concept Change. MICAI (SpecialSessions) 2012: 58-62. ISBN 978-1-4673-4731-0.

8. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz, Yanet Rodríguez Sarabia. Algorithm for concept change treatment. Informática 2013. ISBN 978-959-7213-02-4.Cuba

- Contribuciones en revistas:

1. Dayrelis Mena Torres, Jesús S. Aguilar Ruiz. A similarity-based approach for data stream classification. Expert Systems With Applications, 41(9), pp. 4224-4234. 2014.
http://dx.doi.org/10.1016/j.eswa.2013.12.041

1.5 Estructura de la Tesis

La memoria está organizada según el esquema siguiente:

- En el capítulo 2, se introducen los conceptos básicos y características fundamentales acerca de los flujos de datos, así como los principales problemas que enfrentan los sistemas de aprendizaje en estos entornos y las técnicas más utilizadas para dar solución a los mismos.
- En el capítulo 3, se hace referencia a las Técnicas del Aprendizaje Basado en Instancias (IBL), haciendo énfasis en los principales elementos para su correcto desempeño, a partir de la caracterización y descripción de algunas medidas de distancia, analizadas para el diseño del modelo propuesto.
- En el capítulo 4, se describen las principales técnicas que se utilizan en la minería de flujos de datos, haciendo énfasis en los modelos que utilizan las técnicas del Aprendizaje Basado en Instancias (IBL), especialmente aquellos algoritmos utilizados para los análisis comparativos.
- En el capítulo 5, se describe con detalle el algoritmo propuesto, incluyendo la política de inserción/eliminación y los estimadores que ésta utiliza, así como la metodología para la detección y el tratamiento a los cambios de concepto. Se destacan someramente, además, aspectos de su implementación.
- En el capítulo 6, se presenta una extensa experimentación que demuestra la eficacia del algoritmo propuesto. Se describen los conjuntos de datos utilizados, las medidas de evaluación y los tests estadísticos aplicados para los análisis comparativos.
- En el capítulo 7, se muestran las conclusiones de la presente investigación mostrando cómo se cumplieron cada uno de los objetivos específicos trazados en la tesis. Además se sientan las bases de los trabajos futuros en el desarrollo de nuevos procedimientos para ampliar el espectro de soluciones del modelo propuesto.

Capítulo 2

FLUJOS DE DATOS

2.1 Aprendizaje en Flujos de Datos

Los flujos de datos son una secuencia ordenada de objetos que llegan a una velocidad que no permite almacenarlos de forma permanente en memoria. Son potencialmente ilimitados en tamaño por lo que es imposible analizarlos con la mayoría de los enfoques de la minería de datos existentes. Las principales características de los modelos de flujos de datos implican las siguientes limitaciones:

1. Es imposible almacenar todos los datos en memoria. Sólo pequeños resúmenes de los datos corrientes pueden ser seleccionados y almacenados; el resto de la información se elimina.

2. La velocidad de llegada de la secuencia de datos, obliga a cada elemento, en particular a ser procesado esencialmente en tiempo real y una sola vez.

3. La distribución de los datos de entrada, puede cambiar con el tiempo. Por lo tanto, los datos del pasado pueden resultar irrelevantes o incluso perjudiciales para el resumen actual.

La primera restricción limita la cantidad de memoria que pueden usar, los algoritmos que trabajan con flujos de datos, mientras que la segunda limita el

tiempo de procesamiento de un elemento. La restricción tres, es más importante en algunas aplicaciones que en otras; muchos de los primeros enfoques de minería de flujos de datos no centraron su atención en esta característica y formaron el grupo de algoritmos de aprendizaje de flujos de datos estáticos. Otras investigaciones consideran la restricción número tres, como el elemento clave y dedican su trabajo al aprendizaje de flujos de datos cambiantes.
Con estas nuevas características, las técnicas tradicionales de minería de datos no pueden ser directamente aplicadas a problemas de flujos de datos, pues muchas de ellas necesitan realizar múltiples pasadas por los datos con el objetivo de extraer el conocimiento oculto. Es necesario, entonces, el desarrollo de nuevas técnicas o la adecuación de las ya existentes para enfrentar los nuevos retos que imponen estas corrientes de información.
El modelo de flujo de datos difiere del modelo tradicional de conjunto de datos en los siguientes aspectos (47; 12):

- 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 parcialmente en el mejor 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 2.1.

Características	**Tradicional**	**Flujos de Datos**
Etapas de procesamiento	múltiples	una
Tiempo de procesamiento	ilimitado	restringido
Uso de la memoria	ilimitado	restringido
Tipo de resultado	exacto	aproximado
Concepto	estático	cambiante
Distribuido	no	si
Disponibilidad	total	parcial
Modelo de Conocimiento	constante	variable

Tabla 2.1: Diferencias entre el procesamiento de datos tradicional y el procesamiento de flujos de datos.

En el marco del aprendizaje supervisado, el modelado y la clasificación de flujos de datos puede considerarse una subárea del aprendizaje incremental donde las características antes mencionadas exigen una lógica algorítmica compleja y altamente eficiente para tomar decisiones críticas en tiempo real.
La Figura 2.1 ilustra la arquitectura y el enfoque seguido por los algoritmos para flujos de datos, donde el objetivo no es proporcionar un modelo de gran exactitud, sino garantizar una respuesta suficientemente buena en tiempos suficientemente aceptables.

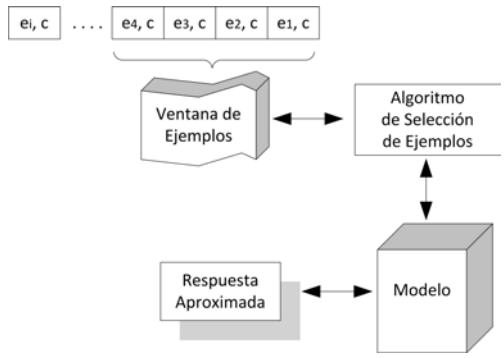

Figura 2.1: Flujo de información en modelos para Flujo de Datos.

Esta nueva filosofía impone considerables retos a muchas de las actuales aplicaciones de procesamiento y minería de datos por los costos computacionales y de

almacenamiento asociados al gran volumen de datos; sin embargo, no son éstos los únicos problemas que deben ser solucionados. El próximo epígrafe describe las principales características que deben tener los sistemas de aprendizaje en la minería de flujos de datos.

2.1.1 Problemas en el Aprendizaje de 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, entre otras, 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 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 dispositivos, podremos ver que el análisis y procesamiento en este tipo de entornos constituye realmente un problema a considerar.

- 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.

Esta última característica, sin duda alguna de vital importancia en entornos de aprendizaje de flujos de datos, será abordada de forma detallada en el próximo epígrafe.

2.1.2 La influencia del contexto

El principal problema al que se enfrentan los sistemas de aprendizaje en flujos de datos, a partir de su naturaleza evolutiva, es el hecho de que la función objetivo puede depender del contexto, el cual no es recogido mediante los atributos de los datos de entrada.
Un ejemplo clásico de ello es el problema de la predicción metereológica, donde los supuestos semánticos pueden variar radicalmente entre las distintas estaciones. Otro ejemplo es la obtención de patrones de compra entre los clientes de las grandes superficies, susceptibles a variaciones permanentes en función del día de la semana, la época del año, factores económicos como la inflación, la disponibilidad de los productos, etc. En tales situaciones, la causa del cambio se dice oculta y el problema se conoce como *hidden context*.
En consecuencia, cambios ocurridos en el contexto pueden inducir variaciones en la función objetivo, dando lugar a lo que se conoce como *concept drift* o *cambios de concepto*. Un factor decisivo en el rendimiento del sistema es el balance entre la robustez al ruido y la sensibilidad al cambio.
Algunos autores consideran el término *concepto* como a toda la distribución del problema en un cierto intervalo de tiempo (3). La distribución de un problema determinado, podemos definirla como una distribución conjunta $p(x, c)$, donde x son los atributos de entrada y c las clases, y puede estar representada por:

- la función de densidad de probabilidad incondicional $(pdf)p(x)$, y probabilidades posteriores $P(c_i|x), i = 1, 2, ..., n$. o
- por probabilidades a priori de las clases $P(c_1), ..., P(c_n)$, donde n es el número de clases existentes, y la clase condicional $(pdf)p(x|c_i), i = 1, ..., n$.

Adoptando esta definición de *concepto*, el término *cambio de concepto* representa cualquier cambio en la distribución del problema (51).
Otros autores consideran que el *concepto* se refiere a la variable que queremos predecir, que en el contexto de los problemas de clasificación, es la clase objetivo. Por lo tanto, se produce un *cambio de concepto* cuando las etiquetas de

clase de los ejemplos cambian con el tiempo (60).
Aunque muchos documentos no proporcionan definiciones explícitas y claras de *concepto* y *cambio de concepto*, podemos considerar la definición, que se utiliza en (46) y varios artículos posteriores, tales como (40; 60; 62; 9).
Para formalizar esta definición, utilizamos la distribución conjunta, considerando que los datos de entrada se pueden descomponer en: los valores de las variables $p(x)$ y la probabilidad de ocurrencia de la clase según el valor del atributo $P(c_i|x)$, tal que: $p(x, c_i) = p(x)P(c_i|x)$. De esta forma, un cambio en la clase objetivo c_i estaría asociado a un cambio en $P(c_i|x)$. Sin embargo, un cambio en $p(x)$ puede ocurrir solamente si existe un cambio en el contexto del problema, por lo que un cambio en $P(c_i|x)$ está siempre asociado a un cambio en $P(c_i)$ o en $p(x|c_i)$. En (60) se menciona que los cambios en la clase objetivo representan los cambios posteriores en las probabilidades $P(c_i|x)$ y en la clase condicional $(pdfs)p(x|c_i)$. Sin embargo, como podemos ver en los ejemplos anteriores, debe quedar claro que un cambio en la clase objetivo también puede estar asociado a un cambio en las probabilidades a priori de las clases $P(c_1), ..., P(c_n)$.
Un cambio en $p(x)$, cuando se trabaja con técnicas del aprendizaje incremental, puede causar un cambio en la decisión límite aprendida por un algoritmo. Un algoritmo puede aprender incrementalmente una frontera de decisión debido a que recibió ejemplos de un particular, $p(x)$. Sin embargo, un cambio en $p(x)$ puede revelar que la frontera de decisión actual ya no es buena, incluso si la clase objetivo de cada ejemplo no cambia. Un algoritmo de aprendizaje incremental debe ser capaz de adaptar su frontera de decisión cuando sea necesario.
Para agravar aún más las dificultades del aprendizaje incremental, en muchos dominios es de esperar que el contexto sea recurrente debido tanto a fenómenos cíclicos (p.ej., las estaciones del año para la predicción climatológica) como a fenómenos irregulares (p.ej., la tasa de inflación o la atmósfera de mercado para el modelado de perfiles de compra). Por tanto, junto a la robustez y la sensibilidad anteriormente mencionadas, el balance de un sistema incremental lo completa la capacidad de reconocer contextos recurrentes.
En función de la frecuencia con la que los nuevos ejemplos descriptivos de la nueva función objetivo son recibidos, en general se distinguen en la literatura dos tipos de *cambio de concepto* o *concept drift*: (1) abrupto (repentino, instan-

táneo) y (2) gradual. El cambio gradual puede dividirse a su vez, en moderado y lento.
En la práctica es irrelevante el tipo de cambio ya que todos producen un impacto en el modelo, en cuanto que aumentan el error del mismo con respecto a los ejemplos actuales y llevan a la necesidad de revisarlo constantemente.
La Figura 2.2 muestra los tres tipos de cambios a los que hacemos referencia.

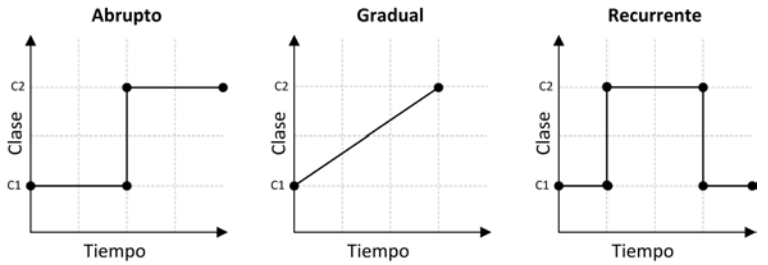

Figura 2.2: Representación de tipos de cambios de concepto.

El primer gráfico muestra un cambio de concepto abrupto, donde $p(x|c_i)$ cambia irreversiblemente, cambiando, en consecuencia, la probabilidad de asignación de la clase objetivo $P(c_i)$; un ejemplo en la vida real de este tipo de cambio, incluye los cambios de temporada en las ventas. El segundo gráfico (cambio incremental o gradual) ilustra los cambios que ocurren cuando las variables cambian lentamente sus valores en el tiempo; un ejemplo típico en este caso son los cambios en las definiciones de spam debido al interés creciente del usuario en las noticias. La tercera gráfica, representa los cambios recurrente, que son situaciones temporales que se repiten cada cierto tiempo. Este tipo de cambio es considerado por algunos investigadores como un cambio local (10) y ocurre cuando los datos de entrada reaparecen en intervalos de tiempo irregulares, sin embargo este cambio no es ciertamente periódico, pues no es seguro que la fuente vuelva a aparecer.
Es importante tener en cuenta que los tipos de cambios de concepto presentados, no son exhaustivos y que en la mayoría de las situaciones de la vida real, los cambios de concepto aparecen como una combinación compleja de muchos tipos de cambios. Sin embargo, es importante identificarlos, ya que el supuesto acerca de los tipos de cambio es absolutamente necesario para el di-

seño de estrategias de adaptabilidad. Los sistemas de aprendizaje en flujos de datos deben estar preparados para reaccionar ante ellos y adaptarse correctamente al nuevo concepto, por lo cual necesitan aprender de forma incremental a partir del flujo de información que llega constantemente.

2.2 Minería de Flujo de Datos

Las técnicas del aprendizaje incremental presentan dos características fundamentales que las hacen indispensable a la hora de trabajar problemas de flujos de datos: su capacidad de incorporar nuevas experiencias a la base de conocimiento (46), y su capacidad de evolucionar esta base de conocimiento, desde una estructura sencilla hacia otra más compleja (57).
Un número de algoritmos han sido propuestos para hacer frente a las características de los flujos de datos, utilizando diferentes técnicas. Dentro de éstas, las más abordadas en la literatura son la clasificación (77), el agrupamiento (35) y el reconocimiento de patrones frecuentes (28; 76; 39).
Dentro del aprendizaje incremental, especialmente los algoritmos para resolver problemas de clasificación, representan una importante tarea, siendo éstos los más tratados por la comunidad de investigadores, dada la necesidad creciente de su aplicación en problemas reales. Especial énfasis se está haciendo en la clasificación distribuida (63; 65), multi-clase (58) e incierta de flujos de datos desclasificados (86; 106; 78; 38).
Los modelos propuestos para clasificación de flujos de datos, en cualquiera de sus variantes, pueden ser clasificados en: simples y combinados, sin embargo todas las tareas de clasificación tienen un clasificador simple como base.
Algunos de los clasificadores más populares propuestos para la minería de datos estacionarios cumplen además con algunos de los requisitos de la minería de flujos de datos, ya que tienen las cualidades del aprendizaje en línea y cuentan con una "política de olvido" (*forgetting policies*). Algunos métodos que sólo son capaces de procesar datos de forma secuencial, pero no se adaptan, pueden ser modificados para reaccionar a cambios de concepto.
El objetivo principal del proceso de clasificación es decidir el tipo o clase de un

objeto, dado un conjunto de entradas. Los atributos que describen un objeto, en general, no necesitan ser directamente físicos, y con frecuencia se requiere algún tipo de pre–procesamiento para que el aprendizaje resulte efectivo.
A continuación se especifican las técnicas que han sido fundamentalmente utilizadas en los algoritmos de aprendizaje incremental para resolver problemas de clasificación en flujos de datos.

- Árboles de decisión: los árboles de decisión y el aprendizaje basado en reglas han sido de las técnicas más favorecidas en este área. El algoritmo más representativo de esta técnica es Very Fast Decision Tree (VFDT) (80). VFDT es un algoritmo de aprendizaje basado en árboles de decisión que se ajusta dinámicamente de acuerdo con la disponibilidad de datos. En esta técnica el problema principal es la decisión de cuándo extender el árbol, realizando pruebas de fraccionamiento y generando hojas nuevas. La idea básica consiste en usar un juego pequeño de casos para seleccionar el fraccionamiento adecuado a incorporar en un nodo del árbol de decisión. Si después de ver un juego de casos, la diferencia de mérito entre los dos mejores fraccionamientos no satisface una prueba estadística (el límite de Hoeffding), VFDT procede examinando más casos. Sólo toma una decisión (es decir, agrega un fraccionamiento en ese nodo), cuando hay suficiente evidencia estadística a favor de un fraccionamiento particular. El soporte estadístico disminuye mientras el árbol crece, y la regularización es obligatoria. En VFDT - como en otros algoritmos el número de ejemplos necesarios para crecer un nodo se define sólo por la importancia estadística de la diferencia entre las dos mejores alternativas. Los nodos más profundos del árbol podrían requerir más ejemplos que aquéllos usados en la raíz. Posteriormente algunas mejoras de este modelo han sido planteadas (52; 17; 86). Las mejoras propuestas descubren ciertas propiedades interesantes, sin embargo, el tiempo requerido para actualizar el árbol de decisión puede ser muy significativo, pues se necesita una gran cantidad de muestras para construir un clasificador con una precisión razonable. Cuando el tamaño del conjunto de entrenamiento es pequeño, el rendimiento del modelo puede resultar

insatisfactorio. Además, su estructura es inestable, pues un cambio ligero de la distribución de los datos puede traer como consecuencias cambios sustanciales en el árbol. Ejemplos de modelos basados en árboles de decisión lo constituyen además los trabajos: FlexDT (96) y eFTP (7).

- Redes Neuronales Artificiales: los modelos basado en redes neuronales han sido escasamente utilizados en entornos de flujos de datos, por sus limitaciones conceptuales. En una red neural, por ejemplo, una nueva observación causa una actualización en los pesos de la red, y esta influencia no puede ser cancelada simplemente después; en el mejor de los casos, su impacto puede reducirse gradualmente con el curso del tiempo. Además el costo computacional del aprendizaje del modelo hace que esta técnica no sea apropiada para entornos de flujos de datos. Algunas aplicaciones en el mundo real a través de redes neuronales, para la minería de flujo de datos son presentadas en LEARN (90) y Fuzzy–UCF (4).

- Aprendizaje simbólico (Reglas): en el aprendizaje basado en reglas, uno de los primeros modelos propuestos fue FACIL (Fast and Adaptive Classifier by Incremental Learning) (59), algoritmo dirigido a la clasificación de flujos de datos con atributos numéricos. Posteriormente otros algoritmos han sido publicados como OGA (83) y AC-DS (67). Estos sistemas aprenden reglas de forma incremental y emplean ventanas dinámicas para proporcionar mecanismos de olvido. Sin embargo el proceso de actualización de las reglas puede ser muy complejo y computacionalmente costoso.

- Aprendizaje basado en instancias: estos modelos, también llamados del vecino más cercano o de aprendizaje perezoso, proporcionan una forma precisa de aprendizaje, a travéz del mantenimiento de los ejemplos típicos de cada clase, teniendo en cuenta tres aspectos fundamentales (26): la función de similitud, la selección de los casos y la función de clasificación. La dificultad principal de esta técnica es combinar, de forma óptima, estos tres elementos. Un método de esta familia (IB3) presentado en (26), limita el número de datos históricos a almacenar, sólo a los más "útiles" para el proceso de clasificación. Aparte de reducir el

tiempo y los requisitos de memoria, esta limitación de tamaño del conjunto de referencia proporciona un mecanismo de olvido natural, ya que elimina ejemplos obsoletos para el modelo. Estos clasificadores, son fáciles de probar, conceptualmente sencillos, y son capaces de segmentar el espacio de manera compleja. Esta técnica ha sido ampliamente utilizada en sistemas de aprendizaje de flujos de datos, ejemplos lo constituyen los modelos, TWF y LWF (74), SlidingWindows (88), IBL-DS (48) e IBLStreams (6).

- Modelos Combinados (*Ensemble*): algunos clasificadores combinados han sido desarrollados recientemente (42; 107; 110; 41; 21); estas técnicas mantienen una cantidad fija de modelos y utilizan el voto combinado para clasificar nuevos casos de consulta. Sin embargo, tienen como desventaja, que aumentan el coste computacional y los requerimientos de memoria, necesitando la aplicación de métodos de eliminación de redundancias y paralelización de procesos; además, la inestabilidad del modelo paga un alto precio por aumentar la precisión y el riesgo de un tratamiento inadecuado podría causar pesos exagerados en ejemplos incorrectamente clasificados. Ejemplos recientes lo constituyen los algoritmos: AE (84) y EM (69).

2.2.1 Técnicas de Minería de flujos de datos

Para el correcto rendimiento de un algoritmo de clasificación para flujos de datos es necesario, de forma general, tener en cuenta algunos aspectos como: preprocesamiento o resumen del flujo de datos que permita mantener un conjunto representativo de casos en memoria para su consulta en cualquier momento; detección de cambios de concepto graduales o abruptos; y un mecanismo de adaptación a estos cambios que permita al modelo ajustarse a la corriente de información. A continuación son tratados algunos de estos aspectos.

2.2.2 Técnicas de Resumen

Para hacer frente a las limitaciones de recursos en entornos de flujos de datos, muchas técnicas de resumen de datos se han adoptado desde el campo de la estadística y de la teoría computacional.
Las técnicas de resumen se utilizan para producir respuestas aproximadas en grandes volúmenes de información. Éstas se refieren al proceso de transformación de los datos para su adecuado análisis, e incluyen las técnicas de reducción de datos, la construcción de sinopsis y las técnicas de ventanas, entre otras. Una excelente revisión de las técnicas de resumen de datos es presentada en (31). En las siguientes secciones se abordan algunas técnicas de interés para el presente trabajo.

2.2.2.1 Técnicas de ventana

Un enfoque muy popular dentro de las técnicas de resumen, para tratar flujo de datos cambiantes en el tiempo, es el uso de ventanas deslizantes. Estas proporcionan una manera de limitar la cantidad de ejemplos introducidos para el aprendizaje, eliminando los datos que provienen de un concepto antiguo.
El procedimiento básico de uso de ventanas deslizantes para minería de flujo de datos se presenta en el Algoritmo 1.

Algoritmo 1 Algoritmo Básico de Ventana

entrada S: Flujo de datos de ejemplos; W: Ventana de ejemplos
salida C: Clasificador construido con el flujo de datos en la ventana W

1: Inicializar ventana W
2: **para todo** ejemplo $x_i \in S$ **hacer**
3: $W \leftarrow W \cup \{x_i\}$
4: eliminar ejemplos antiguos de W, si es necesario
5: reconstruir/actualizar C utilizando W
6: **fin para**

Este algoritmo de ventana básico es sencillo, cada ejemplo actualiza la ventana y luego el clasificador es actualizado por esa ventana. La clave de este algoritmo radica en la definición de la ventana, es decir, en la forma en que se modela el proceso de olvido. En el enfoque más simple las ventanas deslizantes

son de tamaño fijo e incluyen sólo los ejemplos más recientes del flujo de datos. Con cada nuevo ejemplo se elimina el ejemplo más antiguo que no cabe en la ventana.
Con el uso de ventanas de tamaño fijo, el usuario se ve atrapado en una disyuntiva. Si elige un tamaño pequeño de ventana el clasificador reaccionará rápidamente a los cambios, pero puede perder precisión en los períodos de estabilidad; por otra parte, la elección de un tamaño grande dará lugar a una precisión cada vez mayor en períodos de estabilidad, pero no se adaptará a los cambios de conceptos rápidos, respondiendo más lentamente. Ésa es la razón por la cual han sido propuestas algunas formas dinámicas de modelar el proceso de olvido.

2.2.2.2 Ventanas con pesos

Una manera sencilla de hacer el proceso de eliminación más dinámico es proporcionando a la ventana una función que asigna un peso a cada ejemplo. Los ejemplos más antiguos reciben los pesos más pequeños y son tratados como menos importantes por el clasificador base. En (30) se analiza el uso de diferentes funciones para el cálculo de los pesos en flujos de datos. Las ecuaciones 2.1, 2.2, 2.3, presentan las funciones propuestas.

$$w_{exp}(t) = e^{-\lambda t}, \lambda > 0 \tag{2.1}$$

$$w_{poly}(t) = \frac{1}{t^{\alpha}}, \alpha > 0 \tag{2.2}$$

$$w_{chord}(t) = 1 - \frac{t}{|W|} \tag{2.3}$$

La Ecuación 2.1 presenta una función exponencial, 2.2 una función polinómica, y 2.3 una función de acordes. Para cada una de las funciones, t representa la edad de un ejemplo. Un nuevo ejemplo tendría $t = 1$ mientras que el último ejemplo que encaja cronológicamente en una ventana tendría $t = |W|$, donde $|W|$ es la cantidad de ejemplos de la ventana.
El uso de estas funciones permite ponderar gradualmente los ejemplos, ofreciendo un balance entre ventanas fijas, grandes y pequeñas. El Algoritmo 2 presenta el proceso de obtención de una ventana con asignación de pesos.

Algoritmo 2 Algoritmo de Ventanas con Pesos

entrada S: Flujo de datos de ejemplos; k: Tamaño de la Ventana; $w(\cdot)$: Función de Peso
salida W: Ventana de ejemplos

```
1: W ← {}
2: para todo ejemplo x_i ∈ S hacer
3:    si |W|= k entonces
4:       eliminar ejemplo más antiguo de W
5:    fin si
6:    W ← W ∪ {x_i}
7: fin para
8: para todo ejemplo x_j ∈ W hacer
9:    calcular peso del ejemplo w(x_j)
10: fin para
```

2.2.3 Detectores de Cambio de Concepto

Después de las técnicas de resumen, otra variante que permite a cualquier modelo adaptarse al contexto son los detectores de cambio de concepto. Su tarea consiste en detectar los cambios de concepto y alertar al clasificador que su modelo debe ser reconstruido o actualizado.

Una forma muy popular es utilizando pruebas estadísticas, que verifican si el error de ejecución o distribución de clases se mantienen constantes en el tiempo. Para secuencias numéricas, las primeras pruebas estadísticas propuestas fueron: la Suma Acumulada (CUSUM) (32) y la Media Móvil Geométrica (GMA) (99). La prueba CUSUM plantea una alarma si la media de los datos de entrada es significativamente diferente de cero, mientras que GMA comprueba si el promedio ponderado de los ejemplos de una ventana es mayor que un umbral dado. La prueba de Kolmogorov-Smirnov también ha sido propuesta, para poblaciones más complejas. En (82) se presenta un detector de cambios de concepto no supervisado que utiliza el test no paramétrico de Kolmogorov-Smirnov, para comprobar si un segmento de datos tiene la misma distribución que los datos de referencia.

En (51) se propone un método de detección de cambio de concepto (DDM, Drift Detection Method) que se basa en el hecho de que, en cada iteración, un clasificador predice la clase de un ejemplo; esa predicción puede ser verdadera o

falsa, por tanto, para un conjunto de ejemplos el error es una variable aleatoria de Bernoulli, por eso los autores modelan el número del error de clasificación con una distribución binomial.
Denotemos P_i como la probabilidad de una predicción *falsa* y s_i como su desviación estándar calculada como propone la Ecuación 2.4

$$s_i = \sqrt{\frac{P_i(1-P_i)}{i}} \tag{2.4}$$

Los autores utilizan el hecho de que, para un número suficiente de ejemplos ($n > 30$), la distribución binomial se aproxima a una distribución normal con igual media y varianza. Cada ejemplo del flujo de datos actualiza dos registros en la tasa de error: p_{min} y s_{min}. Estos valores se utilizan para calcular una condición de advertencia que se presenta en la Ecuación 2.5 y una condición de alarma que se presenta en la Ecuación 2.6. Cada vez que un nivel de advertencia se alcanza, los ejemplos se guardan en una ventana separada. Si posteriormente la tasa de error disminuye por debajo del umbral de advertencia, este caso se trata como una falsa alarma y la ventana separada se elimina. Sin embargo, si el nivel de alarma es alcanzado, el modelo previamente aprendido se elimina y uno nuevo es creado, pero sólo utilizando los ejemplos almacenados en la ventana apartada de "advertencia".

$$p_i + s_i \geq p_{min} + \alpha * s_{min} \tag{2.5}$$

$$p_i + s_i \geq p_{min} + \beta * s_{min} \tag{2.6}$$

Los valores α y β en las condiciones anteriores, definen los niveles de confianza de las señales de advertencia y alarma. Los autores proponen $\alpha = 2$ y $\beta = 3$, dando aproximadamente el 95% de confianza de advertencia y el 99% de confianza de alarma.
Este método funciona mejor en flujos de datos con cambios de concepto abruptos, pues los cambios graduales pueden pasar sin desencadenar el nivel de alarma. Cuando no se detectan cambios, DDM amplia constantemente el tamaño de la ventana, lo que puede conducir a que el límite de memoria se exceda.
En (70) se propone una modificación al modelo DDM llamado EDDM (Early

Drift Detection Method). En este trabajo los autores utilizan el mismo mecanismo de advertencia-alarma, pero en lugar de utilizar la tasa de error del clasificador, proponen la tasa de error de la distancia. Denotan P'_i como la distancia media entre dos errores consecutivos y s'_i como su desviación estándar. Con estos valores las nuevas condiciones de advertencia y alarma están dadas por las Ecuaciones 2.7 y 2.8.

$$\frac{P'_i + 2 * s'_i}{P'_{max} + 2 * s'_{max}} < \alpha \tag{2.7}$$

$$\frac{P'_i + 3 * s'_i}{P'_{max} + 3 * s'_{max}} < \beta \tag{2.8}$$

EDDM funciona mejor que DDM para cambios de concepto graduales, pero es más sensible al ruido. Otro inconveniente de este método es que considera los umbrales y la búsqueda de los cambios de concepto cuando se han producido un mínimo de 30 errores. Esto es necesario para aproximar la distribución binomial a una distribución normal, pero puede tomar una gran cantidad de ejemplos para que suceda.

2.2.4 Mecanismos de Olvido

Una tarea importante es definir qué hacer luego de detectar un cambio de concepto. Un clasificador de flujo de datos debe ser capaz de reaccionar a estos cambios, olvidando comportamientos obsoletos, mientras aprende nuevas descripciones de clase. El problema principal es cómo seleccionar el rango de datos que debe recordar o los que debe eliminar. La solución más simple es olvidar objetos de entrenamiento a una frecuencia constante y utilizar una ventana de los ejemplos más recientes para entrenar el clasificador. Este enfoque, se ve atrapado en una disyuntiva entre estabilidad y flexibilidad, asociada a las técnicas de ventana con parámetros fijos, descritas anteriormente.
Un enfoque diferente consiste en el envejecimiento a una tasa variable. Esto puede significar cambiar el tamaño de la ventana cuando se detecta un cambio de concepto o el uso de funciones de asignación de pesos para diferenciar el impacto de los datos de entrada, tratando de modificar dinámicamente los parámetros de la ventana para encontrar el mejor equilibrio entre la precisión

y la flexibilidad. Por lo general, es más adecuado para entornos con cambios de concepto abruptos, cuando el cambio es más fácil de detectar.
Otro enfoque para olvidar datos obsoletos y potencialmente perjudiciales es la selección de ejemplos de acuerdo a su distribución de clases. Esto es especialmente útil en situaciones en las que los datos antiguos pueden ser más relevantes que los datos recientes. Este enfoque, considerado en los modelos del vecinos más cercano, utiliza pesos basados en el tiempo, pero puede ser modificado dependiendo de la vecindad de los ejemplos más recientes.
No hay una solución genérica para implementar un mecanismo de olvido. Dependiendo del entorno y los tipos de cambio de concepto esperados, los diferentes enfoques pueden funcionar en mejor o peor medida.
Para flujos de datos con cambios de concepto graduales, las ventanas estáticas deben dar la máxima precisión. Mientras que para cambios abruptos se necesitaría un sistema de envejecimiento a una tasa variable.

2.3 Aplicaciones

En esta sección se presentan algunos de los principales dominios de aplicación de los flujos de datos en que el cambio de concepto juega un papel importante. Junto con las características de cada dominio se analizan las fuentes del cambio en estos problemas. Una presentación más detallada sobre los dominios de aplicación de flujos de datos con cambios de concepto se puede encontrar en (45).

- Sistemas de monitorización. Los sistemas de vigilancia se caracterizan por grandes flujos de datos que necesitan ser analizados en tiempo real. Las clases en estos sistemas suelen ser muy desbalanceadas, lo que hace que la tarea sea aún más complicada. El objetivo principal de un sistema de monitorización consiste en distinguir situaciones "no deseadas" de "conducta normal". Esto incluye el reconocimiento de acciones adversas y la alerta antes de la ocurrencia de posibles estados críticos del sistema.

- Seguridad de la red. La detección de accesos no deseados a un ordenador, también llamado *detección de intrusos*, es uno de los problemas típicos de monitorización. Los sistemas de detección de intrusos filtran el tráfico de red entrante en busca de comportamientos sospechosos. Los cambios de concepto en este problema están relacionados sobre todo con los ataques que pueden producirse. Las acciones realizadas por el intruso pueden evolucionar con el tiempo, superando los también cambiantes sistemas de seguridad. El progreso tecnológico es otra fuente de cambios de concepto, ya que los sistemas modernos que ofrecen más funcionalidades a menudo ofrecen más posibilidades de ataque (100; 5).

- Finanzas. Las transacciones financieras pueden ser analizadas para alertar sobre posibles fraudes en tarjetas de crédito y en transacciones bancarias por Internet. Los mercados de valores también utilizan técnicas de minería de flujos de datos para evitar operaciones con privilegios. En ambos casos, el etiquetado de los datos podría ser impreciso debido a los fraudes no notificados y a transacciones legítimas mal interpretadas. Al igual que en los ejemplos anteriores, la fuente del cambio de concepto es la evolución del comportamiento de los usuarios. Esto es especialmente difícil con operaciones privilegiadas donde el intruso posee información no pública sobre una compañía y trata de distribuir sus transacciones de una manera no trivial (93).

- Transporte. Las técnicas de minería de flujos de datos pueden emplearse para monitorizar y predecir los estados de tránsito y transporte público en tiempo real. Esta información puede ser útil para la programación y planificación del tráfico, logrando reaccionar dinámicamente a los atascos de tráfico. Los patrones de tráfico pueden cambiar estacionalmente, así como permanentemente, por lo tanto los sistemas tienen que ser capaces de manejar estos cambios de concepto. Los factores humanos de conducción también pueden ser una importantes fuente de cambios (56).

- Vigilancia Industrial. Hay varias aplicaciones emergentes en el área del control de sensores, donde un gran número de sensores se distribuyen

físicamente y generan flujos de datos que necesitan ser combinados, monitorizados, y analizados. Tales sistemas se utilizan para controlar el trabajo de los operadores de máquinas y para detectar los fallos del sistema. En el primer caso, los factores humanos son la principal fuente de cambios de concepto, mientras que en el segundo, lo es el cambio del contexto de los sistemas (79).

- Filtrado de spam. El filtrado de spam es un tipo más complicado de filtrado de información, debido a que está más abierto a las acciones del adversario (spamming). Los adversarios se adaptan y cambian rápidamente de spam para superar los filtros. La cantidad y tipos de correo ilegítimo están sujetos a la estacionalidad, pero también cambian irregularmente en el tiempo. La definición de spam también puede diferir entre unos usuarios y otros (44).

- Economía. Las previsiones macroeconómicas y las series de tiempo financieras también tienen aplicación en la minería de flujos de datos. Los datos de estas aplicaciones están sujetos a cambios de concepto debido a un gran número de factores que no están incluidos en el modelo. La información de conocimiento público sobre las empresas puede formar sólo una pequeña parte de los atributos necesarios para modelar correctamente previsiones financieras como un problema estacionario. Es por ello que la principal fuente del cambio es un contexto oculto.

- Aplicaciones biomédicas. Las aplicaciones biomédicas presentan un interesante campo de investigación en problemas con cambios de concepto, dada la naturaleza adaptativa de los microorganismos, debido a sus mutaciones, su resistencia a los antibióticos, cambia. Los pacientes tratados con antibióticos, cuando no es necesario, pueden llegar a ser "inmunes" a su acción cuando realmente se necesita.

- Hogares inteligentes y realidad virtual. Los electrodomésticos inteligentes del hogar deben ser adaptables a cambios en el entorno y a las necesidades de los usuarios. También la realidad virtual necesita mecanismos para tener en cuenta los cambios de concepto. Por ejemplo, en los juegos de

ordenador y simuladores de vuelo, la realidad virtual debe adaptarse a las capacidades de los diferentes usuarios y evitar acciones adversas, como hacer trampa (22).

2.4 Conclusiones Parciales

A partir del análisis realizado podemos concluir que:

- La minería de flujos de datos, es un campo de estudio que supone nuevos desafíos para la comunidad de investigadores a nivel mundial. La diseminación de este fenómeno, necesita el desarrollo de métodos donde la utilización de las técnicas de Inteligencia Artificial podría significar un salto cualitativo en la optimización de los procesos.

- El problema principal, al que se enfrentan los algoritmos para clasificación en el contexto de flujos de datos, son las posibles variaciones en la función objetivo, las cuales han sido clasificadas según diferentes criterios. Un sistema para el tratamiento de estos conjuntos de datos debe estar preparado para detectar y resolver estos cambios de conceptos.

- Para hacer frente a las características de los flujos de datos es necesario definir tareas para: el preprocesamiento o resumen del flujo de datos, la detección de cambios de concepto graduales o abruptos; y un mecanismo de adaptación a estos cambios que permita al modelo ajustarse a la corriente de información.

- Aplicaciones reales de flujos de datos en la industria, la economía, las finanzas y la medicina, entre otras, demandan el desarrollo de algoritmos y modelos capaces de enfrentar los retos impuestos por esta nueva filosofía.

Capítulo 3

APRENDIZAJE BASADO EN INSTANCIAS

3.1 El Aprendizaje Basado en Instancias

Los algoritmos de Aprendizaje Basados en Instancias (IBL) (26), se engloban dentro de las técnicas del Aprendizaje Perezoso donde el proceso de inducción es diferido hasta la fase de clasificación, es decir, a diferencia del resto de las técnicas de clasificación, el conjunto de entrenamiento no es sometido a ningún proceso de inducción para crear un modelo, puesto que son los ejemplos del mismo, el propio modelo.
A pesar de su sencillez conceptual y su facilidad de implementación, las técnicas basadas en el vecino más cercano presentan dos severos inconvenientes: elevada complejidad computacional y fuerte dependencia paramétrica.
La elevada complejidad computacional es debido a que, por cada nueva predicción hay que procesar todo el conjunto de entrenamiento para encontrar los k ejemplos más próximos al nuevo ejemplo de test. Para acelerar este proceso han sido propuestas numerosas técnicas que responden a tres estrategias:

- Búsquedas aproximadas.
- Estructuras de datos eficientes.
- Reducción del número de ejemplos de entrenamiento.

Las técnicas basadas en búsquedas aproximadas no calculan la distancia del ejemplo a clasificar a todos los demás sino que, en base a una heurística, ignoran varios de ellos, calculando así un vecino que se aproxima a su más cercano. Dentro del segundo grupo cabe destacar las propuestas basadas en KD-Tree, que consiguen reducir la complejidad de n^2 a $n \log n$.
Por último, las técnicas pertenecientes al último grupo intentan reducir en la medida de lo posible el número de ejemplos del conjunto de entrenamiento. Para ello existen dos enfoques: selección de ejemplos internos o centrales y selección de ejemplos externos o fronterizos. Los ejemplos internos son aquellos que pueden ser clasificados correctamente con la regla k-NN para un valor reducido de k, es decir, están rodeados en su mayoría por ejemplos de su misma etiqueta. Por el contrario, los ejemplos externos son aquellos que no pueden clasificarse correctamente para varios valores de k. En ambos enfoques, el conjunto de ejemplos seleccionados clasifica correctamente a los que son descartados.
La fuerte parametrización se debe a que el resultado obtenido depende en gran medida del valor de k y de la métrica de similitud utilizada, convirtiéndola en una técnica que exige al usuario el conocimiento del problema. Para suavizar esta dependencia suele aplicarse validación cruzada con conjuntos de entrenamiento de tamaño reducido para encontrar el valor de k que mayor exactitud proporciona.
El éxito de esta técnicas y su aplicación en un dominio particular depende en buena medida de la correcta combinación de sus tres características fundamentales:

- Función de similitud: Esta dice al algoritmo cómo de cercanas son dos instancias. Existe una gran complejidad en la elección de la función de similitud, especialmente en situaciones en las que algunas entradas son simbólicas.

- Función de selección de las instancias típicas: Esto indica al algoritmo cuál de las instancias retener como ejemplos típicos.

- Función de clasificación: Esta función es la que cuando se administra un nuevo caso, decide cómo se relaciona con los casos aprendidos. Por ejemplo, podría definirse como la instancia más cercana a su ubicación (1-NN).

3.1.1 El vecino más cercano

La regla del vecino más cercano 1-NN formulada en (101) para su aplicación en problemas de clasificación, es enunciada como sigue:
Definición 1: Dados un conjunto de entrenamiento Γ *y un ejemplo de test* e *en un espacio métrico m-dimensional, el vecino más cercano de* e *es aquel ejemplo* $e' \in \Gamma$ *a menor distancia de* e*, siendo* e *clasificado según la etiqueta asociada a* e'*.*
El esquema de aprendizaje del vecino más cercano se basa en una simple suposición: definida una medida de similitud d, para cualesquiera dos ejemplos $e_p = (x_p; y_p)$, $e_q = (x_q; y_q)$ del conjunto de entrenamiento se representa en la exprexión 3.1:

$$P(y_p|x_p) \approx P(y_q|x_q) \Leftrightarrow d(x_p, x_q) \approx 0 \tag{3.1}$$

Bajo esta hipótesis, toda información descriptiva - adicional o desconocida - que quiera extraerse de un individuo puede observarse en otras instancias similares, sus vecinos más cercanos.
Una generalización inmediata es la regla k-NN, según la cual el ejemplo de test es clasificado con la etiqueta mayoritaria de sus k vecinos más cercanos. El valor del parámetro k suele ser impar para evitar los empates y pequeño para observar la región del espacio próxima al nuevo ejemplo a clasificar.
Para medir la similitud entre dos ejemplos la mayoría de las técnicas utilizan una función de distancia para los atributos continuos y otra para los atributos nominales, así el modelo puede aplicarse a todo tipo de atributos. Las funciones de distancia han sido muy tratadas en la literatura, proponiéndose una variedad de ellas.

3.1.2 La Similitud Basada en la Distancia

La distancia y la similitud son dos conceptos relacionados (105; 11), y desde el punto de vista matemático son conceptos duales.
En (87) se abordan dos principios relacionados estrechamente con la definición de la similitud. El primero se refiere al principio local-global, el cual asume que las descripciones de los objetos se construyen por partes (generalmente atributos), y que una medida global (en el espacio m-dimensional) también se obtiene por medidas locales (unidimensional).
Con esta finalidad se pueden utilizar las métricas de Minkowsky (13); de las cuales una de la más utilizada es la distancia Euclideana $d_E : X \times X \longrightarrow R^+$, cuya expresión se presenta en la ecuación 3.2:

$$d_E(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{3.2}$$

El conocimiento que encierra la medida de similitud, hace que en su definición siempre se trate de aproximar a la utilidad (87). Es decir, mientras mayor sea la similitud entre la solicitud y un ejemplo, más útil es este ejemplo para la solicitud. El aprendizaje en una función de similitud es generalmente atendido por la adaptación de parámetros, y un típico parámetro es el peso de los rasgos (27).
Muchos investigadores han focalizado su atención en el uso de métricas locales (25; 81; 19) con el objetivo de superar los sistemas basados en métricas globales. Por un lado las métricas locales generan clasificadores que pueden ser más sensibles a cambios locales de los datos y consecuentemente más precisos, y por otro lado, las métricas globales tienen pocos parámetros y los clasificadores resultan menos propensos al efecto de datos ruidosos. El punto crítico radica en precisar el grado de localidad de la métrica en áreas diferentes del espacio de entrada.
Las métricas locales, son generalmente dependientes de la naturaleza simbólica, ordinal o numérica del rasgo en cuestión.
Sean $x, y \in X$ dos valores posibles de un atributo arbitrario. La ecuación 3.3 se refiere a la distancia Euclideana normalizada (local), y solamente puede ser

utilizada si el atributo es numérico, donde l y u se refieren a los valores mínimo y máximo del dominio del atributo respectivamente:

$$d_{norm}(x, y) = \frac{|x - y|}{l_a - u_a} \tag{3.3}$$

La normalización se requiere cuando estos criterios de comparación locales se utilizan en el contexto de funciones de comparación entre dos instancias para evadir la situación de que el rasgo a comparar tenga un rango relativamente grande que lo puede hacer prevalecer sobre el resto de los atributos que se consideren a los efectos de cuantificar la similitud entre dos instancias. En (81) se hace referencia al significado geométrico de estas distancias y a las características del modelo que harían factible su aplicación.
Otros casos se refieren a los atributos simbólicos, donde no se puede definir un orden entre los valores asociados a sus categorías ontológicas e incluso a sus códigos formales. Una distancia de comparación local muy simple y frecuentemente utilizada es la ecuación 3.4:

$$d_{overlap}(x, y) = \begin{cases} 0, & \text{si } x = y \\ 1, & \text{en otro caso} \end{cases} \tag{3.4}$$

El término atributos mezclados o heterogéneos se refiere a la coexistencia de atributos numéricos y simbólicos en la descripción de una instancia en más de una dimensión. La distancia Heterogeneus Euclidean-Overlap Metric (HEOM) definida en la ecuación 3.5 permite manejar esta situación al combinar las ecuaciones 3.4 y 3.3.

$$d_{HEOM}(x, y) = \begin{cases} 1, & \text{si } x =? \text{ ó } y =? \\ d_{overlap}(x, y), & \text{si el atributo es simbólico} \\ d_{norm}(x, y), & \text{si el atributo es numérico} \end{cases} \tag{3.5}$$

Nótese que en este criterio si alguno de los valores a comparar es desconocido, se adopta que la distancia es la máxima posible. La comparación de atributos mezclados por la combinación de estos criterios define una métrica heterogénea y poco consistente.

3.1.3 Métricas Basadas en la Relación con el Objetivo

En el contexto del Aprendizaje Basado en Instancias, muchas propuestas se basan también en la estimación de probabilidades (25; 19; 29; 92). La distancia más conocida de este tipo fue propuesta por (19), denominada Value Difference Metric (VDM). El criterio de comparación local de esta función para definir la distancia entre dos valores x, y de un atributo se especifica en la ecuación 3.6:

$$d_{vdm}(x,y) = \left(\sum_{l=1}^{c} |P_r\left(C_l \mid x\right) - P_r\left(C_l \mid y\right)| \right)^{\frac{1}{q}} \tag{3.6}$$

donde c es el número de clases del problema, q es una constante (con valores 1 ó 2 generalmente), y C_l un valor posible para el atributo clase. El término $P_r(C \mid x)$ se refiere a la probabilidad condicional de que la clase de salida sea C, conocido que el atributo tiene valor x; que se puede estimar a partir de un conjunto de instancias conocidas de acuerdo a la ecuación 3.7:

$$P_r\left(C \mid x\right) = \frac{N_{x,C}}{N_C} \tag{3.7}$$

cuya interpretación se refiere al número de instancias en las cuales los valores x y C aparecen simultáneamente (N_x, C), relativo a la presencia de x (frecuencia relativa).

VDM tiene en cuenta la relación de cada rasgo con el rasgo objetivo; la cual se referencia en (27) como uno de los criterios de similitud más sofisticados para comparar atributos discretos. Un estudio experimental (36) utilizando la regla de los k vecinos más cercanos y la variante no pesada (todos los rasgos influyen de igual manera en la similitud global) muestra porcentajes de clasificaciones correctas para VDM al menos tan buenos como con el modelo C4.5 (57) (técnica basada en árboles de decisión); mientras que en (102) se constata que VDM reduce el impacto de atributos irrelevantes sobre la precisión de la clasificación sin necesidad de pre-procesamiento de los datos.

El criterio de comparación local de VDM tiene las ventajas siguientes: tiene como imagen un valor normalizado en el intervalo $[0, 1]$; mientras la mayoría

de las funciones de distancia sobre valores discretos requieren valores binarios o simplemente cuentan el número de no coincidencias, el resultado de VDM depende del valor individual que tome el rasgo (es más informada), y el valor ausente se trata como si éste fuese otra categoría de las posibles. Sin embargo, esta función sólo es aplicable a problemas de aprendizaje supervisado con atributos simbólicos.

En (25) se hace notar que un problema de VDM es que no precisa qué hacer cuando en la solicitud se presentan valores que no aparecen en el conjunto de entrenamiento, y se proponen varias extensiones a VDM para extender su aplicación a atributos numéricos. La variante más simple para lograr lo anterior es emplear la discretización (64). En este caso el dominio X de un rasgo de tipo numérico se particiona en intervalos, transformando el atributo a tipo ordinal por la asociación de cada valor a uno de los intervalos definidos. Esta distancia se nombra como Discretized VDM (DVDM), y se define en la ecuación 3.8:

$$d_{DVDM}(x,y) = \sqrt{\sum_{l=1}^{c} \left(P_r\left(C_l \mid A\right) - P_r\left(C_l \mid B\right)\right)^2} \tag{3.8}$$

donde c es el número de clases del problema y A, B representan intervalos correspondientes a x e y definidos sobre el conjunto X de valores posibles. Nótese que la medida $\mathrm{d}(x,y)$ para todos los valores $x \in A$, $y \in B$ por la expresión anterior es la misma. Luego, DVDM tiene como desventaja fundamental que con la discretización se pierde información importante disponible en los valores continuos, lo cual puede llevar a que, por ejemplo, los valores extremos de un mismo intervalo sean considerados iguales; repercutiendo negativamente en la precisión de la generalización.

Otra extensión es Heterogeneous VDM (HVDM), que combina la distancia Euclideana para atributos numéricos y VDM para rasgos nominales, de manera similar a HEOM. Esta variante maneja de forma efectiva conjuntos de datos con ambos tipos de atributos, según la cual, dados dos ejemplos x y y, la distancia entre ellos viene dada por la ecuación 3.9:

$$d_{HVDM}(x,y) = \sqrt{\sum_{j=1}^{m} d_{A_j}(x_j, y_j)^2} \tag{3.9}$$

$$d_{A_j}(x_j, y_j) = \begin{cases} 1, & \text{si } x_j \text{ ó } y_j \text{ es desconocido} \\ d_{vdm}(x_j, y_j), & \text{si } A_j \text{ es nominal} \\ \frac{|x_j - y_j|}{4\sigma_{A_j}}, & \text{si } A_j \text{ es continuo} \end{cases} \tag{3.10}$$

donde x y y son dos ejemplos, σ_{A_j} es la desviación típica de los valores del atributo A_j según los ejemplos del conjunto de entrenamiento. Esta medida requiere de un especial cuidado con la normalización, de manera que permita lograr una escala en un mismo rango a partir de dos tipos de medidas diferentes.

Las restantes generalizaciones: Interpolated VDM (IVDM) y Windowed VDM (WVDM), utilizan la interpolación a los efectos de evadir los problemas de la discretización y se diferencian esencialmente en las técnicas para determinar los valores de probabilidad para cada clase.

Se han propuesto algunas métricas locales, que además de estimar algunas probabilidades, basan su efectividad en la optimización de un criterio dado: la métrica de (91), y la Métrica de Riesgo Mínimo (29).

Ambas variantes se aplican a un espacio de entrada multidimensional y continua; entre éstas la variante presentada en (91) tiene una fuerte fundamentación teórica; aunque inicialmente fue utilizada sólo para problemas con atributos numéricos, hace posible su extensión a atributos simbólicos o heterogéneos considerando estimadores de probabilidad más generales.

En (29) se mencionan varias ventajas de este enfoque para definir métrica, entre las que se destacan: las métricas resultantes tienen una clara expresión analítica y motivación, se pueden calcular utilizando técnicas de estimación de densidad estandarizadas, pueden ser definidas sobre conjuntos de datos uniformes con rasgos tanto numéricos como simbólicos. Sin embargo, este último aspecto se hace posible por transformar los datos al tipo que cumple la métrica a usar. Por ejemplo, numerar y ordenar rasgos nominales para aplicar una métrica continua o por la combinación de dos métricas (ejemplo: Euclideana y de Hamming) en una métrica heterogénea difícil de adaptar de una manera consistente; con sus correspondientes desventajas (43).

Una extensión más reciente a VDM denominada NCM (Neigborhood Coun-

ting Metric) (43) se basa en interpretar el concepto de vecindad utilizando el concepto de hipertuplas, en lugar de la distancia, y no se limita a problemas de clasificación.

3.2 Conclusiones Parciales

A partir del análisis realizado podemos concluir que:

- Las técnicas del Aprendizaje Incremental, específicamente el Aprendizaje Basado en Instancias, puede ser aplicado ampliamente a la problemática de la clasificación en flujos de datos con el objetivo de apoyar el proceso de toma de decisiones en tiempo real, ampliando así las funcionalidades de sus algoritmos y aplicaciones.

- Existen muchas formas de medir la similitud y no existe una implementación adecuada de este concepto para todos los dominios de aplicación. En particular, el algoritmo de los k vecinos más cercanos es muy sensible a la definición del criterio de cercanía, y aunque cuenta con algunas desventajas, existen muchas variantes para maximizar su eficacia.

- La métrica basada en la diferencia de valores (VDM) ha mostrado su efectividad para comparar atributos simbólicos en problemas de clasificación, y ha motivado el desarrollo de nuevas versiones que la extienden para comparar también atributos numéricos. Esta medida tiene en cuenta conocimiento del dominio de aplicación profundizando en la relación con el objetivo, y se implementa utilizando la estimación de probabilidades condicionales a partir de un conjunto de instancias conocidas.

- La métrica definida como Heterogeneous VDM (HVDM), que combina la distancia Euclideana para atributos numéricos y VDM para rasgos nominales, hereda las ventajas de las medidas que la componen y maneja de forma efectiva conjuntos de datos con ambos tipos de atributos.

Capítulo 4

ESTADO DEL ARTE

4.1 Introducción

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 los retos que imponen estos 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 como la extracción implícita, no trivial, previamente desconocida y potencialmente útil de patrones y conocimiento de un gran volumen de datos (54). En el contexto de los flujos de datos, la utilización de técnicas como el agrupamiento, clasificación y detección de patrones frecuentes, se han convertido en temas de interés activo para la investigación en las últimas décadas.
El objetivo de este capítulo es hacer un estudio sobre las principales características de los modelos propuestos para la minería de flujos de datos.

4.2 Algoritmos de Minería de flujos de datos

Varios algoritmos y modelos han sido propuestos para trabajar en el entorno de flujos de datos, dentro de los cuales destacan por su importancia las técnicas

desarrolladas para los procesos de agrupamiento y clasificación. La figura 4.1 muestra una cronología, desde mediados de la década del 90, de algunos de los métodos propuestos en estas áreas del conocimiento. Es importante resaltar, que en los últimos años existe un incremento sustancial en la cantidad de trabajos relacionados con esta temática.

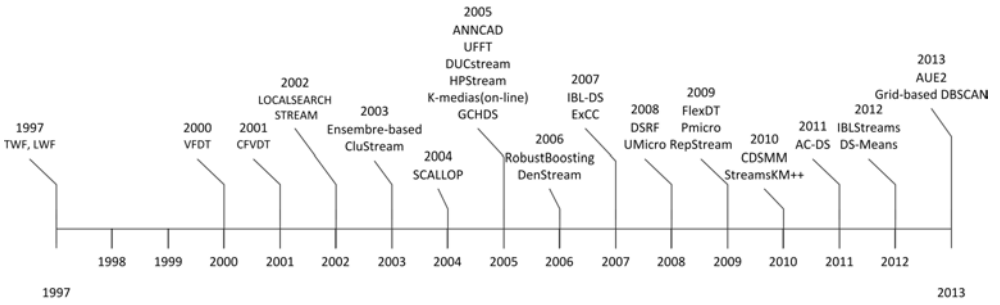

Figura 4.1: Cronología de métodos propuestos para minería de flujos de datos

Los próximos epígrafes hacen referencia a algunos trabajos relacionados, teniendo en cuenta las técnicas más abordadas en la literatura en la minería de flujos de datos.

4.2.1 Algoritmos de Agrupamiento

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 automático. 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 (66; 14; 15). 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.
Durante mucho tiempo el centro de atención de los investigadores, en esta temática, ha sido el problema de las K-medias *(K-means)* propuesto por MacQueen en el año 1968 (55). Este algoritmo toma el parámetro de entrada k y escoge aleatoriamente k objetos haciendo que éstos representen el centro del clúster. Cada uno de los objetos restantes se va asignando al clúster que sea más similar basándose en la distancia del objeto a la media del clúster. Entonces calcula la nueva media de cada clúster y el proceso sigue iterando hasta que se consigue la convergencia (se minimiza el error cuadrático medio). El método es relativamente escalable y eficiente para el procesado de conjuntos de datos grandes ya que la complejidad computacional del algoritmo es $O(nkt)$, donde n es el número de objetos, k el número de clúster y t el número de iteraciones. El K-medias se puede aplicar sólo cuando la media de un clúster puede ser definida, esto significa que no es aplicable en los casos en que los atributos sean categóricos. Otro inconveniente es su sensibilidad al ruido y a los outliers. Además, la necesidad de dar el valor de k a priori resulta uno de sus mayores puntos débiles.
Inicialmente, fueron propuestos muchos algoritmos para agrupamiento incremental, que se basan en el método anteriormente descrito (95; 12; 71; 18), dentro de los cuales destacan, el trabajo presentado por O'Challaghan en (66), donde propone el algoritmo LOCALSEARCH que utiliza técnicas de búsqueda local, demostrando mejoras con respecto al algoritmo K-medias. Para su adaptación a flujos de datos utiliza el algoritmo STREAM que divide los datos de entrada en subconjuntos que son analizados por LOCALSEARCH.
Para esta combinación los autores asumen que en la realidad los datos llegan en fragmentos $X_1, ...X_n$ cada uno de los cuales es mantenido en memoria. Para cada fragmento X_i, primero se determina si está compuesto mayormente por

puntos que se repiten una y otra vez, si esto ocurre, el fragmento se representa como un conjunto con pesos, donde cada punto aparece sólo una vez y su peso es igual a la frecuencia de aparición de ese punto en ese fragmento (o si los punto originales tienen pesos, el nuevo peso será, el correspondiente a ese punto en ese fragmento). Luego realiza el agrupamiento aplicando LOCALSEARCH a X_i, reteniendo solo los centros de los clúster encontrados, donde el peso de estos centros es la suma de los pesos de todos sus miembros. Finalmente se aplica LOCALSEARCH a todos los centros retenidos desde $X_1, ...X_i$, para obtener un conjunto (pesado) de centros del flujo de datos completo $X_1 \bigcup .... \bigcup X_i$.
Otro caso interesante es el propuesto por Aggarwal et al. en (14), donde presentan el algoritmo CluStream, técnica que divide el proceso de agrupamiento, en dos componentes: el almacén de componentes online, que contiene un resumen estadístico sobre la corriente de datos y el almacén de componentes off-line, que efectúa el agrupamiento de los datos resumidos de acuerdo al número que el usuario prefiera, en correspondencia con el tiempo y el número de grupos. El problema con CluStream es que, al igual que el algoritmo K-medias necesita predefinir un número constante de micro-clúster.
También destaca el trabajo presentado por Beringer y Hullermeier en (47), los cuales han desarrollado una versión on-line del algoritmo de K-medias. La principal característica de este método está dada por la transformación on-line de los datos, teniendo en cuenta un cálculo rápido de la distancia entre ellos. En otras palabras, la estructura de los datos actuales es tomada como inicialización de los próximos. Una adaptación importante del método planteado, con respecto al clásico, es que se incrementa el número de clústers.
Además de los problemas asociados a la minería de flujos de datos como: la memoria y tiempo de procesamiento, la naturaleza evolutiva y secuencial de los flujos, y la velocidad y multidimensionalidad de los elementos de un flujo; los algoritmos de agrupamiento, en particular, deben ser diseñados teniendo en cuenta otros requerimientos (33):

- 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 in-

terferencia electromagnética, fallos 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 clúster de forma arbitraria: Algunos algoritmos propuestos tienen el inconveniente de sólo detectar grupos de forma esférica, debido a la función de distancia que utilizan para llevar a cabo el agrupamiento (15; 103). 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 clústers: En flujos de datos de naturaleza evolutiva, el número de grupos puede variar con el progreso del tiempo. Fijar el número de clústers 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, la Tabla 4.1 muestra las fortalezas y debilidades de los algoritmos más representativos de los últimos años en la solución de los problemas antes mencionados.

Algoritmo	**Técnica**	**Cambios de Concepto**	**Outliers**	**Clúster arbitrarios**
(2002) LOCALSEARCH-STREAM (66)	Particiones			
(2003) CluStream (14)	Modelos	X		
(2005) DUCstream (50)	Densidad	X		
(2005) HPStream (15)	Modelos	X		
(2005) k-medias (*on line*) (47)	Modelos			
(2005) GCHDS (109)	Mallas	X		X
(2006) DenStream (33)	Densidad	X	X	X
(2007) ExCC (104)	Densidad	X	X	X
(2008) UMicro (16)	Modelos	X		
(2009) PMicro (20)	Modelos	X		
(2009) RepStream (97)	Grafos	X	X	X
(2010) StreamsKM++ (73)	Modelos			X
(2011) WOD(35)	Modelos		X	
(2012) DS-Means(2)	Modelos			X
(2013) Grid-based DBSCAN(85)	Mallas			X

Tabla 4.1: Algoritmos de agrupamiento para flujos de datos.

4.2.2 Algoritmos de Clasificación

Como principal problema dentro del aprendizaje supervisado, la clasificación consiste en inducir, con el mayor grado de exactitud posible, a partir de la observación de un conjunto finito de casos resueltos, el valor de un caso incompleto que lo identifica dentro de una categoría fijada previamente por el usuario (supervisor).
La clasificación en flujo de datos ha tomado fuerzas en las últimas décadas al igual que otras técnicas de minería ya analizadas. Los algoritmos de clasificación no están exentos de 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 de concepto en el flujos de datos.
Varias técnicas como las redes neuronales, los árboles de decisión, el vecino más cercano y las reglas de decisión han sido utilizadas para clasificar elementos de un flujo de datos. En los últimos años la tendencia ha sido utilizar la combinación de clasificadores, en lugar de utilizar técnicas simples de clasificación. Básicamente, estos algoritmos combinados mantienen una cantidad fija de modelos en memoria y utilizan el voto combinado para clasificar nuevos casos de consulta. Sin embargo, tienen como desventajas que aumentan el costo computacional y los requerimientos de memoria, necesitando la aplicación de métodos de eliminación de redundancias y paralelización de procesos; además, la inestabilidad del modelo paga un alto precio por aumentar la precisión y el riesgo de un tratamiento inadecuado podría causar pesos exagerados en ejemplos incorrectamente clasificados.
En la Tabla 4.2 se muestra una colección de algunos algoritmos de clasificación desarrollados en los últimos años, presentando 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.
Entre los algoritmos más recientes y abarcadores se encuentra el denominado Flexible Decision Tree (FlexDT) que extiende la teoría de lógica difusa a la

Algoritmo	Técnica	Alta Velocidad	Cambio de Concepto	Ruido
(1997) LWF (74)	kNN	X	X	
(1997) TWF (74)	kNN	X	X	
(2000) VFDT (80)	Árboles	X		
(2001) CVFDT (37)	Árboles	X	X	
(2003) Ensemble-based (42)	Combinado		X	
(2004) SCALLOP (34)	Reglas		X	
(2005) ANNCAD (107)	Combinado		X	
(2005) UFFT (52)	Árboles	X	X	
(2006) RobustBoosting (110)	Combinado		X	
(2007) IBL-DS (48)	kNN	X	X	X
(2008) DSRF (41)	Combinado		X	X
(2009) FlexDT (96)	Árboles	X	X	X
(2010) CDSMM (108)	Árboles		X	X
(2011) AC-DS (67)	Reglas		X	
(2011) OGA (83)	Reglas		X	
(2012) IBLStreams (6)	kNN	X	X	X
(2013) AUE2 (21)	Combinado	X	X	
(2013) EM (69)	Combinado	X	X	

Tabla 4.2: Algoritmos de Clasificación en Flujos de Datos.

clasificación de flujos de datos. Este método basado en árboles de decisión puede manejar el ruido y los cambios de concepto ocurridos en la función objetivo de forma eficiente. FlexDT incluye dos fases, en la primera se clasifican los nuevos casos que llegan y en la segunda se adapta el modelo a la aparición de nuevos conceptos, ajustando parámetros difusos mediante el uso de un método de back-propagation. Por otra parte, el uso de conjuntos difusos permite a FlexDT manejar eficazmente contextos imprecisos, como la ausencia de valores.

FlexDT es robusto al ruido, lo que se consigue realizando particiones borrosas de los atributos; enviando cada nuevo ejemplo a lo largo de más de una rama con el grado de pertenencia correspondiente, y teniendo en cuenta todas las hojas para clasificar una instancia. Como resultado, el ruido moderado en el flujo de datos no afecta de forma general el rendimiento de la clasificación de FlexDT. Así puede evitar que el ruido interfiera con la precisión, y la caída de exactitud puede atribuirse a cambios de concepto.

FlexDT clasifica los casos que contienen valores perdidos y a la vez aprende de ellos, ya que pueden participar plenamente en la construcción del clasificador. Si una instancia tiene un valor desconocido para un atributo asociado con un nodo, esta instancia se envía de manera uniforme a todas las ramas con grados

de pertenencia correspondientes, para ser clasificada.
Sin embargo, este método presenta tiempos de ejecución elevados, principalmente en conjuntos de datos de alta dimensionalidad. Comparado con algoritmos del vecino más cercano, es más lento debido a los cálculos para la obtención automática de los valores de los parámetros definidos para realizar las particiones borrosas.

4.2.2.1 Algoritmos basados en modelos combinados

La tendencia más actual en la clasificación de flujos de datos es el diseño de modelos combinados. Básicamente los algoritmos combinados son conjuntos de clasificadores individuales (componentes), cuyas decisiones son consideradas a partir de una regla de votación.
La decisión combinada de muchos clasificadores individuales suele ser más precisa que la que propone un único componente. Los estudios muestran que para obtener este aumento de la precisión, es necesario diversificar los algoritmos a combinar. Los componentes pueden diferir unos de otros por los datos con que han sido entrenados, por los atributos que utilizan, o por el clasificador base con que han sido creados. Para un nuevo ejemplo, la clasificación generalmente se establece por votación. Un algoritmo combinado básico se presenta en el Algoritmo 3.

Algoritmo 3 Algoritmo Combinado de Entrenamiento

entrada S: flujos de datos de ejemplos; ε: Conjunto de clasificadores; k: Número de clasificadores combinados
salida ε: Conjunto de clasificadores
1: $\varepsilon \leftarrow k$ clasificadores
2: **para todo** clasificador C_i en el conjunto ε **hacer**
3: asignar un peso a cada ejemplo en S para crear la distribución de pesos D_m
4: construir/actualizar C_i con S modificado por la distribución del pesos D_m
5: **fin para**

La formación de algoritmos combinados es un proceso costoso, pues se requiere, al menos, k veces más procesamiento que al entrenar un clasificador individual, además la selección de los componentes y la asignación del peso a los ejemplos

suele hacer el proceso aún más largo. En la minería de flujos de datos, los modelos de clasificadores individuales trabajan mejor, porque puede que no haya tiempo para ejecutar y actualizar un algoritmo combinado. Por otro lado, si el tiempo no es importante, pero se requiere muy alta precisión, un algoritmo combinado sería la solución más natural.

Dentro de los modelo combinados diseñados para la clasificación de flujos de datos, destaca el algoritmo denominado Ensemble-based (42). Este trabajo combina múltiples clasificadores, ponderados por la precisión de la clasificación esperada, en cada uno, con los datos de prueba actuales. En comparación con los modelos incrementales entrenados con los datos de la ventana más reciente, este enfoque combina las ventajas de conjuntos de expertos sobre la base de su credibilidad y se ajusta muy bien a los cambios de concepto ocurridos. Este método introduce la técnica de clasificación dinámica [9] para entornos de flujos de datos cambiantes, permitiendo seleccionar dinámicamente un subconjunto de clasificadores sin perder precisión.

El flujo de datos entrante es dividido en fragmentos secuenciales, $S_1, S_2 ... S_n$, donde S_n es el fragmento de información más actual, y cada fragmento es del mismo tamaño. Se entrena cada clasificador C_i para cada fragmento S_i. De acuerdo con la propiedad de la reducción de error, dados T ejemplos de prueba, se da a cada clasificador C_i un peso inversamente proporcional al error esperado de C_i en la clasificación de T. El peso del clasificador C_i es obtenido mediante la estimación del error de predicción esperado en los ejemplos de prueba. El modelo asume que la distribución de las clases de S_n, es igual a la distribución de los datos de prueba más recientes. Por lo tanto, los pesos de los clasificadores se pueden aproximar mediante el cálculo de su error de clasificación en S_n.

Aunque este modelo puede ajustarse a flujos de datos con cambios de concepto, consume muchos recursos en el proceso de entrenamiento de cada algoritmo y en la clasificación, y no logra manejar las altas velocidades de los datos de entrada ni hacer tratamiento al ruido o a la ausencia de información. Además la segmentación del flujo afectaría el aprendizaje de un nuevo concepto, en los casos en que los segmentos no describan totalmente el nuevo cambio.

4.2.2.2 Algoritmos que utilizan las técnicas del Aprendizaje Basado en Instancias

Los algoritmos de Aprendizaje Basados en Instancias pueden segmentar el espacio de atributos en una forma mucho más flexible, que les permite hacer frente con mayor facilidad a los conceptos no ortogonales. En este epígrafe se profundiza en ejemplos de esta familia de algoritmos que han sido exitosos en la clasificación de flujos de datos.

Inicialmente los modelos propuestos no contaban con un tratamiento directo a los cambios de concepto, sino que ésta adaptación se lograba a través de las políticas de olvido definidas. Tal es el caso de los algoritmos LWF (74)(Locally Weighted Forgetting) y TWF (74) (Time Weighted Forgetting), ambos de Aprendizaje Incremental, que utilizan la técnica del Aprendizaje Basado en Instancias.

LWF es uno de los mejores algoritmos de aprendizaje adaptativo. Esta técnica reduce el peso de los k vecinos más cercanos $e_1...e_k$ (en orden creciente de acuerdo a su distancia) a un nuevo ejemplo e_0, utilizando el factor $\tau + (1 - \tau)d_i^2/d_k^2$ si $d_i^2 \leq d_k^2$, dónde $d_i^2 = d^2(e, e_i)$ es el cuadrado de la distancia Euclideana desde el i-ésimo vecino más cercano al ejemplo e, y τ es un parámetro definido previamente. Un ejemplo es eliminado, si su peso está por debajo de un umbral determinado. El radio d_k que contiene la esfera de los k vecinos más cercanos del ejemplo e, se adapta con la influencia de la densidad local de ejemplos alrededor del nuevo caso. Una forma simple de determinar el límite del número de ejemplos a almacenar en la base del caso es poner k, el número de vecinos más cercanos, como $k = \lceil \beta |D| \rceil$, donde $|D|$ es el tamaño actual de la base de casos y β es un parámetro $0 < \beta \leq 1$. El valor de k controla el volumen de espacio para el tratamiento de un nuevo ejemplo.

Como alternativa evidente a LWF, se considera TWF (Time Weighted Forgetting), algoritmo que determina el peso de los casos de acuerdo a su edad (74). TWF es una técnica de ventana deslizante que guarda sólo los ejemplos más recientes presentados en el aprendizaje. En este modelo los ejemplos son pesados utilizando una función de asignación de pesos w_e, inicializado en $w_e = 1$, para cada ejemplo. El peso de un ejemplo e es recursivamente actua-

lizado para obtener una función de peso exponencial usando la regla siguiente: $w_e \longleftarrow \gamma w_e$, con $0 < \gamma \leq 1$, donde w_e es el peso asociado a un ejemplo. Un ejemplo consiste en un vector de entrada X, un valor de salida y y un peso $w : e =< X_e; y_e; w_e >$. Cuando este último decrece por debajo de un valor, el ejemplo correspondiente es eliminado, esto es el equivalente a primero que llega-primero que sale.
En el algoritmo Sliding Windows (88), los autores proponen adaptar el tamaño de la ventana de tal manera que se minimice el error estimado de la generalización de los datos de entrenamiento de esa ventana. Para ello, se dividen los datos en lotes, sucesivamente (es decir, para $k = 1, 2...$) las ventanas de pruebas por lotes, serían $t - k, t - k_1 ... t$. En cada una de estas ventanas, se entrena el clasificador y se valora su predicción. La ventana o combinación de modelos que produce la precisión más alta es finalmente seleccionada.
En (89), este enfoque es más generalizado, al permitir la selección de subconjuntos arbitrarios de lotes, en lugar de sólo secuencias ininterrumpidas. A pesar de la atractiva idea de este enfoque, de ajustar la ventana (conjunto de entrenamiento), las sucesivas pruebas con diferentes longitudes de ventana, son computacionalmente costosas y por lo tanto no aplicables inmediatamente en un escenario de flujo de datos, con limitaciones de tiempo.
Más recientemente es presentado el algoritmo IBL-DS (48) que optimiza la composición y el tamaño de la base de casos de forma automática. Cuando un nuevo ejemplo llega $e(x, c)$, primero se adiciona a la base de casos a través de una política de reemplazo basada en los siguientes aspectos:

- Relevancia Temporal: Las observaciones recientes son consideradas más útiles que otras menos recientes.

- Relevancia Espacial: Un ejemplo nuevo en una región del espacio de instancias ocupada por otros ejemplos es menos relevante que un ejemplo en una región poco ocupada.

- Consistencia: Un ejemplo debe ser eliminado si es inconsistente con el concepto actual.

Además se verifica la posibilidad de eliminar otros ejemplos, cuya información es redundante o ruidosa. Para este objetivo se mantiene un conjunto W de ejemplos cercanos a e, como candidatos, formado por los k_{cand} vecinos más cercanos de e. Los ejemplos más recientes se excluyen de este proceso, debido a la dificultad para distinguir los datos potencialmente ruidosos del principio de un cambio de concepto.

Como función de distancia se utiliza la variante SVDM (8) que es una versión de la medida de distancia simplificada VDM (19; 24), diseñando dos estrategias de edición que se utilizan en combinación con el fin de afrontar con éxito los cambios graduales o abruptos de concepto.

IBL-DS utiliza tests estadísticos para detectar cambios de concepto abruptos. Una vez detectado el cambio, un número de instancias deben ser eliminadas instantáneamente de la base de casos. El test estadístico funciona de la siguiente forma: se mantienen el error de predicción p y la desviación estándar $s = \sqrt{p(1-p)/100}$ de las últimas 100 instancias de entrenamiento. Denotando p_{min} como el error de predicción más pequeño y s_{min} su desviación estándar asociada. Un cambio de concepto es detectado si el valor de p es significativamente mayor que p_{min}. Aquí, la significatividad estadística es probada utilizando el estándar $z - test$. La condición de la prueba sería $p + s > p_{min} + z_{\alpha} s_{min}$ donde α es el nivel de confianza (utilizando $\alpha = 0.001$). Finalmente, si un cambio es detectado, se trata de estimar su extensión, con el objetivo de determinar el número de instancias que deben ser eliminadas. Más específicamente se elimina el porcentaje p_{dif} de todos los ejemplos almacenados, dónde p_{dif} es la diferencia entre p_{min} y el error de clasificación de las últimas 20 instancias; este último sirve como una estimación del error de clasificación actual. Los ejemplos a eliminar son seleccionados de forma aleatoria según una distribución que es espacialmente uniforme pero temporalmente sesgada.

Este algoritmo es capaz de adaptarse a cambios de concepto, además de mostrar una gran precisión para datos que no presentan esta característica. Sin embargo, estas dos situaciones dependen en cierta medida del tamaño de la base de casos. Si el cambio de concepto es estable, la exactitud de la clasificación aumentará con el tamaño de la base de casos, pero los procesos de clasificación

y actualización del modelo serán más costosos. Por otro lado, una gran base de casos resulta ser desfavorable cuando se producen cambios de concepto. IBL-DS obtiene buenos resultado de precisión, pero presenta tiempos elevados en el proceso de actualización del modelo.

Recientemente otros trabajos han sido publicados. El algoritmo AES (68) es un modelo de vecinos más cercanos basado en hormonas para la clasificación de flujos de datos. El Sistema Endocrino Artificial (AES) es una técnica que simula la manera en que se procesa la información en los sistemas biológicos endocrinos (98), donde se permite a las células comunicarse e interactuar entre sí, formando un sistema completo. Los registros podrían verse como las ubicaciones en el espacio de características y cada ubicación puede contener sólo una célula endocrina.

El clasificador AES consiste en células endocrinas en los límites de diferentes clases y la cantidad de límites de prueba K_{total} se decide a priori. Las muestras interiores (los puntos) en una clase se ven como puntos insignificantes que pueden desecharse en este proceso. Con los cambios producidos en los flujos de datos, las células endocrinas continúan cambiando sus posiciones para ajustarse a los nuevos límites de cada clase. Cada vez que un nuevo caso llega la célula que reside en la situación menos importante se moverá hacia el nuevo registro. De esta manera, los límites cambiantes entre las diferentes clases están dados por las ubicaciones en donde residen las células endocrinas. El modelo puede actualizarse cada vez que un nuevo caso llega por lo cual no se necesita un gran buffer de datos para entrenar el calsificador. Este modelo utiliza la distancia Euclídea para calcular las distancias entre el nuevo caso y todas las células endocrinas, con la regla del vecino más cercano $1NN$.

En el próximo capítulo se presenta el algoritmo SimC que se enmarca dentro de la familia de algoritmos de Aprendizaje Basado en Instancias para la clasificación de flujos de datos. Este nuevo enfoque logra resolver algunas de las cuestiones anteriormente planteadas.

4.3 Conclusiones Parciales

A partir del análisis realizado podemos concluir que:

- En las últimas décadas existe un incremento sustancial en la cantidad de técnicas desarrolladas para los procesos de agrupamiento y clasificación en la minería de flujos de datos.

- 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. En algunos casos dichas técnicas pueden llegar a solaparse.

- Inicialmente el centro de atención de los investigadores, en esta temática, fue el problema de las K-medias *(K-means)* propuesto por MacQueen en el año 1968 (55). Muestras de ellos lo son los trabajos propuestos en (47; 12; 66; 14; 95; 71; 18).

- Los algoritmos de agrupamiento en flujos de datos, deben ser diseñados para tratar los problemas asociados a la manipulación de outliers, el descubrimiento de clúster de forma arbitraria, y la necesidad de no asumir el número de clústeres a priori; además de todos los requerimientos asociados a la minería de flujos de datos.

- La clasificación en flujos de datos ha tomado fuerzas en las últimas décadas, estos algoritmos de clasificación deben ser capaces de procesar información a altas velocidades, consumir limitados recursos y actualizar los modelos de clasificación mediante la detección de cambios de concepto en el flujos de datos.

- Técnicas como las redes neuronales, los árboles de decisión, el vecino más cercano y las reglas de asociación han sido utilizadas para clasificar elementos de un flujo de datos. En los últimos años la tendencia ha sido utilizar la combinación de clasificadores, en lugar de utilizar técnicas simples de clasificación.

- Los modelos combinados son conjuntos de clasificadores individuales (componentes), cuyas decisiones son consideradas a partir de una regla de votación. La decisión combinada de muchos clasificadores individuales suele ser más precisa que la que propone un único componente, sin embargo la formación de estos algoritmos es un proceso costoso, pues se requiere, al menos, k veces más procesamiento, que al entrenar un clasificador individual, además la selección de los componentes y la asignación del peso a los ejemplos suele hacer el proceso aún más largo. En esta área los modelos de clasificadores individuales trabajan mejor, porque puede que no haya tiempo para ejecutar y actualizar un algoritmo combinado.

- Dentro de los algoritmos de clasificación, propuestos en la literatura, que utilizan las técnicas del Aprendizaje Basado en Instancias, existen aún problemas sin resolver, como el tratamiento al ruido y su diferenciación de los cambios de concepto ocurridos, la optimización en los procesos de actualización de los modelos, el balance entre la adaptación a cambios graduales y cambios abruptos, y la precisión en flujos de datos de alta dimensionalidad, entre otros.

Capítulo 5
METODOLOGÍA

5.1 Introducción

En el aprendizaje por lotes o *batch learning*, los sistemas de decisión modelan el conjunto de entrenamiento pudiendo procesar o tener en cuenta varias veces un mismo ejemplo, bien de forma recursiva como en el caso de los árboles o bien mediante un número finito de iteraciones como en el caso de las técnicas basadas en reglas. Denominados por ello algoritmos multi-pass a los esquemas que satisfacen los siguientes tres supuestos:

1. Todos los ejemplos necesarios para describir el dominio del problema están disponibles antes del proceso de aprendizaje.
2. Todos los ejemplos de entrenamiento pueden ser cargados en memoria.
3. Tras obtener un modelo para el conjunto de entrenamiento, el aprendizaje puede darse por finalizado.

Bajo esta filosofía, el conocimiento ha sido tradicionalmente extraído a posteriori, por lo que los algoritmos multi-pass son también conocidos hoy día como técnicas *off-line*. Sin embargo, la necesidad de desarrollar nuevos sistemas capaces de procesar bases de datos reales de gran escala (very large databases) y los avances recientes en telecomunicaciones inalámbricas y dispositivos empotrados han hecho posible disponer de un número ilimitado de fuentes para la adquisición de datos, dando lugar a un tráfico permanente y a un incremento

vertiginoso de la velocidad con la que éstos son recibidos, surgiendo los denominados flujos de datos o *data streams*. En estos entornos de aprendizaje, no hay tiempo para aplicar un tratamiento previo y "limpiar" todos los ejemplos recibidos, lo que exige a los sistemas operar *on-line* y aprender en tiempo real. En entornos de flujos de datos un sistema de clasificación debe partir de un modelo vacío que va siendo ampliado y simplificado simultáneamente conforme se reciben nuevos ejemplos, por lo que aunque éste puede realizar predicciones en todo momento, estas últimas no poseerán un nivel de confianza satisfactorio en épocas tempranas. Por otro lado, cuando los ejemplos son dependientes del tiempo, puede ser necesario tener en cuenta su momento de llegada, añadiendo la dimensión temporal bien como atributo extra del conjunto de entrenamiento o bien como un parámetro del algoritmo. A este respecto, una característica de estos sistemas, consiste en organizar el conocimiento en base a la novedad, dando mayor prioridad para la construcción del modelo a los últimos ejemplos recibidos.

Otro elemento importante es la posibilidad real de que la función objetivo puede depender del contexto, no siendo recogido éste, en su totalidad, mediante los atributos de entrada. Consecuentemente, cambios ocurridos en el contexto pueden inducir variaciones en la función objetivo, de forma que conceptos que eran válidos en el pasado pueden dejar de serlo en la actualidad. Este problema se conoce como *concept drift* (descrito en capítulos anteriores) y supone el reto principal para los algoritmos de aprendizaje en flujos de datos, ya que éstos deben ser capaces de detectar tales cambios y adaptarse a ellos con la mayor brevedad posible.

El problema se acentúa además cuando los datos presentan ruido, ya que sistemas demasiado sensibles reaccionarían rápidamente frente a éste como si de un cambio en la función objetivo se tratase, mientras que sistemas demasiado robustos se adaptarían de forma tardía ignorando tales cambios. Así, un aspecto decisivo en el rendimiento de estos algoritmos es el balance entre la robustez al ruido y la sensibilidad al cambio.

La realidad actual demanda el desarrollo de nuevos algoritmos de clasificación en entornos de flujos de datos, cuyos esquemas inductivos suponen un nuevo planteamiento con respecto a la lógica algorítmica multi-pass de décadas ante-

riores para poder así abordar problemas y fenómenos gobernados por el cambio y la infinitud en los datos.
Para hacer frente a las características de este nuevo escenario, es preciso definir un conjunto de tareas encaminadas a dar solución a los principales momentos dentro del proceso de clasificación en flujos de datos. Las políticas de actualización del modelo, la detección y tratamiento a cambios de concepto graduales y abruptos, los mecanismos de olvido y la selección de la técnica a utilizar son aspectos fundamentales a la hora de obtener un modelo que describa la función objetivo real y, por tanto, poder realizar predicciones con un grado de exactitud aceptable.
La propuesta de este trabajo se centra en la obtención y evaluación de un algoritmo de clasificación para flujos de datos basado en similitud. Los próximos epígrafes ofrecen una descripción detallada de la solución presentada para cada una de las etapas del modelo.

5.2 Clasificador de Flujo de Datos Basado en Similitud

La tendencia más actual en el desarrollo de algoritmos de clasificación en flujos de datos lo constituyen los modelos combinados, los cuales agrupan un número de algoritmos de características diversas que mejoran las predicciones y abarcan una mayor cantidad de problemáticas reales. Sin embargo, estos modelos, en su mayoría complejos, pagan un alto precio en términos de tiempo, en la actualización y mantenimiento de todos los clasificadores a la vez, siendo en ocasiones, el proceso de clasificación también muy costoso.
La utilización de un único clasificador base garantiza de antemano la eficiencia del rendimiento del sistema, en comparación con estos modelos combinados. Por lo cual la selección de la técnica a utilizar es un elemento de gran importancia en el diseño de un sistema de aprendizaje.
Diferentes técnicas han sido previamente analizadas, destacando por sus ventajas y amplia utilización en algoritmos de agrupamiento y clasificación, la técnica del Aprendizaje Basado en Instancias, cuyas características y propiedades

han sido anteriormente presentadas. Esta familia de algoritmos brinda una forma sencilla y natural de realizar el aprendizaje incremental sobre un flujo de información constante, donde el conocimiento almacenado lo constituyen los propios ejemplo, sin necesidad de realizar múltiples pases por el conjunto de información, ni construir reglas para cada concepto, ni actualizar pesos de neuronas y parámetros a cada inserción.
Este capítulo ofrece una descripción detallada del algoritmo SIMilarity-based Classifier (SimC), que utiliza la técnica del Aprendizaje Basado en Instancias para la clasificación de flujos de datos.

5.2.1 Descripción general de (SimC)

SimC utiliza las técnicas del Aprendizaje Basado en Instancias (IBL), por lo cual el éxito de su aplicación depende en buena medida de la correcta combinación de tres características fundamentales: la función de similitud, la función de selección de instancias y la función de clasificación.
Estas características se solapan en los principales métodos que implementa SimC, proporcionando un buen rendimiento al modelo. Un esquema general del algoritmo, que incluye estos métodos, es presentado en la figura 5.1:

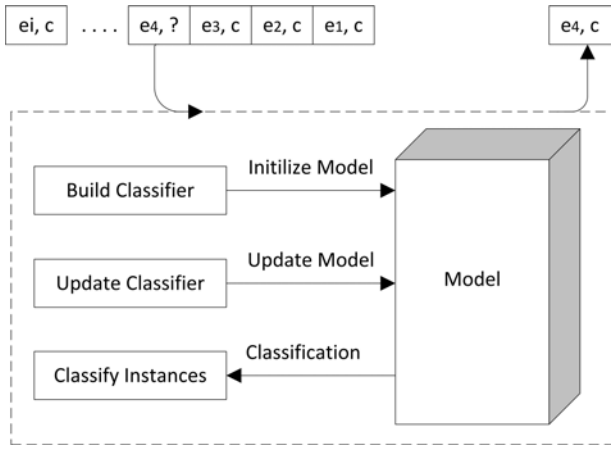

Figura 5.1: Esquema general de SimC.

Partiendo de un modelo vacío, inicialmente se construye el clasificador (Cons-

truir Clasificador) con los primeros ejemplos que llegan, luego, la corriente de información actualiza el modelo en memoria (Actualizar Clasificador) y los casos de prueba son clasificados e incluidos en el modelo (Clasificar Instancias), para contribuir al aprendizaje.
El objetivo principal de nuestro sistema de aprendizaje es lograr mantener un conjunto de instancias en memoria, que puedan describir el concepto real de forma general. Para lograrlo SimC analiza las instancias, una a una, a través de tres procedimientos, que abarcan los procesos de clasificación y actualización, y mantienen una base de casos actualizada en todo momento. Estos procedimientos se describen a continuación:

- Construir Clasificador: este procedimiento estructura el modelo, con las primeras 100 instancias que llegan, las cuales son almacenadas formando un grupo por cada clase. No es preciso que estas 100 instancias iniciales abarquen todo el dominio de clases del problema en cuestión, ya que el algoritmo está preparado para asimilar nuevas clases en cualquier momento. El valor inicial de 100 instancias puede variar, en dependencia de la cantidad máxima de casos que puede almacenar el modelo, pues debe ser menor que este valor. En este procedimiento se inicializa, además, la función de distancia utilizada.

- Clasificar Instancias: este procedimiento devuelve el valor del atributo clase, para una nueva instancia de consulta. La función de clasificación que implementa SimC es básicamente el método del vecino más cercano (1-NN) analizado en capítulos anteriores, utilizando la variante actualizable de la medida de distancia HVDM (25).

- Actualizar Clasificador: una vez construido el clasificador, este método es el responsable de mantener actualizada la base de casos, para lo cual se apoya en la política de inserción y eliminación diseñada, además del procedimiento para la detección y tratamiento de los cambios de concepto.

5.2.1.1 La función de similitud

Existen muchas formas de medir la similitud y no existe una implementación adecuada de este concepto para todos los dominios de aplicación. En particular, el Aprendizaje Basado en Instancias es muy sensible a la definición del criterio de cercanía, existiendo muchas variantes para garantizar su eficiencia. Varias medidas de similitud fueron analizadas en capítulos anteriores, describiendo sus ventajas y desventajas. Con el objetivo de obtener un clasificador que combinara, de forma óptima, atributos continuos y discretos, se realizaron experimentos con las funciones de distancia heterogéneas, teniendo en cuenta las medidas: HEOM, HVDM y DVDM. Los resultados demostraron la superioridad de la medida HVDM.
Una característica fundamental para que una medida de distancia pueda ser utilizada en problemas de flujos de datos es que debe ser actualizable, lo cual es logrado, incluyendo métodos para adicionar nuevos ejemplos y eliminar otros que sean eliminados de la base de casos, modificando así sus parámetros. La implementación de la medida de distancia HVDM, para SimC presenta esta característica, lo que facilita que se tengan en cuenta, en cada momento sólo los ejemplos que están incluidos en el modelo, y no todos los casos vistos.

5.2.2 Política de actualización del modelo

La política de actualización del modelo está estrechamente vinculada a la función de selección de instancias. Es a través de ésta que el algoritmo escoge los ejemplos que va a almacenar y decide dónde ubicarlos. Este método de actualización está diseñado para que el modelo se adapte a cambios de concepto graduales, sin hacer grandes eliminaciones de ejemplos de la base de casos, lo cual corresponde a los casos en que ocurren cambios de concepto abruptos.
En la base de casos se almacenan los ejemplos según la clase a la que pertenecen, formando grupos por cada clase. Para la correcta selección y ubicación de los nuevos ejemplos que llegan, el algoritmo se auxilia de algunos estimadores, diseñados y probados para facilitar el proceso de actualización, los cuales se

describen a continuación:
Para cada grupo de instancias se mantienen los siguientes estimadores:

- La instancia media de cada grupo, obtenida a partir de calcular el valor medio de cada atributo numérico y la moda de cada atributo simbólico, de todas las instancias del grupo.
- La edad del grupo, definida por el orden de creación.

y para cada instancia de cada grupo:

- La utilidad, definida como el número de instancias que ha clasificado correctamente esta instancia.

SimC implementa una política de inserción y eliminación basada en la combinación de estos estimadores. La figura 5.2 muestra un esquema de la heurística de este procedimiento. Cada camino de esta política, se encuentra ejemplificado en el siguiente epígrafe.

Esta política analiza las instancias que llegan, una a una, y sólo una vez. Los casos nuevos siempre son adicionados, lo cual garantiza que la base de instancias este actualizada en cada momento. Debido a esto, con cada inserción, por lo general, es necesaria una eliminación, y el sistema debe decidir qué caso eliminar.

5.3 Formalización de la propuesta

El algoritmo SIMilarity base Classifier (SimC) que propone este trabajo, funciona de la siguiente forma:
Sea B un modelo formado por n clases tal que $\{C_1, C_2, ..., C_n\} \in B$, donde cada clase C esta compuesta por un conjunto de grupos, de forma que:
$\{G_1, G_2, ..., G_{z_1}\} \in C_1$;
$\{G_{z_1+1}, G_{z_1+2}, G_{z_1+3}, ..., G_{z_2}\} \in C_2$;
$\{G_{z_2+1}, G_{z_2+2}, G_{z_2+3}, ..., G_{z_3}\} \in C_3$ y

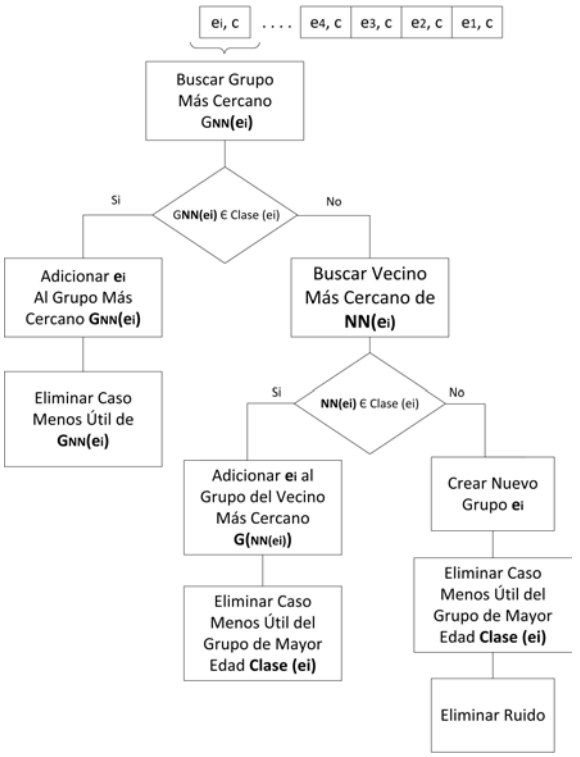

Figura 5.2: Política de inserción/eliminación.

$\{G_{z_n+1}, G_{z_n+2}, G_{z_n+3}, ..., G_j\} \in C_n$, donde j es la cantidad máxima de grupos en B.

Cada grupo tiene una estructura que almacena las instancias y los estimadores pertenecientes a él:

Sea $G_j \leftarrow \{\{m_j, a_j\}; \{e_{j_1}, e_{j_2}, e_{j_3}, ..., e_{j_i}\}\}$ un grupo formado por los estimadores $\{m_j, a_j\}$, donde m_j es la instancia media del grupo; a_j es la edad del grupo y $\{e_{j_1}, e_{j_2}, e_{j_3}, ..., e_{j_i}\}$ es el conjunto de instancias que pertenecen a ese grupo.

Inicialmente el modelo B se encuentra vacío y se inicializa con las primeras 100 instancias que llegan, en el procedimiento Construir Clasificador, almacenándose por cada clase C_n, sus grupos G_j.

Este método está implementado de forma incremental, por lo cual, no es estrictamente necesario que el clasificador se construya con 100 casos, perfectamente pueden ser más, o incluso, no ser ninguno, en esta última variante el sistema

iría incluyendo casos según el método de Actualizar Clasificador. Otro elemento importante es que, estas 100 instancias iniciales no deben contener la totalidad de las clases del problema, pues el modelo puede incluir una clase nueva en cualquier momento del aprendizaje.

Para la construcción del modelo, su estructuta y política de inserción/eliminación, fueron analizados posibles casos a ocurrir, los cuales son descritos a continuación.

Condiciones iniciales.

Tomando como condiciones iniciales un modelo B, formado por dos clases $\{C_A, C_B\}$, donde cada clase esta compuesta por un grupo $\{G_1, G_2\}$ respectivamente, y cada grupo tiene sus estimadores y un conjunto de 4 instancias, como se representa en la figura 5.3. La figura incluye además un ranking de utilidad, donde se encuentran ubicadas todas las instancias ordenadas de mayor a menor por su utilidad.

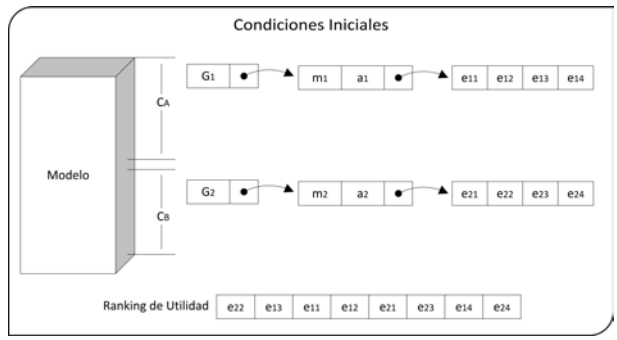

Figura 5.3: Estructura del modelo. Condiciones iniciales para ejemplos posteriores.

Caso A.

Partiendo de este estado, el primer caso se presenta cuando llega una nueva instancia $e^* \leftarrow \{x_i^*, C^*\}$, donde x_i^* representa el conjunto de valores para cada atributo y C^* el valor de la clase. Si la clase de esta nueva instancia no se encuentra en el modelo, tal que $C^* \leftarrow C_C$ y $C_C \notin B$, entonces se adiciona la nueva clase al modelo y se crea un nuevo grupo para esa clase, de forma que $G_3 \in C_C$, el cual es adicionado en B como representa la figura 5.4.

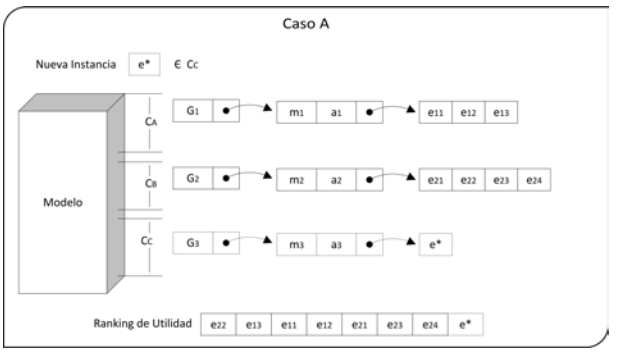

Figura 5.4: Política de inserción-eliminación. Caso A.

Caso B.
Si la clase de la nueva instancia ya se encuentra en el modelo ($e^* \in C_A$), es necesario encontrar el grupo en que debe ser incluida. En este caso se calcula la distancia de e^* hacia el centro de todos los grupos almacenados en B, utilizando la instancia media de cada grupo. Una vez encontrado el grupo más cercano $G_{NN}(e^*)$, si este pertenece a la misma clase de e^*, de forma que $G_{NN}(e^*) \in C_A$, entonces la nueva instancia es adicionada en este grupo. Si el modelo B tiene el número máximo de instancias permitidas, se elimina el elemento menos útil del grupo $G_{NN}(e^*)$ (teniendo en cuenta el ranking de utilidad), como muestra la figura 5.5, de lo contrario e^* es adicionado sin realizarse ninguna eliminación.
Caso C.
En caso de que $e^* \in C_B$ y $G_{NN}(e^*) \in C_A$, existe una diferencia entre la clase de la instancia nueva y la clase de su grupo más cercano, entonces es necesario buscar el vecino más cercano $NN(e^*)$. Si su vecino más cercano es de la misma clase, de forma que $NN(e^*) \in C_B$, entonces e^* es adicionado al grupo de su vecino más cercano, en este caso $G_{NN}(NN(e^*)) \leftarrow G_2$. En esta situación, si el modelo B tiene el número máximo de instancias permitidas, se elimina el elemento menos útil del grupo de mayor edad de la clase C_B, de lo contrario e^* es adicionado sin realizarse ninguna eliminación. En el ejemplo que muestra la figura 5.6, el grupo de mayor edad coincide con el grupo del vecino más

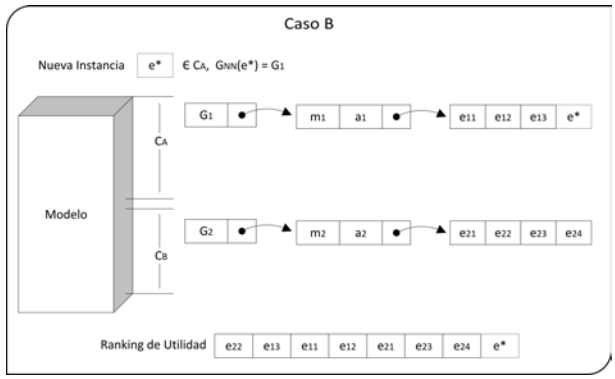

Figura 5.5: Política de inserción-eliminación. Caso B.

cercano $G_{NN}(e^*)$, pues la clase C_B solo tiene un grupo.

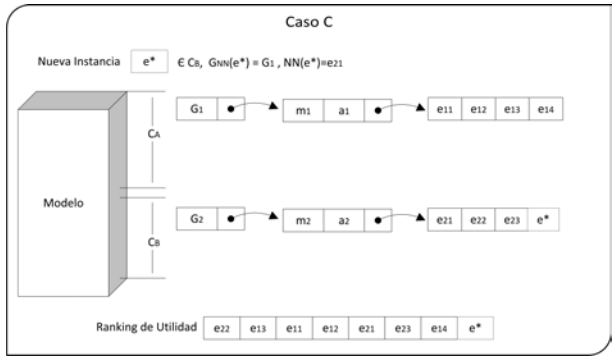

Figura 5.6: Política de inserción-eliminación. Caso C.

Caso D.

En caso de que el vecino más cercano $NN(e^*)$ y el grupo más cercano $G_{NN}(e^*)$, corresponden a una clase diferente a la de e^*, como muestra la figura 5.7 donde $e^* \in C_A$; $NN(e^*) \in C_B$ y $G_{NN}(e^*) \in C_B$, se crea un nuevo grupo con esta instancia y se elimina el caso menos útil del grupo de mayor edad de la clase C_A. Esta última situación puede ser síntoma de ruido o de la ocurrencia de cambios de concepto, por lo que el modelo debe ir adaptándose a partir de la creación de nuevos grupos. En caso de tratarse de ruido, cuando se crea un nuevo grupo con una instancia de ruido, este grupo persiste en el modelo,

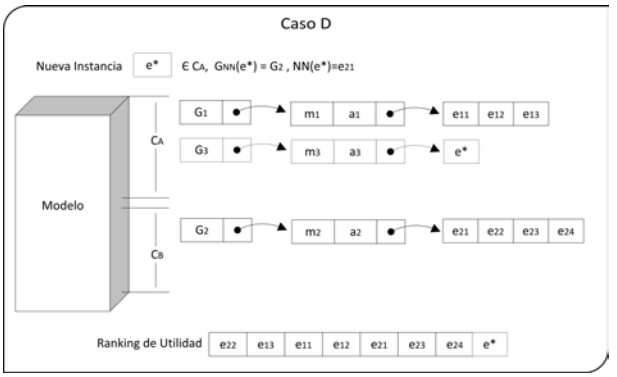

Figura 5.7: Política de inserción-eliminación. Caso D.

por un tiempo, hasta que es creado otro grupo, en este momento el modelo comprueba la existencia de otros grupos de la misma clase de edad 1, y los elimina, lo cual contribuye al control del ruido y los "outliers".
En caso de que se trate de un cambio de concepto, este grupo creado continúa recibiendo casos, que lo fortalecen, y con el tiempo, son eliminados otros grupos que ya no son útiles.
El modelo contiene un número máximo de instancias (este parámetro puede ser definido por el usuario), por lo que cada inserción, conlleva, por lo general, a la eliminación de otro ejemplo. Cuando todas las instancias de un grupo son eliminadas, éste desaparece.
Otra variable que se controla es la cantidad total de grupos en el modelo. El número máximo de instancias permitidas en el modelo dividido por la cantidad total de grupos, nos devuelve la cantidad de instancias máxima que puede tener un grupo, garantizando así el balance de todos los grupos y clases, elemento que influye de forma positiva en la precisión del modelo. Estos valores se actualizan con cada creación y eliminación de grupos.
El algoritmo 4 presenta el método de Actualizar Clasificador, que siguiendo esta política logra mantener el modelo debidamente actualizado.

Algoritmo 4 Actualizar Clasificador en SimC

entrada $e^* = (x_i^*, C^*)$: nueva instancia; d: función de distancia; $NoInstanciasModelo$: número máximo de instancias en el modelo; $NoGrupos$: cantidad de grupos;
salida B: modelo

1: $(g, C_g) \leftarrow G_{NN}(e^*)$
2: $(n_i, C_n) \leftarrow NN(e^*)$
3: $NoInstanciaGrupo \leftarrow (NoInstanciasModelo/NoGrupos)$
4: $Actualizar(d(e^*))$
5: **si** $C^* \notin B$ **entonces**
6: $CrearNuevaClase(C^*)$
7: $G \leftarrow CrearNuevoGrupo(x_i^*, C^*)$
8: $Adicionar(G, C^*)$
9: $Actualizar(B)$;
10: **si no**
11: **si** $C_g = C^*$ **entonces**
12: **si** $B.CantidadInstancias \geq NoInstanciasModelo$ **entonces**
13: $EliminarElementoMenosUtilGrupoMayorEdad(C^*)$
14: **si no**
15: **si** $(g, C_g).CantidadInstancias \geq NoInstanciaGrupo$ **entonces**
16: $EliminarElementoMenosUtilGrupo(g, C_g)$
17: **fin si**
18: **fin si**
19: $Adicionar(e^*, (g, C_g))$
20: **si no**
21: **si** $C_n = C^*$ **entonces**
22: **si** $B.CantidadInstancias \geq NoInstanciasModelo$ ó $G_{NN}(n_i, C_n).CantidadInstancias \geq NoInstanciaGrupo$ **entonces**
23: $EliminarElementoMenosUtilGrupoMayorEdad(C^*)$
24: **fin si**
25: $Adicionar(e^*, G_{NN}(n_i))$
26: **si no**
27: $EliminarRuido(C^*)$
28: $G \leftarrow CrearNuevoGrupo(x_i^*, C^*)$
29: $Adicionar(G, C^*)$
30: $Actualizar(B)$;
31: $NoGrupos++$;
32: **fin si**
33: **fin si**
34: **fin si**

5.4 Detección y tratamiento del ruido en SimC

Un aspecto importante en los sistemas de aprendizaje en flujos de datos es el tratamiento del ruido, como ya ha sido puntualizado anteriormente. En el caso de SimC, las instancias ruidosas pueden ser almacenadas en la base de casos, en situaciones como la que describe el caso D, del epígrafe anterior (figura 5.7). Cuando un nuevo grupo es creado para un ejemplo cuya clase difiere de la clase de su vecino más cercano y de su grupo más cercano, puede significar que un cambio de concepto está sucediendo, o que la instancia es un ejemplo ruidoso. En caso de tratarse de ruido, este grupo creado no recibirá más ejemplos, y quedará con edad 1, ya que ha contenido sólo una instancia. Al ejecutarse la acción de crear un grupo, primero, todos los demás grupos son revisados, buscando precisamente grupos de edad 1 que se encuentren en la base de casos, para ser eliminados. Por lo cual, las instancias ruidosas, aunque se mantienen almacenadas por un tiempo en la base de casos, finalmente se eliminan cuando se comprueba que no constituyen cambios de concepto.
Esta política permite diferenciar los cambios de concepto de los ruidos, característica ésta muy deseable en los sistemas de aprendizaje en flujos de datos. En las situaciones que ocurren cambios de concepto abruptos, los nuevos grupos creados continúan recibiendo ejemplos que acreditan la veracidad del cambio, permitiendo así que SimC pueda afrontar y actuar ante los cambios de concepto, con la certeza de que éstos están ocurriendo.

5.5 Detección y tratamiento de cambios de concepto en SimC

Con la política de actualización, descrita en epígrafes anteriores, el modelo logra adaptarse a los cambios de concepto graduales, sin embargo, carece de un tratamiento directo para los cambios de concepto abruptos. Para este objetivo, SimC utiliza un método que se basa en las caídas de los valores de precisión del modelo. Específicamente, SimC combina dos medidas de precisión: la precisión absoluta (c_a) y la precisión del flujo (precisión de las últimas 100 instancias

analizadas por el sistema)(c_s). Si la precisión del flujo cae por debajo de un umbral con respecto a la precisión absoluta, entonces se asume que está ocurriendo un cambio de concepto.
En este caso la base de instancias se limpia y se utilizan las últimas 100 instancias vistas para construir de nuevo el clasificador. El procedimiento es el que muestra el algoritmo 5.

Algoritmo 5 Tratamiento del Cambio de Concepto en SimC

entrada ca: precisión absoluta; cs: precisión del flujo; $lista100instancias$: últimas 100 instancias; B: base de instancias
salida B: nueva base de instancias

```
1: si ((c_a − c_s)/c_a) >= τ entonces
2:     Limpiar(B)
3:     B ← CrearNuevaBaseInstancias(lista100instancias)
4: fin si
```

Para la selección del valor del parámetro τ una extensa experimentación ha sido realizada, teniendo en cuenta medidas de evaluación de precisión y tiempo, así como diferentes conjuntos de datos utilizados igualmente para el análisis comparativo con otros modelos. Este valor puede ser ajustado teniendo en cuenta la problemática que se trabaje. El próximo capítulo muestra la experimentación realizada y sugiere valores adecuados para este parámetro.
Utilizando este procedimiento, SimC se adapta a los cambios de concepto de forma rápida, ya que todas las instancias nuevas son incorporadas a la base de casos y los grupos creados son actualizados.
Al detectar un cambio de concepto, todo el conocimiento almacenado en la base de casos es eliminado. Esta característica, en particular, podría ser beneficiosa para la aplicación de SimC en problemas de series temporales, pues bastaría almacenar en memoria los casos eliminados, para reutilizarlos en momentos en que se detecta un cambio de concepto, de un conocimiento que vuelve. Esta posibilidad constituye una línea de investigación para trabajos futuros.
SimC produce buenos resultados con flujos de datos de diferentes características, especialmente tiene los mejores valores con conjuntos de datos desbalanceados que incluyen cambios de concepto, como podrá comprobarse con la experimentación que muestra el próximo capítulo.

5.6 Conclusiones Parciales

El algoritmo SIMilarity-based Classifier (SimC) presenta las siguientes ventajas:

- Combina de forma óptima atributos numéricos y nominales, a partir de la utilización de la medida de distancia actualizable HVDM.
- Almacena un conjunto de casos en memoria de forma organizada, lo cual garantiza el fácil acceso a las instancias y la eficiencia del modelo.
- La política de inserción y eliminación que implementa está basada en el uso de estimadores diseñados para apoyar la correcta selección de los casos que deben ser almacenados.
- El algoritmo Actualizar Clasificador, garantiza el control del ruido y outliers, y se adapta a los cambios de concepto graduales que puedan ocurrir en la función objetivo.
- Logra diferenciar entre el ruido y los cambios de concepto ocurridos, lo que le permite adaptarse a estos cambios de concepto abruptos a través de un método que combina dos medidas de precisión.
- La flexibilidad de su diseño le permitirá adaptarse a contextos recurrentes, en caso de que la problemática lo requiera, almacenando los casos que son sustituidos, cuando es detectado un cambio de concepto, para utilizarlos con posterioridad.

Capítulo 6

EXPERIMENTACIÓN

6.1 Diseño de la Experimentación

El esquema diseñado para evaluar el algoritmo propuesto, incluye técnicas de validación cruzada, y un diseño específico orientado al objetivo del estudio, donde se consideran conjuntos de datos representativos de las situaciones a resolver por el sistema.

Se utiliza la variante de comparación con otros enfoques, con el objetivo de concluir, de ser posible, qué caracteriza los conjuntos de datos para los cuales el nuevo modelo propuesto arroja mejores resultados.

En (49) se presenta un resumen de los procedimientos a seguir en la validación de nuevos algoritmos de Inteligencia Artificial (IA), en una muestra de artículos seleccionados; y se hace énfasis en la importancia de la evaluación estadística comparativa de los resultados experimentales (utilizando pruebas de comparación de poblaciones), en lugar de llegar a conclusiones sobre estudios comparativos considerando solamente el valor promedio de alguna medida (ejemplo, la precisión) o la cantidad de archivos de datos para los cuales se obtiene el mejor resultado respecto a los algoritmos considerados en el experimento.

La comparación que presenta este trabajo, se apoya sobre estadísticos descriptivos, pero realiza una evaluación más profunda, que fundamenta la diferencia entre unos modelos y otros, sobre distintos conjuntos de datos.

6.1.1 Conjuntos de Datos

Para la experimentación fueron seleccionados conjuntos de datos de diferentes procedencias, con el objetivo de evaluar el modelo en el espectro más amplio posible.
De forma general, podemos clasificar los conjuntos de datos utilizados en sintéticos y reales. Dentro de los conjuntos de datos sintéticos se encuentran los generadores de flujos de datos que están incluidos en la librería de la herramienta Masive Online Analysis (MOA) (1), como Agrawal, Distrib, Gauss, Hyperplane, Led, Means, Mixed, RandomRBF, RDG, Sine, Stagger y Waveform. Para los conjuntos de datos reales fueron utilizados algunos datasets de la UCI- MLR (61), que han sido utilizados en múltiples experimentos de estas características, como Balance, Breast, Car, Letter, Nursery, Pages-bloks, Pen Digits, Vehicle y Vowel, además fueron utilizados flujos de datos reales publicados en internet por investigadores de esta temática, como Bank, Covetype, Elect, KDD, Poker Hand, Sea y Usenet recurrent. Las características generales de todos los conjuntos de datos utilizados se muestran en la tabla 6.1.

6.1.2 Medidas de evaluación

Las métricas que evalúan el rendimiento de los clasificadores, fueron seleccionadas a partir de su utilización en múltiples trabajos (96; 48):

- Precisión Absoluta de un algoritmo A con un flujo de datos D: se obtiene de dividir el número de instancias correctamente clasificadas por A en D entre la cantidad total de instancias clasificadas.

- Precisión del flujo de un algoritmo A con un flujo de datos D: se obtiene de dividir el número de instancias correctamente clasificadas de las últimas 100 dividido entre 100.

- Promedio de tiempo de clasificación: es el promedio de tiempo que el sistema demora en clasificar una instancia.

Origen	Nombre	No. Atributos	Tipo	No. Clases	No. Instancias	Cambio de Concepto
MOA	Agrawal	9	Mix	2	100000	
	Distrib	2	Num	2	20000	X
	Gauss	2	Num	2	20000	X
	Hyperplane	2	Num	2	100000	X
	Led24	24	Nom	7	100000	X
	Means	2	Num	5	20000	X
	Mixed	4	Mix	2	20000	X
	RandomRDF	10	Num	2	100000	
	RDG	10	Nom	2	20000	
	Sine	2	Num	2	20000	X
UCI- MLR	Annealing	39	Mix	5	20000	
	Balance	5	Num	3	20000	
	Breast	10	Num	6	20000	
	Car	7	Nom	4	20000	X
	Nursery	8	Nom	5	20000	X
	Pages-bloks	11	Num	5	20000	
	Pen Digits	17	Num	10	20000	
	Solar-Flare	13	Nom	6	20000	
	Vehicle	19	Num	4	20000	
	Vowel	14	Mix	11	20000	
Datos Reales	Bank	16	Mix	2	45210	
	Covtype	56	Mix	7	581012	
	Elec	4	Num	2	45312	X
	KDD	41	Mix	2	125972	
	Poker Hand	10	Mix	10	25010	
	Sea	3	Num	2	60000	
	Usenet recurrent	658	Nom	2	5931	

Tabla 6.1: Características de los Conjuntos de Datos Utilizados en la experimentación.

- Promedio de tiempo de actualización: es el promedio de tiempo que el sistema demora en actualizar la base de casos con una nueva instancia.

El rendimiento de los clasificadores también fue evaluado por clases, utilizando las siguientes métricas (75):

- Precisión Positiva, P: porcentaje de ejemplos que el clasificador ha predicho como positivos y que en realidad lo son.
- Proporción de Verdaderos Positivos (Recall), TPR: porcentaje de ejemplos correctamente clasificados de la clase positiva con respecto al número total de elementos existentes de esa clase.
- F-Measure: es una medida combinada basada en P y el TPR. Se calcula utilizando la siguiente fórmula: $F - Measure = 2 * ((P * TPR)/(P +$

TPR)).

6.1.3 Validación estadística

Para la evaluación y comparación de los diferentes esquemas, fueron utilizados estadísticos descriptivos como los valores de media y desviación estándar, además de algunos tests estadísticos, como se recomienda en (94):

- Test de Friedman, es un test no paramétrico que se utiliza para la comparación de múltiples clasificadores sobre múltiples conjuntos de datos. Este test ordena los algoritmos por su comportamiento, con cada conjunto separadamente. El algoritmo de mejor comportamiento toma el lugar 1, y así sucesivamente. En caso de empates, se asigna el promedio del orden establecido. A partir de esto, el test decide cuándo rechazar la hipótesis nula o aceptarla en caso de que el ordenamiento sea igual.

- Pruebas Post-hoc, una vez que se ha determinado que existen diferencias entre las medias, las pruebas post hoc permiten determinar qué medias difieren. En este trabajo han sido utilizados los procedimientos de Nemenyi, Holm y Shafter. Estas pruebas indican qué esquemas tienen diferencias significativas estadísticamente (en todos los tests se utiliza un nivel de significatividad de 0.05).

6.2 Evaluación de SimC

Con el objetivo de evaluar el rendimiento del algoritmo propuesto, incluyendo la metodología para el tratamiento a los cambios de concepto y las comparaciones con otros algoritmos como IBL-DS, LWF, TWF y Sliding Windows (WIN400), que fueron previamente descritos, se realizaron diferentes experimentos.
Inicialmente se presenta un experimento para seleccionar el valor más adecuado del parámetro τ, del algoritmo 5 para cambios de concepto abruptos.
A continuación, se evalúa este algoritmo, a través de la comparación de SimC in-

cluyendo este método y sin incluirlo, utilizando sólo los conjuntos de datos sintéticos que presentan esta característica. Luego se realizan experimentos para comparar SimC, con los demás modelos, de forma separada, con los conjuntos de datos sintéticos y reales.
Todos los experimentos evalúan todas las medidas de rendimiento propuestas e incluyen los valores de media y desviación estándar.
Los algoritmos de la comparación tienen parámetros que deben ser configurados también. En este trabajo se utilizan los valores que proponen los artículos correspondiente a cada modelo. Por ejemplo: LWF utiliza $\beta = 0.04$, $\tau = 0.8$; TWF utiliza $w = 0.996$ y para todos los experimentos se utiliza $k = 1$ y una ventana máxima de 400 instancias.
Los conjuntos de datos sintéticos fueron generados con un 5% de ruido y una cantidad maxima de 20.000 y 100.000 instancias cada uno, con el objetivo de estandarizar la experimentación con otros trabajos publicados anteriormente.

6.2.1 Seleccionando valores de parámetros

Cuando un cambio de concepto abrupto ocurre, la precisión de los últimos ejemplos clasificados cae precipitadamente, debido a que el modelo no está preparado para responder correctamente al nuevo flujo de información. Cuando esto ocurre se evidencia una diferencia significativa entre las medidas de la precisión absoluta (total de instancias correctamente clasificadas/total de instancias) y la precisión de flujo (total de instancias bien clasificadas de las últimas 100/100). Esta diferencia la definimos como el parámetro τ, para determinar cuándo ocurre un cambio de concepto abrupto, y para la selección de un valor adecuado realizamos una experimentación que compara el comportamiento del algoritmo para diferentes valores de τ, diferenciando los conjuntos de datos que tienen la característica de tener cambios de concepto abrupto, los cuales podemos monitorear más fácilmente y los conjuntos de datos que no tienen cambios de concepto abruptos.

Utilizamos para el análisis medidas de evaluación de precisión y tiempo, y conjuntos de datos sintéticos y de la UCI/MLR, descritos anteriormente.
Los valores de τ que proponemos son: el 10% y 30% de diferencia entre la

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.9$
Distrib	.9124 ± .0015	.9137 ± .0013	.9128 ± .0025	.9116 ± .0034	.9070 ± .0033
Gauss	.8356 ± .0048	.8402 ± .0045	.8439 ± .0045	.8440 ± .0044	.8450 ± .0042
Hyperplane	.8600 ± .0103	.8554 ± .0115	.8422 ± .0118	.8363 ± .0114	.8212 ± .0152
Means	.4642 ± .0340	.4579 ± .0326	.4542 ± .0338	.4440 ± .0333	.4416 ± .0358
Mixed	.8391 ± .0033	.8442 ± .0052	.8445 ± .0036	.8453 ± .0038	.8446 ± .0044
Random	.6417 ± .0057	.6435 ± .0050	.6406 ± .0039	.6413 ± .0027	.6419 ± .0029
Sine	.8375 ± .0029	.8419 ± .0029	.8429 ± .0027	.8444 ± .0025	.8442 ± .0033
Car	.6777 ± .0033	.6782 ± .0023	.6800 ± .0029	.6796 ± .0023	.6775 ± .0068
Nursery	.4062 ± .0036	.4039 ± .0080	.3965 ± .0152	.3855 ± .0052	.3881 ± .0039
Media	.7194	**.7199**	.7175	.7147	.7123
Desv.	**.1836**	.1860	.1875	.1912	.1894

Tabla 6.2: Precisión absoluta para diferentes valores del parámetro τ con conjuntos de datos que tienen cambio de concepto.

precisión del flujo y la precisión absoluta.
Las tablas 6.2 y 6.3 muestran los resultados obtenidos con la medida de evaluación precisión absoluta, con conjuntos de datos que tienen cambios de concepto abruptos y conjuntos de datos que no tienen esta característica, respectivamente. Analizando los valores de la media y la desviación estándar obtenidos, para los dos tipos de conjuntos de datos, los mejores resultados se presentan para un 10% ($\tau = 0.1$) de diferencia entre las medidas de precisión.
De igual forma las tablas 6.4 y 6.5 muestran los resultados obtenidos con la medida de evaluación precisión del flujo. Los valores de la media para los conjuntos que tienen cambios de concepto presentan los mejores resultados para un 30% ($\tau = 0.3$) de diferencias entre las medidas de precisión, y para los conjuntos que no tienen cambio de concepto presentan mejores resultados para una diferencia del 10% ($\tau = 0.1$).

Los resultados para la medida de tiempo medio de actualización (el tiempo promedio, en milisegundos, que demora el algoritmo en actualizar el modelo con cada instancia nueva) se muestran en las tablas 6.6 y 6.7, respec-

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.9$
Balance	.6831 ± .0021	.7630 ± .0044	.7616 ± .0073	.7651 ± .0042	.7638 ± .0025
Agrawal	.6941 ± .0033	.7118 ± .0107	.7140 ± .0045	.7199 ± .0058	.7266 ± .0047
Led24	.6120 ± .0057	.6141 ± .0055	.6156 ± .0048	.6163 ± .0042	.6178 ± .0044
Ramdomrbf	.9064 ± .0071	.9191 ± .0068	.9247 ± .0045	.9296 ± .0042	.9317 ± .0042
Rdg	.9074 ± .0549	.9182 ± .0504	.9278 ± .0447	.9278 ± .0472	.9365 ± .0440
Breast	.9996 ± .0002	.7317 ± .1062	.6879 ± .0019	.7241 ± .0060	.7280 ± .0023
Letter	.4305 ± .0029	.4336 ± .0023	.4379 ± .0021	.4395 ± .0038	.4451 ± .0024
Page	.9553 ± .0023	.9467 ± .0026	.9431 ± .0024	.9374 ± .0040	.9304 ± .0024
Pen	.9343 ± .0010	.9521 ± .0014	.9591 ± .0009	.9588 ± .0017	.9610 ± .0005
Vehicle	.6544 ± .0006	.6582 ± .0008	.6592 ± .0013	.6618 ± .0016	.6662 ± .0024
Vowel	.8009 ± .0034	.8043 ± .0025	.8138 ± .0044	.8172 ± .0061	.8095 ± .0058
Media	**.7798**	.7684	.7677	.7725	.7742
Desv.	.1784	.1627	.1652	.1629	**.1614**

Tabla 6.3: Precisión absoluta para diferentes valores del parámetro τ con conjuntos de datos que no tienen cambio de concepto.

tivamente. Los valores de la media para los conjuntos que tienen cambios de concepto presentan los mejores resultados para un 30% ($\tau = 0.3$) de diferencias entre las medidas de precisión, y para los conjuntos que no tienen cambios de concepto presentan mejores resultados para una diferencia del 10% ($\tau = 0.1$).

Para definir el valor del parámetro τ, resulta interesante conocer también la cantidad de veces que el algoritmo propuesto detecta un cambio de concepto, por lo cual comparamos esta cantidad para los valores de τ propuestos, que obtienen mejores resultados. La tabla 6.8 muestra la cantidad de cambios de concepto que encuentra SimC para diferentes valores de τ con los conjuntos de datos que presentan esta característica.

Los valores de τ propuestos muestran resultados muy competitivos, pero de forma general los mejores resultados se presentan para $\tau = 0.1\%$ con conjuntos de datos que no presentan cambios de concepto abruptos y $\tau = 0.3\%$ para

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.9$
Distrib	.9320 ± .0239	.9200 ± .0200	.9080 ± .0449	.9220 ± .0268	.9180 ± .0303
Gauss	.8780 ± .0311	.8540 ± .0546	.8820 ± .0217	.8840 ± .0251	.8840 ± .0344
Hyperplane	.8940 ± .0546	.8920 ± .0502	.8180 .2167	.9020 ± .0507	.9000 ± .0158
Means	.4900 ± .0596	.4940 ± .0783	.4440 ± .0937	.4580 .1228	.4300 ± .0778
Mixed	.8780 ± .0522	.8860 ± .0462	.8700 ± .0515	.8760 ± .0483	.8740 ± .0422
Random	.5520 ± .0773	.5300 ± .0718	.5900 ± .0975	.6500 ± .1065	.6520 ± .1171
Sine	.8720 ± .0342	.8740 ± .0358	.8820 ± .0148	.8840 ± .0114	.8860 ± .0134
Car	.5840 ± .0647	.6320 ±.1203	.7100 ±.1626	.7600 ±.2288	.5320 ±.1080
Nursery	.4260 .1594	.4800 ± .0648	.4240 ± .0911	.2220 ± .0877	.3540 ± .1857
Media	.7229	**.7291**	.7253	.7287	.7144
Desv.	.2044	**.1906**	.1938	.2435	.2257

Tabla 6.4: Precisión del flujo para diferentes valores del parámetro τ con conjuntos de datos que tienen cambio de concepto.

conjuntos de datos que sí presentan cambios de concepto abruptos. Por otra parte, la diferencia entre la cantidad de cambios de concepto detectados por SIMC utilizando $\tau = 0.1\%$ y $\tau = 0.3\%$, es significativa, reduciéndose con este último valor, en la mayoría de los casos, a menos del 70%, lo cual unido al hecho de que los resultados en las medidas de precisión y tiempo no presentan diferencias significativas, proponemos el valor de $\tau = 0.3\%$, para todos los conjuntos de datos, siendo utilizado este valor en lo sucesivo, en los restantes experimentos.

6.2.2 Evaluando el tratamiento directo al cambio de concepto

Para evaluar el rendimiento del método para el tratamiento directo de los cambios de concepto se comparan los resultados del algoritmo, incluyendo este método (SIMCτ) y sin incluirlo (SIMC), usando sólo conjuntos de datos que tienen esta característica. Los resultados de las dos variantes, con todas las medidas de rendimiento, incluyendo los valores de media y desviación estándar,

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.9$
Balance	.6580	.7600	.7720	.7600	.7760
	.1123	± .0374	± .0669	± .0574	± .0623
Agrawal	.6920	.7100	.7040	.7060	.7220
	± .0669	± .0430	± .0321	± .0627	± .0589
Led24	.6160	.6100	.6140	.6400	.6260
	± .0773	± .0718	± .0783	± .0822	± .0764
Ramdomrbf	.9040	.9380	.9360	.9500	.9440
	± .0207	± .0228	± .0230	± .0255	± .0270
Rdg	.8900	.8880	.9120	.9360	.9220
	± .0809	± .0669	± .0614	± .0594	± .0581
Breast	1,0000	.7280	.6860	.7120	.7040
	± .0000	±.1246	± .0503	± .0356	± .0428
Letter	.4120	.4040	.4100	.3920	.4020
	± .0798	± .0207	± .0141	± .0164	± .0295
Page	.9740	.9380	.9180	.9460	.9540
	± .0167	± .0179	± .0482	± .0483	± .0089
Pen	.9480	.9400	.9800	.9820	.9640
	± .0327	± .0316	± .0141	± .0084	± .0152
Vehicle	.6560	.6800	.6460	.6380	.6460
	± .0472	± .0682	± .0483	± .0311	± .0288
Vowel	.8360	.8380	.8380	.8620	.8460
	± .0568	± .0327	± .0427	± .0726	± .0404
Media	**.7805**	.7667	.7651	.7749	.7733
Desv.	.1858	.1664	.1729	.1813	**.1757**

Tabla 6.5: Precisión del flujo para diferentes valores del parámetro τ con conjuntos de datos que no tienen cambio de concepto.

se muestran en la tabla 6.9. Para referirnos a cada variante utilizaremos, en lo sucesivo, las notaciones: SimCτ y SimC. Para el análisis estadístico de los resultados, se aplica inicialmente el test de Friedman, con el que se rechaza la hipótesis de igualdad y se infiere que existen diferencias significativas entre ambos modelos para todas la medidas de rendimiento empleadas. A continuación, se aplica el test de Wilcoxon por pares, resultando existir también diferencias significativas entre los modelos en todas las medidas. Como resultado de este análisis podemos concluir que SimCτ es estadísticamente superior que SimC, en conjuntos de datos que tienen esta característica, con lo cual comprobamos la efectividad del método propuesto.

La figura 6.1 muestra la Curva de Aprendizaje de los flujos de datos Gauss y Mixed, con las dos variantes del modelo. Al principio del experimento las dos variantes presentan un comportamiento inestable, debido a que la base de casos no está completa aún, y esta situación afecta los valores de la precisión

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.9$
distrib	.0203 ± .0079	.0205 ± .0038	.0252 ± .0021	.0261 ± .0066	.0191 ± .0038
gauss	.0163 ± .0019	.0189 ± .0037	.0202 ± .0034	.0206 ± .0039	.0234 ± .0039
hyperplane	.0244 ± .0048	.0210 ± .0051	.0217 ± .0053	.0230 ± .0022	.0225 ± .0044
means	.0200 ± .0041	.0153 ± .0032	.0175 ± .0050	.0173 ± .0026	.0169 ± .0054
mixed	.0363 ± .0065	.0275 ± .0037	.0469 ± .0066	.0423 ± .0044	.0409 ± .0060
random	.0312 ± .0194	.0261 ± .0066	.0238 ± .0032	.0272 ± .0039	.0229 ± .0020
sine	.0211 ± .0075	.0235 ± .0060	±.0131	.0267 ± .0070	.0254 ± .0068
car	.0438 ± .0073	.0364 ± .0053	.0352 ± .0069	.0359 ± .0025	.0335 ± .0069
nursery	.0655 ± .0110	.0671 ± .0063	.0565 ± .0063	.0553 ± .0047	.0546 ± .0061
media	.0310	**.0285**	.0306	.0305	.0288
desv	.0157	.0157	.0132	.0120	**.0122**

Tabla 6.6: Tiempo Medio de Actualización para diferentes valores del parámetro τ con conjuntos de datos que tienen cambio de concepto.

de la clasificación. SimC cae precipitadamente cada vez que ocurre un cambio de la función objetivo, y toma más tiempo para volver a su estado anterior, sin embargo SimCτ no cae abruptamente y se recupera del cambio mucho más rápido. La combinación de la política de actualización, que abarca los cambios graduales, y el método para el tratamiento directo de los cambios de concepto abruptos, permiten a SimCτ mantener altos índices de precisión en cada momento y hacen de éste un algoritmo eficiente para la clasificación de flujos de datos con estas características.

6.2.3 Evaluando SimCτ con conjuntos de datos sintéticos

Esta sección evalúa el comportamiento de SimCτ en comparación con otros modelos que utilizan el Aprendizaje Basado en Instancias (IBL), descritos anteriormente.

La tabla 6.10 presenta los valores de precisión absoluta y precisión del flujo, de cada algoritmo con los conjunto de datos que presentan cambio de concepto,

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$	$\tau = 0.5$	$\tau = 0.7$	$\tau = 0.9$
balance	.0191 ± .0045	.0272 ± .0075	.0224 ± .0026	.0238 ± .0049	.0258 ± .0043
agrawal	.0476 ± .0062	.0602 ± .0076	.0646 ± .0057	.0637 ± .0058	.0587 ± .0110
led24	.3376 ± .0234	.3620 ± .0257	.3511 ± .0281	.3578 ± .0057	.3589 ± .0199
ramdomrbf	.0514 ± .0068	.0631 ± .0093	.0718 ± .0071	.0773 ± .0090	.0825 ± .0076
rdg	.0744 ± .0054	.0811 ± .0057	.0870 ± .0153	.0943 ± .0070	.1014 ± .0176
breast	.0411 ± .0057	.0855 ± .0231	.0731 ± .0071	.0843 ± .0065	.0859 ± .0109
letter	.1295 ± .0131	.1218 ± .0114	.1110 ± .0122	.1107 ± .0185	.1041 ± .0057
page	.0469 ± .0106	.0484 ± .0096	.0403 ± .0061	.0380 ± .0063	.0391 ± .0105
pen	.0649 ± .0071	.0675 ± .0074	.0710 ± .0086	.0665 ± .0138	.0608 ± .0122
vehicle	.0757 ± .0112	.0592 ± .0055	.0538 ± .0049	.0605 ± .0065	.0619 ± .0049
vowel	.1232 ± .0247	.1110 ± .0116	.1134 ± .0175	.1055 ± .0142	.0981 ± .0186
media	**.0919**	.0988	.0963	.0984	.0979
desv	**.0881**	.0914	.0887	.0900	.0902

Tabla 6.7: Tiempo Medio de Actualización para diferentes valores del parámetro τ con conjuntos de datos que no tienen cambio de concepto.

además de los valores de la desviación estándar y la media.

Con un 77% de precisión y la más baja desviación estándar, SimCτ presenta los mejores resultados. Para el análisis estadístico se aplica el test de Friedman, con 5 algoritmos y 7 conjuntos de datos, con lo cual la hipótesis nula es rechazada y se infiere que existen diferencias significativas entre los esquemas comparados. Para encontrar exactamente estas diferencias, se aplica el test de Wilcoxon, por pares, y como resultado se obtiene que, con esta medida, SimCτ es significativamente superior a IBL-DS, TWF y WIN400 y no presenta diferencias con el modelo LWF.

Es necesario destacar los valores de precisión obtenidos por SimCτ, con el conjunto Mixed, de atributos mixtos, lo que demuestra la eficacia de la medida de distancia utilizada; y Means, de mayor dimensionalidad con respecto al número de clases.

Los valores obtenidos con la medida de la precisión del flujo, se muestran en

Flujo de datos	$\tau = 0.1$	$\tau = 0.3$
Distrib	94	45
Gauss	114	71
Hyperplane	94	59
Means	165	147
Mixed	112	72
Random	142	116
Sine	116	72
Car	139	110
Nursery	157	149

Tabla 6.8: Cantidad de cambios de concepto detectados por SimC para diferentes valores de τ con flujos de datos sintético.

la tabla 6.10. Una vez más el rendimiento de SimCτ es el mejor, con una precisión de 77% y la más baja desviación estándar. El mismo procedimiento estadístico se repite. El test de Friedman rechaza la hipótesis nula, y se infiere que existen diferencias significativas. El test de Wilcoxon, compara cada par, y demuestra que con esta medida SimCτ es significativamente mejor que los modelos IBL-DS, TWF y WIN400, y que no presenta diferencias con respecto a LWF.

En esta medida de evaluación, SimCτ también muestra buenos resultados con el conjunto de atributos mezclados Mixed y con el conjunto de mayor dimensionalidad Means. Es necesario destacar que, teniendo en cuenta los valores medios, SimCτ obtiene los mejores porcentajes de precisión.

Un objetivo importante en el contexto de flujo de datos es lograr actualizar y clasificar nuevos casos de prueba, en tiempos apropiados. La tabla 6.11 muestra el comportamiento de todos los modelos con respecto a los tiempos medios de actualización y clasificación de cada algoritmo con cada conjunto de datos. Win400 y TWF son los modelos más rápidos, sin embargo éstos no tienen buenos resultados en la medidas de precisión. Los algoritmos IBL-DS y LWF son muy competitivos, dado que presentan resultados similares a SimCτ en la precisión, pero no resultan eficientes a la hora de actualizar el modelo. El análisis estadístico demuestra que, para la medida de tiempo medio de actualización, SimCτ es significativamente mejor que IBL-DS y LWF. Por otra parte, los modelos TWF y WIN400 presentan los mejores resultados para esta medida.

	Precisión Absoluta		Precisión del Flujo		Tiempo Medio de Clasificación		Tiempo Medio de Actualización	
Flujo de datos	SimC	SimCτ	SimC	SimCτ	SimC	SimCτ	SimC	SimCτ
Distrib	.7210 ± .0784	.9070 ± .0033	.4240 ± .3029	.9180 ±.0303	.1679 ± .0043	.1156 ± .0060	.0241 ± .0024	.0191 ± .0038
Gauss	.5062 ± .0031	.8450 ± .0042	.8420 ± .0507	.8840 ± .0344	.1754 ± .0096	.1444 ± .0186	.0293 ± .0041	.0234 ± .0039
Hyperplane	.6083 ± .0388	.8212 ± .0152	.5600 ± .0700	.9000 ± .0158	.1659 ± .0187	.1105 ± .0087	.0342 ± .0031	.0225 ± .0044
Means	.2129 ± .0021	.4416 ± .0358	.1700 ± .0406	.4300 ± .0778	.1931 ± .0071	.0968 ± .0076	.0227 ± .0046	.0169 ± .0054
Mixed	.5010 ± .0009	.8446 ± .0044	.8000 ± .0274	.8740 ± .0422	.4598 ± .0221	.4184 ± .0116	.0446 ± .0111	.0409 ± .0060
Random	.5094 ± .0042	.6419 ± .0029	.5000 ± .0339	.6520 ± .1171	.1758 ± .0080	.1068 ± .0078	.0315 ± .0016	.0229 ± .0020
Sine	.5034 ± .0014	.8442 ± .0033	.8500 ± .0394	.8860 ± .0134	.1749 ± .0111	.1948 ± .0150	.0299 ± .0039	.0254 ± .0068
Car	.5684 ± .0348	.6775 ± .0068	.5120 ± .1413	.5320 ± .1080	.7524 ± .0713	.4163 ± .0099	.0501 ± .0054	.0335 ± .0069
Nursery	.2365 ± .0635	.3881 ± .0039	.4200 ± .2595	.3540 ± .1857	1.553 ± .0360	.5849 ± .0137	.0815 ± .0110	.0546 ± .0061
Media	.4852	.7123	.5642	.7144	.4243	.2432	.0387	.0288
Desv.	.1640	.1894	.2283	.2257	.4684	.1815	.0183	.0122

Tabla 6.9: Medidas de precisión Absoluta y precisión del Flujo de SimC y SimCτ.

SimCτ presenta mejores tiempos actualizando el modelo que en el proceso de clasificación, pues durante éste, todas las instancias en la base de casos deben ser analizadas, como en un k-NN estándar, por lo cual demora más. Sin embargo en la medida de tiempo medio de actualización presenta buenos resultados, y si éstos son combinados con los valores de precisión que obtiene, SimCτ puede resultar un método efectivo para la clasificación de flujo de datos con cambio de concepto.

6.2.4 Evaluando SimCτ con conjuntos de datos de la UCI-MLR

Los conjuntos de datos de la UCI-MLR, están basados en problemas reales, y son utilizados para la validación de todos los algoritmos de minería de datos. Los conjuntos seleccionados han sido utilizados para evaluar algunos clasificadores de flujo de datos como (48; 96), sin embargo, necesitan adaptarse a la problemática, pues estos conjuntos por lo general tienen un número pequeño de casos. Con el objetivo de evaluar el comportamiento del modelo propuesto,

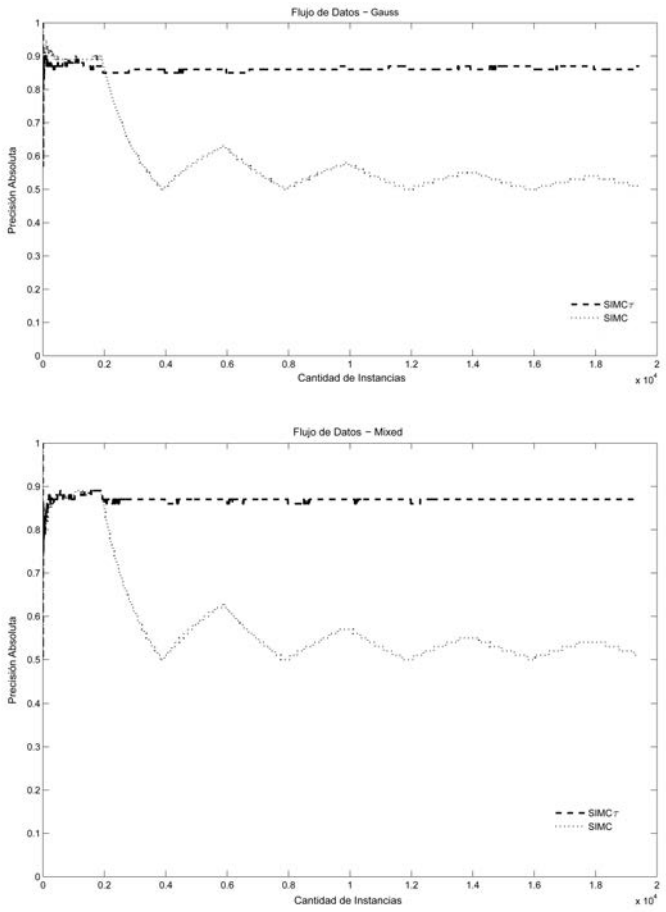

Figura 6.1: Curva de Aprendizaje de SimC y SimCτ, con los conjuntos de datos sintéticos: Gauss y Mixed

en todas las situaciones posibles, estos conjuntos de datos presentan características diferentes en cuanto a dimensionalidad, tipo de atributos y no presentan cambio de concepto.

La tabla 6.12 muestra los valores obtenidos para las medidas de precisión. El análisis estadístico realizado para la medida de precisión absoluta demuestra que con estos conjuntos de datos SimCτ es significativamente superior a los algoritmos IBL-DS, LWF y WIN400, y no presenta diferencias con el modelo

	Precisión Absoluta					Precisión del Flujo				
Flujo de datos	SimCτ	IBL-DS	LWF	TWF	Win400	SimCτ	IBL-DS	LWF	TWF	Win400
Distrib	.9200 ± .0200	.9116 ± .0036	.8861 ± .0050	.8618 ±.0021	.8622 ± .0028	.9200 ± .0200	.9180 ± .0268	.8800 ± .0748	.9020 ± .0349	.9020 ± .0349
Gauss	.8540 ± .0546	.8039 ± .0199	.5516 ± .0044	.5498 ± .0065	.5486 ± .0034	.8540 ± .0546	.7860 ± .0261	.5320 ± .0466	.5820 ± .0249	.5780 ± .0268
Hyperplane	.8920 ± .0502	.8319 ± .0194	.8627 ± .0285	.7997 ± .0139	.7814 ± .0077	.8920 ± .0502	.8200 ± .1782	.9000 ± .0700	.8700 ± .0539	.8440 ± .0650
Means	.4940 ± .0783	.3926 ± .0149	.3232 ± .0192	.2311 ± .0075	.2283 ± .0058	.4940 ± .0783	.3460 ± .0522	.3240 ± .0799	.2320 ± .0217	.2280 ± .0327
Mixed	.8860 ±.0462	.8367 ± .0017	.7586 ± .0036	.7512 ± .0032	.7515 ± .0022	.8860 ± .0462	.8320 ± .0444	.8000 ± .0539	.7860 ±.0336	.8020 ± .0130
Random	.5300 ± .0718	.6299 ± .0075	.6755 ± .0040	.6185 ± .0035	.6198 ± .0026	.5300 ± .0718	.5980 ± .1434	.7320 ± .1274	.5920 ± .0898	.6640 ± .1159
Sine	.8740 ± .0358	.8335 ± .0051	.6760 ± .0048	.6767 ±.0012	.6754 ±.0030	.8740 ± .0358	.8540 ± .0434	.7300 ± .0406	.7100 ± .0400	.7340 ± .0518
Media	.7710	.7486	.6762	.6413	.6382	.7786	.7363	.6997	.6677	.6789
Desv.	.1617	.1790	.1940	.2098	.2087	.1835	.1987	.2057	.2290	.2269

Tabla 6.10: Precisión absoluta y Precisión del flujo para cada algoritmo con cada flujo de datos sintéticos.

TWF. Además es necesario destacar que obtiene buenos resultados con conjuntos de datos de alta dimensionalidad, con respecto al número de atributos y clases, como Nursery, Balance, Breast, Pen-Digit y Vowel, este último además presenta atributos mezclados. El análisis estadístico realizado para la medida de precisión del flujo demuestra que con estos conjuntos de datos SimCτ es significativamente superior a los algoritmos IBL-DS y LWF, y no presenta diferencias con los modelos TWF y WIN400. También con esta medida se obtienen buenos resultados con conjuntos de datos de alta dimensionalidad, como Nursery, Balance, Pen-Digit y Vowel.

La tabla 6.13 muestra los resultados de las medida del tiempo medio de actualización y tiempo medio de clasificación. En la medida de tiempo medio de actualización, SimCτ es significativamente superior a los algoritmos IBL-DS y LWF, pero presenta tiempos más elevados comparados con los modelos TWF y WIN400. Estos últimos presentan los mejores tiempos de ejecución, como se evidenció en la experimentación del epígrafe anterior. En la medida del tiempo medio de clasificación, SimCτ es significativamente superior al algoritmo LWF, pero presenta tiempos más elevados comparados con los modelos IBL-DS, TWF y WIN400.

	Tiempo medio de actualización					Tiempo medio de clasificación				
Flujo de datos	SimCτ	IBL-DS	LWF	TWF	Win400	SimCτ	IBL-DS	LWF	TWF	Win400
Distrib	.0205 ±.0038	1.141 ± .4810	.1160 ± .0108	.0003 ± .0004	.0094 ± .0032	.0933 ± .0036	.0974 ± .0134	.0541 ± .0039	.0803 ± .0115	.0294 ± .0065
Gauss	.0189 ± .0037	.8055 ± .0095	.0671 ± .0093	.0018 ± .0009	.0132 ± .0023	.1094 ± .0107	.1114 ±.0066	.0530 ± .0083	.0810 ± .0043	.0403 ± .0048
Hyperplane	.0210 ±.0051	.7418 ± .0843	.1069 ± .0590	.0016 ± .0013	.0097 ± .0019	.1038 ± .0149	.0810 ± .0138	.0767 ± .0150	.0945 ± .0155	.0369 ± .0052
Means	.0153 ± .0032	.7693 ± .0820	.0475 ± .0061	.0017 ± .0003	.0106 ± .0044	.0832 ±.0030	.0823 ± .0062	.0566 ± .0147	.1028 ± .0082	.0391 ± .0121
Mixed	.0275 ±.0037	.9861 ± .0815	.2985 ± .0252	.0119 ± .0038	.0152 ± .0031	.3131 ± .0176	.1537 ± .0070	.2474 ± .0498	.1744 ± .0265	.0533 ±.0147
Random	.0261 ± .0066	.7772 ± .0167	.0737 ± .0108	.0028 ± .0015	.0124 ± .0013	.1121 ± .0192	.0908 ± .0148	.0561 ±.0081	.1090 ± .0086	.0439 ± .0162
Sine	.0235 ± .0060	.7893 ± .0308	.0745 ± .0071	.0037 ±.0026	.0122 ± .0050	.1575 ± .0229	.0941 ± .0144	.0713 ± .0097	.1129 ± .0116	.0448 ± .0035
Media	.0218	.8587	.1120	.0034	.0118	.1389	.1015	.0879	.1078	.0411
Desv.	.0042	.1485	.0855	.0039	.0021	.0803	.0252	.0709	.0320	.0074

Tabla 6.11: Tiempo medio de actualización y Tiempo medio de clasificación (en milisegundos) de cada algoritmo con cada flujo de datos sintético.

6.2.5 Evaluando SimCτ con conjuntos de datos reales.

Los flujos de datos reales KDD, SEA y Usenet recurrent utilizados se encuentran publicados en el siguiente sitio web http://www.liaad.up.pt/kdus, el resto están publicados en la UCI-MLR disponibles para todos los investigadores.
Estos conjuntos de datos presentan diferentes características en cuanto a dimesionalidad, balance de clases y cambios de concepto. Para el análisis del rendimiento de SimCτ con estos conjuntos de datos se establece una comparación con los modelos IBL-DS, TWF y Win400. No se incluye el algoritmo LWF, debido a que los tiempos de ejecución para los procesos de clasificación y actualización son muy elevados, resultando inadecuada su aplicación en este contexto.
La tabla 6.14 muestra los valores obtenidos para las medidas de precisión. El análisis estadístico realizado para las dos medidas de precisión acepta la hipótesis nula y concluye que no existen diferencias significativas entre los modelos comparados. El rendimiento del modelo en las medidas de tiempo es similar a los experimentos anteriores.
La evaluación del rendimiento por clases es algo decisivo para obtener una medida del rendimiento de un clasificador. Generalmente en problemas binarios

	Precisión Absoluta					Precisión del Flujo				
Flujo de datos	SimCτ	IBL-DS	LWF	TWF	Win400	SimCτ	IBL-DS	LWF	TWF	Win40
Car	.6782 ± .0023	.6500 ± .0042	.6027 ± .0429	.5978 ± .0026	.6242 ± .0036	.6320 ±.1203	.5160 ±.0904	.6180 ± .2190	.4980 ±.1497	.6560 ± .2274
Nursery	.4039 ± .0080	.3410 ± .0113	.3989 ± .0262	.2535 ± .0015	.3756 ± .0006	.4800 ±.0648	.3200 ± .0596	.3820 ± .1608	.2380 ± .0370	.3880 ±.0311
Balance	.7630 ± .0044	.7130 ± .0042	.0753 ± .0105	.6829 ± .0065	.5454 ± .0110	.7600 ±.0374	.6940 ±.1092	.0720 ± .0396	.7080 ±.0460	.5900 ±.0992
Agrawal	.7118 ±.0107	.6214 ± .0027	.8601 ± .0082	.7928 ± .0022	.5600 ± .0031	.7100 ±.0430	.5620 ±.0311	.8180 ±.0526	.7900 ±.0400	.5340 ± .0808
Led24	.6141 ± .0055	.3953 ± .0029	.6259 ± .0033	.5541 ± .0051	.3690 ± .0028	.6100 ±.0718	.3940 ±.0720	.6100 ±.0914	.5760 ±.0404	.3980 ±.0507
Ramdomrbf	.9191 ± .0068	.8003 ± .0293	.8803 ± .0242	.9224 ± .0051	.9251 ± .0057	.9380 ±.0228	.7920 ±.0444	.8060 ± .0799	.9420 ±.0179	.9400 ±.0245
Rdg	.9182 ± .0504	.8896 ± .0133	.8825 ± .0666	.9200 ± .0582	.8849 Rdg ± .0511	.8880 ± .0669	.8680 ± .0249	.8800 ±.0624	.9280 ±.0958	.9060 ±.0639
Breast	.9998 ± .0003	.8546 ± .0164	.7995 ± .0008	.9947 ± .0002	.9996 ± .0004	1.000 ±.0000	.8520 ±.0409	.8080 ±.0110	1.000 ± .0000	1.000 ± .0000
Letter	.4336 ± .0023	.2653 ± .0096	.4158 ± .0029	.6326 ± .0014	.6350 ± .0010	.4040 ± .0207	.2380 ±.1003	.2620 ± .0545	.6260 ±.0358	.6300 ±.0354
Page	.9467 ± .0026	.9330 ± .0021	.9377 ± .0028	.9442 ± .0023	.9467 ± .0016	.9380 ±.0179	.9240 ±.0737	.9220 ±.0841	.9420 ±.0402	.9520 ±.0507
Pen-Digit	.9521 ± .0014	.8575 ± .0030	.6093 ± .0056	.9559 ± .0006	.9578 ± .0004	.9400 ±.0316	.7920 ±.0427	.4020 ± .0335	.9500 ± .0308	.9580 ±.0228
Vehicle	.6582 ± .0008	.6286 ± .0052	.5942 ± .0081	.6720 ± .0010	.6181 ± .0009	.6800 ± .0682	.6420 ± .0228	.5320 ±.0335	.6780 ±.0217	.6420 ±.0327
Vowel	.8043 ± .0025	.5828 ± .0121	.2224 ± .0249	.7956 ± .0050	.8546 ± .0037	.8380 ±.0327	.5360 ±.3344	.0160 ±.0089	.8260 ± .0555	.8840 ±.0518
Media	.7541	.6563	.6080	.7476	.7151	.7541	.6563	.6080	.7476	.7151
Desv.	.1948	.2165	.2696	.2111	.2232	.1948	.2165	.2696	.2111	.2232

Tabla 6.12: Precisión absoluta y Precisión del flujo para cada algoritmo con cada data set de UCI-MLR.

existe un desbalance considerable entre las clases, permitiendo al clasificador obtener cerca del 100 % de precisión en la clase mayoritaria, pero un bajo redimiento en la clase minoritaria. Un buen clasificador consigue buenos resultados para ambas clases en términos de recall, lo cual garantiza su efectividad para identificar los elementos de cada clase. La tabla 6.15 muestra los resultados de Precisión/Recall/f-Measure para cada clasificador, con flujos de datos reales binarios, por clases. Las clases minoritarias son marcadas con * y los mejores resultados de la medida f-measure se encuentran resaltados.

SimCτ obtiene buenos resultados para cada clase, en especial para las clases más desbalanceada. En todos los casos SimCτ obtiene los mejores resultados con las clases minoritarias superando al resto de los algoritmos de la comparación que obtienen rendimientos muy pobres en cada caso. Estos resultados se

	Tiempo medio de actualización					Tiempo medio de clasificación				
Flujo de datos	SimCτ	IBL-DS	LWF	TWF	Win400	SimCτ	IBL-DS	LWF	TWF	Win400
Car	.0364 ± .0053	.6782 ± .0408	3.9607 ± .1895	.0335 ± .0033	.0272 ±.0101	.3926 ±.0507	.3205 ±.0321	1.177 ±.2659	.1535 ±.0096	.1788 ±.0194
Nursery	.0671 ± .0063	.7137 ± .0311	.6668 ± .0902	.0524 ± .0153	.0328 ± .0046	.6078 ±.0235	.3390 ±.0479	.4505 ± .0971	.2238 ± .0290	.2162 ± .0222
Balance	.0272 ± .0075	1.162 ± .2668	.7477 ± .0905	.0016 ± .0008	.0133 ± .0047	.1483 ± .0140	.1175 ±.0125	.2513 ±.0583	.1071 ±.0070	.0362 ±.0079
Agrawal	.0602 ± .0076	3.132 ± 1.366	2.641 ± .5041	.0526 ± .0081	.0241 ± .0034	.5167 ±.0889	.7173 ±.1166	1.873 ± .2163	.2567 ± .0066	.0877 ±.0090
Led24	.3620 ± .0257	1.784 ± .0475	3.273 ± .0900	.1781 ± .0134	.0747 ± .0138	4.016 ±.1421	.8781 ±.0234	3.022 ±.0983	.5413 ±.0308	.9096 ±.0478
Ramdomrbf	.0631 ± .0093	5.484 ± 2.474	5.101 ± .9327	.0028 ±.0013	.0308 ±.0039	.4657 ±.0085	.8657 ±.1660	4.763 ±.7224	.2581 ±.0149	.3872 ±.0127
Rdg	.0811 ± .0057	1.066 ± .0259	1.528 ± .1314	.0310 ± .0054	.0310 ± .0035	1.058 ±.2420	.5107 ±.0123	.8501 ± .1400	.2539 ± .0140	.3114 ±.0177
Breast	.0430 ± .0099	2.729 ± .5503	.7347 ± .1014	.0030 ± .0007	.0247 ± .0059	.5512 ±.0086	.1717 ± .0177	.4565 ± .1121	.2400 ± .0202	.0749 ±.0058
Letter	.1218 ± .0114	3.015 ± .2336	1.205 ± .0772	.0039 ± .0030	.0464 ± .0072	.4849 ±.0383	.1995 ±.0191	1.129 ± .1104	.3995 ±.0148	.4830 ±.0398
Page	.0484 ± .0096	1.897 ± .1483	2.744 ± .1951	.0017 ±.0007	.0282 ±.0043	.2071 ±.0125	.3629 ±.0224	1.558 ± .0619	.2716 ± .0217	.1063 ±.0160
Pen-Digit	.0675 ± .0074	2.887 ± .3444	1.830 ± .0991	.0048 ± .0013	.0517 ± .0105	.9061 ± .0297	.5064 ± .0539	1.405 ±.0463	.4603 ± .0232	.2697 ±.0440
Vehicle	.0592 ± .0055	3.155 ± .1610	1.426 ± .1149	.0073 ± .0021	.0427 ± .0046	.3739 ± .0068	.4347 ±.0310	1.035 ±.0601	.5499 ± .0242	.2179 ±.0249
Vowel	.1110 ± .0116	1.787 ± .1379	1.361 ± .0581	.0779 ± .0124	.0508 ± .0039	1.340 ± .0667	.2126 ± .0135	.8670 ±.0615	.4623 ± .0136	.330 ± .0330
Media	.0949	2.554	2.054	.0331	.0380	.9154	.4525	1.564	.3455	.2922
Desv.	.0929	1.240	1.302	.0543	.0172	1.0899	.2737	1.300	.1439	.2497

Tabla 6.13: Tiempo medio de actualización y Tiempo medio de clasificación (en milisegundos) para cada algoritmo con cada data set de UCI-MLR.

deben a que SimCτ, a diferencia del resto de los modelos de la comparación, logra mantener un balance entre todas las clases que componen el problema, evitando que el desbalance favorezca a la clase mayoritaria. La estructura flexible de SimCτ, formada por grupos en cada clase y su política de inserción/eliminación garantizan la permanencia de los ejemplos más significativos desde el punto de vista espacial y temporal, lo que resulta muy efectivo en conjuntos de datos desbalanceados, donde el principal problema radica en mejorar el rendimiento del clasificador con la clase minoritaria.

El método para el tratamiento de cambios de concepto está basado en un hecho simple, cuando existen diferencias entre las principales medidas de precisión, se asume que está ocurriendo un cambio de concepto y el modelo se prepara para recibir la nueva información que trae este cambio. Cuando el cambio es

	Precisión Absoluta				Precisión del Flujo			
Flujo de datos	SimCτ	IBL-DS	TWF	Win400	SimCτ	IBL-DS	TWF	Win400
Airlines	.5877	.6075	.1984	.5575	.58	.57	.21	.55
Bank	.8822	.8669	.8798	.8766	.63	.64	.57	.58
Poker Hand	.4438	.4469	.4652	.4719	.33	.47	.48	.58
Covtype	.9325	.9287	.2869	.9541	1.0	1.0	1.0	1.0
KDD	.9628	.9479	.9685	.9539	.98	.90	.95	.97
Sea	.7922	.7886	.8508	.8045	.81	.80	.81	.78
Usenet recurrent	.5666	.5852	.6002	.5870	.60	.59	.59	.52
Elec	.8094	.8018	.7904	.7823	.89	.77	.84	.83
Media	.7471	.7466	.6300	.7484	.72	.71	.68	.72
Desv.	.1908	.1805	.2887	.1867	.23	.18	.26	.19

Tabla 6.14: Precisión absoluta y Precisión del flujo para cada algoritmo con cada flujo de datos real.

detectado, la estructura del modelo mantiene el balance entre las clases, agregando los nuevos casos que llegan y actualizando los grupos por cada clase. Los resultados demuestran que SimCτ constituye una propuesta eficiente y eficaz para la clasificación de flujos de datos desbalanceados y cambiantes.

Clases	SimCτ	IBL-DS	TWF	WIN400
	P/TPR/f-Measure	P/TPR/f-Measure	P/TPR/f-Measure	P/TPR/f-Measure
Bank				
Yes*	.49/.49/**.49**	.59/.20/.30	.70/.12/.20	.76/.07/.13
No	.93/.93/.93	.90/.98/.94	.89/.99/.94	.89/.99/.94
Elect				
UP*	.78/.85/**.81**	.88/.68/.77	.95/.52/.68	.95/.51/.66
Down	.88/.82/.85	.80/.93/.86	.73/.98/.84	.73/.98/.83
KDD				
Anomaly*	.96/.95/**.96**	.98/.91/.94	.99/.87/.93	.99/.85/.92
Normal	.96/.96/.96	.92/.99/.95	.90/.99/.94	.89/.99/.94
Sea				
Clase0*	.71/.73/**.72**	.83/.59/.69	.89/.49/.63	.89/.49/.63
Clase1	.83/.82/.83	.793/.93/.85	.76/.96/.85	.76/.96/.85

Tabla 6.15: Rendimiento por clases.

6.3 Conclusiones Parciales

SimCτ ha sido probado con conjuntos de datos de diferentes características y evaluado estadísticamente con varias medidas de rendimiento. A partir del análisis realizado se puede concluir que:

- Con el experimento realizado para obtener un valor adecuado del parámetro τ en el algoritmo 5, propuesto para el tratamiento a los cambios de concepto abruptos, concluimos en la utilización del valor $\tau = 0.3$, para el resto de los experimentos, obteniéndose buenos resultados en la mayoría de los casos. Sin embargo, específicamente se propone $\tau = 0.3$ para flujos de datos con cambios de concepto abruptos y $\tau = 0.1$ para los que no presentan cambios o presentan cambios de concepto graduales.

- El algoritmo 5, es un método efectivo, que resuelve directamente los cambios de concepto abruptos que ocurren en la función objetivo. Su combinación con el algoritmo 4, presenta resultados estadísticamente superiores a la aplicación de éste último únicamente.

- Para conjuntos de datos sintéticos:

  - En las medidas de precisión SimCτ obtiene los mejores porcentajes de clasificaciones correctas. Siendo estadísticamente superior a los algoritmos IBL-DS, TWF y WIN400 y no presentando diferencias significativas con el modelo LWF. En estas medidas obtiene buenos resultados, con flujos de datos de alta dimensionalidad y de atributos mezclados.
  - En la medida de tiempo medio de actualización, SimCτ es significativamente superior a IBL-DS y LWF.

- Para conjuntos de la UCI-MLR:

  - En la medida de precisión absoluta, SimCτ es estadísticamente superior a los algoritmos IBL-DS, LWF y WIN400 y no presenta diferencias significativas con TWF. Obtiene los mejores resultados con conjuntos de datos de alta dimensionalidad y de atributos mezclados.
  - En la medida de precisión del flujo, SimCτ es estadísticamente superior a los algoritmos IBL-DS y LWF, y no presenta diferencias significativas con TWF y WIN400.

  - En la medida de tiempo medio de actualización, SIMCτ es significativamente superior IBL-DS y LWF.
  - En la medida de tiempo medio de clasificación, SIMCτ es significativamente superior a LWF y no presenta diferencias con respecto a IBL-DS. TWF y WIN400 presentan los mejores resultados para esta medida.

- Para flujos de datos reales:
  - En las medida de precisión, SIMCτ no presenta diferencias significativas con el resto de los modelos comparados.
  - En el análisis del rendimiento por clases, SIMCτ obtiene los mejores resultados en flujos de datos desbalanceados, demostrando ser más efectivo para identificar los elementos de la clase minoritaria, que el resto de los modelos de la comparación.

Capítulo 7

CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

Como resultado de la presente tesis se llegó a las siguientes conclusiones:

- Hemos realizado un estudio de los retos y problemas asociados a la minería de flujos de datos y las principales técnicas de aprendizaje utilizadas para enfrentarlos (ver capítulo II).
- Hemos realizado un análisis de las técnicas del Aprendizaje Basado en Instancias, con los principales elementos que la constituyen (ver Capítulo III).
- Hemos realizado un estudio de los algoritmos para minería de flujos de datos propuestos en la bibliografía, específicamente los modelo de agrupamiento y clasificación y sus principales ventajas y desventajas, haciendo énfasis en aquéllos que utilizan la técnica del Aprendizaje Basado en Instancias y sus métodos para tratar los problemas asociados al aprendizaje en flujos de datos (ver Capítulo IV).
- Hemos propuesto un algoritmo de clasificación para flujo de datos que combina de forma óptima atributos numéricos y nominales, a partir de la utilización de la medida de distancia actualizable HVDM. El modelo propuesto almacena un conjunto de casos en memoria de forma organizada,

formando grupos de instancias según su similitud. La política de inserción/eliminación que implementa está basada en el uso de estimadores diseñados para apoyar la correcta selección de los casos que deben ser almacenados y adaptarse a los cambios de concepto graduales que puedan ocurrir en la función objetivo. El algoritmo propuesto logra diferenciar entre el ruido y los cambios de concepto ocurridos, lo que le permite adaptarse a estos cambios de concepto abruptos a través de un método propio que combina dos medidas de precisión. Además la flexibilidad de su diseño le permitirá adaptarse a contextos recurrentes, en caso de que la problemática lo requiera (ver Capítulo V).

- Hemos validado la propuesta con datos sintéticos que simulan la ocurrencia de cambios de concepto, con bases de datos internacionales de la UCI Machine Learning Repository (ver Capítulo VI) y con problemas de flujos de datos reales disponibles en la web, donde se verificó la superioridad estadística del modelo propuesto, con respecto a otros algoritmos de la bibliografía, utilizandos en la clasificación de flujos de datos. Además se concluye que SimCτ obtiene los mejores resultados en conjuntos de datos desbalanceados y que presentan cambios de concepto.

7.2 Trabajo futuro

- En cuanto al algoritmo de clasificación para flujos de datos propuesto, hay varias direcciones de trabajo que se podrían llevar a cabo en el futuro:

    1. Diseñar y validar una estrategia que permita al modelo adaptarse a entornos recurrentes, a partir de la utilización de los casos eliminados.
    2. Diseñar y validar un mecanismo que permita al modelo ajustar el tamaño de la base de casos, en cada momento. Mejorando así la eficiencia y eficacia del algoritmo.
    3. Diseñar y validar un mecanismo que permita incluir nuevas características del problema en cualquier momento, utilizando además algún método de selección de atributos, para evaluar la importancia de los mismo en el proceso de clasificación.

- En cuanto a la aplicación del algoritmo de clasificación para flujos de datos propuesto:

    1. Obtener una herramienta informática para el apoyo a la toma de desiciones en los servicios agrometeorológicos del Centro Provincial de Meteorología de Pinar del Río, utilizando el modelo propuesto.

Bibliografía

[1] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive Online Analysis,J. Mach. Learn. Res., vol. 11, pp. 1601–1604, 2010.

[2] A. Guerrieri, A. Montresor, Ds-means: distributed data stream clustering, En *Proceedings of the 18th international conference on Parallel Processing, Euro-Par'12*, vol. 7484, pp. 260–271, 2012.

[3] A. M. Narasimhamurthy, L. I. Kuncheva, A framework for generating data to simulate changing environments, *Proceedings of the 25th Conference on Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications, AIAP'07*, pp. 384–389, 2007.

[4] A. Orriols-puig, J. Casillas, E. Bernadó-Mansilla, Fuzzy-UCS: A Michigan-style Learning Fuzzy-Classifier System for Supervised Learning, *IEEE Transactions on Evolutionary Computation*, vol. 13(2), pp. 260–283, 2009.

[5] A. Patcha, J. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, *Computer Networks*, vol. 51(12), pp. 3448–3470, 2007.

[6] A. Shaker, E. Hullermeier, IBLStreams: A System for Instance-Based Classification and Regression on Data Streams, Evolving Systems, vol. 3(4), pp. 235-249, 2012.

[7] A. Shaker, R. Senge, E. Hullermeier, Evolving fuzzy pattern trees for binary classification on data streams, Information Sciences, vol. 220, pp. 34–45, 2013.

[8] A. Skowron, A. Wojna, K nearest neighbor classification with local induction of the simple value dierence metric, En *In Proceedings of the Fourth Internatinal Conference on Rough Sets and Current Trends in Computing, Lectures Notes in Artificial Intelligence, Springer-Verlag*, vol. 3066, pp. 229–234, 2004.

[9] A. Tsymbal, The problem of concept drift: Definitions and related work, Technical report, Trinity College Dublin, Ireland, 2004.

[10] A. Tsymbal, M. Pechenizkiy, P. Cunningham, S. Puuronen, Dynamic integration of classifiers for handling concept drift, *Inf. Fusion*, vol. 9(1), pp. 56–68, 2008.

[11] A. Wojna, Center-based indexing in vector and metric spaces, *Fundamenta Informaticae XX*, vol. 56(3), pp. 1–26, 2002.

[12] B. Babcock, M. Datar, R. Motwani, L. O'Callaghan, Maintaining variance and k- medians over data stream windows, *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '03*, pp. 234–243, 2003.

[13] B. G. Batchelor, Pattern recognition: Ideas in practice, *Springer Dordrecht*, 1978.

[14] C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, *Proceedings of the 29th International Conference on Very Large Data Bases, VLDB '03*, vol.29, pp. 81–92, 2003.

[15] C. Aggarwal, On high dimensional projected clustering of data streams, *Data Mining and Knowledge Discovery*, vol. 10(3), pp. 251–273, 2005.

[16] C. Aggarwal, P.S. Yu, A framework for clustering uncertain data streams, *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, ICDE '08*, pp. 150–159, 2008.

[17] C. Li, Y. Zhang, X. Li, OcVFDT: One-class very fast decision tree for one-class classification of data streams,Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data, SensorKDD '09, pp. 79–86, 2009

[18] C. Ordonez, Clustering binary data streams with k-means, *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03*, pp. 12–19, 2003.

[19] C. Stanfill, D. Waltz. Toward memory based reasoning, *Communications of the ACM*, vol. 29(12), pp. 1213–1228, 1986.

[20] C. Zhang, C. Jin, A. Zhou, Efficiently clustering probabilistic data streams, *Advances in Data and Web Management: Proceedings of Joint International Conferences*, vol. 5446, pp. 273–284, 2009.

[21] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25(1), pp. 81–94, 2013.

[22] D. Charles, A. Kerr, M. McAlister, M. McNeill, J. Kucklich, M.M. Black, A. Moore, K. Stringer, Player-centred game design: Adaptive digital games, *Digital Games Research Association, Changing Views - Worlds in Play*, 2005.

[23] D.J. Newman, S. Hettich, C.L. Blake, y C.J. Merz, UCI repository of machine learning databases, http://www.ics.uci.edu/∼mlearn/MLRepository.html, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[24] D. Randall-Wilson, T.R. Martinez, Value difference metrics for continuously valued attributes, *Proceedings of the International Conference on Artificial Intelligence, Expert Systems and Neural Networks, AIE '96*, pp. 11–14,1996.

[25] D. Randall-Wilson, T.R. Martinez, Improved Heterogeneous Distance Functions. *Artificial Intelligence*, vol. 6, pp. 1–34, 1997.

[26] D.W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, *Machine Learning*, vol. 6, pp. 37–66, 1991.

[27] D. Wettschereck, D. W. Aha, T. Mohri, Review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, *Artif. Intell. Rev.*, vol. 11, pp. 273–314, 1997.

[28] D. Yang, E.A. Rundensteiner, M.O. Ward, Mining neighbor-based patterns in data streams, Information Systems, vol. 38, pp. 331–350, 2013.

[29] E. Blanzieri, F. Ricci, A minimum risk metric for nearest neighbor classification, *Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99*, pp. 22–31, 1999.

[30] E. Cohen, M. J. Strauss, Maintaining time-decaying stream aggregates, *J. Algorithms*, vol. 59(1), pp. 19–36, 2006.

[31] E. Ikonomovska, S. Loskovska, D. Gjorgjevik, A survey of stream data mining, PhD thesis, University of California Santa Cruz, 2007.

[32] E. S. Page, Continuous inspection schemes, *Biometrika*, vol. 41(1/2), pp. 100–115, 1954.

[33] F. Cao, Density-based clustering over an evolving data stream with noise, *Proceedings of the 6th SIAM International Conference on Data Mining*, pp. 326–337, 2006.

[34] F.J. Ferrer, J.S. Aguilar, J.C. Riquelme, Discovering decision rules from numerical data streams, *Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04*, pp. 649–653, 2004.

[35] F. Masseglia, A. Marascu, Atypicity detection in data streams:a self adjusting approach, *Intelligent Data Analysis*, vol. 15, pp. 89–105, 2011.

[36] G. Gora, A. Wojna, Riona: A classifier combining rule induction and k-nn method with automated selection, *Proceedings of the 13th European Conference on Machine Learning, ECML '02*, pp. 111–123, 2002.

[37] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01*, pp. 97–106, 2001.

[38] G. Jing, K. Latifur, H. Jiawei, W.H. Kevin, C.O. Nikunj, M.M. Mohammad, W. Clay, Facing the reality of data stream classification: coping with scarcity of labeled data, *Knowl Inf Syst*, vol. 33, pp. 213–244, 2012.

[39] G. Lee, U. Yun, K.H. Ryu, Sliding window based weighted maximal frequent pattern mining over data streams, Expert Systems with Applications, vol. 41, pp. 694–708, 2014.

[40] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, *Machine Learning*, vol. 23(1), pp. 69–101, 1996.

[41] H. Abdulsalam, D.B. Skillicorn, P. Martin, Classifying evolving data streams using dynamic streaming random forests, *Database and Expert Systems Applications: Proceedings of 19th International Conference*, vol. 5181, pp. 643–651, 2008.

[42] H. Wang, W. Fan, P. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03*, pp. 226–235, 2003.

[43] H. Wang, W. Dubitzky, A flexible and robust similarity measure based on contextual probability, *Proceedings of IJCAI'05*, pp. 27–32, 2005.

[44] I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, I. P. Vlahavas, An adaptive personalized news dissemination system, *J. Intell. Inf. Syst.*, vol. 32(2), pp. 191–212, 2009.

[45] I. Zliobaitye, Adaptive training set formation, *PhD thesis, Vilnius University*, 2010.

[46] J.C. Schlimmer, R.H. Granger, Incremental learning from noisy data, *Machine Learning*, vol. 1(3), pp. 317–354, 1986.

[47] J. Beringer, E. Hullermeier, Online clustering of parallel data streams, *Data Knowl. Eng.*, vol. 58(2), pp. 180–204, 2006.

[48] J. Beringer, E. Hullermeier, Efficient Instance-Based Learning on Data Streams, *Intelligent Data Analysis*, vol. 11(6), pp. 627–650, 2007.

[49] J. Demsar, Statistical comparisons of clasiffiers over multiple data sets, *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[50] J. Gao, An incremental data stream clustering algorithm based on dense units detection, *Advances in Knowledge Discovery and Data Mining: Proceedings of 9th Pacific- Asia Conference*, vol. 3518, pp. 420–425, 2005.

[51] J. Gama, P. Medas, R. Rocha. Forest Trees for On-line Data. *Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04*, pp. 632–636, 2004.

[52] J. Gama, P. Medas, P. Rodrigues, Learning decision trees from dynamic data streams, *Proceedings of the 2005 ACM Symposium on Applied Computing, SAC '05*, pp. 573–577, 2005.

[53] J. Gama, P. Pereira, Stream-based electricity load forecast, *Lecture Notes in Computer Science*, vol. 4702, pp. 446–453, 2007.

[54] J. Han, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers Inc., 2005.

[55] J. MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[56] J. Mendes-Moreira, C. Soares, M. J. Alypio, J.F. deSousa, The effect of varying parameters and focusing on bus travel time prediction, *Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science Volume*, vol. 5476, pp. 689–696, 2009.

[57] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers Inc., 1994.

[58] J. Read, A. Bifet, G. Holmes, B. Pfahringer, Scalable and efficient multi-label classification for evolving data streams, *Machine Learning*, vol. 88, pp. 243–272, 2012.

[59] J. Riquelme, F.J. Ferrer, J.S. Aguilar, Incremental rule learning and border examples selection from numerical data streams, *Proceedings of the 2005 ACM Symposium on Applied Computing, SAC '05*, pp. 568–572, 2005.

[60] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, *Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

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

[62] K.O. Stanley, Learning concept drift with a committee of decision trees, Technical Report AI03-302, Department of Computer Sciences, The Uni-

versity of Texas at Austin, 2003.

[63] K. Pripuzic, I. Podnar, K. Aberer, Distributed processing of continuous sliding-window k-NN queries for data stream filtering, World Wide Web, vol. 14(5-6), pp. 465–494, 2011.

[64] L. Kurgan, K. Cios, Caim discretization algorithm, *IEEE Transactions on Knowledge and Data Engineering*, vol. 16(2), pp. 145–153, 2004.

[65] L. Lap-Kei, T. Hing-Fung, C. Ho-Leung, L. Tak-Wah, Continuous monitoring of distributed data streams over a time-based sliding window, *Algorithmica*, vol. 62, pp. 1088–1111, 2012

[66] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani, Streaming-data algorithms for highquality clustering, *Proceedings of the 18th International Conference on Data Engineering, ICDE '02*, pp. 685–695, 2002.

[67] L. Su, H. Liu, Z. Song. A new classification algorithm for data stream, *IJMECS*, vol. 3(4), pp. 32–39, 2011.

[68] L. Zhao, L. Wang, Q. Xu, Data stream classification with artificial endocrine system,Appl Intell, vol. 37, pp. 390–404, 2012.

[69] M. A. Hossain, C. M. Rahman, R. Strachan, G. Sexton, K.P. Dahal, D. Farid, L. Zhang, An adaptive ensemble classifier for mining concept drifting data streams, *Expert Syst. Appl.*, vol. 40(15), pp. 5895–5906, 2013.

[70] M. Baena, J. del Campo, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales, Early drift detection method, *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

[71] M. Charikar, L. O'Callaghan, R. Panigrahy, Better streaming algorithms for clustering problems, *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing, STOC '03*, pp. 30–39, 2003.

[72] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: An update, *SIGKDD Explorations*, vol. 11(1), 2009.

[73] M. R. Ackermann, C. Lammersen, M. Märtens, C. Raupach, C. Sohler, K. Swierkot, Streamkm++: A clustering algorithms for data streams, En

ACM Journal of Experimental Algorithmics, vol. 17(1), 2012.

[74] M. Salganicoff, Tolerating Concept and Sampling Shift in Lazy Learning Using Prediction Error Context Switching, *Artificial Intelligence Review*, vol. 11, pp. 133–155, 1997.

[75] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Information Processing and Management, vol. 45, pp. 427–43, 2009.

[76] M. Toyoda, Y. Sakurai, Y. Ishikawa, Pattern discovery in data streams under the time warping distance, The VLDB Journal, vol. 22, pp. 295–318, 2013.

[77] N.M. Adams, D.J. Hand, N.G. Pavlidis, D.K. Tasoulis, λ-perceptron: An adaptive classifier for data streams, *Pattern Recognition*, vol. 44, pp. 78–96, 2011.

[78] N. Mozafari, S. Hashemi, A. Hamzeh, A Precise Statistical approach for concept change detection in unlabeled data streams, Computers and Mathematics with Applications, vol. 62, pp. 1655–1669, 2011.

[79] O. A. Omitaomu, R. R. Vatsavai, A.R. Ganguly, N.V. Chawla, J. Gama, M.M. Gaber, Knowledge discovery from sensor data (SensorKDD), *SIGKDD Explor. Newsl.*, vol. 11(2), pp. 84–87, 2009.

[80] P. Domingos, G. Hulten, Mining High-Speed Data Streams, *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00*, pp. 71–80, 2000.

[81] P. Kontkanen, J. Lahtinen, P. Myllymaki, H. Tirri, An unsupervised bayesian distance measure, *Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning, EWCBR '00*, pp. 148–160, 2000.

[82] P. Sobolewski, M. Wozniak, Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors, *Journal of Universal Computer Science*, vol. 19(4), pp. 462–483, 2013.

[83] P. Vivekanandan, R. Nedunchezhian, Mining data streams with concept drifts using genetic algorithm, Artif Intell Rev, vol. 36, pp. 163–178, 2011.

[84] P. Zhang, X. Zhu, Y. Shi, L. Guo, X. Wu, Robust ensemble learning for mining noisy data streams, Decision Support Systems, vol. 50, pp. 469–479, 2011.

[85] Q. Quan, C. Xiao, R. Zhang, Grid-based data stream clustering for intrusion detection, *International Journal of Network Security*, vol. 15(1), pp. 1–8, 2013.

[86] Q. Xiangju, Z. Yang, L. Chen, L. Xue, Learning from data streams with only positive and unlabeled data, Intell Inf Syst, vol. 40, pp. 405–430, 2013.

[87] R. Bergmann, M. M. Richter, S. Schmitt, A. Stahl, I. Vollrath, Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning, *Proceedings of the 1st Conference on Professional Knowledge Management*, pp. 264–274, 2001.

[88] R. Klinkenberg, T. Joachims, Detecting Concept Drift with Support Vector Machines, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pp. 487–494, 2000.

[89] R. Klinkenberg, Meta-Learning, Model Selection and Example Selection in Machine Learning Domains with Concept Drift, *Annual Workshop of the Special Interest Group on Machine Learning, Knowledge Discovery, and Data Mining, FGML '05*, 2005.

[90] R. Polikar, L. Udpa, S.S. Udpa, V. Honavar, LEARN ++: an Incremental Learning Algorithm For Multilayer Perceptron Networks, *IEEE Transactions on System, Man and Cybernetics (C), Special Issue on Knowledge Management*, vol. 31, pp. 497–508, 2000.

[91] R. Short, K. Fukunaga, The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory*, vol. 27(5), pp. 622–627, 1981.

[92] S. Cost, S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features, *Machine Learning*, vol. 10, pp. 57–78, 1993.

[93] S. Donoho, Early detection of insider trading in option markets, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pp. 420–429, 2004.

[94] S. Garcia, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[95] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: Theory and practice, *IEEE Trans. on Knowl. and Data Eng.*, vol. 15(3), pp. 515–528, 2003.

[96] S. Hashemi, Y. Yang. Flexible decision tree for data stream classification in the presence of concept change, noise and missing values, *Data Mining and Knowledge Discovery*, vol. 19, pp. 95–131, 2009.

[97] S. Luhr, M. Lazaresc Incremental clustering of dynamic data streams using connectivity based representative points, *Data Knowl. Eng.*, vol. 68(1), pp. 1–27, 2009.

[98] S. Miyamoto, K. Mori, H. Ihara, Autonomous decentralized control and its application to the rapid transit system, Comput Ind, vol. 5, pp. 115–124, 1984.

[99] S. W. Roberts, Control chart tests based on geometric moving averages, *Technometrics*, vol. 42(1), pp. 97–101, 2000.

[100] T. Lane, C. E. Brodley, Temporal sequence learning and data reduction for anomaly detection, *ACM Trans. Inf. Syst. Secur.*, vol. 2(3), pp. 295–331, 1999.

[101] T. M. Cover, P. E. Hart, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.

[102] T. R. Payne, P. Edwards, Interface agents that learn: An investigation of learning issues in a mail agent interface, *Applied Artificial Intelligence*, vol. 11(1), pp. 1–32, 1997.

[103] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, *SIGMOD Rec.*, vol. 25(2), pp. 103–114, 1996.

[104] V. Bhatnagar, S. Kaur, Exclusive and complete clustering of streams, *Database and Expert Systems Applications: Proceedings of 18th International Conference*, vol. 4653, pp. 629–638, 2007.

[105] W. Duch, Similarity based methods: a general framework for classification, approximation and association, *Control And Cybernetics*, vol. 29(4), pp. 937–968, 2000.

[106] W. Xindong, L. Peipei, H. Xuegang, Learning from concept driftig data streams with unlabeled data, Neurocomputing, vol. 92, pp. 145–155, 2012.

[107] Y. Law, C. Zaniolo, An adaptive nearest neighbor classification algorithm for data streams, *Proceedings of the Ninth European Conference on Principles and Practice of Knowledge Discovery in Databases*, vol. 3721, pp. 108–120, 2005.

[108] Y. Li, Y. Zhang, H. Xuegang, L. Peipei, A classification algorithm for noisy data streams, 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, FSKD '10, vol. 5, pp. 2239–2244, 2010.

[109] Y. Lu, A grid-based clustering algorithm for high-dimensional data streams, *Advanced Data Mining and Applications: Proceedings of First International Conference*, vol. 3584, pp. 824–831, 2005.

[110] Y. Wang, Z. Li, Y. Zhang, Classifying noisy data streams, *Fuzzy Systems and Knowledge Discovery: Proceedings of 3rd International Conference*, vol. 4223, pp. 549–558, 2006.