UNIVERSIDAD DE GRANADA

DOCTORADO IBEROAMERICANO EN INFORMÁTICA (SOFT COMPUTING)

Tesis Doctoral

"ALGORITMO MULTICLASIFICADOR CON APRENDIZAJE INCREMENTAL QUE MANIPULA CAMBIOS DE CONCEPTOS"

Autor

Agustín Alejandro Ortiz Díaz [(1)].

Directores:

Doctor Gonzalo Ramos Jiménez [(2)],

Doctor Rafael Morales Bueno [(2)],

(1) Universidad de Granma, Cuba.

(2) Universidad de Málaga, España.

Granada, España

2014

Editor: Editorial de la Universidad de Granada
Autor: Agustín Alejandro Ortiz Díaz
D.L.: GR 2346-2014
ISBN: 978-84-9083-390-2

El doctorando Agustín Alejandro Ortíz Díaz y los directores de la tesis Gonzalo Ramos Jiménez y Rafael Morales Bueno garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro 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, España. Octubre de 2014.

Directores de la Tesis.

Gonzalo Ramos Jiménez
Fdo.:

Rafael Morales Bueno
Fdo.:

Doctorando

Agustín Alejandro Ortíz Díaz

Fdo.:

DEDICATORIA.

A mi familia,

en especial a mis padres,

a mi hermana y a Laura.

AGRADECIMIENTOS.

En primer lugar me gustaría agradecer infinitamente a mis tutores, Gonzalo Ramos Jiménez y Rafael Morales Bueno, pues sin su dirección no hubiese sido posible ni siquiera empezar esta investigación. El mismo nivel de agradecimiento es para José del Campo Ávila y Yailé Caballero Mota, que aunque no están registrados como tutores, en la práctica se han comportado como tales, brindando toda la ayuda que les ha sido posible.

Agradecer al CEI, en la Universidad Central “Martha Abreu” de Las Villas; al DECSAI, en la Universidad de Granada; al Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga y a la Asociación Universitaria Iberoamericana de Postgrado (AUIP), principalmente por el apoyo desde el punto de vista investigativo, pero también por el apoyo económico.

Un aparte especial debo hacer para todos los profesores del Departamento de Informática de la Universidad de Granma, fundamentalmente por el tiempo que me han proporcionado, responsabilizándose de mis tareas docentes por un largo periodo de tiempo; pero además, por el apoyo científico y anímico que me han brindado.

Ya en lo particular, deseo agradecer a todos mis amigos y entre ellos destacar a mis antiguos alumnos, hoy amigos y compañeros de trabajo, Alberto verdecia, David La O y Ramón Ramírez Taset.

Y por último, pero muy especialmente, a mis padres, mi hermana y mi eterna compañera Laura María.

El tratamiento de grandes flujos de datos en presencia de cambios de conceptos es uno de los principales retos en el área de la minería de datos, particularmente cuando los algoritmos tienen que tratar con conceptos recurrentes. Diferentes modelos de clasificación han sido adaptados para manipular flujos de datos bajo estas condiciones, destacándose en los últimos tiempos los sistemas multiclasificadores, con los cuales se procesan un conjunto de ejemplos, teniendo en cuentas la combinación de las predicciones independientes de varios clasificadores básicos, iguales o diferentes. La presente investigación propone un nuevo algoritmo llamado “Multiclasificador de Adaptación Rápida” (FAE) que cuenta con mecanismos capaces de actualizar sus estructuras de forma acelerada para el tratamiento de cambios de conceptos graduales, abruptos o recurrentes.

Para llegar a esta propuesta se ha partido básicamente del algoritmo MultiCIDIM-DS, desarrollado por un grupo de investigadores de la Universidad de Málaga y a través de un estudio de las principales tendencias, fundamentalmente de los algoritmos multiclasificadores para trabajo en línea, se han logrado mejoras sustanciales en el tratamiento de los diferentes tipos de cambios de conceptos. FAE procesa los ejemplos de entrenamiento en bloques de igual tamaño, pero no tiene que esperar que el próximo bloque de datos esté completo para adaptar sus mecanismos básicos de clasificación. El nuevo algoritmo incorpora un detector de cambio con vista a mejorar la manipulación de cambios de conceptos abruptos. Además, almacena un conjunto de clasificadores inactivos que representan a viejos conceptos, los cuales se activan de forma muy rápida cuando estos conceptos reaparecen.

Paralelamente se ha realizado un estudio de las principales metodologías, herramientas de software y conjuntos de datos utilizados para la evaluación y comparación de algoritmos incrementales. El nuevo algoritmo ha sido implementado bajo las condiciones del entorno de trabajo MOA y dentro de este, ha sido probado junto a propuestas anteriores sobre conocidos conjuntos de datos, tanto artificiales como reales, alcanzando resultados promisorios.

ÍNDICE GENERAL.

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

En las últimas décadas, el almacenamiento, organización y recuperación de la información se ha automatizado gracias a los sistemas de bases de datos, pero la ubicuidad de la información en formato electrónico ha empezado a ser patente a finales de siglo XX con la irrupción de Internet. Como resultado de este tremendo crecimiento, los datos brutos, se han convertido en una vasta fuente de información. Gran parte de esta información es histórica, es decir, representa transacciones o situaciones que se han producido. Aparte de su función de "memoria", la información histórica es útil para explicar el pasado, entender el presente y predecir la información futura. La mayoría de las decisiones de empresas, organizaciones e instituciones se basan en información sobre experiencias pasadas extraídas de fuentes muy diversas por lo que el verdadero valor de los datos radica en la posibilidad de extraer de ellos información útil para la toma de decisiones o la exploración y comprensión de los fenómenos que le dieron lugar [1]. Debido a tal crecimiento y variedad de la información se hace necesario e importante el análisis de datos en ramas como: bioinformática, medicina, economía y finanzas, industria, medio ambiente, entre otras. Más importante aún es, además del conocimiento que puede inferirse y la capacidad de poder usarlo, tener un conjunto de “estructuras” que a partir de antecedentes, comportamiento y otras características de los datos nos permitan predecir su comportamiento futuro [2].

Para el análisis de datos con características especiales en cuanto a su dimensión (cientos de tablas con muchas columnas, millones de registros, tamaño de varios gigabytes, etc), almacenados en grandes cantidades o que llegan en tiempo real desde diversas fuentes, normalmente se usan técnicas de minería de datos (del inglés, *data mining*) que no es más que la búsqueda de patrones e importantes regularidades en bases de datos de gran volumen [3]. La minería de datos es un paso esencial dentro de un proceso mucho más amplio cuyo propósito es el descubrimiento de conocimiento en bases de datos (del inglés, *Knowledge Discovery in Databases, KDD*). Asociado a este proceso existe otro campo de la

ciencia de la computación llamado aprendizaje automático (*machine learning*), que trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos [4]. Las técnicas utilizadas en el aprendizaje automático pueden dividirse en dos grandes grupos: descriptivas y predictivas. Las primeras están en correspondencia con las tareas de detección de agrupamientos y correlaciones, y las segundas con las tareas de **clasificación** y regresión.

La clasificación es una tarea de la minería de datos ampliamente abordada desde distintas áreas del aprendizaje automático. En ella, cada instancia (o ejemplo) pertenece a una clase, la cual se indica mediante el valor de un atributo que es conocido como la clase de la instancia. Este atributo puede tomar diferentes valores discretos, cada uno de los cuales corresponde a una clase. El resto de los atributos de la instancia se utilizan para predecir la clase. El objetivo de la tarea de clasificación es predecir la clase de nuevas instancias de las que se desconoce esta.

Dos de las principales familias de técnicas de clasificación están formadas por los sistemas de inducción de modelos, fuera de línea (del inglés *off–line*) y en línea (del inglés *on-line*). Los primeros necesitan que todos los ejemplos necesarios para describir el dominio del problema estén disponibles antes del proceso de aprendizaje y el aprendizaje concluye cuando ha sido procesado el conjunto de entrenamiento. Las técnicas de aprendizaje de este primer grupo no pueden ser aplicadas en un gran número de problemas donde los datos proceden de entornos dinámicos y van siendo adquiridos a lo largo del tiempo, lo que obliga a procesarlos secuencialmente en sucesivos episodios, de forma incremental, con técnicas de clasificación en línea [5]. Estas grandes secuencias de datos, potencialmente infinitas, que se van adquiriendo a lo largo del tiempo son conocidas como **flujos de datos** (del inglés, *datastream*).

El proceso de aprendizaje incremental a partir de grandes volúmenes de datos, mucho más si no es posible disponer de todos los datos al iniciar el trabajo, acarrea ciertas restricciones [6]. Primero, debido a que la cantidad de datos a

procesar tiene un tamaño potencialmente infinito, es imposible almacenarlos completamente, por lo tanto, solo parte de estos datos puede ser procesada y almacenada, mientras que el resto suele ser desechado. Segundo, como los datos van llegando en el tiempo y con frecuencia a alta velocidad, estos deben ser procesados en tiempo real e irse descartando de forma permanente. El tercer aspecto a tener en cuenta, de gran importancia para el presente trabajo, es que la función de distribución de probabilidad a partir de la cual son generados estos datos puede variar en el tiempo. Esto puede traer como consecuencia que el aprendizaje logrado a partir de datos pasados puede tornarse inconsistente con respecto a la información que trasmiten los datos actuales. Como consecuencia, cambios ocurridos en el contexto pueden inducir variaciones en el modelo de aprendizaje utilizado, dando lugar a lo que se conoce como **cambio de concepto** (de inglés, *concept drift*) [5].

Debido a la presencia de cambios de conceptos al procesar flujos de datos, situación que es muy frecuente observar en problemas reales, y debido al impacto que estos cambios pueden causar en el modelo de aprendizaje, en general, los algoritmos de aprendizaje incremental deben incorporar mecanismos adicionales para manipular cambios de conceptos.

Existen muchos modelos de aprendizaje para representar el conocimiento extraído de un conjunto de datos, entre estos se encuentran, sistemas basados en reglas [7], Naïve Bayes [8], máquinas de soporte vectorial [9], redes neuronales artificiales [10], aprendizaje basado en casos [11], árboles de decisión [12] entre otros.

Es difícil, en la actualidad, encontrar un modelo de clasificación eficiente para resolver cada situación planteada. Se ha comprobado en investigaciones anteriores que ciertas partes del espacio de datos son mejor modeladas por un método de clasificación y otra parte, por un clasificador diferente. Esto ha originado que utilizar una combinación de clasificadores (Multiclasificación) sea una buena alternativa. Ya han sido propuestos varios métodos para la creación de combinaciones de clasificadores, aunque, no existe una clara forma de saber qué

método de multiclasificación es mejor que otro, o cuándo emplear un determinado método y cuándo uno diferente [13]. Ejemplos de algoritmos multiclasificadores lo constituyen los algoritmos, SEA (Streaming Ensemble Algorithm) [14], CBEA (Coverage Based Ensemble Algorithm) [15], MultiCIDIM-DS y MultiCIDIM-DS-CFC [13].

Todo lo antes planteado, constituye una problemática a la cual aún la ciencia no ha dado respuestas definitivas, para tratar de dar un paso más en este sentido, en la investigación actual se plantea la siguiente interrogante que constituye su ***problema de investigación***: ¿Cómo obtener un algoritmo multiclasificador con aprendizaje incremental para minar flujos de datos, que sea capaz de manipular cambios de conceptos graduales, abruptos o recurrentes?

Así, el ***objetivo*** de este trabajo es: Desarrollar un algoritmo con aprendizaje incremental para minar flujos de datos que sea capaz de manipular cambios de conceptos graduales, abruptos o recurrentes.

Se considera que: Con la propuesta de un algoritmo de multiclasificación, que parte del análisis de las características fundamentales de la familia de algoritmos MultiCIDIM, con aprendizaje incremental para minar flujos de datos, que sea capaz de manipular cambios de conceptos graduales, abruptos o recurrentes, se podrían obtener modelos más complejos y precisos para la tarea de clasificación. Siendo lo anterior la ***hipótesis*** de esta tesis doctoral.

Para alcanzar el objetivo propuesto el doctorando se propuso las *siguientes* ***tareas de investigación:***

Estudio teórico de conceptos y sistemas relacionados con los campos de investigación: descubrimiento de conocimiento en bases de datos, minería de datos y aprendizaje automático.

Análisis y estudio de los algoritmos de multiclasificación basados en CIDIM (MultiCIDIM-DS, MultiCIDIM-DS-CFC).

Revisión y estudio bibliográfico de trabajos que proponen métodos de clasificación y multiclasificación para aprendizaje incremental.

Análisis de trabajos que proponen métodos de clasificación y multiclasificación para el aprendizaje incremental que se adaptan a cambios de conceptos en los datos.

Selección y propuesta de nuevas ideas, partiendo de la familia de algoritmos MultiCIDIM, para obtener un nuevo algoritmo de multiclasificación capaz de manipular de forma eficiente los diferentes tipos de cambios de conceptos.

Diseño e Implementación de algoritmos basados en las ideas propuestas.

Identificación de las metodologías de evaluación de algoritmos adecuadas para el aprendizaje incremental con cambios de conceptos, así como de las métricas utilizadas por estas.

Análisis y discusión de los experimentos y resultados.

A continuación se describe la **estructura del trabajo,** incluyendo una breve descripción de los principales contenidos de cada capítulo:

El documento ha sido dividido en cinco capítulos, además de la introducción, conclusiones, tareas futuras, bibliografía y anexos.

En el **Capítulo I** se trata de contextualizar la investigación dentro de la Ciencia de la Computación. Se incluyen una breve descripción de los principales conceptos asociados a las áreas de investigación que sirven de sustento al presente trabajo. Entre estos conceptos se encuentran: Extracción de Conocimiento, Minería de Datos, Aprendizaje Automático, entre otros.

En el **Capítulo II**, se describen los principales conceptos vinculados con los flujos de datos y los cambios de conceptos. Luego se realiza un breve bosquejo de los principales modelos utilizados para tratar con cambios de conceptos, dejando un aparte para los sistemas multiclasificadores a los cuales se les dedica luego un espacio especial. Además, se describen algunos detectores de cambios significativos para la investigación. Luego, se realiza una revisión de las metodologías, herramientas de software y conjuntos de datos utilizados para la evaluación y comparación de algoritmos incrementales para la detección de cambios de conceptos.

El **Capítulo III** es un apartado especial para los sistemas multiclasificadores que tienen mecanismos de trabajo en línea y que manipulan cambios de conceptos. En él se incluye una revisión bibliográfica de los principales algoritmos de Multiclasificación, haciendo énfasis en sus características (forma de trabajar los datos de entrada, sistema de votación, tipos de cambios que tratan, uso de detectores de cambios, etc), ventajas y desventajas según opiniones propias y de la literatura revisada.

El **Capítulo IV** comienza con una breve descripción de los algoritmos de la familia MultiCIDIM, MultiCIDIM-DS y MultiCIDIM-DS-CFC, que sirven de punto de partida a los principales aportes de este trabajo. Además, se describen las características principales de un nuevo algoritmo, resultado de realizar varias modificaciones al algoritmo MultiCIDIM-DS con vista a adaptarlo al trabajo con cambios de conceptos. Por último, se incluye una breve experimentación para mostrar los avances de la nueva propuesta en el tratamiento de cambios de conceptos.

En el **Capítulo V** se presentan todos los detalles del algoritmo llamado Multiclasificador de Adaptación Rápida (FAE, del inglés Fast Adapting ensemble) que resulta el principal aporte de este trabajo. Además, se incluye un estudio experimental donde se compara la nueva propuesta con otros conocidos algoritmo implementados sobre el entorno de trabajo MOA.

El aprendizaje incremental para el trabajo sobre flujos de datos en presencia de cambios de conceptos es el tema principal de esta tesis, de aquí que las principales aportaciones de la misma se sustenten en los conocimientos científicos precedentes aportados desde esta dirección.

En este capítulo se incluyen descripciones de los principales conceptos necesarios para contextualizar el presente trabajo dentro de los siguientes campos de investigación de las ciencias de la computación: extracción de conocimiento, minería de datos y aprendizaje automático.

EXTRACCIÓN DE CONOCIMIENTO.

La extracción de conocimiento es el proceso por el cual se identifican, de forma no trivial, patrones válidos, novedosos, potencialmente útiles y comprensibles que se encuentren en los datos [16]. Esta definición es una de las más difundidas porque recoge las principales características que definen al proceso de extracción de conocimiento (del inglés, *Knowledge Discovery in Databases*, *KDD*).

El concepto de extracción de conocimiento suele confundirse con el de *minería de datos* dependiendo del ámbito en que sean usados, pero no se refieren a lo mismo puesto que éste último es, en realidad, parte del primero. Si se quiere hablar del concepto de extracción de conocimiento en términos de minería de datos se debe usar el término de *proceso de minería de datos*. Esta última terminología está más extendida en el ámbito industrial, mientras que el concepto de extracción de conocimiento se utiliza más habitualmente en los entornos académicos [17], pero ambos describen lo mismo: el proceso que hay que seguir para conseguir conocimiento útil, novedoso, válido y comprensible [13].

MINERÍA DE DATOS.

La minería de datos (del inglés, *Data Mining*) puede definirse como la extracción no trivial de información implícita, previamente desconocida y potencialmente útil, a partir de los datos [18]. La minería de datos es, en principio, la fase más característica del KDD y, por esta razón, muchas veces se utiliza esta fase para nombrar todo el proceso, y se distingue porque no obtiene información extensional (datos) sino intencional (conocimiento) y sus resultados son conjuntos de reglas, ecuaciones, árboles de decisión, redes neuronales, grafos probabilísticos u otros modelos. Esta fase tiene dos retos: por un lado, trabajar con grandes volúmenes de datos, procedentes mayoritariamente de sistemas de información, con los problemas que ello conlleva (ruido, datos ausentes, intratabilidad, volatilidad de los datos), y por el otro, usar técnicas adecuadas para analizar los mismos y extraer conocimiento novedoso y útil. En muchos casos la utilidad del conocimiento extraído está íntimamente relacionada con la comprensibilidad del modelo inferido.

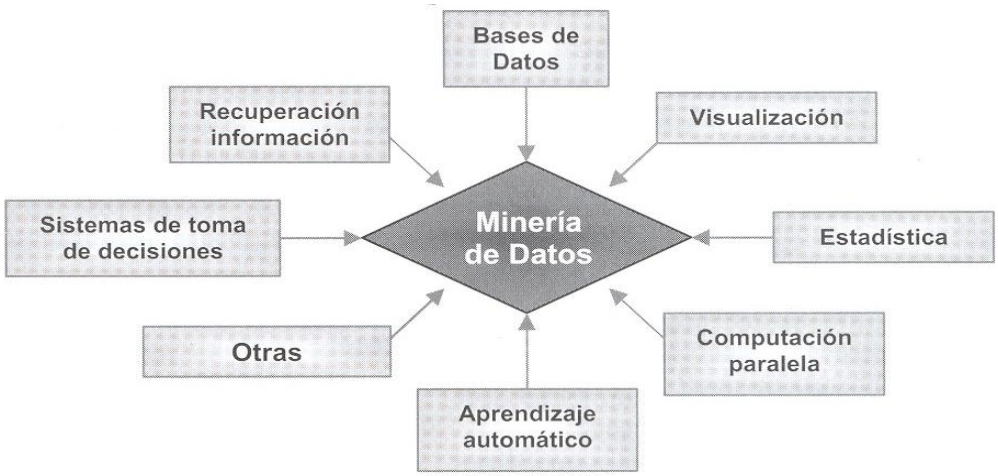


FIGURA 1.1 MINERÍA DE DATOS Y ÁREAS RELACIONADAS.

Dentro de la minería de datos se distinguen tipos de tareas, cada una de las cuales puede considerarse como un tipo de problema a ser resuelto por un algoritmo de minería de datos. Estas distintas tareas pueden ser categorizadas en predictivas o descriptivas. Entre las tareas predictivas se encuentran la clasificación y la regresión, mientras que el agrupamiento, las reglas de

asociación, las reglas de asociación secuenciales y las correlaciones son tareas descriptivas [19].

El objetivo de esta fase es producir nuevo conocimiento que pueda utilizar el usuario. Esto se realiza construyendo un modelo basado en los datos recopilados para este efecto. El modelo es una descripción de los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender mejor los datos o para explicar situaciones pasadas.

Las técnicas de Minería de Datos tienen múltiples aplicaciones en todas las esferas de la sociedad. La integración de las técnicas de minería de datos en las actividades del día a día se está convirtiendo en algo habitual. Los negocios de la distribución y la publicidad dirigida han sido tradicionalmente las áreas en las que más se han empleado los métodos de minería, ya que han permitido reducir costes o aumentar la receptividad de ofertas. Sin embargo, cada vez más, son aplicadas en otras áreas como: financieras, seguros, científicas (medicina, farmacia, astronomía, psicología, etc.), políticas económicas, sanitarias o demográficas, educación, policiales, procesos industriales entre muchas otras.

Un área muy estrechamente relacionada con la Minería de Datos es el aprendizaje y dentro de este, el aprendizaje automático que tiene mayor interés para esta investigación por lo que se describirán sus características fundamentales en las secciones siguientes.

APRENDIZAJE

La Informática tiene relación con innumerables esferas de la sociedad entre ellas las ciencias cognitivas, con la cual se relaciona de manera directa mediante el área de la Inteligencia Artificial. Esta área es muy extensa y comprende, entre otros objetos de estudio, al aprendizaje automático.

Para una mayor comprensión del concepto general de aprendizaje, solamente se tendrá en cuenta su significado dentro de la Inteligencia Artificial. A pesar de que muchas han sido las definiciones de aprendizaje en esta área, se puede resumir que el aprendizaje no es más que la realización de cambios en el sistema que se

adaptan, de manera que permiten llevar a cabo la misma tarea de un modo más eficiente y eficaz. En la práctica, el aprendizaje se usa para resolver problemas y puede representar la diferencia entre la resolución rápida y la imposibilidad de resolverlo. La idea de poder aprender de la propia experiencia en la resolución de problemas conlleva a esperar obtener mejores soluciones en un futuro. Además está relacionado con el conocimiento. Puede definirse como el proceso mediante el cual un ente adquiere conocimiento. Este conocimiento puede ser suministrado por otro ente denominado profesor o puede adquirirse sin la ayuda del mismo. El aprendizaje también puede ser visto como una actualización en el comportamiento, habilidades o conocimiento en general con la finalidad de mejorar el desempeño [13].

A continuación se dará una descripción de algunos conceptos muy relacionados con la adquisición de conocimiento y cuyo significado puede confundirse dependiendo del ámbito en que se empleen: aprendizaje automático y aprendizaje incremental además de los que se han explicado anteriormente: extracción de conocimiento y minería de datos.

APRENDIZAJE AUTOMÁTICO

El Aprendizaje Automático, Aprendizaje Computacional o Aprendizaje de Máquina (del inglés, Machine Learning) tiene como objetivo desarrollar sistemas (programas de computadora) de forma que las computadoras aprendan, o sea, que puedan mejorar su funcionamiento para realizar una tarea de forma automática, basándose en la experiencia [20].

Un concepto importante en aprendizaje es el de función objetivo, que puede entenderse como la definición del tipo de conocimiento que debe ser aprendido. Este conocimiento puede referirse por ejemplo, a la evaluación de una tarea representativa dentro de un dominio particular.

A partir de lo anterior, el Aprendizaje Automático puede entenderse como la búsqueda de la descripción de una función objetivo con el fin de utilizar algún algoritmo para computarla. Comúnmente la descripción de tal función es no

operacional, no puede computarse eficientemente. Como consecuencia, los sistemas de aprendizaje, buscan una descripción operacional de ella, lo que se denomina aproximación de funciones.

Al diseñar un sistema de aprendizaje se deben tomar en cuenta varios elementos: el tipo de experiencia de entrenamiento, la medida de desempeño, la función objetivo y su representación así como el algoritmo para aproximarla.

En función de la clase de experiencia utilizada, el Aprendizaje Automático puede dividirse en tres tipos principales [21]:

- ✓ Supervisado: Consiste en aprender una función a partir de ejemplos de entradas y salidas, es decir, las clases son definidas previamente y con base en ellas se clasifican los datos.
- ✓ No supervisado: Consiste en aprender patrones de entradas cuando no hay valores de salida especificados. Las clases se infieren de los datos, creando grupos diferenciados.
- ✓ Por refuerzo: El aprendizaje se basa en la evaluación de un refuerzo o recompensa para el conjunto de acciones realizadas.

APRENDIZAJE INDUCTIVO

De acuerdo con T. Mitchell [22], un concepto central en el aprendizaje es la inducción, que consiste en aprender funciones generales a partir de ejemplos particulares. De esta manera, el Aprendizaje de Conceptos busca una definición de una categoría general basándose en ejemplos positivos y negativos de ella.

El aprendizaje inductivo es aquel aprendizaje que parte de casos particulares (ejemplos) y obtiene casos generales (modelos o reglas) y es el tipo de aprendizaje más comúnmente usado en el ámbito del aprendizaje automático. Se contrapone al aprendizaje deductivo en el cuál el experto “deduce” que una regla (modelo o hipótesis) puede servir para describir el conocimiento y lo comprueba consultando en el conjunto de datos [23].

De esta forma, un algoritmo puede “aprender” a realizar una tarea, por ejemplo clasificar datos, si obtiene una buena aproximación de la función objetivo tomando en cuenta sólo los ejemplos de entrenamiento.

Como se ha dicho anteriormente, las tareas a abordar desde el aprendizaje automático se pueden dividir en dos grandes grupos: descriptivas y predictivas. Entre las primeras están las tareas de detección de agrupamientos y correlaciones, y entre las segundas están las tareas de clasificación y regresión. Existen distintas técnicas para abordar las tareas citadas y entre estas podemos mencionar: las que generan reglas de asociación [24], estas son más descriptivas; las que entrenan redes neuronales [25], estas son más predictivas; las que inducen árboles de decisión [26], en estas la separación no siempre es tan clara pues estos modelos pueden ser fácilmente comprensibles y también tiene capacidad para ser utilizado como un sistema predictor. Existen multitud de técnicas empleadas para inferir gran diversidad de modelos y éstas pueden usarse de forma conjunta o aislada.

APRENDIZAJE INCREMENTAL

El aprendizaje incremental se caracteriza principalmente por dos aspectos: es un aprendizaje capaz de incorporar la información que aporten nuevas experiencias (que antes no estaban disponibles en el conjunto de datos) al modelo que se está induciendo [7] y capaz de hacerlo evolucionar para que cada vez represente conceptos más complejos [27].

El aprendizaje incremental es necesario cuando se requiere que un sistema lleve a cabo tareas que necesitan aprender de forma serial o dinámica (cuando los ejemplos se presentan de manera secuencial, como un flujo, o están distribuidos en varios depósitos y no como un conjunto de tamaño fijo), cuando se necesita revisar la hipótesis aprendida en presencia de nuevos ejemplos, en lugar de rehacerla cada vez que se tengan datos nuevos [28].

Algunos ejemplos de tareas que necesitan de aprendizaje incremental son: bioinformática, minería de datos, Sistemas multi-Agente e Interfaces Inteligentes.

En las dos primeras la cantidad de datos es tan grande que se hace necesario almacenarla en bases de datos distribuidas en varios sitios y que difícilmente podrían reunirse en un solo punto. En tanto que una interfaz inteligente recibe la información como un flujo discontinuo (a medida que el usuario ocupa la interfaz) y generalmente es escasa. Para tratar el problema de aprendizaje a partir de un conjunto creciente de evidencias, se han desarrollado algoritmos incrementales, los cuales revisan el concepto aprendido al recibir nuevos ejemplos de forma eficiente.

CLASIFICACIÓN Y PREDICCIÓN.

La clasificación es un proceso que consta de dos pasos fundamentales. En el primer paso es donde se construye un modelo que describe un conjunto predeterminado de datos, clases o conceptos. Este modelo se construye tupla a tupla, describiendo en cada caso los atributos de la base de datos que se analiza. Se asume que cada tupla pertenece a una clase predefinida, que está determinada por uno de los atributos llamado atributo clase. Las tuplas dentro de la clasificación son también conocidas como ejemplos, instancias o experiencias (de aquí en adelante ejemplos). El conjunto de ejemplos analizados para construir el modelo forman el conjunto de datos de entrenamiento. Los ejemplos individuales que forman el conjunto de entrenamiento son seleccionados aleatoriamente de la población muestral asignándole a cada uno, una etiqueta de clase, teniendo lugar el aprendizaje supervisado (se indica a que clase pertenece cada experiencia) ya mencionado anteriormente. Contrario al aprendizaje supervisado, el aprendizaje no supervisado, es aquel en el que no se conocen las etiquetas de clase de las experiencias, y el número o conjunto de clases no se tienen a priori. Generalmente el modelo obtenido está representado en la forma de reglas de clasificación, árboles de decisión o fórmulas matemáticas. En el segundo paso, el modelo obtenido en el paso anterior es usado para predicción y se puede estimar la precisión de la predicción del modelo (o clasificador) [17].

Según Ye [29], a manera de resumen podemos decir que la clasificación tiene como objetivo la construcción de modelos concisos que representen la distribución

del atributo dependiente (clase) en función de los atributos predictores (atributos). El modelo resultante se usará, principalmente, para determinar la clase a la que pertenecen las observaciones de las que se conocen todos los valores de sus atributos con excepción del valor de clase, es decir, su tarea será preferentemente predictiva. Dependiendo del modelo generado, éste además puede presentar características descriptivas, lo que lo hará aún más deseable desde el punto de vista de la extracción de conocimiento.

DIFERENCIAS ENTRE LA PREDICCIÓN Y LA CLASIFICACIÓN

La predicción puede ser vista como la construcción y uso de un modelo para evaluar la clase de un objeto sin etiqueta, el valor o estimar los rangos que puede tener un objeto o atributo dado. La clasificación y la regresión son los dos tipos principales de problemas de predicción donde la clasificación se usa para predecir valores discretos o nominales, mientras la regresión se usa para predecir valores continuos u ordenados. Sin embargo en este caso se puede referir al uso de predicción para predecir etiquetas de clase, como clasificación, y el uso de predicción para predecir valores continuos (usando técnicas de regresión) como predicción, este punto de vista es el más aceptado dentro de la Minería de Datos [17]. La predicción tiene aplicaciones numerosas dentro de las cuales podemos encontrar la aprobación de crédito, la predicción de diagnóstico médico, predicción de desempeño, y mercadeo selectivo.

CONCLUSIONES DEL CAPÍTULO I.

En el presente capítulo se incluye una breve descripción de los principales conceptos asociados a las áreas de investigación que sirven de sustento al presente trabajo. Entre estos conceptos se encuentran: Extracción de Conocimiento, Minería de Datos, Aprendizaje Automático, entre otros. Además, se diferenciaron distintos tipos de aprendizajes, según conceptos dados en la literatura científica desde el punto de vista de la ciencia de la computación; entre estos se especificaron: Aprendizaje automático, aprendizaje inductivo y aprendizaje incremental. Por último, se establecieron diferencias entre los conceptos de predicción y clasificación que muchas veces suelen ser confundidos.

Podemos concluir que la presente investigación está enmarcada en el área de aprendizaje automático, en particular profundiza en la tarea de clasificación dentro de aprendizaje supervisado.

Como hemos mencionado anteriormente, los procesos incrementales son aplicados en muchas áreas, no solo exclusivas a tareas de aprendizaje y minería de datos, lo que lo convierte en una poderosa herramienta en muchos problemas actuales que son presentados en formato de flujos de datos [30]. Las grandes secuencias de datos, potencialmente infinitas, que se van adquiriendo a lo largo del tiempo son conocidas como **flujos de datos** (del inglés, *datastream*) [5].

Dentro del aprendizaje incremental [31], el problema de clasificación es generalmente definido para una secuencia, posiblemente infinita de ejemplos (también conocidos por instancias o experiencias) $S = e_1, e_2, \dots, e_i, \dots$ que se obtienen en el tiempo, normalmente uno a la vez. Cada ejemplo de entrenamiento $e_i = (\vec{x_i}, y_i)$ está formado por un vector $\vec{x_i}$ y un valor discreto y_i, llamado etiqueta clase y tomado de un conjunto finito *Y* llamado Clase. Cada vector $\vec{x_i} \in \vec{X}$ tiene la misma dimensión, cada dimensión es llamada atributo y cada componente $x_{i,j}$ es un valor de atributo (numérico o nominal). Es supuesto que existe una función $y = f(\vec{x_i})$ y el objetivo es obtener un modelo a partir de S que aproxime a f como $\hat{f}$ para clasificar o predecir la etiqueta clase de ejemplos no etiquetados (también conocidos como observaciones), tal que $\hat{f}$ maximice la precisión en la predicción [32]. Algunas veces también se asume que los ejemplos se obtienen en lotes de igual tamaño. Consideraremos concepto como el término que se refiere a la función de distribución de probabilidades completas del problema en un punto determinado en el tiempo [33]. Este concepto puede ser caracterizado por la distribución conjunta $P(\vec{X}, Y)$. De esta forma, un cambio en la función de distribución de probabilidades del problema (también conocido como contexto [34]) trae consigo un cambio de concepto.

Uno de los problemas fundamentales del aprendizaje incremental se debe a que la función objetivo puede depender del contexto, el cual no es recogido mediante los atributos de los datos de entrada. En tales situaciones, la causa del cambio se dice oculta y el problema se conoce como contexto oculto (del inglés, *hidden context*). Como consecuencia, cambios ocurridos en el contexto pueden inducir

variaciones en la función objetivo, dando lugar a lo que se conoce como cambio de concepto (del inglés, *concept drift*) [35,36].

La dependencia del contexto no sólo puede inducir variaciones en la función objetivo sino que además puede cambiar la propia distribución de los atributos de los ejemplos de entrenamiento. Cuando el cambio contextual afecta únicamente a los datos de entrada éste se dice virtual (del inglés, *sampling shift*) [37] mientras que el cambio es real cuando únicamente induce un movimiento del concepto objetivo (del inglés, *concept shift*) [36]. En la práctica es irrelevante el tipo del cambio ya que ambos producen un impacto en el modelo, en cuanto aumentan el error del mismo con respecto a los ejemplos actuales y lleva a la necesidad de revisarlo constantemente.

Un problema de gran dificultad está relacionado con la velocidad del cambio. En la literatura se distinguen dos tipos, relacionado con la frecuencia con la que se reciben los ejemplos que describen a la nueva función objetivo: abrupto (repentino, instantáneo) y gradual. Algunos autores, como Stanley [38], dividen el cambio gradual en moderado y lento, dependiendo de la velocidad del mismo.

De forma general, se pretende que un sistema que maneje cambios de conceptos sea capaz de adaptarse rápidamente al cambios, ser robusto en la distinción entre un verdadero cambio de concepto y el ruido, así como reconocer y tratar contextos recurrentes. La distinción entre un verdadero cambio de concepto y el ruido es un problema de gran dificultad para la manipulación del cambio de concepto. Algunos algoritmos pueden ser muy susceptibles al ruido, interpretándolo erróneamente como un cambio de concepto, mientras que otros pueden ser robustos al ruido pero se ajustan al cambio muy lentamente [31]. Adicionalmente, en algunos dominios el contexto puede ser recurrente. Para adaptarse rápidamente al cambio de concepto, las descripciones de los conceptos pueden ser almacenadas para luego reexaminarlas y usarlas posteriormente. Algunos sistemas que usan esta estrategia para manipular cambios de conceptos son: FLORA3 [37], PECS [39], SPLICE [40] y Local Weights and Batch Selection [41].

CARACTERÍSTICAS DE ALGUNOS MODELOS PARA MANIPULAR CAMBIOS DE CONCEPTOS

En esta sección se analizarán ejemplos de cómo algunos modelos de aprendizaje son capaces de manipular cambios de conceptos teniendo en cuenta sus características específicas.

ÁRBOLES DE DECISIÓN.

Los algoritmos para inducir árboles de decisión fueron inicialmente propuestos por Hunt, Marin y Stone [42] y, posteriormente, empezaron a tomar relevancia a partir de los trabajos de Quinlan [27] (desde la perspectiva del aprendizaje inductivo) y del trabajo de Breiman, Friedman, Olshen y Stone [26] (desde una perspectiva estadística).

Una familia de algoritmo muy popular para manipular cambios de conceptos basado en inducción de árboles de decisión combinado con la desigualdad de Hoeffding está basada en el algoritmo principal VFDT (del inglés, Very Fast Decision Tree) [43]. Sin embargo, Rutkowski y otros [44] demostraron recientemente que la desigualdad de Hoeffding no es una herramienta adecuada para resolver problemas de calcular un intervalo de confianza para cualquier medida heurística (ganancia de información, etc.), por lo que los métodos y algoritmos desarrollados en la literatura que usan la cota de Hoeffding en este sentido deben ser revisados. Un uso correcto de las cotas de Hoeffding y Chernoff es presentado en la familia de algoritmos IADEM (Inducción de Árboles de Decisión por Muestreo) [13, 45].

Una estrategia habitual para manipular cambios de conceptos sobre la inducción incremental de árboles de decisión es llevada a cabo en dos etapas:

(1) estimar en cada nodo de decisión si el subárbol con raíz en ese nodo es consistente con los datos actuales

(2) si se estima una inconsistencia en cualquier nodo de decisión se ejecutan algunas acciones para reconstruir el subárbol correspondiente; con esta estrategia se tienen en cuenta cambios locales.

Por ejemplo, en el algoritmo CVFDT (del inglés, *Concept-adapting Very Fast Decision Tree*) [46] los subárboles alternativos cambian entre los modos entrenamiento y prueba por medio de parámetros ajustados por el usuario. En la fase de prueba, CVFDT elimina subárboles alternativos cuya precisión no aumenta en el tiempo.

Otro método diferente, dentro de la misma estrategia, es propuesto por Natwichai y Li [47]. Ellos usan una tabla de decisión ambigua como intermediaria para la representación del conocimiento. Cada fila en dicha tabla es una regla de decisión obtenida desde CVFDT cuando este genera un subárbol alternativo en respuesta a un posible cambio de concepto.

Son muchas las estrategias y ejemplos de algoritmos que usan como modelo base para manipular cambios de conceptos los árboles de decisión. Adicionalmente, algunas técnicas de hibridación recientes que envuelven árboles de decisión han probado ser una herramienta poderosa en la tarea del aprendizaje incremental [48, 49]. La extensión de estas técnicas al aprendizaje en flujos de datos no estacionarios es un área de investigación prometedora.

REGLAS DE DECISIÓN

En comparación con los árboles, las reglas de decisión pueden tener una adaptación más rápida al cambio de concepto ya que se pueden eliminar reglas individuales sin la necesidad de tener que reconstruir otras partes del modelo. Sin embargo, las reglas de decisión han recibido poca atención en la minería de flujo de datos [50, 51].

La mayoría de los sistemas basados en reglas que manipulan cambios de conceptos adaptan el aprendizaje continuamente sin considerar que ha ocurrido un cambio de concepto. En estos casos la adaptación es probablemente menos efectiva que cuando se usa la detección de cambio explícita. Entre los algoritmos basados en reglas de decisión más populares que manipulan cambios de conceptos encontramos a STAGGER [52], la familia FLORA [31] y la familia de algoritmos AQ-PM [53].

Por ejemplo, STAGGER [52] induce fórmulas booleanas (caracterizaciones) que se mueven desde caracterizaciones más generales (disyunción de pares atributo-valor) a más particulares (conjunciones de pares atributo-valor). Sobre estas caracterizaciones son definidos tres operadores de búsqueda (para crear una fórmula más general, una más particular, o negar una fórmula existente), los cuales son ejecutados cuando el modelo comete un error en la clasificación.

Mientras que el sistema anterior solo manipula atributos no numéricos, FACIL (del inglés, *Fast and Adaptive Classifier by Incremental Learning*) [32] aprende reglas de decisión incrementalmente a partir de ejemplos en un flujo de datos que pueden tener atributos numéricos. Existen muchas otras ideas interesantes para manipular cambios de conceptos con reglas de decisión.

MÁQUINAS DE SOPORTE VECTORIAL

Las máquinas de soporte vectorial son un método robusto y popular en muchas áreas del aprendizaje. Por ejemplo, ellas tienen un rendimiento muy alto para la clasificación de textos y también han sido favorecidas para la detección de correos basuras basada en el contenido [54]. Sin embargo, el entrenamiento de las máquinas de soporte vectorial es con frecuencia computacionalmente costoso, y hasta ahora presenta algunos retos que no han sido resueltos en el aprendizaje en línea [54].

Algunos ejemplos de algoritmos que utilizan máquinas de soporte vectorial para manipular cambios de conceptos son:

Syed [55] propone un algoritmo que entrena una máquina de soporte vectorial tomando los vectores de soporte calculados a partir de un lote de instancias. Cuando un nuevo lote de instancias es obtenido, los vectores de soporte son actualizados usando los anteriormente calculados y los nuevos a partir de este nuevo lote.

Un método que manipula cambios de conceptos puramente con máquinas de soporte vectorial es descrito en el trabajo de Dries y Rückert [56]. Su idea es

evaluar una función de decisión fija *f* en dos secuencias (lotes) de datos y usar alguna prueba estadística en las secuencias resultantes de la evaluación, calculando un valor de probabilidad bajo la hipótesis nula que estas secuencias corresponden a la misma distribución.

REDES NEURONALES ARTIfiCIALES

Las redes neuronales artificiales son un poderoso modelo computacional que puede representar cualquier relación no lineal entre los datos de entrada y el espacio objetivo. Las redes neuronales se han usado para resolver muchos problemas del mundo real incluyendo áreas del aprendizaje supervisado. Sin embargo, la mayoría de los modelos de redes neuronales requieren varias pasadas sobre los datos de entrenamiento, y su extensión a escenarios del aprendizaje incremental no es una tarea obvia [57]. Por ejemplo, algunos de los acercamientos más conocidos para manipular cambios de conceptos básicamente usan un algoritmo de aprendizaje por lotes basados en una ventana deslizante [10], en multiclasificadores [58] o en búsquedas heurísticas [59] con un alto coste computacional.

Otro ejemplo de un sistema que utiliza redes neuronales para manipular cambios de conceptos es FRANN (del inglés, *Floating Rough Approximation in Neural Network*) [10], uno de los sistemas más conocidos relacionado con esta tarea. Dados p ejemplos con n atributos, FRANN realiza un mapeo no lineal del espacio de las instancias R^n a R^m de m neuronas ocultas de funciones de base radial. Usando una filosofía de ventana, el sistema toma cada uno de los p ejemplos en la ventana y trata de crear una neurona oculta quitando al ejemplo seleccionado. Se selecciona la neurona que provea mayor precisión sobre la ventana de ejemplos.

MULTICLASIFICADORES

Los métodos multiclasificadores (en inglés es usado, *ensemble*) han recibido en los últimos tiempos una gran atención para el modelado y la clasificación de flujos de datos. En general, y aunque dentro de un dominio incremental, las nuevas propuestas siguen el mismo esquema que las técnicas de multiclasificación

aplicadas en el aprendizaje por lotes. Estas se basan en heurísticas y medidas de interés para eliminar, reactivar o añadir dinámicamente nuevos algoritmos de aprendizaje en respuesta a las variaciones ocurridas en la consistencia del modelo con respecto a los nuevos ejemplos recibidos. Existen una gran variedad de acercamientos, ya que la idea de combinar muchos clasificadores [97] provee de una buena arquitectura para manipular los problemas que surgen cuando se aprende con cambios de conceptos.

Debido a que los algoritmos de multiclasificación son un tema central en el presente trabajo, más adelante se les dedica un apartado especial.

SISTEMAS INDEPENDIENTES PARA DETECTAR CAMBIOS DE CONCEPTOS.

Una estrategia ampliamente usada para la manipulación de cambios de conceptos monitoriza constantemente alguna medida de rendimiento del modelo de aprendizaje en el tiempo. Si esta medida se deteriora significativamente, se asume un cambio de concepto y algunas acciones son definidas para actualizar el modelo de aprendizaje acorde a los ejemplos más recientes. En esta estrategia, los detectores de cambios de conceptos desempeñan un papel fundamental, ellos se incorporan en el algoritmo de aprendizaje, ya sea para monitorizar la consistencia de partes del modelo o la del modelo completo.

En general, los detectores de cambios operan sobre un flujo de datos de valores reales correspondientes a alguna medida de rendimiento del modelo de aprendizaje en el tiempo. De esta forma, el problema de detectar cambios de conceptos se reduce a estimar cambios distribucionales en un flujo de valores reales.

En esta sección se presentarán tres detectores de cambios que por sus características particulares han sido utilizados en el presente trabajo en combinación con la principal propuesta de algoritmo del mismo. Estos tres detectores tienen en común que ofrecen tres posibles salidas:

ESTABLE, cuando parece no haber cambio.

ALERTA, cuando se estima que un posible cambio de concepto puede aparecer.

CAMBIO, cuando el cambio se identifica claramente.

DETECTOR DE CAMBIO DDM.

El detector de cambio DDM (del inglés, *Drift Detection Method*), propuesto por Gama y otros [34] en 2004, usa una distribución binomial para su funcionamiento. Esta distribución ofrece una forma general de la probabilidad para una variable aleatoria que representa el número de errores al experimentar sobre un conjunto de ejemplo. Para cada punto *i*, de la secuencia de ejemplos que está siendo examinada, la tasa de error es la probabilidad de ejemplos mal clasificados (p_i) con desviación estándar dada por $s_i = \sqrt{p_i(1-p_i)/\,i}$. Se asume que la tasa de error del algoritmo de aprendizaje (p_i) disminuirá si el número de ejemplos aumenta y la distribución de los ejemplos se mantiene estacionaria. Por otra parte, un aumento significativo en el número de errores del algoritmo, sugiere que la distribución de clases está cambiando y por lo tanto, el modelo de decisión actual es inapropiado. Los valores de p_i y s_i son almacenados cuando $p_i + s_i$ alcanza su valor mínimo durante el proceso, obteniéndose p_{min} y s_{min}. Con estos valores almacenados los autores definen tres posible estados para el sistema:

En-control (del inglés, *in-control*) $p_i + s_i < p_{min} + \beta * s_{min}$: El sistema es estable, se asume que los ejemplos están siendo generados bajo la misma distribución.

Fuera-de-control (del inglés, *out-of-control*) $p_i + s_i > p_{min} + \alpha * s_{min}$: El error del sistema está aumentando y ha alcanzado un nivel significativamente alto. Con probabilidad $1 - \alpha/2$ los ejemplos actuales están siendo generados bajo una distribución diferente.

Alerta (del inglés, *warning*): Cuando el error del sistema está entre los márgenes anteriores. El error está aumentando (alcanzó el primer margen) pero no ha alcanzado el nivel considerado significativamente alto para concluir un cambio (segundo margen).

En los experimentos, los autores definieron como nivel de confianza para *alerta (warning) el 95%* $p_i + s_i \geq p_{min} + 2 * s_{min}$ *($\beta = 2$); y para cambio (out-of-control) el 99%* $p_i + s_i \geq p_{min} + 3 * s_{min}$ *($\alpha = 3$).*

Esta propuesta tiene buenos resultados al detectar cambios abruptos y graduales cuando el cambio no es muy lento, pero tiene algunas dificultades para cambios graduales muy lentos [60].

DETECTOR DE CAMBIO EDDM.

El detector de cambio EDDM (del inglés, *Early Drift Detection Method*), propuesto por Baena y otros [60] en 2006, fue desarrollado para mejorar la detección de cambios graduales manteniendo los mismos resultados que DDM en la detección de cambios abruptos. La idea básica es considerar la distancia entre dos errores de clasificación en lugar de considerar sólo el número de errores. A medida que el método de aprendizaje es entrenado, éste mejorará las predicciones y la distancia entre dos errores aumentará. Podemos calcular la media de las distancias entre dos errores ($p_i^{'}$) y su desviación estándar ($s_i^{'}$) cuando $p_i^{'} + 2s_i^{'}$ alcanza sus valores máximos, obteniéndose $p_{max}^{'}$ y $s_{max}^{'}$. Al igual que el método anterior, usando estos valores almacenados ($p_{max}^{'}$ y $s_{max}^{'}$) se definen dos umbrales:

Nivel de *alerta (del inglés, warning)* $(p_i^{'} + 2s_i^{'})/(p_{max}^{'} + 2s_{max}^{'}) < \alpha$: Más allá de este nivel, los ejemplos se almacenan antes de un posible cambio de contexto.

Nivel de cambio (del inglés, *concept drift*) $(p_i^{'} + 2s_i^{'})/(p_{max}^{'} + 2s_{max}^{'}) < \beta$: Más allá de este nivel el cambio de concepto se supone que es cierto. En este punto, los valores $p_{max}^{'}$ y $s_{max}^{'}$ son reiniciados.

Para la sección experimental, los valores utilizados para α y β fueron 95 y 90.

DETECTOR DE CAMBIO HDDM.

El detector de cambio HDDM (del inglés, *Hoeffding Drift Detection Method*), propuesto por Frías y otros [61] en 2014, tiene un funcionamiento similar al detector de cambios DDM pero utiliza un acercamiento basado en desigualdades de Hoeffding para realizar la prueba estadística.

El algoritmo recibe como parámetros de entrada un flujo de valores potencialmente infinito acotados en un intervalo [a, b] y dos niveles de significación $\propto_W \in (0,1]$ y $\propto_D \in (0,1]$ para determinar los estados de "alerta" (posible cambio) y "cambio" (cambio seguro) respectivamente. Como salida, el algoritmo ofrece tres posibles estados separados: ESTABLE, cuando parece no haber cambio; ALERTA, cuando se estima que un posible cambio de concepto puede aparecer; y CAMBIO cuando el cambio se identifica claramente.

Básicamente el método trabaja calculando tres estadísticos: $\hat{X}_{corte}$, $\hat{Y}_{n-corte}$ y $\hat{Z}_n$.

Entonces: Si H_0: $E[\hat{X}_{corte}] \geq E[\hat{Y}_{n-corte}]$ es rechazada con nivel de significación $\propto_D$,
ESTADO ← CAMBIO

De lo contrario, Si H_0 : $E[\hat{X}_{corte}] \geq E[\hat{Y}_{n-corte}]$ es rechazada con nivel de significación $\propto_W$,

ESTADO ← ALERTA

De lo contrario, ESTADO ← ESTABLE

MÉTODOS PARA EVALUAR LOS ALGORITMOS INCREMENTALES.

En esta sección se presentan algunas de las metodologías de evaluación usadas hasta el momento en el aprendizaje incremental con cambio de concepto, se analiza la evolución de los primeros acercamientos y se proponen algunos detalles que pueden ser beneficiosos.

La evaluación de los métodos y algoritmos con el objetivo de validar su rendimiento es un aspecto muy importante en el campo de la minería de datos. Este aspecto toma particular importancia cuando el proceso de aprendizaje se realiza en presencia de cambios de conceptos. Debido a las propiedades particulares de este contexto y la relativa novedad de esta área, las estrategias tradicionales no son válidas [62] y son necesarias varias modificaciones y mejoras. Aunque algunos métodos incluyen garantías matemáticas y cotas cuando son presentados [63], estas usualmente se refieren al rendimiento en el peor de los casos [64] o incluso calculados en relación con el rendimiento actual del clasificador básico [65]. En adición, los supuestos usados para calcular esas cotas son en muchos casos muy restrictivos o lejanos de lo real, por lo que es difícil determinar si están presentes en el problema en estudio (o conjunto de datos), el que puede incluir ruido y sesgo en el muestreo. Debido a estos inconvenientes, es difícil encontrar conclusiones significativas relacionadas con el rendimiento exclusivamente en términos de cotas. Las técnicas de evaluación ofrecen una aproximación diferente con un mayor margen de aplicación a casos variados, independientemente de la naturaleza particular del problema.

MÉTRICAS

Por lo general, para analizar el rendimiento de los algoritmos que llevan a cabo tareas de clasificación en presencia de cambios de conceptos las principales dimensiones a evaluar son: el poder de generalización del modelo inducido y los requerimientos básicos de cómputo del algoritmo: tiempo de ejecución y memoria utilizada [62].

El poder de generalización, usualmente se mide por medio de la precisión (del inglés, *accuracy*) o por el error producidos en un escenario de predicción. No obstante, se debería notar que en los últimos tiempos la precisión y la rememoración (del inglés, *recall*) están tomando mayor uso [9]. También, algunas medidas basadas en las anteriores, como el área bajo la curva de precisión-rememoración (AUC-PR) [56], el área bajo la curva de la característica operativa receptora (AUC) [66], o la medida F-score [67], han sido utilizadas para evaluar el rendimiento.

Por otro lado, debido a la gran cantidad de datos con los que se trabaja y la tasa de llegada de estos, los requerimientos computacionales del algoritmo, medidos en tiempo y espacio, suelen tener gran importancia.

La relevancia de estos aspectos se considera en los primeros estudios, donde los requerimientos de memoria son medidos usando el número de elementos necesitados para definir el concepto. Obviamente, la intensidad del proceso de evaluación se ha incrementado y actualmente se analizan más detalles. Desde el punto de vista de los requerimientos de memoria, es usual calcular la memoria como el número de componentes que forman el modelo y que describen el concepto aprendido (número de reglas en el modelo [32], clasificador básico en un multiclasificador [68], etc.) aunque esta puede no ser la mejor aproximación ya que diferentes componentes pueden usar distintas cantidades de memoria.

En cuanto al tiempo de ejecución, se pueden controlar diferentes niveles de especificación. De esta manera, se pueden considerar reportes donde solo se obtiene el tiempo global (periodo de entrenamiento [35], mezclando los periodos de entrenamiento y prueba [68]) o descripciones detalladas, donde se puede observar el incremento del tiempo acumulado cuando nuevos ejemplos son procesadas por el modelo en cuestión [69].

También existen otras medidas adicionales que suelen ser consideradas con el propósito de complementar las métricas anteriores. Por ejemplo, se pueden considerar métricas que prueban la resistencia al ruido [35], la complejidad del modelo inducido [32], la robustez en presencia de atributos irrelevantes [10], la sensibilidad al orden en que arriban los ejemplos de entrenamiento [7] o con más frecuencia la combinación de algunos de estos [31, 33, 10].

Por otro lado, a veces es importante priorizar las características relacionadas con la detección del cambio. Sobre todo, se puede estudiar el rendimiento de los métodos bajo diferentes suposiciones para los cuales los algoritmos son diseñados en particular. Por ejemplo, para este trabajo se han tomado en cuenta algunas de las siguientes características:

1- Respuesta de los algoritmos atendiendo a diferentes tipos de cambios de conceptos (abrupto o gradual) con diferentes niveles de la extensión del cambio [37, 35].
2- El número de ejemplos necesarios para estabilizar el proceso de aprendizaje, o para recuperarse de una fase donde el rendimiento ha empeorado [7, 55];
3- Ventajas de adaptar el modelo para manipular cambios de conceptos recurrentes [70];
4- Controlar el tiempo necesario para detectar cambios, complementado con las medidas de falsos positivos y falsos negativos en dicha detección, con el objetivo de evaluar la sensibilidad y especificidad de los detectores de cambio [60, 62, 71].

METODOLOGÍA PARA TRATAR CAMBIOS DE CONCEPTOS.

En esta sección se presentarán algunos de los procedimientos más generalizados en la literatura, a través de los cuales, son calculadas las métricas que se mencionaron anteriormente.

Una de las formas de calcular las métricas es considerar solo un punto de medición al finalizar del proceso de aprendizaje. Este procedimiento es más usual cuando se trabaja con de datos reales, debido a que la información que estos proporcionan es más limitada [72]; muchas veces no hay forma de saber si existen

cambios, ni cuál es su tipo. Un posible problema de este proceder, es que no muestra el desarrollo intermedio del proceso de aprendizaje en el tiempo, lo cual suele ser muy importante por las informaciones que aporta. De esta forma, es mucho más informativo presentar curvas de aprendizaje expresando el rendimiento de diferentes métricas en línea [10] a medida que los ejemplos son procesados. Un punto interesante es cómo las métricas individuales que forman la curva de aprendizaje son calculadas con una perspectiva secuencial, esto es, qué configuración usar a cada paso de aprendizaje para calcular los valores deseados (precisión, memoria, tiempo, etc.).

En este sentido existen dos propuestas fundamentales [62, 72], una que separa ejemplos de entrenamiento y de prueba (holdout) y otro que utiliza el mismo conjunto para ambas tareas (test-then-train), primero hace pruebas y luego entrena.

El primer acercamiento, *holdout*, fija en un paso precedente al menos un conjunto de prueba que contiene los ejemplos que únicamente serán usados para evaluar el funcionamiento del algoritmo de aprendizaje. Es posible usar solo un conjunto de prueba [55], aunque esta configuración no suele ser la más apropiada porque utilizando un solo conjunto de prueba no se puede saber cuál es el concepto evaluado. Es más provechoso utilizar varios conjuntos de prueba que serán usados en diferentes períodos [31, 73]. Si los conceptos y los períodos de cambios no son conocidos (como es frecuente en conjunto de datos reales), la selección del conjunto de prueba no puede ser guiada, pero las especificaciones del proceso serán mayores que cuando sólo se usa un conjunto de entrenamiento. Por otro lado, si el concepto y los períodos de cambio son conocidos (lo que es posible en los conjuntos de datos artificiales que son definidos con propósitos específicos), la creación de conjuntos de prueba es inmediata, y también es conocido cuáles conjuntos de prueba en cada período corresponden al concepto actual.

El segundo acercamiento, *test-then-train*, también conocido como *prequential* (formado de las palabras inglesas, predictive sequential), se deriva a partir del error predictivo secuencial. Este consiste básicamente en calcular las medidas de interés (usualmente la precisión) a la llegada de cada ejemplo (prueba);

seguidamente, el ejemplo es utilizado por el algoritmo de aprendizaje para continuar con su entrenamiento (entrenamiento) [46]. Esta metodología está basada en la suma acumulada de los valores de una función dada. De esta forma el valor del error predictivo secuencial tiende a ser dudoso a medida que más ejemplos son vistos, debido a que no hay mecanismo de olvido y el rendimiento actual es diluido en las medidas pasadas de rendimiento. La primera variante propuesta para mejorar esta deficiencia fue el uso de ventanas deslizantes. Así, el error predictivo secuencial se calcula considerando solamente los últimos ejemplos [74], olvidando argumentos antiguos y ofreciendo una estimación que refleja mejor el estado actual. Una desventaja del uso de las ventanas deslizantes es el cómo determinar el tamaño de ventana, entonces surge una segunda propuesta para mantener el mecanismo de olvido: factores de desvanecimiento [74] o uso del estadístico EWMA. Con este método, una función de decremento de pesos se aplica y la importancia (o peso) de los valores vistos anteriormente decrece, dando más importancia a los valores más actuales. El uso de este método es más eficiente debido a que no requiere almacenar ningún ejemplo.

Otro aspecto importante que diferencia las metodologías de evaluación en este contexto con respecto a las metodologías tradicionales es el soporte estadístico que puede ser garantizado. En el aprendizaje tradicional, por lotes, donde los conjuntos de datos son finitos y se asume que los ejemplos son independientes, muchas técnicas (validación cruzada, bootstrapping, etc.) pueden ser utilizadas para lograr algún soporte estadístico de los resultados. Pero en el contexto que se trabaja, donde los conjuntos de datos pueden tender a ser infinitos y existen cambios de conceptos, la situación puede cambiar y los ejemplos pueden presentar correlaciones con respecto al tiempo, por lo que no es factible el uso de estas técnicas mencionadas anteriormente para lograr el mismo resultado [71]. Algunos experimentos han sido diseñados para utilizar acercamientos inspirados por repetición [10, 31] y validación cruzada [55] pero, configurados en su forma original, no es posible realizar alguna prueba estadística que pueda asegurar las diferencias significativas a las que se aspiran. De esta forma, nuevas estrategias de evaluación tendrán que ser desarrolladas.

CONJUNTOS DE DATOS

Por lo general, los conjuntos de datos se dividen en dos grandes grupos, aquellos que son generados de forma artificial (ya sea basado en problemas reales o no) y aquellos datos recogidos estrictamente del mundo real. Cada uno de ellos tiene sus propias ventajas y desventajas. Los conjuntos de datos artificiales posibilitan probar los algoritmos bajo situaciones controladas, simulando todos los aspectos que los algoritmos deben superar en el futuro (requerimientos de tiempo y espacio, resistencia al ruido, etc.) y relacionado con posibles cambios de conceptos (el tiempo necesario para detectar cambios, su conducta al enfrentar diferentes tipos de cambios, etc.). Los conjuntos de datos reales posibilitan extender el proceso de aprendizaje a situaciones reales para las cuales los algoritmos son diseñados.

Aunque la mayoría de los conjuntos de datos para controlar el rendimiento de los algoritmos de aprendizaje son dinámicas, es usual empezar el proceso de aprendizaje con algún conjunto de datos grande y estático donde no se incluyan cambios de conceptos [32, 75]. El principal propósito es garantizar que el algoritmo de aprendizaje sea estable en la ausencia de cambios, aparte de la comparación que se puede realizar con algunos algoritmos tradicionales. El repositorio de la UCI (del inglés, *University of California Irvine*) [76] es una de las fuentes de conjuntos de datos más utilizada para esta tarea.

CONJUNTOS DE DATOS ARTIFICIALES.

Los conjuntos de datos que son construidos de forma artificial tienen el beneficio de poder controlar diferentes escenarios donde los algoritmos pueden demostrar su rendimiento. Por lo general se encuentran dos tipos diferentes de conjuntos de datos, los dirigidos a probar cambios abruptos y los que introducen cambios graduales. Adicionalmente pueden ser añadidas otras características como ruido artificial, atributos irrelevantes, etc.

Relacionado con estos conjuntos de datos donde se introducen cambios abruptos, una idea frecuente que soporta sus operaciones es la generación de conceptos distintos activos en diferentes períodos. El paso de un concepto a otro es inmediato (abrupto) aunque la extensión del periodo de cambio puede simular un cambio gradual de alguna forma. Así, si dos conceptos consecutivos son muy

parecidos, se podría considerar que la velocidad del cambio es lenta. De cualquier forma, se considera que la activación repentina de un nuevo concepto y la desactivación inmediata del concepto previo producen un cambio abrupto. Existen varios conjuntos de datos que operan de esta forma. Uno de los más populares es el generador de conceptos STAGGER, propuesto por Schlimmer y Granger [7], el cual es usado con frecuencia con la misma configuración [10, 34, 75]. Este generador crea ejemplos compuestos por valores de tres atributos:

Color $\in$ {verde, azul, rojo}

Forma $\in$ {triángulo, círculo, rectángulo}

Tamaño $\in$ {pequeño, mediano, grande}

Además, define una clase binaria, usando tres posibles funciones booleanas (por ejemplo):

"$color = rojo \wedge tamaño = pequeño$"

"$color = verde \vee forma = circulo$"

"$tamaño = mediano \vee tamano = l\arg o$"

La clase tiene valor 1, siempre que una de las expresiones anteriores (la expresión que está en uso para el concepto actual) es verdadera y 0 en caso contrario. Cada una de estas expresiones corresponde a un concepto diferente. Estos conceptos son fijados y tienen un particular solapamiento, pero la idea puede ser extendida para obtener otras funciones que determinen más conceptos [33].

Tomando a los conceptos STAGGER como un punto de partida, es fácil generalizar el proceso de definir conceptos diferentes y más complejos. Así, Widmer y Kubat [31] incluyen modificaciones simples para tratar con conceptos recurrentes o ruido, pero ellos proponen también un generador basado en funciones booleanas que combina más de tres atributos nominales. En este sentido, LED [77] puede ser considerado como un generador que también usa funciones booleanas, ya que este modela la combinación de atributos (segmentos individuales en una pantalla digital de siete segmentos) que tienen que ser activados en forma de dígito (diez etiquetas de clase). Este generador es una

variante de la versión original donde el patrón de activación de segmentos individuales es estable (sin cambio de concepto).

Bifet y otros [77] definen diferentes patrones de activación para segmentos individuales (para ser usadas en diferentes períodos) por medio del intercambio de atributos que forman parte de los siete segmentos de la pantalla por otros atributos que son irrelevantes. Este generador es usado con mucha frecuencia [79, 69].

Los generadores vistos hasta el momento solo incluyen atributos nominales, pero también existen generadores para incluir atributos numéricos. Las funciones descritas por Agrawal y otros [80] para determinar cuál préstamo debe ser aprobado pueden ser usadas para generar diferentes conceptos. Cada función modela un concepto diferente y se puede simular nuevamente cambios abruptos considerando una de ellas en diferentes períodos de tiempo [77]. Una aproximación similar es presentada por Kubat y Widmer [10] en el generador mixto (MIXED), donde son utilizados dos atributos booleanos y dos atributos numéricos para describir el problema. Como se ha mostrado, las funciones booleanas pueden ser utilizadas o adaptadas con facilidad para generar problemas artificiales con atributos nominales. Si los atributos son exclusivamente nominales, pueden ser usados otros tipos de funciones. Aparte de la definición del generador mixto, Kubat y Widmer [10] se describen otro tipo de funciones como SINE, CIRCLES y GAUSSIAN [10], que usan funciones matemáticas y estadísticas para determinar los conceptos. Todos estos generadores producen problemas con dos valores de clases (positivos o negativos) y dos formas diferentes de simular cambios abruptos: invirtiendo la clasificación, donde los ejemplos positivos se convierten en negativos y viceversa (SINE y GAUSSIAN); o definiendo conceptos diferentes, incluso si ellos se solapan (CIRCLES) y el cambio pueda parecer gradual.

Aunque estos generadores son relativamente simples, permiten probar los algoritmos desde otra perspectiva que propicia su uso [34, 60]. Otro generador ampliamente usado es SEA, presentado por Street y Kim [73]. El valor de la clase binaria que es asignada a los ejemplos generados depende de una función que chequea si la suma de dos atributos numéricos excede un umbral determinado. Pueden ser simulados varios conceptos cambiando el nivel del umbral. Esta idea

es fácilmente reproducida [74] y puede ser extendida para la evaluación del rendimiento de los algoritmos en presencia de ruido [72].

Similar al generador LED que fue modificado para incluir cambio de concepto. El generador de formas de ondas (waveform) también ha sido adaptado [77]. En este caso, el generador resultante usa atributos numéricos y diferencia entre tres clases, cada una de las cuales es generada a partir de una combinación de dos o tres funciones base con formas de ondulación. La forma de simular cambio es similar al método usado con el generador LED, este intercambia el papel que desempeñan los atributos en la descripción del problema: algunos atributos relevantes son intercambiados por atributos irrelevantes y viceversa.

Todos los generadores presentados hasta el momento simulan tipos de cambio abruptos. El cambio puede ser suavizado si la extensión del cambio es pequeña, pero existen mejores formas de simular cambios graduales. El acercamiento más usado y adaptado es el propuesto por Widmer y Kubat [31]. En vez de producir cambio entre dos conceptos inmediatamente, este acercamiento combina los conceptos por medio de su ponderación durante un período de tiempo. Así, antes de que el cambio comience, el concepto actual tiene una presencia total (su peso es 1), mientras que el concepto final no tiene presencia (su peso es 0). Existe una etapa de transición entre conceptos (t_{cambio}) que termina con la configuración opuesta (el concepto antiguo no tiene presencia y solo está presente el concepto final). El método más extendido propone un cambio uniforme caracterizado por la pendiente de una función (α), mientras la presencia de un concepto previo decrece, la presencia del segundo concepto aumenta (1 - α). Se puede observar cómo la presencia de ambos conceptos siempre representa el 100% del conjunto de datos resultante. Esta propuesta es muy conveniente cuando se necesita la simulación de cambios graduales, e incluso más cuando el cambio deseado debe ser muy lento [60]. Para suavizar el extremo de la función previa se propone la función sigmoide [77]. Note que la combinación de ambos conceptos se mantiene igual al 100%.

Adicionalmente se han definido algunas notaciones y propiedades para describir cuán diferentes son los conceptos cuando se combinan en fases de cambio

usando la función sigmoide [77]. Una notación equivalente puede extrapolarse para la función con pendiente. El caso usual considera solamente dos conceptos, pero otras propuestas recientes [78] sugieren el incremento de este número para combinar más de dos conceptos. En adición a estas propuestas para simular cambios de concepto graduales desde conceptos separados (con mayor o menor solapamiento), existen otros generadores que logran la misma conducta. Ellos están definidos para problemas con atributos numéricos. El generador más conocido es el hiperplano rotante usado por Hulten y otros [46]. Un hiperplano con d dimensiones espaciales es denotado por la ecuación:

$$\sum_{i=1}^{d} a_i x_i = a_0$$

Son etiquetados como ejemplos positivos los que satisfacen que:

$$\sum_{i=1}^{d} a_i x_i \geq a_0$$

Y como ejemplos negativos aquellos que satisfacen:

$$\sum_{i=1}^{d} a_i x_i \leq a_0$$

Los hiperplanos han sido utilizados para simular cambios de conceptos porque la orientación y posición del hiperplano puede ser cambiada suavemente cambiando la magnitud de sus pesos. Este generador ha recibido mucho interés y los elementos que determinan el tipo de cambio (número de atributos, extensión del cambio, etc.) fueron rápidamente parametrizados [35, 32].

Otro generador que puede crear conjuntos de datos con cambio gradual está basado en funciones de base radial (RBF) [77]. El mismo genera un número diferente de centroides que son usados para definir funciones de base radial. Cada centroide tiene una clase asociada a él, entonces el número máximo de clases en este problema está limitado por el número de centroides. La generación de ejemplos lo guían hiperesferas distribuidas normalmente alrededor de cada centroide. El cambio de concepto es simulado moviendo los centroides de una forma suave. Hasta ahora se han mostrado los generadores más importantes, se

han destacado sus detalles más importantes y explicado sus principales características y alcance.

Los generadores de conjuntos de datos previos pueden ser adaptados para adicionarles nuevas características, lo que es importante para diferentes métodos y algoritmos. Algunos de ellos están diseñados para tratar con diferentes grados de ruido, o para asumir varios niveles de cambio, por lo que es importante comprobar cómo ellos son afectados cuando ocurren tales situaciones. Por ejemplo, existen alternativas para extender los conjuntos de datos con atributos irrelevantes.

Así, la asignación de valores aleatorios a los nuevos atributos que no influyen en el concepto es una opción para su extensión [10, 31]. Otra opción es asignar pesos a cada atributo y configurar a cero el peso de los atributos irrelevantes [46]. Si el objetivo es adicionar ruido, se puede cambiar la etiqueta de clase de los ejemplos de forma aleatoria e independientemente [31, 46, 72], o seleccionar aleatoriamente la etiqueta de clase si existiera una mezcla de conceptos [10]. La extensión del cambio puede variar también, dependiendo del acercamiento seleccionado. Podemos encontrar definiciones de conceptos que cambia a su opuesto, como el caso de SINE y GAUSSIAN, de conceptos que son fijos y se solapan parcialmente, así como STAGGER o SEA, cuya extensión del cambio puede ser calibrada. Teniendo en cuenta la calibración, pueden ser consideradas las siguientes acciones: selección guiada de conceptos (funciones booleanas), variación de los pesos de los atributos (hiperplano rotante) o determinación del número de atributos relevantes que pueden intercambiarse con atributos irrelevantes (LED o waveform).

Como se ha señalado anteriormente, una de las principales ventajas de este tipo de conjunto de datos es la posibilidad de crear cualquier cantidad de datos, con variables grados de configuración. Para facilitar esta tarea, algunos autores ofrecer su propia implementación de estos generadores. Por ejemplo, MOA [81] es uno de los marcos de trabajo más completos que soporta muchos conjuntos de datos (STAGGER, LED, funciones de préstamo, SEA, waveform, hiperplano rotante y RBF aleatorio). Narasimhamurthy y Kuncheva [78] proveen herramientas

que ofrecen otros generadores (STAGGER, GAUSSIAN, hiperplano rotante). Minku y otros [33] también facilitan implementaciones alternativas para generar conjuntos de datos con cambio de concepto (STAGGER, conceptos booleanos, SEA, SINE, CIRCLES, hiperplano rotante).

CONJUNTOS DE DATOS REALES

El aprendizaje a partir de flujo de datos está directamente relacionado con el cambio de concepto porque esta es una característica inherente que aparece usualmente cuando los datos son adquiridos a lo largo del tiempo. Para tener una idea aproximada de problemas reales donde es importante el cambio de concepto, se puede observar cuáles conjuntos de datos reales ha sido usado con el propósito de la evaluación. A pesar que estos conjuntos de datos pueden ser usados indistintamente con respecto al modelo de aprendizaje, en específicos campos de aplicación, algunos modelos de aprendizajes tienen ventajas con respecto a otros [11, 54].

Internet es uno de los escenarios más motivadores para las tareas de aprendizaje incremental con la presencia de cambios de concepto. En este contexto, múltiples fuentes generan enormes cantidades de datos con una tasa de llegada alta: datos de correos, flujos de clics, peticiones de usuarios, registros web, servidores web, descargas punto a punto, etc. Aquí podemos diferenciar entre dos categorías fundamentales: aquellas relacionadas con sistemas de comunicación (capas de red, capas de transporte, etc.) y las relacionadas con el contenido que se intercambia. Desde un punto de vista de la administración, es importante la extracción de conocimientos interesantes que posibiliten la predicción de futuras situaciones (saturación de la red, intrusión en la red, etc.). Con respecto al contenido, una tarea muy común es el filtrado de correos basuras. El objetivo final del diseño de algoritmos y métodos que puedan aprender cuando los conceptos cambian es aplicarlos a situaciones reales. Lógicamente, en este punto la variedad de conjuntos de datos se convierte mucho más extensiva. Desde el punto de vista de aplicar procesos de evaluación a diferentes algoritmos, la principal diferencia de estos conjuntos de datos es la capacidad de reproducirlas.

En estos problemas reales son comunes los datos privados (rastro web, datos de fraude de créditos [35], etc.), ya que estos no pueden publicarse para el uso abierto debido a cuestiones de derechos de autor, limitaciones técnicas, etc. Pero en otros casos, aunque la reproducción exacta no es posible, se pueden obtener conjuntos de datos similares desde la misma fuente de datos y siguiendo un proceso específico (subconjunto de documentos seleccionados a partir de un conjunto de datos [9], simulación de juegos de video, etc.). Lo más importante en nuestro caso es la identificación de conjuntos de datos reales que puedan ser considerados como puntos de referencia, porque ellos han sido usados ampliamente en diferentes trabajos investigativos. Como sugerimos anteriormente, el repositorio de la UCI [76] es una de las fuentes más comunes y algunos de los conjuntos de datos que allí se encuentran han sido estudiados desde la perspectiva del cambio de concepto (Adult, Poker hand, Ozone level detection [69], etc.).

Otra fuente de conjunto de datos es facilitada por competiciones organizadas con el soporte de conferencias bien conocidas. La copa KDD, organizada por la ACM Special Interest Group on Knowledge Discovery and Data Mining, es especialmente destacable. Desde 1997 han propuesto tareas a resolver, muchas de las cuales pueden ser manipuladas desde un punto de vista del aprendizaje con la presencia de cambio de concepto [69]. Otra competición anual, PAKDD Data Mining Competition, organizada por la conferencia Pacific-Asia Knowledge Discovery and Data Mining ofrece problemas y conjuntos de datos en diferentes sitios siempre que esta tiene lugar.

Un área muy interesante que produce enormes cantidades de datos, en la cual los algoritmos son evaluados, es la relacionada con correos basuras. Así, Katakis y otros [68] tratan de simular y detectar cambio de concepto (abrupto o gradual) adaptando conjuntos de datos originales [68] y haciendo disponible los conjuntos de datos finales usados. Delany y otros [83] también ofrecen los conjuntos de datos generados, conocidos como ECUE Spam Datasets. Es común que los investigadores ofrezcan los conjuntos de datos que usan. Así, Gama y otros [34] han popularizado el conjunto de datos Electricity propuesto por Harries [84], el cual

ha sido extensamente usado [60, 69]. Otros investigadores que publican algunos conjuntos de datos en sus propios sitios son Zhu (conjuntos de datos enfocados en sensores y suministro de energía) y Polikar (conjunto de datos climáticos).

HERRAMIENTAS DE SOFTWARE DE CÓDIGO ABIERTO

En general, es útil tener la implementación de los algoritmos y métodos en el área del aprendizaje automático. Desde un punto de vista práctico y enfocándose en aquellos que pueden aprender en la presencia de cambio de concepto, esta disponibilidad permite a los investigadores desarrollar nuevas ideas, evaluar el rendimiento de distintas alternativas, etc. En este contexto los algoritmos pueden ser ofrecidos en un escenario independiente y aislado, como ocurre con la herramienta VFML (Very Fast Machine Learning) [85], el cual incluye el algoritmo CVFDT; o el software ADWIN [62], que implementa un algoritmo basado en ventanas deslizantes para detectar cambio. La integración de estas herramientas con otras puede ser una tarea más o menos simple, pero tomar marcos de trabajo más generales como punto de partida ofrece ventajas adicionales.

El sistema popular Weka (del inglés, Waikato Environment for Knowledge Analysis) [86] contiene una colección de herramientas de visualización y algoritmos para el análisis de datos y el modelado predictivo. En la actualidad este es usado en muchas áreas, fundamentalmente con objetivos educacionales e investigativos, pero el mismo ha sido adoptado por algunas compañías. Weka soporta pre-procesado, agrupamiento, clasificación, regresión, visualización y selección de atributos. Los algoritmos que están más cercanos al contexto del cambio de concepto son aquellos que implementan la interfaz updateableclassifier, ya que ella induce modelos de clasificación incremental que pueden aprender usando una instancia a la vez.

Entre los software que están más específicamente orientados al aprendizaje en presencia de cambio de concepto, podemos encontrar al entorno de trabajo MOA (del inglés, Massive Online Analysis) [81]. Este está relacionado con el proyecto WEKA. MOA incluye una colección de algoritmos de aprendizaje automático (clasificación, regresión y agrupamiento) orientados a la minería de flujos de datos, donde el cambio de concepto es un aspecto relevante. Este dispone de una gran

variedad de algoritmos inherentes al aprendizaje incremental, métodos para detectar cambio de concepto, herramientas para la evaluación y muchos generadores de conjuntos de datos artificiales con la posibilidad de incluir varios tipos de cambio de concepto.

Otro sistema que considera parcialmente cambio de concepto es RapidMiner (previamente YALE) [82]. Este sistema ofrece una gran variedad de métodos y permite una rápida fase de prototipado, reduciendo el costo de nuevas aplicaciones. Adicionalmente, posee una amplia funcionalidad para la evaluación y optimización de procesos. Para manipular cambio de concepto, RapidMiner necesita un software adicional (plugin) que aunque no está disponible para la versión actual, se puede obtener para una versión anterior. Así, este sistema es especialmente notorio porque sus desarrolladores planifican soportar minería de flujos de datos y aprendizaje en línea en versiones futuras.

CONCLUSIONES DEL CAPÍTULO II.

En el presenta capítulo se describieron los principales conceptos vinculados con el trabajo sobre flujos de datos y los cambios de conceptos.

Desde que la comunidad científica internacional tomó conciencia de la existencia de cambios de conceptos en la mayoría de los flujos datos reales que son obtenidos a partir del trabajo diario humano, ha dedicado gran parte de su esfuerzo a la construcción de modelos que sean capaces de trabajar bajo estas nuevas condiciones. Han sido adaptados un conjunto de modelos conocidos, entre los que se incluyen los árboles de decisión, las reglas decisión, las redes neuronales, las máquinas de soporte vectorial entre otros.

Debido al uso de tan disimiles modelos se realizó un estudio de las principales metodologías, herramientas de software y conjuntos de datos utilizados para la evaluación y comparación de algoritmos incrementales para la detección de cambios de conceptos. De este estudio podemos concluir que se han llegado a establecer para la evaluación de estos algoritmo un conjunto de métricas, metodologías, se han determinado desde el punto de vista práctico muchos

conjuntos de datos para pruebas, tanto artificiales como reales, pero no se ha llegado a un consenso global de que parámetros y pasos de evaluación utilizar en cada caso particular.

Se han desarrollado algunos entornos para el trabajo sobre flujos de datos en presencia de cambios de conceptos entre los que podemos mencionar a MOA y a RapidMiner. MOA incluye una colección de algoritmos de aprendizaje automático (clasificación, regresión y agrupamiento) orientados a la minería de flujos de datos.

En trabajos muy recientes, que abordan temas de clasificación en flujos de datos con presencia de cambios de conceptos, se ha prestado especial atención a los sistemas multiclasificadores ya que proporcionan un mecanismo que combina de manera eficaz un conjunto de clasificadores obteniendo un modelo más complejo que un clasificador simple pero también más preciso. Una de las tareas de los sistemas multiclasificadores es la mejora de la precisión con respecto a un clasificador individual (Clasificador Básico) mediante la combinación con otros modelos tratando así de eliminar errores en la clasificación de observaciones con el modelo obtenido [87]. Esta mejora tiene su explicación lógica, podemos decir que un clasificador individual puede equivocarse ante una observación en concreto, mientras que al usar un conjunto de clasificadores existe la posibilidad de que la mayoría de los clasificadores logren alcanzar la predicción correcta, estas predicciones dadas por la mayoría se pueden combinar para así evitar y reducir errores individuales.

El principal objetivo de los sistemas multiclasificadores es aumentar el poder predictivo, pero se consigue a costa de mermar el poder descriptivo. A pesar de aumentarse por lo general la precisión con estos modelos, la interpretabilidad de los clasificadores básicos es mayor que la que puede alcanzarse con la combinación de varios modelos. Aunque estos modelos multiclasificadores presentan dificultades de interpretación existe una ventaja, y es que se aumenta el poder expresivo, es decir, el modelo multiclasificador puede sintetizar conocimiento que antes no podía ser recogido en un único modelo básico [13].

Algunas ventajas de los multiclasificadores sobre los clasificadores simples, según Wang y otros [35] son:

- Los multiclasificadores ofrecen una mejor precisión en la predicción.
- Construir un multiclasificador es más eficiente que construir un modelo simple.

- Los multiclasificadores por naturaleza dan mayores posibilidades de trabajar en paralelo y sobre grandes bases de datos online.

MULTICLASIFICADORES PARA MINAR FLUJOS DE DATOS.

A la hora de diseñar algoritmos que generen sistemas multiclasificadores hay que considerar dos puntos fundamentales [23, 88]: cómo son generados los distintos clasificadores básicos que componen el sistema multiclasificador y cómo son combinados para realizar una predicción conjunta.

Por otra parte, en el trabajo de Gama [34] se distinguen dos categorías donde se ubican las estrategias para enfrentar el problema del cambio de concepto: estrategias que adaptan el aprendizaje en intervalos de tiempo regulares sin considerar que ha ocurrido un cambio en el concepto; y estrategias que primero detectan el cambio de concepto, y luego el aprendizaje es adaptado al cambio. Los multiclasificadores por lo general se incluyen dentro de la primer estrategia, ya que cuentan con mecanismos (actualizar los clasificadores existentes, eliminar los clasificadores con bajo rendimiento, insertar nuevos clasificadores, etc.) que le permiten ir evolucionando sin necesidad de detectar directamente los puntos de cambio. Sin embargo, hoy en día existen varios trabajos que insertan en los muticlasificadores mecanismos de detección directa de cambios de conceptos, lo que los incluye también dentro de la segunda estrategia. La ventaja de incorporar los detectores de cambios es aprovechar la capacidad de los multiclasificadores para adaptarse a los cambios graduales, combinado con el trabajo natural del detector de cambios para los cambios abruptos.

A continuación se describen varios trabajos relacionados a la investigación tomando en cuenta las tres características mencionadas anteriormente.

Una de las primeras propuestas para el trabajo con flujos de datos fue SEA (del inglés, *Streaming Ensemble Algorithm*) [73]. SEA va dividiendo el conjunto de entrenamiento en bloques del mismo tamaño y con cada uno de estos bloque construye un nuevo clasificador básico que agrega al conjunto. El algoritmo cuenta con un límite máximo de clasificadores que al ser alcanzado y como mecanismo

de adaptación, obliga a sustituir a clasificadores básicos anteriores siguiendo cierto criterio de remplazo. Para unificar las predicciones de los clasificadores básicos utiliza votación por mayoría no ponderada y como generador de los clasificadores básicos al algoritmo C4.5. SEA se adapta a los cambios graduales del entono pero tiene problemas para adaptarse a los cambios de conceptos abruptos; en estos resultados influye el mecanismo de votación utilizado y que los clasificadores dejan de aprender una vez que son creados.

Bajo el mismo esquema de división del conjunto de entrenamiento, Wang [35] propone el método AWE (del inglés, *Accuracy Weighted Ensemble*). Esta propuesta, para combinar las respuestas de los clasificadores básicos utiliza votación por mayoría ponderada. La ponderación de los clasificadores básicos está en función de la precisión obtenida por los mismos, al utilizar como test los propios ejemplos del bloque de entrenamiento actual. Al igual que SEA, tiene buenos resultados frente a cambios graduales del entorno, pero esto no ocurre así cuando los cambios son abruptos. Uno de los motivos de esta ineficiencia es que se tienen que esperar al próximo bloque de entrenamiento para actualizar los pesos de los clasificadores básicos. Desafortunadamente reducir el tamaño de los bloques no resuelve el problema pues traería como consecuencia que la precisión general del sistema descendiera mucho.

Una familia de algoritmos que sigue el mismo esquema de división del conjunto de entrenamiento tiene como exponentes fundamentales a los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC propuestos por José del Campo [13]. Estos multiclasificadores, al igual que SEA, utilizan votación por mayoría no ponderada; como diferencias utiliza al algoritmo CIDIM [89] (Control de Inducción por División Muestral) para inducir los clasificadores básicos, y una medida de calidad mucho más simple para sustituir estos. Además, el algoritmo MultiCIDIM-DS-CFC, utiliza un segundo clasificador simple incremental llamado IADEM (Inducción de Árboles de Decisión por Muestreo) [13] como filtro de corrección, donde este clasificador incremental interviene en la votación del resultado final. Ambas propuestas tienen dificultades para manipular cambios de conceptos, en especial del tipo abrupto.

Una extensión propuesta por los autores como línea futura consiste en incluir un detector de cambio de concepto para que los ajustes ante cambios abruptos sean más eficientes.

Una propuesta más reciente es el algoritmo BWE (del inglés, *Batch Weighted Ensemble*) [90]. BWE también construye sus clasificadores básicos dividiendo el conjunto de entrenamiento en bloques de igual tamaño, utiliza votación por mayoría ponderada para unificar las predicciones parciales y tiene como antecedente fundamental al algoritmo AWE. Esta propuesta es una de las que incorpora un detector de cambio al modelo; este detector es llamado BDDM (del inglés, *Batch Drift Detection Method*) y utiliza un modelo de regresión como ayuda para determinar la presencia de cambios de conceptos. El detector de cambios se utiliza fundamentalmente para determinar si es necesario crear un nuevo clasificador básico debido a cambios de conceptos, o si el concepto se mantiene estable y no se modifica el multiclasificador. La idea que se defiende es combinar la capacidad de los multiclasificadores para adaptarse a los cambios graduales con el trabajo natural del detector de cambios para detectar cambios abruptos.

De acuerdo con Gonçalves and Barros [91] el algoritmo AUE (del inglés, *Accuracy Updated Ensemble*) [92] es una mejora al algoritmo AWE. Ambos utilizan clasificadores básicos ponderados, cuyos pesos son actualizados con los nuevos datos que van arribando. La principal diferencia entre estos es el uso de clasificadores incrementales en lugar de clasificadores estáticos. AUE propone una simple función para evitar que se haga cero el peso de todos los clasificadores básicos, una situación que puede pasar en el algoritmo AWE. Además, AUE solo actualiza los clasificadores si han obtenido una alta precisión sobre los datos recientes.

Otra idea para manipular el conjunto de datos de entrenamiento es trabajar los ejemplos de entrenamiento uno a uno a medida que van llegando.

Un algoritmo que utiliza este sistema para actualizar sus clasificadores básicos es DWM (del inglés, *Dynamic Weighted Majority*) [75]. DWM está basado en el

algoritmo WMA (del inglés, *Weighted Majority Algorithm*) [93] del cual toma la idea de trabajar con un conjunto de expertos a los que se les asigna un peso inicial; luego, cuando llega una nueva instancia, el algoritmo básico recibe una predicción de cada experto y toma una decisión final combinando las predicciones y los pesos de cada uno de los expertos; finalmente, los expertos que no coincidieron con la mayor votación son penalizados en sus pesos multiplicándolos por una constante entre 0 y 1. DWM para adaptarse al trabajo con flujos de datos y para manipular los cambios de conceptos incluye mecanismos para agregar, actualizar y eliminar clasificadores básicos. Cada cierto periodo p se realiza una prueba y si la salida del sistema es incorrecta, se agrega un nuevo clasificador con valor de peso 1; además, el sistema elimina todo clasificador básico cuyo peso desciende de un umbral θ. Uno de los posibles problemas de este algoritmo es que penaliza a los clasificadores básicos cuando fallan pero no los bonifica cuando aciertan, esto hace que sus pesos pueden descender rápidamente y permanecer poco tiempo dentro del multiclasificador. Por lo tanto el algoritmo olvida rápidamente los viejos conceptos lo cual no lo hace adecuado para el tratamiento de conceptos recurrentes; además, puede influir en que sea poco robusto al ruido.

También Kolter y Maloof [65], proponen el algoritmo llamado AddExp (del inglés, *Additive Expert Ensemble*). Este sistema es muy parecido a DWM y tienen mecanismos comunes como el tipo de votación, la forma de insertar nuevos clasificadores y el mecanismo para eliminar de forma rápida varios clasificadores al mismo tiempo. Como diferencia propone dos métodos para sustituir los clasificadores; el primero está basado en eliminar el más viejo, para lo que incluye una constante que controla el tiempo que el experto lleva dentro del multiclasificador y el segundo está basado en el más débil, pues elimina el que tiene valor de peso más bajo. AddExp, por su similitud a DWM, hereda las mismas deficiencias en la manipulación de conceptos recurrentes.

Con la misma estrategia de trabajo sobre el flujo de datos de entrada, fue propuesto ICEA (del inglés *Incremental Classification Ensemble Algorithm*) [94]. La idea que sigue a esta propuesta, es que cada clasificador básico aprende de

forma incremental, agregando automáticamente y lo más pronto posible el resultado de su aprendizaje. Según los autores, se obtiene como resultado una más rápida detección del cambio de concepto, en comparación con algunos algoritmos basados en bloques. ICEA utiliza mecanismos de adaptación muy parecidos a los de DWM, votación ponderada, una constante entre 0 y 1 para penalizar los fallos de los clasificadores básicos, y un valor de umbral θ para eliminar de forma conjunta clasificadores básicos. Al igual que en DWM es posible que los clasificadores básicos permanezcan poco tiempo dentro del multiclasificador no haciéndolo eficiente para tratar conceptos recurrentes; además de que los clasificadores básicos son readaptados constantemente olvidando los conceptos viejos.

El algoritmo DWM-WIN [95], propone algunas modificaciones al algoritmo DWM. La primera modificación está basada en una característica de la versión del algoritmo Winnow implementada por Blum [96]. Winnow es similar a WMA en la idea de modificar el peso de los expertos según su predicción individual, la diferencia es que incluye una nueva constante multiplicativa ($\eta > 1$) para recompensar el peso de los expertos cuando su predicción es correcta. Agregando esta característica, DWM-WIN logra que cada experto tenga más posibilidades de permanecer dentro del multiclasificador si su comportamiento mejora con el tiempo; esto lo hace más flexible en el tratamiento de conceptos recurrentes. Otra modificación es que en alguna variante del algoritmo propuesto se toma en cuenta, a la hora de eliminar expertos, la edad de estos.

Un multiclasificador que utiliza estrategias muy particulares para mantener una alta diversidad y adaptarse a los cambios de conceptos es DDD (del inglés, *Diversity for Dealing with Drifts*) [97]. DDD mantiene varios multiclasificadores con diferentes niveles de diversidad. Si en los datos no se detecta la presencia de cambios de conceptos, el sistema estará compuesto por dos multiclasificadores, uno con baja diversidad y otro con alta diversidad. Cuando un cambio de concepto es detectado dos nuevos multiclasificadores son construidos, uno con baja diversidad y otro con alta diversidad; los viejos multiclasificadores son mantenidos

ya que, según los autores, esto garantiza mejor explotación de la diversidad, el uso de la información aprendida de viejos conceptos y la robustez ante falsas alarmas. Los cuatro multiclasificadores son mantenidos mientras se cumplan dos condiciones específicas que chequean la situación de cambio, de lo contrario, utilizando un mecanismo de combinación se pasa a trabajar con dos multiclasificadores nuevamente. Los autores refieren que DDD es capaz de mantener mejor precisión que otras propuestas como DWM.

ADAPTACIONES DE LOS ALGORITMOS BAGGING Y BOOSTING.

En esta sección se presentan algunas propuestas recientes que adaptan los conocidos algoritmos Bagging [98] y Boosting [99] para el trabajo con flujos de datos.

Bagging y Boosting son algoritmos que mediante heurísticas de votación y ponderación, van creando modelos intermedios en base a los cuales forman un único modelo final cuya exactitud mejora la exactitud de cualquiera de ellos. Mediante Bagging el modelo final es compuesto a partir de las reglas más frecuentes dentro de varios modelos individuales y mediante Boosting se generan varios clasificadores que son votados de acuerdo a su tasa de error; sin embargo, a diferencia de Bagging, no se obtienen a partir de diferentes muestras sino secuencialmente sobre el mismo conjunto de entrenamiento.

Versiones incrementales de los algoritmos Bagging y Boosting fueron propuestas por N. Oza y S. Russel [100] desde el año 2001. Sin embargo, otras versiones que se adaptan a los cambios de conceptos han aparecido en fechas más recientes.

Una versión adaptativa del algoritmo Bagging fue llamada OzaBagADWIN [77]. La idea de esta variante es agregar a la versión incremental del algoritmo Bagging [98] un detector de cambio llamado ADWIN (del inglés, Adaptive Windowing) [101]. El mecanismo de adaptación utilizado se basa en la sustitución del peor de los clasificadores en un instante de tiempo por un nuevo clasificador básico creado más recientemente.

Una versión adaptativa del algoritmo Boosting para trabajar sobre flujos de datos fue llamada AEC (del inglés, Adaptive Ensemble Boosting Classifier) [102]. Esta nueva versión utiliza a Boosting como método de multiclasificación combinado con una ventana deslizante adaptativa y un árbol de Hoeffding para detectar cambios de conceptos existentes, y de ser necesario agregar un nuevo clasificador básico. Como clasificador básico utiliza una versión adaptativa del conocido algoritmo Naive Bayes. Según muestran los resultados, el algoritmo funciona bien en entornos de datos con cambios de conceptos, ya que se adapta dinámica y rápidamente a los cambios y además, necesita poca memoria para su funcionamiento.

Otra versión adaptativa del algoritmo Boosting para trabajar sobre flujos de datos, que sigue una idea similar al AEC, fue propuesta en 2013 [103]. La nueva adaptación tiene como idea combinar al conocido algoritmo boosting con el detector de cambios ADWIN [101]. La propuesta utiliza a Boosting como método de multiclasificación y a ADWIN para detectar cambios de conceptos existentes, y de ser necesario manipular la ventana de datos de entrada y agregar nuevos clasificadores básicos. Los resultados muestran que el método propuesto toma menos tiempo, menos memoria y eleva más la precisión que otros métodos conocidos.

ALGORITMOS PARA TRATAR CONCEPTOS RECURRENTES.

Hasta ahora, ninguno de los clasificadores mencionados anteriormente toman en cuenta la posible presencia de conceptos recurrentes, debido a ello, estos algoritmos no están adaptados a trabajar con este tipo de cambios de conceptos.

ACE (del inglés, *Adaptive Classifiers Ensemble*) [104] es un algoritmo capaz de manipular cambios de conceptos recurrentes mejor que un sistema convencional. Este multiclasificador está acompañado por cuatro elementos; primero, un clasificador simple que trabaja los datos de entrada uno a uno de forma incremental, este clasificador sustituye al multiclasificador, en las labores de predicción, cuando ocurren cambios de conceptos abruptos debido a que el multiclasificador demora mucho en actualizarse pues tiene que esperar al próximo

bloque; segundo, un detector de cambios de concepto, éste es otro de los multiclasificadores que incluye dentro de su estructura un detector de cambio; tercero, una ventana deslizante utilizada para almacenar los resultados de precisión predictiva y los intervalos de confianza de cada clasificador sobre los datos más recientes; y por último, un búfer utilizado para almacenar recientes ejemplos de entrenamientos y para construir los nuevos clasificadores.

Un multiclasificador especializado en el tratamiento de conceptos recurrentes fue presentado por Ramamurthy and Bhatnagar [70]. Esta propuesta construye un conjunto histórico global de clasificadores (árboles de decisión) a parir de una secuencia de bloques de igual tamaño. Cada clasificador individual de este conjunto representa a un concepto diferente. Por lo tanto, un nuevo clasificador básico es solo incluido cuando hay un cambio de concepto en el flujo de datos y cuando este concepto no está representado dentro del conjunto histórico global. Los clasificadores históricos nunca son eliminados debido a que el concepto que ellos representan puede reaparecer. No todos los clasificadores básicos participan en el proceso de clasificación al mismo tiempo. El sistema utiliza un filtro que permite que solo los clasificadores relevantes al concepto participen en el proceso de clasificación. Esta propuesta, al igual que AWE, tiene que esperar por el próximo bloque para adaptar todo el mecanismo del sistema.

Aunque no es un multiclasificador, se incluye el algoritmo para manipular flujos de datos llamado RCD (del inglés, *Recurring Concept Drifts*) [91]. RCD no es ni un clasificador simple ni un multiclasificador, es una colección de clasificadores (podría clasificarse como un metaclasificador) de la cual se selecciona cual clasificador utilizar en cada momento basado en la distribución de los datos actuales; para esto son utilizadas pruebas estadísticas no paramétricas. A la colección se agrega un nuevo clasificador, unido a una muestra significativa de los datos con que fue creado, cada vez que un nuevo concepto detectado no coincide con ninguno de los conceptos de distribuciones almacenados previamente. Los autores declaran que sus resultados son superiores a los de los otros algoritmos

cuando se enfrentan a cambios bruscos y que obtienen resultados igualados cuando se enfrentan a cambios graduales.

TABLA 3.1. CARACTERÍSTICAS DE LOS MULTICLASIFICADORES PARA TRABAJO EN LÍNEA (Y RCD).

ALGORITMO	AÑO	REFERENCIA	CAMBIOS (ABRUPTO, GRADUAL)	CONCEPTOS RECURRENTES	EJEMPLOS (BLOQUE, UNO_A_UNO)	VOTACIÓN (PONDERADA, NO PONDERADA	DETECTOR DE CAMBIO
SEA	2001	[14]	GRADUAL	NO	BLOQUE	NO PONDERADA	NO
AWE	2003	[35]	GRADUAL	NO	BLOQUE	PONDERADA	NO
AUE	2011	[93]	AMBOS	NO	UNO_A_UNO	PONDERADA	NO
MULTICIDIM-DS	2007	[13]	GRADUAL	NO	BLOQUE	NO PONDERADA	NO
MULTICIDIM-DS-CFC	2007	[13]	GRADUAL	NO	BLOQUE	NO PONDERADA	NO
BWE	2011	[90]	AMBOS	NO	BLOQUE	PONDERADA	SI
DWM	2003	[75]	AMBOS	NO	PERIODO P. (P= 1)	PONDERADA	NO
ADDEXP	2003	[65]	AMBOS	NO	PERIODO P. (P= 1)	PONDERADA	NO
ICEA	2007	[94]	AMBOS	NO	UNO_A_UNO	PONDERADA	NO
DWM-WIN	2012	[95]	AMBOS	NO	PERIODO P. (P= 1)	PONDERADA	NO
ACE	2005	[104]	AMBOS	SI	AMBOS	PONDERADA	SI
DDD	2012	[97]	AMBOS	SI	UNO_A_UNO	PONDERADA	SI
OZABAG-ADWIN	2009	[77]	AMBOS	NO	BLOQUE	NO PONDERADA	SI
AEC	2012	[102]	AMBOS	NO	VENTANA DESLIZANTE	PONDERADA	SI
BOOSTING-ADWIN	2013	[103]	AMBOS	NO	VENTANA DESLIZANTE	PONDERADA	SI
RCD	2013	[91]	AMBOS	SI	VENTANA DESLIZANTE	NO PROCEDE	SI

Finalmente, se ha decidido incluir otras dos propuestas que aunque no son multiclasificadores, si están especializadas en el tratamiento de conceptos recurrentes y sus mecanismos de adaptación pueden resultar de interés.

Li y otros [105] propusieron un algoritmo de clasificación llamado REDLLA para trabajar con flujos de datos en presencia de cambios de conceptos recurrentes y datos donde su atributo clase puede ser desconocido. Este algoritmo fue construido para aprendizaje semi-supervisado y utiliza un árbol de decisión como

modelo de clasificación. Cuando un árbol crece un algoritmo de agrupamiento basado en el método de k-medias es instalado para producir grupos de conceptos y etiquetar datos no etiquetados en las hojas del árbol. Si existen desviaciones entre grupos de conceptos históricos y los nuevos grupos potenciales cambios de conceptos son distinguidos y cambios de conceptos recurrentes son tomados en cuenta. De acuerdo con los autores, el algoritmo REDLLA es eficiente y efectivo para minar cambios de conceptos recurrentes sobre grandes volúmenes de datos no etiquetados.

Gama y Kosina [79] presentan un método que memoriza modelos de decisión ya creados cada vez que un cambio de concepto es encontrado. El sistema utiliza técnicas de meta-aprendizaje que caracterizan el dominio de aplicabilidad de modelos aprendidos anteriormente. A través de las técnicas de meta-aprendizaje se pueden detectar la ocurrencia de previos conceptos y de esta forma activar modelos aprendidos previamente. De acuerdo con los autores, el principal beneficio de esta propuesta es que el sistema es capaz de seleccionar conceptos históricos similares sin el conocimiento de las clases verdaderas de los ejemplos.

CONCLUSIONES DEL CAPÍTULO III.

Después de realizar un estudio de los principales y más recientes trabajos de investigación relacionados con los algoritmos de multiclasificación podemos concluir que:

En los últimos años ha existido un notable aumento de la cantidad de investigaciones científicas relacionadas con los sistemas multiclasificadores vinculados a la minería de grandes flujos de datos en presencia de cambios de conceptos. Esto se debe a la especial eficiencia y adaptabilidad de este tipo de modelos para el trabajo en línea, sobre datos que van arribando en el tiempo.

Nuevos mecanismo de adaptación están siendo probados con vista de adaptar estos modelos a los diferentes tipos de cambios de conceptos. Varios trabajos han combinado los sistemas multiclasificadores con detectores de cambios (como ADWIN, DDM, EDDM, etc.) con el objetivo de reforzarlos para manipular cambios de conceptos abruptos. La idea es, combinar el trabajo natural de los algoritmos multiclasificadores para detectar cambios de conceptos graduales, con la habilidad de los detectores de cambio para detectar cambios de conceptos abruptos.

El número de trabajos científicos que utilizan los sistemas multiclasificadores para la manipulación de cambios de conceptos recurrentes es bastante reducido. Por su importancia, esta línea de investigación debería de ser más abordada; sobre todo, por la posible presencia de este tipo de cambios de conceptos en flujos de datos reales y la necesidad de realizar un trabajo eficiente sobre estos.

CAPÍTULO IV. DESDE LA FAMILIA MULTICIDIM HASTA UNA NUEVA PROPUESTA.

En este capítulo se describirán algunas características de los principales algoritmos de la familia MultiCIDIM. Se tomarán como punto de partidas los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC, algoritmos diseñados para trabajar con flujos de datos, para lograr construir otras propuestas que además de trabajar con flujos de datos, sean capaces de manipular cambios de conceptos.

ALGORITMOS MULTICIDIM-DS Y MULTICIDIM-DS-CFC.

El algoritmo CIDIM (Control de Inducción por División Muestral), propuesto por Ramos [89], ha sido objeto de posteriores estudios [12] y aplicado a casos reales con resultados satisfactorios [107], debido principalmente a sus propiedades: inducción de árboles de decisión precisos y de reducido tamaño.

Aprovechando estas características y añadiéndole la particularidad de que, por el propio diseño del algoritmo CIDIM, es sencillo obtener diferentes árboles de decisión, aun usando el mismo conjunto de datos, resulta apropiado diseñar métodos que utilicen CIDIM como algoritmo de aprendizaje para inducir los clasificadores básicos. Como consecuencia de diversos enfoques se han diseñado diferentes sistemas multiclasificadores entre los que tenemos los que siguen un enfoque más clásico, como M-CIDIM [12], E-CIDIM [108] o FE-CIDIM [109], o los que usan múltiples sistemas multiclasificadores en sucesivas capas para refinar la clasificación, como ML_m-CIDIM [111] o ML-bin_m-CIDIM [110].

ALGORITMO MULTICIDIM-DS.

El algoritmo MultiCIDIM-DS [13] es un multiclasificador que utiliza como algoritmo de aprendizaje para inducir los algoritmos básicos al algoritmo CIDIM y está preparado para trabajar con flujos de datos (del inglés, *DS por datastream*). El proceso básico que lo define es el que se muestra en la Figura 4.1.

Entrada: E (conjunto de datos),
$algoritmo\_aprendizaje$ (algoritmo para inducir los clasificadores básicos),
tam_bloque (tamaño del bloque de datos por iteración),
max_num_clasif (máximo número de clasificadores básicos)

1. $Conjunto = \emptyset$
2. **mientras** $E \neq \emptyset$ **hacer:**
2.1 $Entrenamiento \leftarrow$ Extraer las primeras tam_bloque experiencias de E (son eliminadas de E)
2.2. $Clasificador_i \leftarrow$ Inducir un nuevo clasificador con $algoritmo\_aprendizaje$ usando $Entrenamiento$
2.3. $Conjunto = Conjunto \cup \{Clasificador_{i-1}\}$
2.4. **Para todo** $x \in Conjunto$ **hacer:**
2.4.1 Actualizar $tasa\_acierto(x)$ usando $Entrenamiento$
2.5. **si** $|Conjunto| > max\_num\_clasif$ **entonces:**
2.5.1. $Peor\_clasificador = \{x \in Conjunto \mid tasa\_acierto(x) < tasa\_acierto(y) \; \forall y \in Conjunto \wedge x \neq y\}$
2.5.2. $Conjunto = Conjunto - \{Peor\_clasificador\}$

Salida: $Conjunto$ (sistema multiclasificador inducido)

FIGURA 4.1 SEUDOCÓDIGO DEL MÉTODO GENERAL PARA INDUCIR SISTEMAS MULTICLASIFICADORES. (TOMADA DE TESIS DE, DEL CAMPO [13]).

Para la generación de este sistema multiclasificador se usa el método de *bagging*. La primera razón que ha conducido al uso de este método es que el algoritmo de inducción básico que se usa (CIDIM) no está diseñado para minimizar los errores ponderados de la forma en que los presenta *boosting*. La segunda razón, es que el enfoque seleccionado no presenta la limitación de que el sistema multiclasificador pueda requerir cada vez más memoria por el hecho de añadir nuevos clasificadores básicos y no eliminar ninguno antiguo. Por último, en cada momento considera cuáles son los clasificadores básicos más precisos y no los elimina simplemente por el hecho de ser más antiguos [13]. Los elementos más destacados que diferencian éste, de otros algoritmos como SEA [14], son la elección del algoritmo CIDIM en lugar del algoritmo C4.5 [27] y la modificación del criterio para comparar los modelos.

Este algoritmo recibe como argumentos, además del algoritmo de aprendizaje que se usa como clasificador básico y el del propio conjunto de datos, otros dos argumentos que determinan el comportamiento del algoritmo: el número de experiencias que compondrán cada uno de los subconjuntos que serán usados para inducir cada uno de los clasificadores básicos (bloques de ejemplos) y el número máximo de clasificadores básicos que se permitirá almacenar.

FILTROS CORRECTORES PARA LA CLASIFICACIÓN (CFC).

Este trabajo no tiene como objetivo adentrarse en el estudio de los filtros correctores para clasificación por lo que solamente se describirán las características fundamentales del método de mejora propuesta por del Campo-Ávila **[13]**.

Este método para la corrección incluye algunos clasificadores que tratan de aprender cuáles ejemplos son clasificados correctamente por los clasificadores básicos correspondientes, al igual que el método propuesto por Ortega usando sistemas clasificadores no incrementales [112].

Estos clasificadores nuevos incluidos (llamados filtros de corrección) pueden ser inducidos por cualquier tipo de algoritmo incremental [43, 109]. Así, el conjunto será constituido por un número máximo de clasificadores básicos y el mismo número de filtros de corrección para la clasificación (CFC). Los clasificadores nuevos son llamados filtros de corrección porque se usarán para filtrar la salida de los clasificadores básicos cuando son usados para predecir. Estos clasificadores básicos permanecen inalterables, lo que se altera son sus medidas de calidad y sus filtros de corrección. Los ejemplos pasados al algoritmo que induce los filtros de corrección son los mismos ejemplos usados para inducir clasificadores básicos nuevos (o para actualizar la medida de calidad de los clasificadores básicos existentes) pero con una diferencia, el atributo de clase. El atributo original de clase es reemplazado por un atributo binario que representa el éxito o fracaso del clasificador básico correspondiente cuando se usa para clasificar el ejemplo

original. Así, el filtro de corrección aprenderá cuáles ejemplos están correctamente clasificados por su clasificador básico correspondiente y cuál está mal clasificado.

Del Campo Ávila propuso un nuevo uso de los filtros de corrección para desarrollar un algoritmo concreto llamado *MultiCIDIM-DS-CFC* [13], de éste se describen sus características principales a continuación.

ALGORITMO MULTICIDIM-DS-CFC.

A partir del método descrito anteriormente para mejorar la precisión alcanzada por los sistemas multiclasificadores orientados a realizar aprendizaje incremental, se definió el algoritmo MultiCIDIM-DS-CFC [13]. Como se puede deducir del nombre, este algoritmo usa como básico el algoritmo MultiCIDIM-DS (descrito anteriormente) y le incorpora la capacidad de clasificar usando filtros correctores.

Este algoritmo usa como algoritmo incremental para actualizar la información de los filtros correctores el IADEM-2. El MultiCIDIM-DS-CFC, para cada uno de los clasificadores básicos que se incluyan en el sistema multiclasificador creará, un modelo que recogerá el conocimiento sobre qué tipo de experiencias son correctamente clasificadas por el clasificador básico. Su función será que cada filtro corrector aprenda, de forma incremental, que experiencias son las que el clasificador básico que le corresponde clasifica correctamente.

Se necesita que este aprendizaje sea incremental porque es posible que un clasificador básico, que no es incremental y permanece inalterable desde que se induce, permanezca en el sistema multiclasificador durante muchas iteraciones. Cuanto más tiempo se encuentre en el sistema multiclasificador, más conjuntos de entrenamiento habrán servido para incrementar el conocimiento y la única forma de incorporarlo todo, sin desechar nada, es usando un algoritmo incremental [13].

ANÁLISIS DE LOS ALGORITMOS BASADOS EN SEA CON VISTA A ADAPTARSE A LOS CAMBIOS DE CONCEPTOS.

Como se ha mencionado anteriormente, los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC tienen como base al algoritmo SEA. Aunque existen notables diferencias como: el algoritmo utilizado para inducir los clasificadores básicos, la

medida de calidad para sustituir los clasificadores básicos y el uso de los filtros correctores en la última versión; también existen algunos puntos de coincidencia como: la forma de dividir el conjunto de entrada para formar los clasificadores (en bloques), el sistema de votación (por mayoría no ponderado) y en parte el mecanismo utilizado para sustituir los clasificadores básicos.

El artículo de Kolter y Maloof [75] y el de Yue y otros [94], analizados en la revisión bibliográfica, destacan algunas deficiencias que puede presentar el algoritmo SEA para responder y adaptarse eficientemente a los cambios de conceptos:

POSIBLES DEFICIENCIAS DEL ALGORITMO SEA SEGÚN ANÁLISIS DE KOLTER Y MALOOF.

1- En el algoritmo SEA, los clasificadores básicos dejan de aprender una vez que son creados. Esto implica que un fijo periodo de tiempo debe de ser suficiente para aprender todos los conceptos, por lo que puede ser posible que no se logre aprender un nuevo concepto en este periodo fijo de tiempo.

2- Además, el método de remplazar clasificadores no ponderados, por uno que mejore la precisión global del multiclasificador suele no converger muy rápido a la hora de adaptarse a un nuevo concepto, sobre todo, si el cambio de un concepto a otro es un cambio abrupto.

POSIBLES DEFICIENCIAS DEL ALGORITMO SEA SEGÚN ANÁLISIS DE YUE Y OTROS.

1- El algoritmo SEA requiere un grupo de instancias para detectar cambios en la cadena de datos (un bloque), por lo que podría no detectar cambios de conceptos bajo ciertas circunstancias. Por ejemplo, asumamos un concepto continuo C_p en la cadena de datos y además existe otro cambio de concepto rápido C_q, puede ser que SEA detecte el cambio de concepto continuo, pero no el cambio rápido.

2- El algoritmo SEA trabaja dividendo los datos en bloques por lo que no se puede adaptar al aprendizaje incremental ya que cuando una nueva instancia llega éste no puede tomar acción inmediata. Este método de división, en bloques, puede

provocar “fluctuación de conceptos”, por ejemplo, un concepto puede ser descartado cuando el correspondiente conjunto de datos es eliminado y sin embargo luego es reconstruido cuando arriban cadenas de datos parecidas en la cadena de datos.

Como resumen, se podría decir que el algoritmo SEA carece de la habilidad de adaptarse a cambios de conceptos abruptos y baja mucho su precisión de predicción global cuando se enfrenta a cadenas de datos con pocas instancias en el tiempo. Esto se debe a: la forma en que inserta y elimina clasificadores, su forma de votación simple y que espera un número fijo de experiencias para crear un nuevo clasificador, el cual no es entrenado nuevamente.

ANÁLISIS MEDIANTE UN EJEMPLO. SEA Y DWM.

A continuación se presenta un análisis hecho por Kolter y Maloof [75], bajo ciertas condiciones y siguiendo las características de cada algoritmo, para mostrar que el algoritmo DWM es capaz de adaptarse con mayor rapidez a un cambio de concepto brusco que el algoritmo SEA.

El método SEA entrena a un nuevo clasificador con un número fijo de experiencias; si el nuevo clasificador mejora la precisión global del multiclasificador, entonces éste es agregado, asumiendo que todavía no se ha llegado al número máximo de clasificadores; en caso contrario, es decir, ya se llegó al máximo, se selecciona el peor clasificador de los ya existentes y se sustituye por el nuevo en caso que éste mejore el funcionamiento global del multiclasificador.

Basado en lo anterior, si todos los clasificadores del multiclasificador basado en SEA han sido entrenados sobre un concepto determinado y ocurre un cambio brusco a un concepto totalmente disjunto al anterior, entonces el algoritmo necesitará remplazar la mitad de los clasificadores antes de poder adaptarse y lograr una buena precisión en la clasificación.

Consideremos lo siguiente, tenemos un multiclasificador con 20 clasificadores y cada uno ha aprendido de un bloque de 500 experiencias, entonces si éste funcionara como SEA y ocurriera un cambio a un concepto totalmente disjunto al anterior, le tomaría unas 5000 experiencias para poderse adaptar al nuevo concepto, pues necesitaría sustituir 10 de los clasificadores viejos para poder votar a favor del nuevo concepto.

En contraste, DWM bajo similares circunstancias requiere solamente 1500 experiencias. Asumamos que tenemos 20 clasificadores, que cada uno entrena con un conjunto de 500 experiencias, que el peso de cada uno de los 20 es 1 (el peso inicial) y además que el criterio de sanción a un clasificador que falla es 0,5. Entonces, si ocurre un cambio de concepto brusco, totalmente disjunto del concepto anterior, ocurre lo siguiente:

En la próxima clasificación, todos los clasificadores predicen de forma equivocada y por lo tanto, siguiendo el algoritmo, todos los pesos son reducidos a la mitad, es decir 0.5 y es creado un nuevo clasificador con un peso inicial igual a 1.

Fueron necesarias 500 experiencias para crear este nuevo clasificador y esta vez el multiclasificador volverá a fallar, porque ahora 20 clasificadores con un peso de 0.5 cometerán un error y sólo el más joven de los clasificadores votará a favor ($20*0.5 = 10 > 1$). Una vez más, siguiendo el algoritmo, todos los pesos son reducidos a la mitad, es decir 0.25, y es creado un nuevo clasificador con un peso inicial igual a 1.

Este nuevo clasificador necesitará 500 nuevas experiencias y ahora cuando ocurra una nueva clasificación, una vez más el multiclasificador ha de fallar, pues 20 de los clasificadores votarían en contra y sólo dos lo harían a favor ($20*0.25 = 5 > 2$). Nuevamente, siguiendo el algoritmo, todos los pesos son reducidos a la mitad, es decir 0.125, luego es creado un nuevo clasificador con un peso inicial igual a 1, para esto serán necesarias otras 500 experiencias.

Sin embargo, en la nueva clasificación ya el multiclasificador estará completamente adaptado al nuevo concepto, pues al realizar la votación, 20

clasificadores lo hacen en contra pero con un peso de 0.125 y tres clasificadores con un peso igual a 1 votan a favor (20*0.125 = 2.5 < 3). Entonces, el algoritmo DWM, sólo necesita de 1500 experiencias para adaptarse al nuevo concepto, dadas estas condiciones iniciales.

Generalmente los multiclasificadores que utilizan mecanismos de pesos convergen mucho más rápido a nuevos conceptos que los multiclasificadores que remplazan clasificadores básicos no ponderados.

PRIMEROS PASOS HACIA UNA NUEVA PROPUESTA.

Partiendo de la familia de algoritmos multiCIDIM (básicamente de MultiCIDIM-DS) y con el propósito de diseñar un nuevo algoritmo que fuese capaz de trabajar con flujos de datos y adaptarse de forma eficiente a los cambios de conceptos, fueron concebidas varias modificaciones.

La nueva propuesta, al igual que varios de los algoritmos analizados en la revisión bibliográfica [13, 14, 35, 90], divide el flujo de datos de entrenamiento en bloques de igual tamaño y con cada uno de estos bloques construye un nuevo clasificador básico que agrega al multiclasificador. El presente algoritmo mantiene la idea de fijar un límite máximo de clasificadores a almacenar, que al ser alcanzado y como mecanismo de adaptación obliga a sustituir a clasificadores básicos anteriores siguiendo ciertos criterios de remplazo.

Este algoritmo, como primera modificación, asocia a cada clasificador básico un peso y utiliza, para unificar los votos parciales, votación por mayoría ponderada. Al igual que Wang [35], para actualizar los pesos, utiliza la precisión obtenida por cada clasificador básico sobre parte del conjunto de entrenamiento actual; como diferencias, propone una nueva fórmula para el ajuste de los pesos y además, no tiene que esperar a completar el nuevo bloque de entrenamiento, sino que va haciendo actualizaciones parciales de los pesos. Debido a las características de la fórmula de actualización, los pesos asociados a cada clasificador básico pueden descender o aumentar según su comportamiento frente a los nuevos datos.

Otra de las modificaciones, de la nueva propuesta, es que agrega un nuevo parámetro que funciona como umbral de eliminación y que le permite actualizar de forma más rápida el conjunto de clasificadores, si alguno de estos ha quedado desadaptado. Todo clasificador básico cuyo peso asociado es menor que el umbral propuesto es eliminado del multiclasificador.

$S = e_1, e_2, \ldots, e_i, \ldots$: Flujo de datos.

$e_i = (\vec{x_i}, y_i)$: Cada ejemplo de entrenamiento está formado por un vector $\vec{x_i}$ y por un valor discreto y_i, llamado etiqueta, que toma valores de un conjunto Y, llamado clase.

$\vec{bc}$: Vector de clasificadores básicos.

$\vec{w}$: Vector de pesos asociados a los clasificadores básicos.

$E = \{\vec{bc}, \vec{w},\}$**:** multiclasificador

block: Conjunto de ejemplos necesarios para construir un nuevo clasificador básicos.

t_block: Conjunto de ejemplos necesarios para actualizar el multiclasificador.

ne: Número de ejemplos necesarios para construir un nuevo clasificador básicos.

nt: Número de ejemplos necesarios para actualizar el multiclasificador.

Algoritmo general.

```
Inicializar_multiclasificador
While (next example)      // Comenzar el entrenamiento del multiclasificador.
  block ← block ∪ {example}
  t_block ← t_block ∪ {example}
  i ← i+1
  if (i mod ne = 0)   //cada ne ejemplos, se crea un nuevo clasificador básico.
    Adicionar_nuevo_clasificador_básico.
    block ← ∅
  end if              // Fin del bloque para agregar clasificadores básicos.
  if (i mod nt = 0)        // cada nt ejemplos, los pesos son actualizados.
    Actualizar_Multiclasificador
    t_block ← ∅
  end if                   // Fin del bloque de actualización.
end while
```

Algoritmo 1. Nuevo algoritmo, primeras modificaciones a MultiCIDIM-DS.

Inicializar el multiclasificador.

Para que el multiclasificador sea funcional, es necesario garantizar que exista al menos un clasificador básico. Por esta razón, el paso inicial es crear con el primer bloque de entrenamiento un clasificador básico cuyo peso inicial es igual a 1. Además, son inicializadas las variables necesarias para el correcto funcionamiento del multiclasificador en el periodo de entrenamiento.

Inicializar_multiclasificador
block ← ∅
for *i = 1...ne*
next *example*
block ← *block* ∪ {*example*}
end for
$\boldsymbol{bc_1}$ ←build_base_classifier *(block)* // un nuevo clasificador básico es construido.
$\boldsymbol{w_1}$ ← *1*
block ← ∅
t_block ← ∅
i ← *0*

Método. Inicializar el multiclasificador (algoritmo 1).

Adicionar un nuevos clasificador básico.

Cada vez que se completa un bloque de ejemplos un nuevo clasificador básico es construido y agregado al muticlasificador. Si se ha alcanzado el número máximo de clasificadores básicos permitidos por el multiclasificador, primeramente se elimina el clasificador que tiene menor peso y luego es agregado el nuevo miembro. Por lo general, el multiclasificador nunca está completo, pues el método de eliminación mediante el umbral suele garantizar esto.

max: cantidad máxima de clasificadores básicos permitida por el multiclasificador.
n: número de clasificadores básicos dentro del multiclasificador en cada instante de tiempo.

Adicionar_nuevos_clasificador_básico.

if *(n = max) // Multiclasificador completo.*
  Eliminar_clasificador_básico // elimina el clasificador básico de menor peso.
  $\boldsymbol{n \leftarrow n - 1}$
end if

$\boldsymbol{n \leftarrow n + 1}$
$\boldsymbol{cb_n}$ ←*build_base_classifier (block) // un nuevo clasificador básico es construido.*
$\boldsymbol{w_n} \leftarrow 1$

Método. Agregar nuevos clasificadores básico (algoritmo 1).

Actualizar el multiclasificador.

n: Número de clasificadores básicos dentro del multiclasificador en cada instante de tiempo.

β_1, β_2 : Factor de ajuste de los pesos asociados a los clasificadores básicos ($\beta_1 < 0$; $\beta_2 < 0$; $\beta_1 + \beta_2 = 1$).

Actualizar_multiclasificador
for *j ← 1…n*
$\boldsymbol{w_j \leftarrow w_j * \beta_1 + accuracy\,(bc_j, t\_block) * \beta_2}$
if *(*$\boldsymbol{w_j < \theta}$ ***and*** *n > 1)*
  eliminar ($\mathbf{bc_1}$); // se elimina también el peso asociado.
end if
end for

Método. Actualizar el multiclasificador (algoritmo 1).

Los pesos de los clasificadores básicos son actualizados cada cierto periodo nt. El valor de nt, es la cantidad de ejemplos necesarios para actualizar los pesos de los clasificadores básicos; este valor debe ser menor que el tamaño definido para un bloque (ne, cantidad de ejemplos necesarios para crear un nuevo clasificador básico) de forma tal que no sea necesario esperar a tener un bloque completo para poder actualizar los clasificadores. Esta es una de las deficiencias del

algoritmo propuesto por Wang [35], razón por la cual le era difícil detectar cambios de conceptos abruptos.

La fórmula utilizada para actualizar los pesos asociados a los clasificadores básicos es inspirada por estudios en disciplinas como las telecomunicaciones [113], fórmula de suavizado para el cálculo de una medida estable de la usabilidad de las líneas de comunicaciones. Esta forma de actualizar los pesos de los clasificadores básicos permite que estos sean penalizados o bonificados según su comportamiento al clasificar partes del conjunto de entrenamiento actual. Con esta forma de actualizar los pesos, se pretende que los clasificadores básicos puedan permanecer más tiempo dentro del multiclasificador.

Las constantes preestablecidas β_1 y β_2 ($\beta_1 + \beta_2 = 1$) representan el nivel de importancia que se les otorga al funcionamiento de los clasificadores básicos frente a los datos antiguos y frente a los datos actuales respectivamente. Un valor elevado de β_1 (en comparación con el valor de β_2) significa que se le dará más importancia al funcionamiento histórico del clasificador que al funcionamiento frente a los datos actuales; la adaptación al cambio será un poco más lenta pero el proceso será más robusto frente a datos ruidosos. Un valor elevado de β_2 (en comparación con el valor de β_1) significa que se le dará más importancia al funcionamiento actual de los clasificadores básicos que al funcionamiento históricos de los mismos; la adaptación al cambio será mucho más rápida pero es muy posible que el algoritmo sea más afectado por datos ruidosos. De aquí la importancia de lograr un balance entre los valores β_1 y β_2 para poder mediar entre cambios de conceptos y datos con ruidos.

BREVE EXPERIMENTACIÓN. MULTICIDIM-DS Y LA NUEVA PROPUESTA.

Para la experimentación no se utilizó la primera versión de implementación del algoritmo MultiCIDIM-DS [13], sino, una nueva versión de implementación que sigue los requerimientos del entorno de trabajo MOA [81]. El nuevo algoritmo propuesto también fue implementado sobre el entorno de trabajo MOA; de esta forma ambas implementaciones están bajo los mismos requerimientos.

La decisión de utilizar para las implementaciones el entorno de trabajo MOA partió del análisis que se realizó de las principales herramientas software de código abierto, el cual aportó que: MOA incluye una colección de algoritmos de aprendizaje automático (clasificación, regresión y agrupamiento) orientados a la minería de flujos de datos, donde el cambio de concepto es un aspecto relevante. Este entorno de trabajo dispone de una gran variedad de algoritmos inherentes al aprendizaje incremental, métodos para detectar cambios de conceptos, herramientas para la evaluación y muchos generadores de conjuntos de datos artificiales con la posibilidad de incluir varios tipos de cambios de conceptos.

En esta parte de los experimentos no se realizó un estudio profundo para determinar los mejores valores globales de los parámetros de entrada; tan solo teníamos el objetivo de comparar los dos algoritmos bajo condiciones bastante homogéneas.

Los valores utilizados para los parámetros de entradas fueron:

Máximo número de clasificadores básicos: 10, en ambos casos.

Tamaño de los bloques de datos: 1000, en ambos caso.

Para la nueva propuesta:

$\beta_1 = \frac{7}{8}$, $\beta_2 = \frac{1}{8}$ y nt = 100, número de ejemplos para actualizar los clasificadores básicos.

Fueron seleccionados tres generadores de conjuntos de datos artificiales con características muy distintas cada uno, con el objetivo de lograr diversidad en las pruebas (ver tabla 4.1). La idea principal, de esta breve experimentación, es mostrar que la nueva propuesta se adapta mejor a los cambios de conceptos abruptos que la versión original.

TABLA 4.1. CARACTERÍSTICAS GENERALES DE LOS CONJUNTO DE DATOS UTILIZADOS.

GENERADOR DE CONJUNTO DE DATOS	NÚMERO DE ATRIBUTOS	ATRIBUTOS RELEVANTES	ATRIBUTOS IRRELEVANTES	NÚMERO DE EJEMPLOS.	TIPO DE DATOS
SEA	**3**	**2**	**1**	**100 000**	**REAL**
LED	**24**	**7**	**17**	**100 000**	**BINARIO**
STAGGER	**3**	**3**	**-**	**100 000**	**NOMINAL**

Con el generador LED se construyó un conjunto de datos con 100 000 ejemplos. Se insertaron tres puntos de cambios (t_1 = 25 000, t_2 = 50 000 y t_3 = 75000). Para ello se alternaron cuatro conceptos simulados variando el número de atributos con cambio en la secuencia LED (la estructura seguida fue: 1, 3, 5, 7 atributos con cambios respectivamente por conceptos). Se agregó un 10% de ruido.

Con el generador SEA se construyó un conjunto de datos con 100 000 ejemplos. Se insertaron tres puntos de cambios (t_1 = 25 000, t_2 = 50 000 y t_3 = 75000). Para ello se alternaron cuatro conceptos simulados variando la función de clasificación SEA (la estructura seguida fue: función 1, función 2, función 3, función 4 respectivamente por conceptos). Se agregó en cada caso un 5% de ruido.

Con el generador STAGGER se construyó un conjunto de datos con 100 000 ejemplos. Se insertaron dos puntos de cambios (t_1 = 30 000 y t_2 = 60000). Para ello se alternaron tres conceptos simulados variando la función de clasificación STAGGER (la estructura fue función1, función 2, función 3 respectivamente por conceptos). No se agregó ruido en este caso.

TABLA 4.2. STAGGER, SEA Y LED. 100000 EJEMPLOS CADA UNO

	STAGGER		SEA		LED	
CLASIFICADORES	**PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)**	**TIEMPO (S)**	**PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)**	**TIEMPO (S)**	**PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)**	**TIEMPO (S)**
MULTICIDIM-DS (MOA)	**92,88**	**2,22**	**90,15**	**6,66**	**70,88**	**35,68**
NUEVA PROPUESTA.	**99,24**	**1,06**	**91,21**	**5,99**	**72,34**	**34,27**

Como se muestra en la tabla anterior y en los siguientes tres gráficos, la nueva propuesta alcanzó resultados muy superiores. Los valores de precisión y tiempo de ejecución siempre favorecieron al nuevo algoritmo.

Es necesario resaltar (ver próximos tres gráficos) que el comportamiento frente a los cambios de conceptos también es favorable para el nuevo algoritmo; en prácticamente todos los casos, experimentó menores caídas de precisión en los alrededores de los puntos de cambios y además, el tiempo de recuperación logrado fue mucho menor.

Es preciso notar que en los datos tipo STAGGER la diferencia de comportamientos es mucho. El algoritmo original tuvo muchas dificultades para readaptarse en cada cambio.

Se puede concluir que: las modificaciones realizadas al algoritmo original constituyen mejoras sustanciales para el tratamiento de cambios de conceptos.

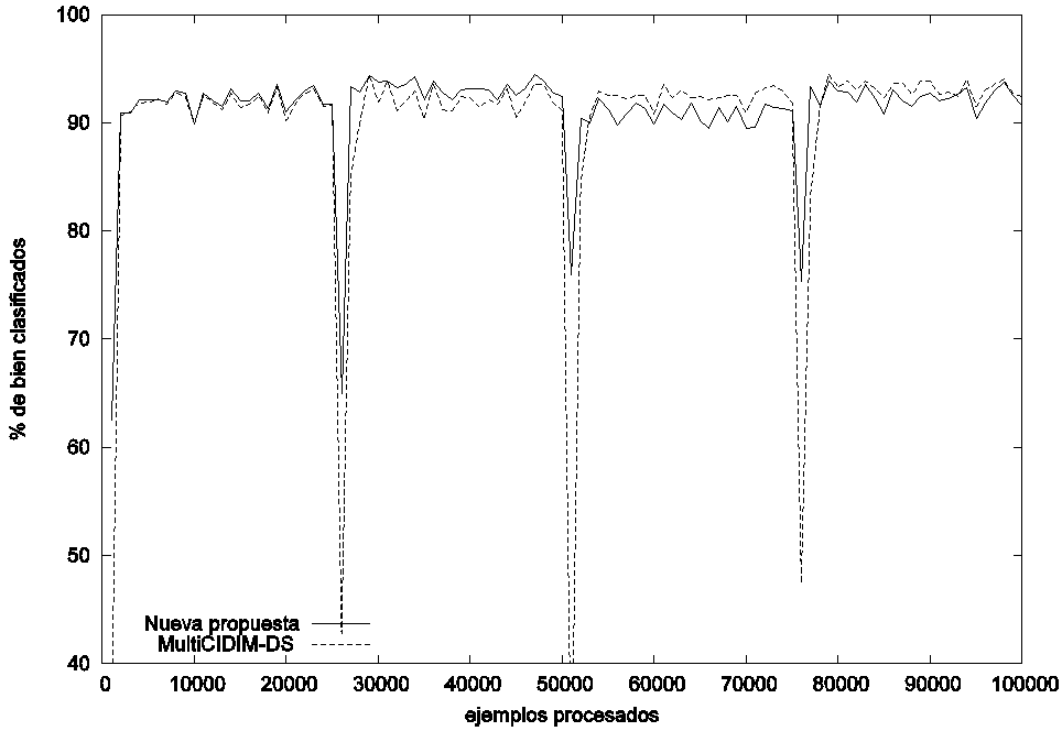
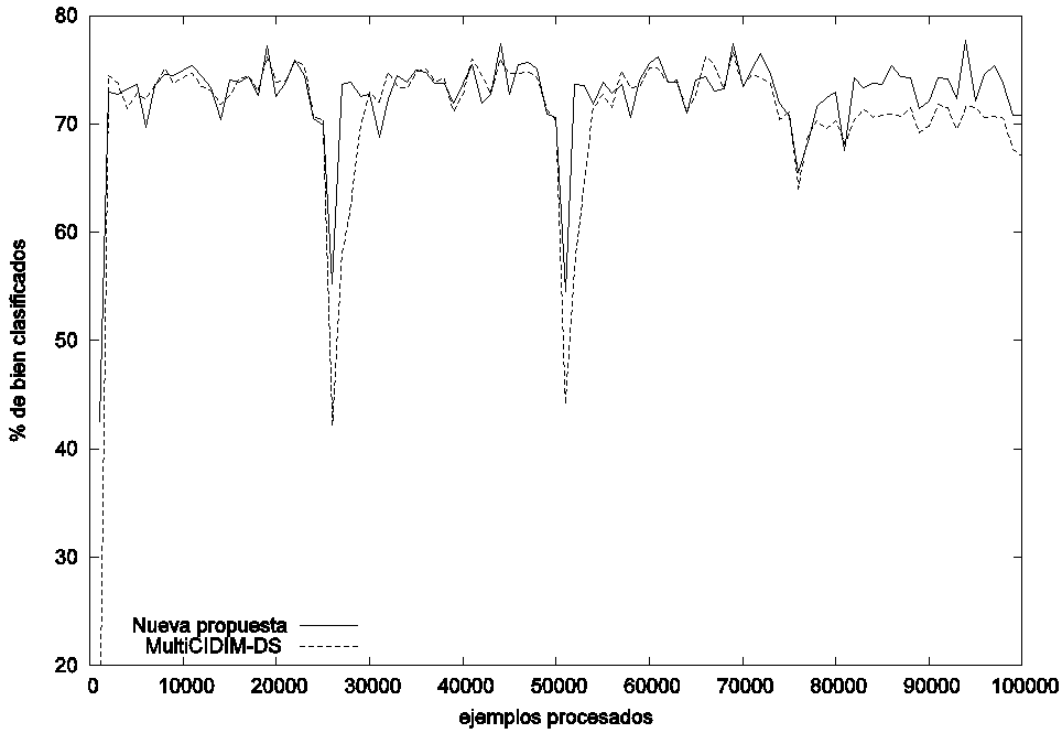


FIGURA 4.2. SEA, 100 000 EJEMPLOS. TRES PUNTOS DE CAMBIO.

.



0.1

FIGURA 4.3. LED, 100 000 EJEMPLOS. TRES PUNTOS DE CAMBIO.

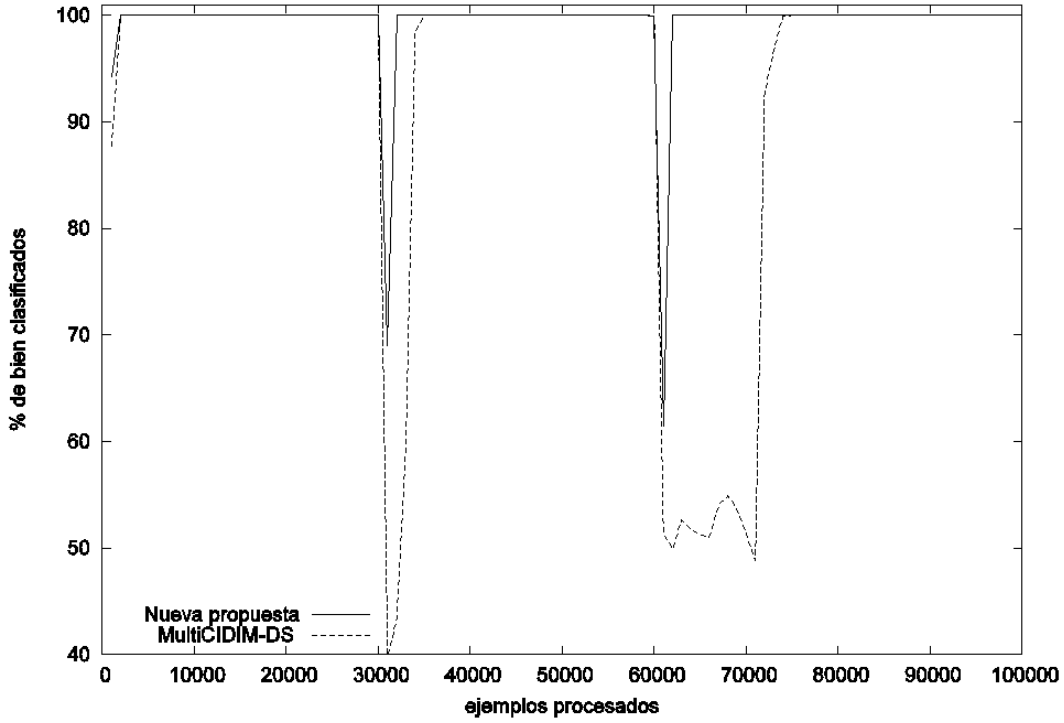


FIGURA 4.4. STAGGER, 100 000 EJEMPLOS. DOS PUNTOS DE CAMBIO.

CONCLUSIONES DEL CAPÍTULO IV.

Después de una breve descripción de los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC, de analizar las características principales de los primeros pasos hacia una nueva propuesta de algoritmo de Multiclasificación, podemos concluir que:

Los algoritmos multiclasificadores, MultiCIDIM-DS y MultiCIDIM-DS-CFC, están perfectamente adaptados para el trabajo sobre grandes flujos de datos. Sin embargo, a pesar de contar con los mecanismos más frecuentes de adaptación para el aprendizaje incremental en multiclasificadores (mecanismos para adicionar, actualizar y eliminar sus clasificadores básicos) tienen ciertas dificultades para adaptarse de forma rápida y eficiente a los cambios de conceptos (sobre todo a cambios abruptos) y no cuentan con mecanismos para el tratamiento de conceptos recurrentes.

Según los trabajos de Kolter y Malof [75] y de Yue y otro [94], el algoritmo SEA (se puede extender a algoritmos con características similares) carece de la habilidad de adaptarse a cambios de conceptos abruptos y baja mucho su precisión de predicción global cuando se enfrenta a cadenas de datos con pocas instancias en el tiempo. Esto se debe a: la forma en que inserta y elimina clasificadores, su forma de votación simple y que espera un número fijo de experiencias para crear un nuevo clasificador, el cual no es entrenado nuevamente.

La nueva propuesta de algoritmo, que toma como punto de partida al algoritmo MultiCIDIM-DS, y le aplica las siguientes modificaciones:

- Cambia el sistema votación por mayoría no ponderado por un sistema de votación por mayoría ponderado. Sgún Kolter y Malof [75], este último sistema de votación es más adecuado para que los algoritmos se adapten de forma rápida a los cambios de conceptos.
- Cada clasificador básico tienen ahora asociado un peso; este peso se actualiza cada cierto periodo utilizando una nueva fórmula que le permite

crecer o decrecer según el comportamiento frente a los datos más actuales.

- Incluye un mecanismo que puede eliminar de forma simultánea a más de un clasificador.

Obtuvo resultados favorables, en el tratamiento de cambios de conceptos, en comparación con el algoritmo original.

CAPÍTULO V. MULTICLASIFICADOR DE ADAPTACIÓN RÁPIDA.

En el presente capítulo se describirán las principales características de un nuevo algoritmo de clasificación llamado Multiclasificador de Adaptación Rápida, de ahora en adelante **FAE** (del inglés, *Fast Adapting Ensemble*). El presente algoritmo se deriva de la propuesta descrita en el capítulo anterior, a la cual se le agregarán varias modificaciones especialmente dirigidas a adaptar la misma al tratamiento de conceptos recurrentes.

Como se ha mencionado en secciones anteriores y fue una conclusión del tercer capítulo, existen muy pocas propuestas de investigación que utilizan sistemas multiclasificadores para el tratamiento de conceptos recurrentes. El uso de clasificadores básicos que se adaptan de forma individual a los cambios de conceptos y que muchas veces permanecen muy poco tiempo dentro del multiclasificador (debido a que son sustituidos por otros clasificadores mejores adaptados) es una estrategia que favorece una rápida adaptación a los cambios de conceptos, sin embargo, en algunas ocasiones puede influir de forma negativa en el correcto tratamiento de conceptos recurrentes.

FAE es un multiclasificador diseñado para adaptarse de forma rápida al cambio de concepto, tanto abrupto como gradual, y especializado en el tratamiento de conceptos recurrentes. Al igual que RCD [91], esta propuesta cuenta con una colección de clasificadores que representan a varios de los conceptos analizados; como diferencia, en este caso los clasificadores están organizados en activos e inactivos según su comportamiento frente a los datos actuales. FAE es un multiclasificador que toma su decisión global a partir de las decisiones parciales de los clasificadores activos; mientras que conserva un grupo de clasificadores inactivos como almacén de antiguos conceptos, los cuales le favorecen en el tratamiento de conceptos recurrentes. Estos clasificadores inactivos son activados de forma muy rápida si reaparece el concepto al cual ellos representan. La reactivación de clasificadores y la inserción, de ser necesaria, de nuevos clasificadores actualizados garantiza una rápida adaptación, sobre todo si existen conceptos recurrentes.

FAE, al igual que varios de los algoritmos analizados [13, 14, 35, 90], incluida la propuesta del capítulo anterior, divide el flujo de datos de entrenamiento en bloques de igual tamaño

y con cada uno de estos bloques construye, de ser necesario, un nuevo clasificador básico que agrega al multiclasificador; de esta forma extrae conocimiento de grandes conjuntos de datos de forma natural. El algoritmo FAE también fija un límite máximo de clasificadores a almacenar, que al ser alcanzado y como mecanismo de adaptación obliga a sustituir a clasificadores básicos anteriores siguiendo ciertos criterios de reemplazo. Esta es la única forma de eliminar clasificadores que tiene la nueva propuesta.

FAE asocia a cada clasificador básico un peso y utiliza, para unificar los votos parciales, votación por mayoría ponderada. Al igual que Wang [35], para actualizar los pesos, utiliza la precisión obtenida por cada clasificador básico sobre parte del conjunto de entrenamiento actual. FAE asume la misma fórmula (que la propuesta del capítulo anterior) para el ajuste de los pesos, por lo tanto, tampoco tiene que esperar a completar el próximo bloque de entrenamiento para ajustar sus mecanismos, sino que se van haciendo actualizaciones parciales. También como antes, debido a las características de la fórmula de actualización, los pesos asociados a cada clasificador básico pueden descender o aumentar, según su comportamiento sobre a los nuevos datos.

Esta propuesta se incluye dentro de la segunda estrategia mencionada por Gama y otros [34] ya que incorpora un detector de cambios al modelo. Al igual que en [90], el detector de cambios se utiliza para determinar cuándo crear un nuevo clasificador básico, según la aparición o no de cambios de conceptos. Si el concepto es estable no se crea un nuevo clasificador innecesario, contribuyendo al ahorro de memoria y favoreciendo la permanencia de clasificadores básicos anteriores que representan a otros conceptos. El propósito de esta idea, agregar un detector de cambios, es aprovechar la capacidad de los multiclasificadores para adaptarse a los cambios graduales combinado con el trabajo natural del detector de cambios para los cambios abruptos. Debido a esto, FAE es capaz de manipular tanto cambios de conceptos graduales como abruptos.

$\mathbf{S} = \mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_i}, \ldots$: Flujo de datos.

$\mathbf{e_i} = (\vec{\mathbf{x_i}}, \mathbf{y_i})$: Cada ejemplo de entrenamiento está formado por un vector $\vec{\mathbf{x_i}}$ y por un valor discreto $\mathbf{y_i}$, llamado etiqueta, que toma valores de un conjunto Y, llamado clase.

$\overrightarrow{\mathbf{bc}}$: Vector de clasificadores básicos.

$\vec{\mathbf{w}}$: Vector de pesos asociados a los clasificadores básicos.

$\overrightarrow{\mathbf{status}}$: Vector de estados asociados a los clasificadores básicos. (Activo o inactivo).

$\overrightarrow{\mathbf{concept}}$: Vector de conceptos asociados a los clasificadores básicos.

$\mathbf{E} = \{\overrightarrow{\mathbf{bc}}, \vec{\mathbf{w}}, \overrightarrow{\mathbf{status}}, \overrightarrow{\mathbf{concept}}\}$: multiclasificador

block: Conjunto de ejemplos necesarios para construir un nuevo clasificador básicos.

t_block: Conjunto de ejemplos necesarios para realizar pruebas al multiclasificador.

ne: Número de ejemplos necesarios para construir un nuevo clasificador básicos.

nt: Número de ejemplos necesarios para realizar pruebas al multiclasificador.

Algoritmo general (FAE).

Inicializar_multiclasificador

While (***next*** *example*) // Comenzar el entrenamiento del multiclasificador.

block ← *block* ∪ {*example*}

t_*block* ← t_*block* ∪ {*example*}

i ← *i+1*

if *(i* ***mod*** *nt = 0)* // cada *nt* ejemplos, los pesos y los estados son actualizados.

Actualizar_peso_clasificador_básico

Actualizar_estado_clasificador_básico

t_*block* ← ∅

end if // Fin del bloque de actualización.

if *(i* ***mod*** *ne = 0)* //cada *ne* ejemplos, se analiza si crear un nuevo clasificador básico.

Adicionar_nuevo_clasificador_básico.

block ← ∅

end if

end while

Algoritmo 2. Multiclasificador de Adaptación Rápida (FAE).

INICIALIZACIÓN DEL MULTICLASIFICADOR.

Para que el multiclasificador sea funcional, es necesario garantizar que exista al menos un clasificador básico activo. Por esta razón, el paso inicial es crear con el primer bloque de entrenamiento un clasificador básico con estado activo y peso inicial igual a 1. Además se inicializa el primer concepto que será analizado y otras variables necesarias en el proceso de entrenamiento.

```
concept: Concepto actual.
Inicializar_multiclasificador
concept ← 1
block ← ∅
 for  i = 1…ne
   next example
   block  ←  block   ∪ {example}
end for
bc_1 ←build_base_classifier (block)     // un nuevo clasificador básico es construido.
w_1 ← 1
status_1 ← active
concept_1 ← concept
block  ← ∅
t_block  ← ∅
i ← 0
```

Método. Inicializar multiclasificador (algoritmo 2).

ACTUALIZAR LOS PESOS Y LOS ESTADOS DE LOS CLASIFICADORES BÁSICOS.

Los pesos y los estados de los clasificadores básicos son actualizados cada cierto periodo *nt*. El valor de *nt,* es la cantidad de ejemplos necesarios para actualizar los pesos y los estados de los clasificadores básicos; este valor debe ser menor que el tamaño definido para un bloque (*ne*, cantidad de ejemplos necesarios para crear un nuevo clasificador básico) de forma tal que no sea necesario esperar a tener

un bloque completo para poder actualizar los clasificadores. Esta es una de las deficiencias del algoritmo propuesto por Wang [35], razón por la cual le era difícil detectar cambios de conceptos abruptos.

La fórmula utilizada para actualizar los pesos asociados a los clasificadores básicos es la misma que la usada en la propuesta anterior [113]. De esta forma, se mantienen las mismas características: los pesos de los clasificadores básicos pueden ser penalizados o bonificados según su comportamiento al clasificar partes del conjunto de entrenamiento actual.

n: Número de clasificadores básicos dentro del multiclasificador en cada instante de tiempo.

$_1$, $_2$: Factor de ajuste de los pesos asociados a los clasificadores básicos ($_1 < 0$; $_2 < 0$; $_1 + \ _2 = 1$).

Actualizar_peso_clasificador_básico

for *j ← 1…n*

$\boldsymbol{w \leftarrow w_j * \beta_1 + accuracy\left(bc_j, t\_block\right) * \beta_2}$

if $\boldsymbol{((status_j = active)\ or\ (w > w_j))}$

$\boldsymbol{w_j \leftarrow w}$

end if

end for

Método. Actualizar peso de los clasificadores básicos (algoritmo 2).

Las constantes preestablecidas $\boldsymbol{\beta_1}$ y $\boldsymbol{\beta_2}$ ($\boldsymbol{\beta_1 + \beta_2 = 1}$) representan el nivel de importancia que se les otorga al funcionamiento de los clasificadores básicos frente a los datos antiguos y frente a los datos actuales respectivamente. Un valor elevado de $\boldsymbol{\beta_1}$ (en comparación con el valor de $\boldsymbol{\beta_2}$ significa que se le dará más importancia al funcionamiento histórico del clasificador que al funcionamiento frente a los datos actuales; la adaptación al cambio será un poco más lenta pero el proceso será más robusto frente a datos ruidosos. Un valor elevado de $\boldsymbol{\beta_2}$ (en comparación con el valor de $\boldsymbol{\beta_1}$) significa que se le dará más importancia al funcionamiento actual de los clasificadores básicos que al funcionamiento

históricos de los mismos; la adaptación al cambio será mucho más rápida pero es muy posible que el algoritmo sea más afectado por datos ruidosos. De aquí la importancia de lograr un balance entre los valores $\boldsymbol{\beta_1}$ $\boldsymbol{y}$ $\boldsymbol{\beta_2}$ para poder mediar entre cambios de conceptos y datos con ruidos.

Es importante resaltar el hecho de que a los clasificadores inactivos no se les penaliza su peso, solo se les bonifica si su comportamiento actual mejora. El propósito de este proceder es no disminuir innecesariamente el peso de un clasificador que ya conocemos no pertenece al concepto actual (por esta razón está inactivo) y además garantizar su rápida activación cuando reaparezca nuevamente el concepto que él representa.

θ: Umbral utilizado para determinar el estado de los clasificadores básicos.

Actualizar_estado_clasificador_básico

for *j ← 1…n*

  if *(* $\boldsymbol{w_j > \theta}$ *)*

    $\boldsymbol{status_j \leftarrow active}$

  else

    $\boldsymbol{status_j \leftarrow inactive}$

  end if

end for

Método. Actualizar estados de los clasificadores básicos (Algoritmo 2).

Los clasificadores básicos pueden tener dos posibles estados, activos o inactivos. Los clasificadores activos son los que mantienen su peso por encima de un umbral $\boldsymbol{\theta}$ preestablecido; solamente estos son los utilizados para predecir en un instante de tiempo, pues se consideran los mejores adaptados al concepto actual. Los clasificadores inactivos son los que mantienen su peso por debajo del umbral $\boldsymbol{\theta}$ preestablecido; estos no intervienen en las predicciones pero se mantienen almacenados, mientras sea posible, pues son los que representan a antiguos conceptos. Los pesos de los clasificadores inactivos también son actualizados cada *nt* ejemplos, solo si este es mejorado.

Cada vez que los pesos de los clasificadores básicos son actualizados, los estados de los mismos son actualizados también; de esta forma se garantiza la activación-inactivación de clasificadores y la actualidad del multiclasificador.
Existen dos detalles de implementación no reflejados en el seudocódigo:
Primero, al actualizar los estados de los clasificadores básicos siempre se garantiza que al menos un clasificador quede en estado activo y además que éste sea el que mejor funcionamiento actual tenga (el de mayor peso).
Segundo, en los experimentos se utiliza un valor algo mayor que el umbral $\underline{\boldsymbol{\theta}}$ para activar un clasificador básico, con este proceder se disminuyen considerablemente los saltos seguidos activación-inactivación o viceversa que resultan molestos y perjudiciales en las predicciones.

AGREGAR UN NUEVO CLASIFICADOR BÁSICO.

Para agregar un nuevo clasificador básico, lo primero que se tiene en cuenta es la información suministrada por el detector de cambios utilizado. Se utiliza un detector de cambios para controlar el proceso de creación de nuevos clasificadores básicos. El detector de cambios a utilizar debe tener como salida tres posibles alertas, sin-cambio, posible-cambio, cambio-seguro. Con estas características (fueron descritas en el capítulo) se encuentran los conocidos detectores de cambios DDM (del inglés, drift detection method) [34] propuesto por Gama y otros autores y EDDM (del inglés, Early drift detection Method) [60] propuesto por Baena y otros autores; además de una reciente propuesta llamada HDDM (del inglés, *Hoeffding drift detection method)* [61], desarrollada por Frías y otros.
Solamente se crea un nuevo clasificador con las alertas de cambio o posible-cambio; la diferencia es, que con la alerta de posible-cambio se crea un nuevo clasificador básico asociado al concepto actual y con la alerta de cambio-seguro se crea asociado a un nuevo concepto. Si la alerta es sin-cambio, el multicalsificador no se modifica.

max: cantidad máxima de clasificadores básicos permitida por el multiclasificador.
n: número de clasificadores básicos dentro del multiclasificador en cada instante de tiempo

Adicionar_nuevos_clasificador_básico.

alert ← Drift detector ( )

if *((alert = "warning")* ***or*** *(alert = "drift"))*

  if *(n = max) // Multiclasificador completo.*

    Eliminar_clasificador_básico

    $\boldsymbol{n \leftarrow n - 1}$

  end if

  $\boldsymbol{n \leftarrow n + 1}$

  $\boldsymbol{cb_n}$ *←build_base_classifier (block) // un nuevo clasificador básico es construido.*

  $\boldsymbol{w_n} \leftarrow 1$

  $\boldsymbol{status_n} \leftarrow$ *active*

  if *(alert = "drift")*

    concept ← concept +1

  end if

  $\boldsymbol{concept_n} \leftarrow$ *concept*

end if

Método. Adicionar nuevo clasificador básico (algoritmo 2).

Esta forma para agregar clasificadores básicos trata de agregar nuevos clasificadores tan solo cuando sea necesario, de esta manera mantiene dentro del multiclasificador por más tiempo a conceptos antiguos que le permite un mejor tratamiento de conceptos recurrentes.

ELIMINAR UN CLASIFICADOR BÁSICO.

El algoritmo elimina a un clasificador básico cuando se necesita agregar un nuevo clasificador y se ha llegado al máximo número de clasificadores básicos preestablecido. El proceso de eliminación tiene en cuenta los siguientes aspectos:

estado de los clasificadores (activo o inactivo), antigüedad o peso de los clasificadores y cantidad de clasificadores asociados a un concepto.

Siempre se trata de eliminar primero un clasificador inactivo, de no existir (todos los clasificador están activos) se procede a eliminar un clasificador activo. Para evitar un seudocódigo engorroso se incluyen explicaciones.

```
Eliminar_clasificador_básico

if (existen clasificadores inactivos)

   borrar un clasificador inactivo // Opción 1.

else if (no existen clasificadores inactivos)

   eliminar un clasificador activo // Opción 2.
```

Método. Eliminar clasificador básico (algoritmo 2).

Opción 1: Se elimina el clasificador inactivo más antiguo que pertenezca a un concepto que tenga asociado más de un clasificador básico. Si todos los conceptos existentes tienen asociado un solo clasificador, se elimina el clasificador inactivo más antiguo y con él al concepto.

Opción 2: Se elimina el clasificador activo con menos peso.

Siempre se trata de eliminar el clasificador más antiguo, pero se tiene en cuenta la permanencia de la mayor cantidad de conceptos representados dentro del multiclasificador. Muchas veces no se elimina el clasificador más antiguo por tal de mantener un clasificador que es el único representante de un concepto dentro del multiclasificador.

Uno de los propósitos del mecanismo empleado para eliminar clasificadores básicos es mantener representados dentro del multiclasificador, el mayor tiempo posible, a todos los conceptos analizados. Este proceder favorece el tratamiento de conceptos recurrentes y mantiene una alta diversidad en el multiclasificador.

En el contexto del aprendizaje automático es muy importante analizar las complejidades espacial y temporal de los algoritmos. Este estudio es mucho más necesario cuando el proceso de aprendizaje es realizado sobre flujos de datos (datos potencialmente infinitos) y realizado en línea (los datos llegan con el transcurso del tiempo y no son almacenados).

Una vez que el algoritmo FAE ha sido analizado en detalles, se ha llegado al punto en que es necesario también detallar las características de su complejidad. Como primer elemento, es necesario notar que el algoritmo puede ser configurado con diferentes clasificadores básicos, por lo tanto los detalles finales acerca de la complejidad van a depender del clasificador básico que sea seleccionado. Por ejemplo, en los experimentos que se analizan en las siguientes secciones es utilizado con frecuencia, como clasificador básico, el algoritmo HoeffdingTree o VFDT [43] por lo que podemos realizar el análisis bajo la asunción de que este ha sido el clasificador escogido. Estudios adicionales para otros clasificadores básicos pueden ser derivados con relativa facilidad.

La **complejidad espacial** está básicamente determinada por el máximo número de clasificadores almacenados dentro del clasificador (max) y el tamaño máximo de cada uno de estos. En este caso, según el ejemplo que se está siguiendo (VFDT), se estaría trabajando con un árbol de decisión; por lo tanto, el tamaño máximo de un modelo de este tipo es un árbol de decisión completamente expandido. Si se asume que la base de datos analizada está definida por un número finito de atributos (n_attr) con un máximo número de valores simbólicos (n_values) la complejidad espacial es $O(max \cdot n\_attr \cdot n\_values)$ que tiene un orden polinómico. Este razonamiento puede ser extendido a atributos numéricos. En el peor de los casos, habría tantos valores diferentes como ejemplos diferentes en el conjunto de datos utilizado para construir el clasificador. Además, aplicando un método de discretización se podrían obtener muchos menos valores [114].

Claramente este es el peor de los casos, pero en general, la cantidad de espacio necesitado debe ser mucho menor.

La **complejidad temporal**, es el contexto que se trabaja, no puede ser estudiada para el proceso completo debido a que este es continuo. Sería factible analizar el tiempo de procesamiento para un ejemplo o en este caso en particular para un bloque de ejemplos (*ne o nt*). El análisis puede ser realizado teniendo en cuenta dos situaciones: construir un nuevo clasificador básico o actualizar el conjunto de estos. En el primero de los casos, la complejidad temporal depende de la complejidad temporal del clasificador básico seleccionado. Según el ejemplo que se está siguiendo (FAE ha sido configurado con VFDT), este requiere un tiempo constante para procesar cada ejemplo (que dependerá del tamaño del árbol completamente expandido), por lo que la complejidad temporal se obtendría con O(ne · n_attr · n_values). En el segundo de los casos, el proceso de actualización, cada ejemplo dentro del bloque de prueba (con tamaño *nt*) es analizado por cada uno de los clasificadores básicos con el objetivo de actualizar todos los pesos y los estados. En el peor de los casos, los clasificadores serían árboles completamente expandidos (cuyas ramas tendrían como nodos tantos atributos existan n_attr), por lo tanto la complejidad temporal sería O(nt · max · n_attr).

ESTUDIO EXPERIMENTAL.

En esta sección se presentarán los algoritmos utilizados para los experimentos, los parámetros empledos por estos, información acerca de los conjuntos de datos utilizados y un estudio empírico de los resultados obtenidos.

El algoritmo FAE fue implementado siguiendo los requerimientos del entorno de trabajo MOA (del inglés, Massive Online Analysis) [81], ya descrito anteriormente.

Los experimentos fueron realizados en un ordenador con las siguientes características: Intel (R), Pentium (R) CPU, P6000 1.87 GHz con 4GB de RAM.

ALGORITMOS.

En los experimentos que a continuación se describen fueron utilizados los siguientes algoritmos: AWE (desarrollado por: Wang y otros en 2003), AUE

(desarrollado por: Brzezinski y Stefanowski en 2011), OzaBagADWIN (desarrollado por: Oza y Russell en 2009) y HoeffdingTree (árboles de decisión con cotas de Hoeffding o VFDT, desarrollado por: Domingos y Hulten en 2000). Las implementaciones de todos estos algoritmos tienen libre acceso en el entorno de trabajo MOA. Las principales características de los mismos fueron descritas en secciones anteriores.

Para todos los multiclasificadores utilizados en los experimentos fue seleccionado como clasificador básico al algoritmo HoeffdingTree o VFDT [43]. Con este proceder se excluye la posible influencia del clasificador básico en la comparación de los multiclasificadores. También se incluyó en la comparación al propio algoritmo HoeffdingTree para tratar de comprobar que su funcionamiento es mejorado por los multiclasificadores que lo utilizan.

Los valores de los parámetros utilizados para cada uno de los algoritmos (AWE, AUE, OzaBagADWIN y HoeffdingTree) utilizados en los experimentos son los valores por defectos definidos en el entorno de trabajo MOA.

Para determinar los mejores valores globales de los parámetros de entrada, con el propósito de maximizar el buen funcionamiento del algoritmo FAE durante el proceso de prueba, se realizaron alrededor de 240 experimentos. En estos experimentos se probaron diferentes configuraciones para los siguientes parámetros de entrada: $\boldsymbol{\beta_1}$ y $\boldsymbol{\beta_2}$, umbral de activación-inactivación ($\boldsymbol{\theta}$), tamaño del bloque de entrada (*ne*) y número de ejemplos para actualizar los pesos (*nt*).

La configuración ideal más estable encontrada fue: $\boldsymbol{\beta_1} = 0{,}5$; $\boldsymbol{\beta_2} = 0{,}5$; $\boldsymbol{\theta} = 0{,}6$; $ne = 500$ y $nt = 50$.

Por lo tanto, el algoritmo FAE utilizó esta configuración en los siguientes experimentos.

También se hicieron pruebas para determinar que detector de cambios usar y los mejores resultados se obtuvieron con DDM. El detector de cambios EDDM encontraba siempre muchos más posibles cambios, por lo que se construían muchos clasificadores básicos innecesarios.

El máximo número de clasificadores básicos permisible por el multiclasificador está relacionado con la memoria disponible en el entorno de trabajo. En los experimentos se utilizó el valor 25.

Conjuntos de datos artificiales.

Se seleccionaron para esta parte de los experimentos dos generadores de conjuntos de datos sintéticos muy utilizados en las investigaciones del área del tratamiento de flujos de datos con presencia de cambios de conceptos. LED fue propuesto por Breiman y otros en 1984 y SEA fue propuesto por Street y Kim en 2001 (ambos fueron descritos anteriormente). Generadores de ambos tipos de datos están disponibles de forma libre en el entorno de trabajo MOA.

LED y SEA tienen características muy diferentes en cuanto a su estructura y tipos de datos. La siguiente tabla muestra las características generales de estos conjuntos de datos.

TABLA 5.1: CARACTERÍSTICAS GENERALES DE LOS CONJUNTOS DE DATOS LED Y SEA.

GENERADORES DE CONJUNTOS DE DATOS	NÚMERO DE ATRIBUTOS	ATRIBUTOS RELEVANTES	ATRIBUTOS IRRELEVANTES	TIPO DE DATOS
LED	24	7	17	BINARIO
SEA	**3**	**2**	**1**	**REAL**

Para simular la presencia de cambios de conceptos, abruptos o graduales, en los flujos de datos el entorno de trabajo MOA utiliza una función sigmoide como una solución práctica para definir la probabilidad que cada nuevo ejemplo de la cadena de datos de entrenamiento tiene de pertenecer al nuevo concepto después del cambio. En este modelo solo se necesita especificar dos parámetros: $\mathbf{t_0}$, el punto de cambio; y ***w,*** la longitud del cambio [81] (número de ejemplos necesarios para la transición de un concepto a otro).

CAMBIOS DE CONCEPTOS ABRUPTOS Y GRADUALES.

En la primera fase de los experimentos, para comprobar el funcionamiento de los algoritmos analizados sobre diferentes tipos de cambios de conceptos se utilizó la siguiente estructura:

100 000 ejemplos, 50 000 pertenecen a un primer conceptos y los 50 000 restantes pertenecen a un segundo concepto ($\mathbf{t_0}$ = 50 000).

La longitud del cambio (**w**) tomará cuatro posibles valores 0, 100, 500 y 1000 para simular desde un cambio de concepto abrupto (**w** = 0) hasta un cambio de concepto más gradual (**w** = 1000). En ambos generadores los datos serán afectados con un 10% de ruido.

La tabla 5.2 muestra los resultados obtenidos al hacer pruebas sobre los conjuntos de datos construidos con el generador LED. Los primeros 50 000 ejemplos fueron generados con un número de atributos con cambios igual a 1 y los 50 000 ejemplos restantes con un número de atributos con cambios igual a 7 (es decir, dos conceptos diferentes). Para simular cambios de conceptos en datos generados con LED solamente es necesario variar el número de atributos con cambio, este número oscila de 0 a 7.

TABLA 5.2: LED, 100 000 EJEMPLOS. UN PUNTO DE CAMBIO: $\mathbf{t_0}$ = = 50 000.

	PRECISIÓN MÁS BAJA ALREDEDOR DEL PUNTO DE CAMBIO (%).				PRECISIÓN. (% DE BIEN CLASIFICADOS)				TIEMPO (S)			
CLASIFICADORES	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
FAE	**45,1**	**50,9**	**55,1**	**54,1**	**73,15**	**73,2**	**73,16**	**73,15**	**63,01**	**64,55**	**66,75**	**68,09**
AUE	19,9	15,6	26,6	31	72,84	72,99	72,95	72,89	128,12	118,81	109,25	125,49
AWE	**45,2**	**44,7**	**27,5**	**29,7**	**73,13**	**73,12**	**72,97**	**72,87**	**234,52**	**238,35**	**230,43**	**204,13**
OZABAGADWIN	33,6	61,6	61,1	57,8	72,68	73,53	73,48	73,44	101,57	102,13	101,43	102,68
HOEFFDINGTREE	**16,9**	**18**	**21,7**	**26,7**	**69,24**	**69,24**	**69,26**	**69,24**	**9,05**	**8,95**	**8,36**	**9,05**

Columnas **A**, w = 0; columnas **B**, w = 100; columnas **C**, w = 500; columnas **D**, w = 1000;

En la tabla 5.2 se puede observar que los valores de precisión se reducen significativamente alrededor del punto de cambio. Este comportamiento es esperado porque en el periodo de transición entre dos conceptos los algoritmos tienden a estar desactualizados.

FAE siempre reporta resultados entre los dos mejores valores de precisión tomados alrededor del punto de cambio (el valor tabulado, es el valor de precisión más bajo alcanzado por cada algoritmo en el periodo de transición de concepto a otro) para todos los valores de w (0, 100, 500 y 1000). FAE también reporta resultados entre los dos mejores para todos los casos de los otros dos parámetros medidos, precisión final y tiempo de ejecución.

Las figuras 5.1 y 5.2 se corresponden con los resultados obtenidos para los valores de w iguales a 0 y 1000 respectivamente. Se pueden ver muy claro en los gráficos las caídas de los valores de precisión en el periodo de transición entre dos conceptos. Es muy importante llamar la atención sobre la profundidad y el ancho de estas caídas de precisión (estas caídas de precisión en los gráficos se observan como picos invertidos, hacia abajo). La profundidad de estas caídas de precisión indican cuánto pierde en precisión cada algoritmo en el paso de un concepto a otro y el ancho de estas caídas indica cuánto tarda en recuperarse. FAE, comparado con los otros algoritmos tanto en los experimentos mostrados como en los no graficados, reporta, por lo general, caídas de precisión más leves y menor tiempo de recuperación.

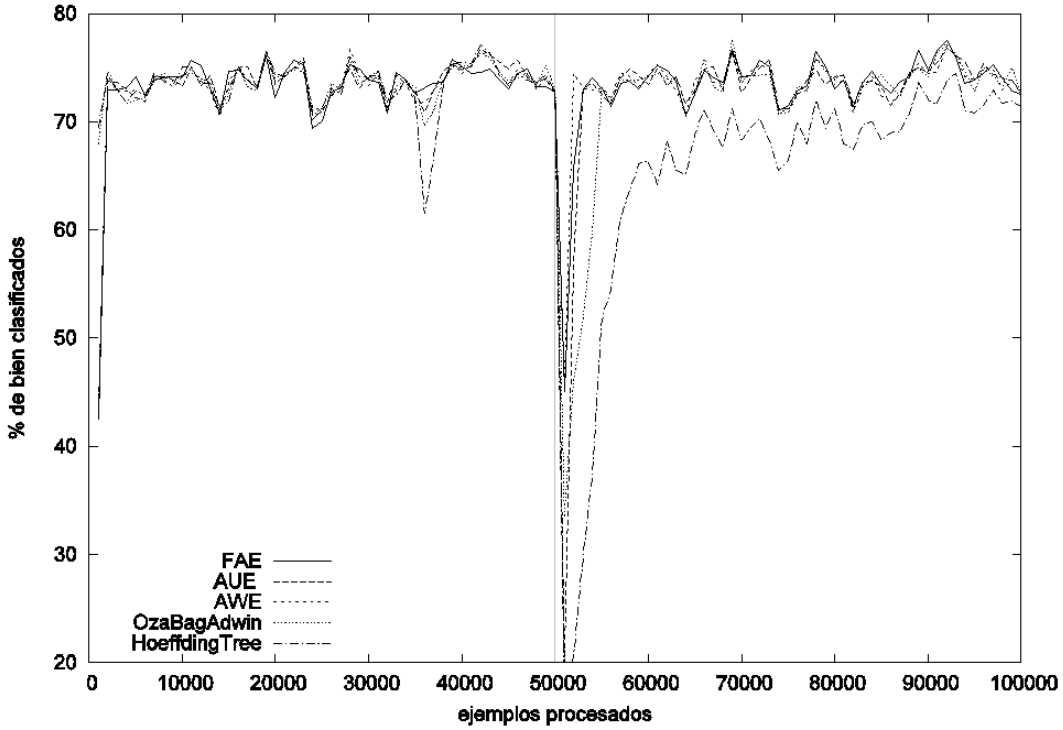


FIGURA 5.1: LED. 100000 EJEMPLOS. UN PUNTO DE CAMBIO: $T_0 = \mathbf{50000}$ **W = 0**.

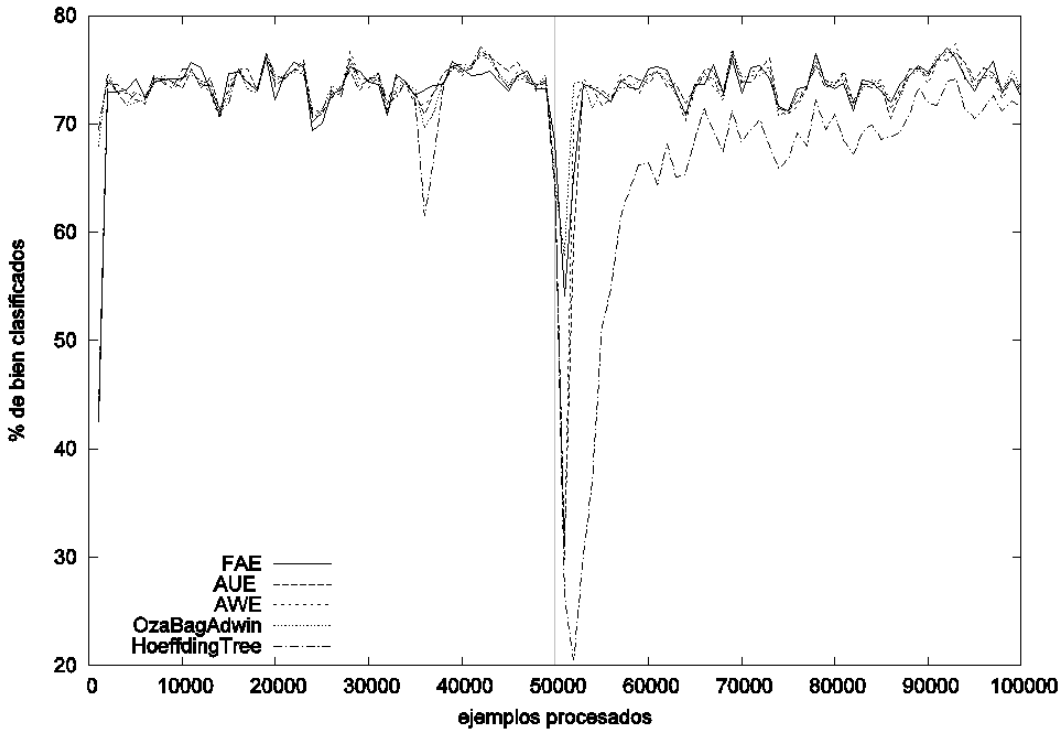


FIGURA 5.2: LED. 100000 EJEMPLOS. UN PUNTO DE CAMBIO: $T_0 = \mathbf{50000, W = 1000}$.

Resultados muy similares a los descritos anteriormente ocurren cuando se realizan pruebas sobre conjuntos de datos generados siguiendo una estructura SEA.

Los primero 50000 ejemplos fueron generados con la primera función de clasificación (f1 + f2 ≤ 9) y los otro 50000 ejemplos con la función de clasificación número cuatro (f1 + f2 ≤ 9.5).

TABLA 5.3: SEA, 100 000 EJEMPLOS. UN PUNTO DE CAMBIO: $\mathbf{t_0 = 50\ 000}$.

	PRECISIÓN MÁS BAJA ALREDEDOR DEL PUNTO DE CAMBIO (%).				PRECISIÓN. (% DE BIEN CLASIFICADOS)				TIEMPO (S)			
CLASIFICADORES	A	B	C	D	A	B	C	D	A	B	C	D
FAE	78,3	79,5	79,6	80,5	88,1	88,14	88,14	88,07	8,19	8,17	8,03	8,05
AUE	78,2	78,6	79,3	79,5	88,11	88,15	88,32	88,09	15,33	14,76	14,57	15,46
AWE	80,7	80,8	81,7	82	87,69	87,68	87,73	87,72	31,04	29,89	26,64	27,85
OZABAGADWIN	78,5	78,8	79,2	79,2	87,99	88,07	87,83	88,25	12,73	13,68	13,88	10,9
HOEFFDINGTREE	78,1	78,5	79	79,1	86,81	86,8	86,8	86,8	1,08	1,12	1,09	1,09

Columnas **A**, w = 0; columnas B, w = 100; columnas **C**, w = 500; columnas **D**, w = 1000;

Sobre ambos tipos de datos, los generados según los patrón LED y los generados según los patrones SEA, y sobre diferentes tipos de cambios, abruptos y

graduales, FAE muestra resultados promisorios en cuanto a: caída de precisión en el periodo de transición de un concepto a otro, tiempo de recuperación de la caída de precisión, precisión final y tiempo de ejecución.

CAMBIOS DE CONCEPTOS RECURRENTES.

En una segunda fase de los experimentos, para probar el comportamiento de los algoritmos sobre conjuntos de datos con cambios de conceptos recurrentes, se han construido 8 conjuntos de datos, cada uno con 100 000 ejemplos y con presencia de conceptos recurrentes. Igualmente se ha agregado un 10% de ruido a cada conjunto de datos.

Conjuntos de datos 1 y 2: Estructura LED, 3 puntos de cambios, cada 25000 ejemplos es cambiado el número de atributos con cambios (se sigue el siguiente esquema 1, 7, 1, 7 atributos con cambios). Para el conjunto de datos 1, w = 0 y para el conjunto de datos 2, w = 1000.

Conjuntos de datos 3 y 4: Estructura SEA, 3 puntos de cambios, cada 25000 ejemplos es cambiado el valor umbral α de la función de clasificación (se sigue el siguiente esquema 9.5, 9, 9.5, 9 valor umbral α). Para el conjunto de datos 3, w = 0 y para el conjunto de datos 4, w = 1000.

Conjuntos de datos 5 y 6: Estructura LED, 7 puntos de cambios, cada 12500 ejemplos es cambiado el número de atributos con cambios (se sigue el siguiente esquema 1, 3, 5, 7, 1, 3, 5, 7; atributos con cambios). Para el conjunto de datos 5, w = 0 y para el conjunto de datos 6, w = 1000.

Conjuntos de datos 7 y 8: Estructura SEA, 7 puntos de cambios, cada 12500 ejemplos es cambiado el valor umbral α de la función de clasificación (se sigue el siguiente esquema 7, 8, 9.5, 9, 7, 8, 9.5, 9 valor umbral α). Para el conjunto de datos 7, w = 0 y para el conjunto de datos 8, w = 1000.

Las tablas 5.4 y 5.5 muestran los resultados de evaluar cada uno de los algoritmos sobre los ocho conjuntos de datos definidos anteriormente. La tabla 5.4 contiene los resultados de cuando se trabaja con tres puntos de cambios mientras que la

tabla 5.5 contiene los resultados de cuando se trabaja con siete puntos de cambios. Cada tabla viene acompañada de cuatro figuras que la complementan. Las etiquetas concepto 1, concepto 2…, fueron adicionadas solamente a las figuras con el objetivo de llamar la atención visual cuando un concepto es recurrente. Conceptos iguales son etiquetados con iguales etiquetas.

TABLA 5.4: LED Y SEA, 100000 EJEMPLOS. TRES PUNTOS DE CAMBIOS: CADA 25000 EJEMPLOS.

	LED				SEA			
	PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)		TIEMPO (S)		PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)		TIEMPO (S)	
CLASIFICADORES	A	B	A	B	A	B	A	B
FAE	72,65	72,36	83,9	82,98	87,52	87,48	14,6	11,72
AUE	71,82	71,79	142,23	143,58	87,51	87,23	15,12	15,23
AWE	72,43	71,52	233,14	239,77	87,3	87,17	31	30,84
OZABAGADWIN	71,39	72,6	99,23	84,26	86,97	87,21	13,29	13,42
HOEFFDINGTREE	61,22	61,79	8,07	5,32	85,61	85,6	1,05	1,15

Columnas A, w = 0; columnas B, w = 1000.

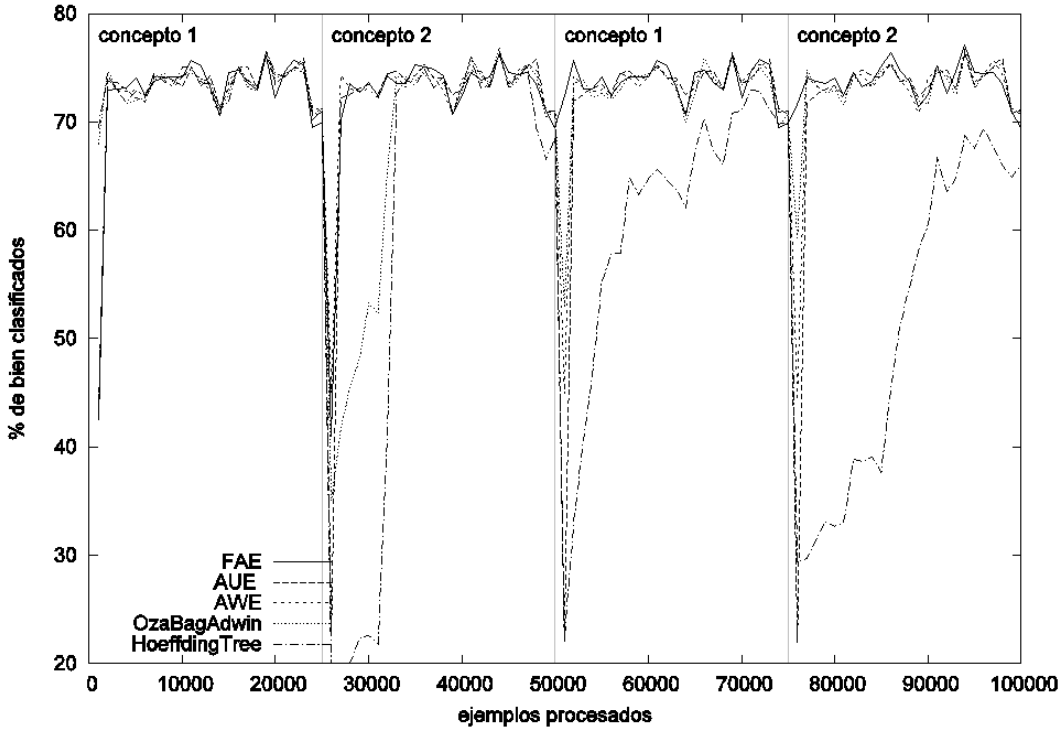


FIGURA 5.3: CONJUNTO DE DATOS 1. LED, TRES PUNTOS DE CAMBIOS: CADA 25000 EJEMPLOS, W = 0.

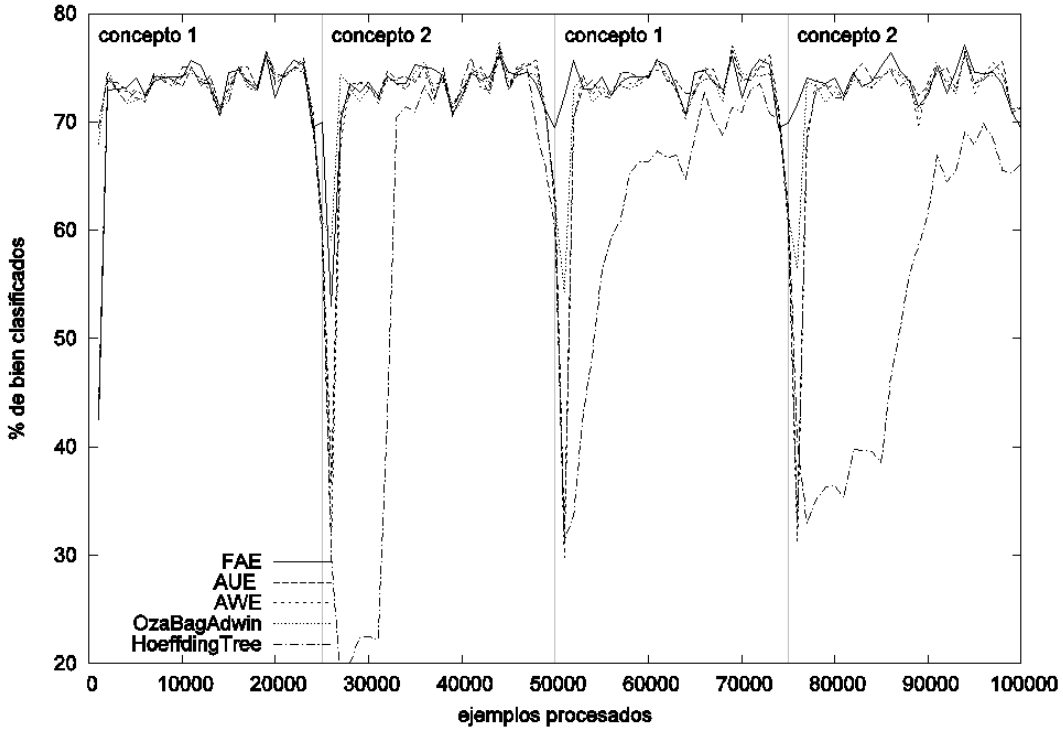


FIGURA 5.4: CONJUNTO DE DATOS 2. LED, TRES PUNTOS DE CAMBIOS: CADA 25000 EJEMPLOS, W = 1000.

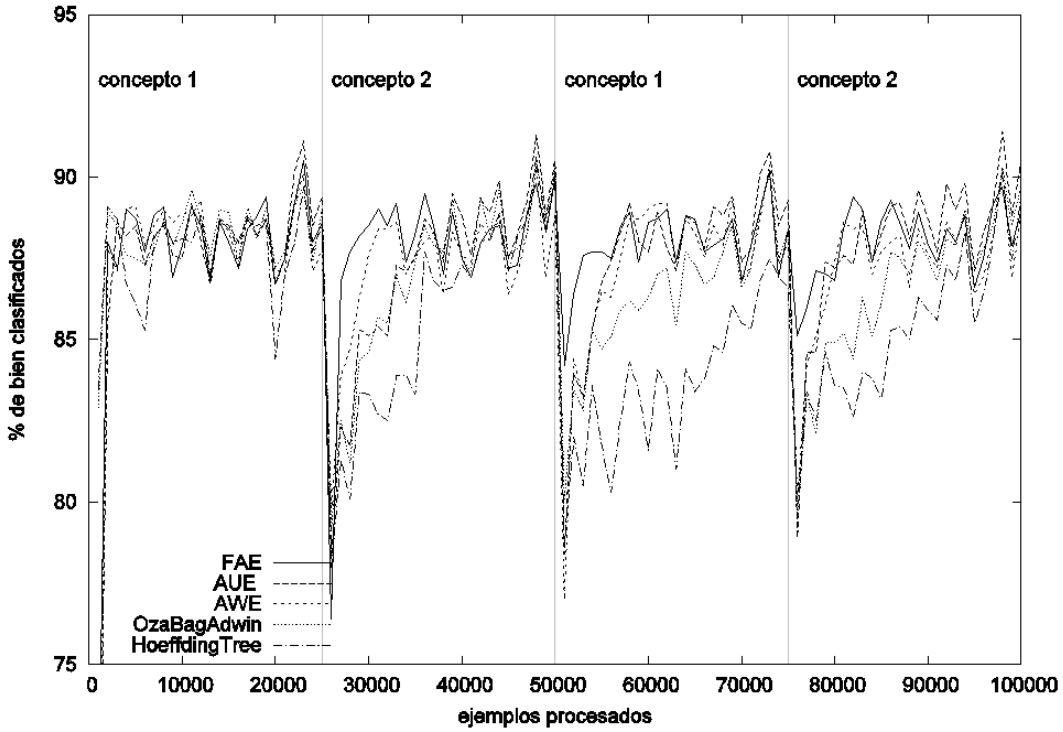


FIGURA 5.5: CONJUNTO DE DATOS 3. SEA, TRES PUNTOS DE CAMBIOS: CADA 25000 EJEMPLOS, W = 0.

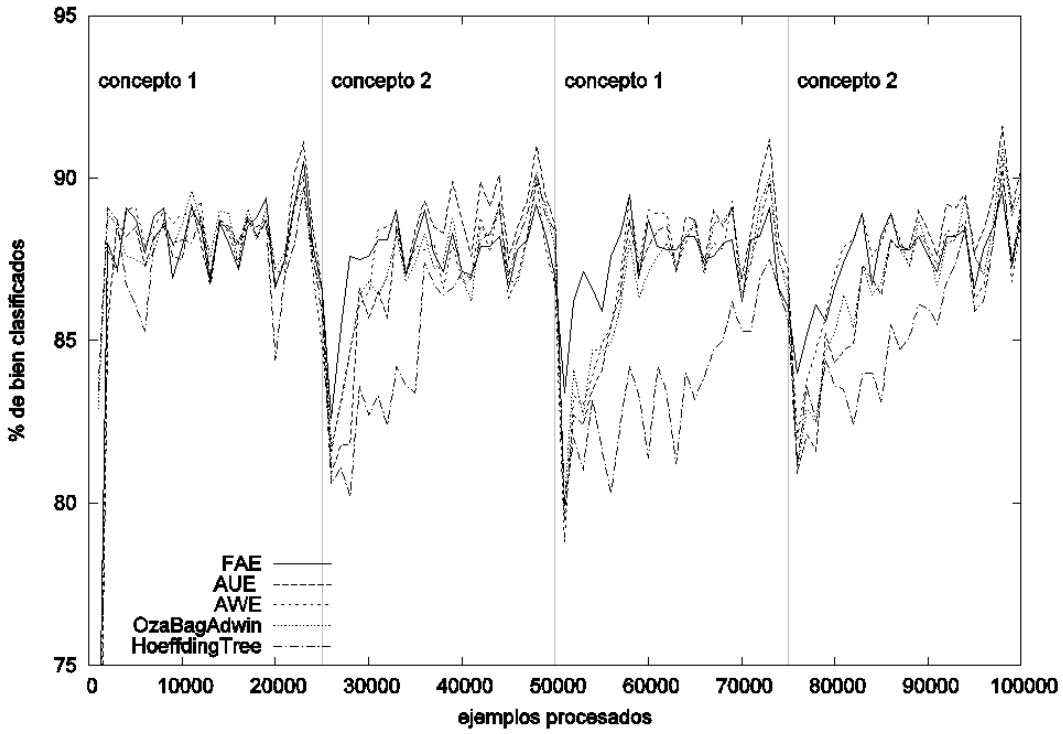


FIGURA 5.6: CONJUNTO DE DATOS 4. SEA, TRES PUNTOS DE CAMBIOS: CADA 25000 EJEMPLOS, W = 1000.

TABLA 5.5: LED, 100000 EJEMPLOS. SIETE PUNTOS DE CAMBIOS: CADA 12500 EJEMPLOS.

	LED				SEA			
	PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)		TIEMPO (S)		PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)		TIEMPO (S)	
CLASIFICADORES	A	B	A	B	A	B	A	B
FAE	72,27	71,95	74,69	81,32	87,46	86,95	11,37	10,67
AUE	71,52	71,59	133,54	120,84	85,99	86,4	15,09	14,96
AWE	70,38	70,06	236,58	210,79	86,58	86,5	30,62	30,17
OZABAGADWIN	71,68	72,08	86,67	91,37	86,13	86,31	12,99	13,23
HOEFFDINGTREE	62,65	62,71	8,5	7,92	84,31	84,35	1,09	1,19

De acuerdo con los resultados mostrados en las tablas 5.4 y 5.5 es interesante resaltar que:

El algoritmo HoeffdingTree (este no es un multiclasificador) siempre alcanza mejores resultados que el resto de los algoritmos en cuanto a tiempo de ejecución, sin embargo en cuanto a la precisión final siempre obtiene los peores resultados. En contraste con este resultado, el algoritmo AWE siempre obtiene los peores

resultados en cuanto a tiempo de ejecución sin embargo sus valores de precisión final son comparablemente buenos, los cuales son incluidos entre los mejores resultados en varios experimentos. También los algoritmo AUE y OzaBagAdwin obtienen resultados comparablemente buenos en cuanto a los valores de precisión final. El algoritmo OzaBagAdwin alcanza resultados muy similares a los del algoritmo FAE en cuanto a tiempo de ejecución, aunque siempre sus valores son más bajos.

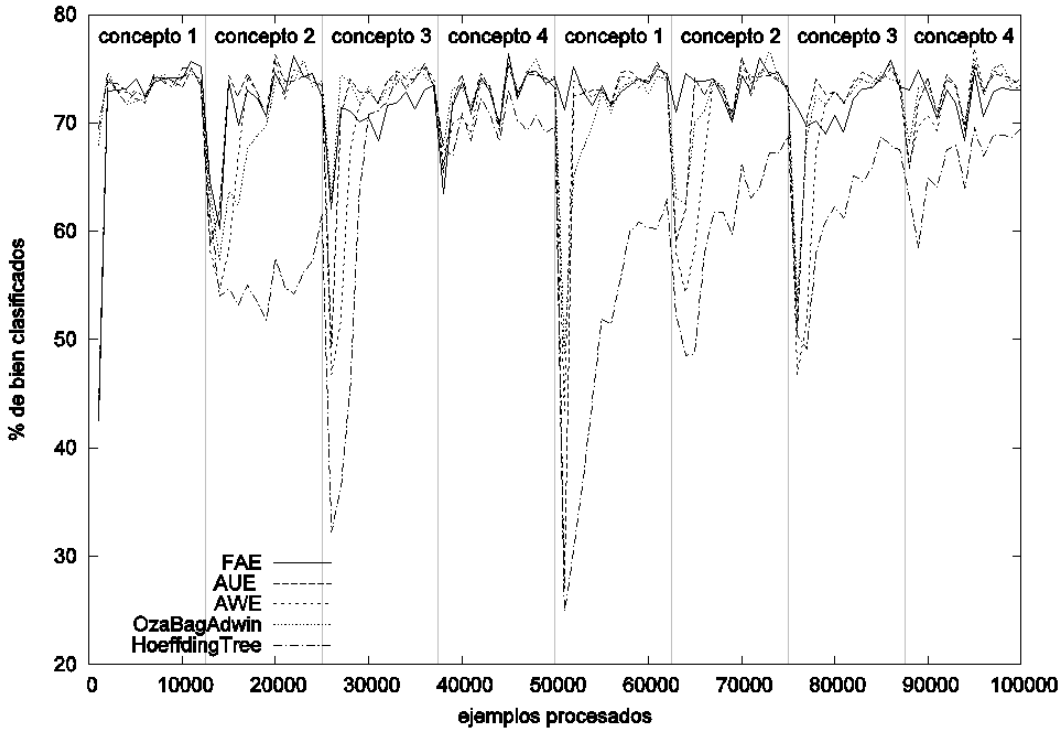


FIGURA 5.7: CONJUNTO DE DATOS 5, LED, SIETE PUNTOS DE CAMBIOS: CADA 12500 EJEMPLOS, W = 0

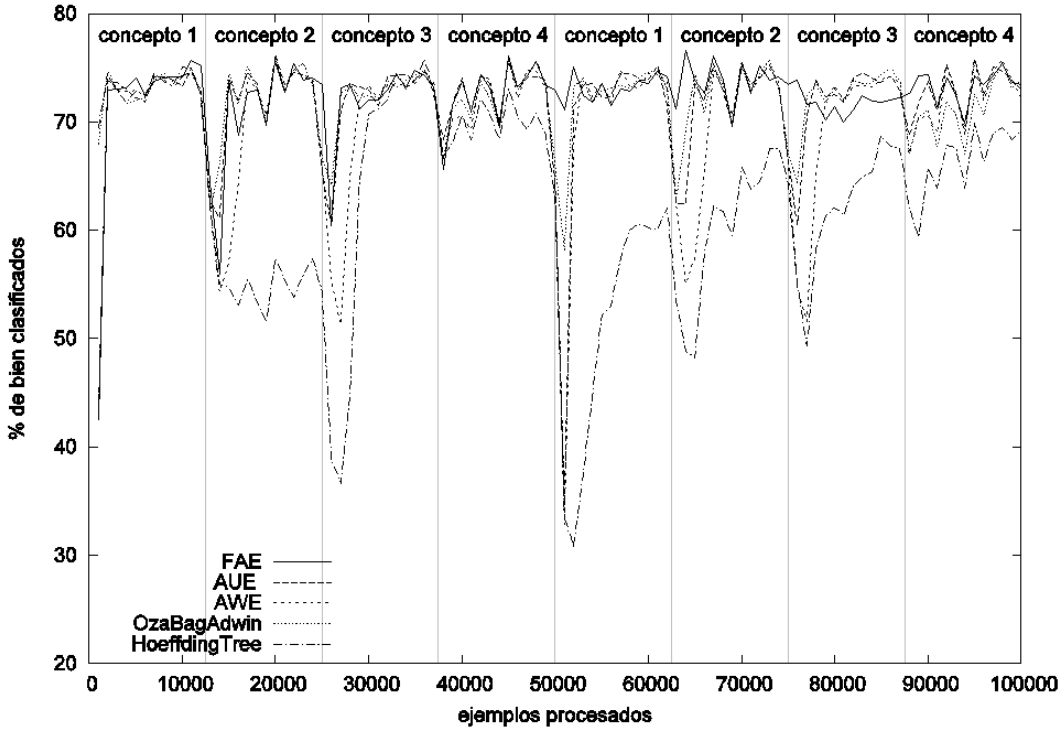


FIGURA 5.8: CONJUNTO DE DATOS 6. LED, SIETE PUNTOS DE CAMBIOS: CADA 12500 EJEMPLOS, W = 1000.

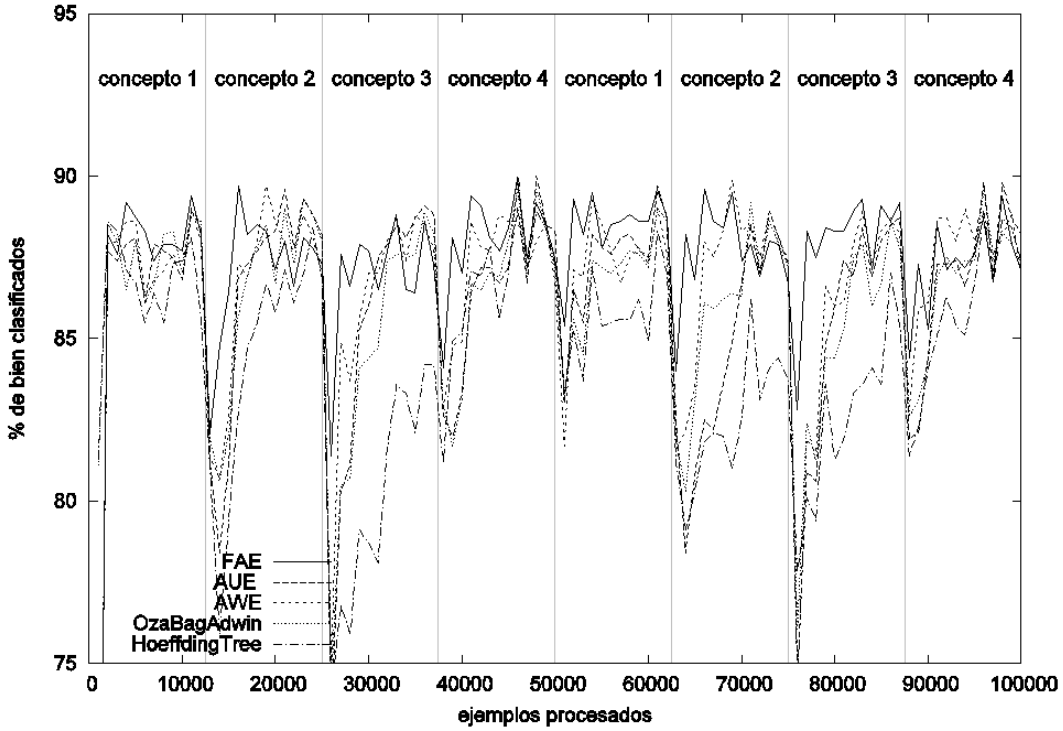


FIGURA 5.9: CONJUNTOS DE DATOS 7. SEA, SIETE PUNTOS DE CAMBIOS: CADA 12500 EJEMPLOS,W = 0.

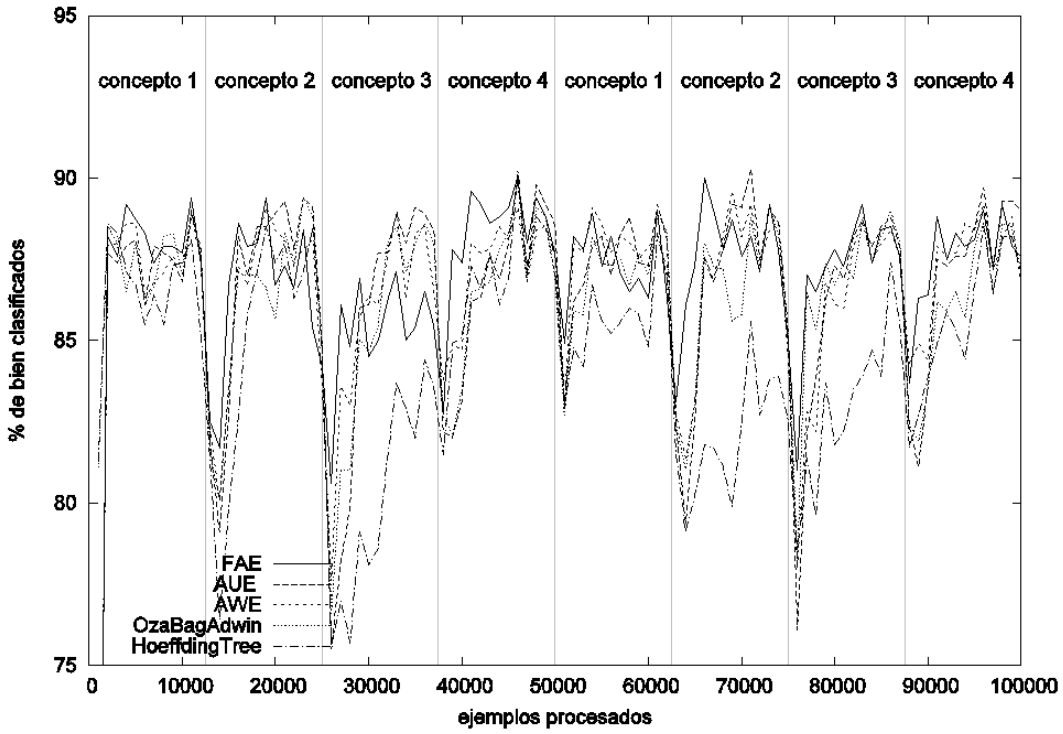


FIGURA 5.10: CONJUNTO DE DATOS 8. SEA, SIETE PUNTOS DE CAMBIOS: CADA 12500 EJEMPLOS, W = 1000.

Como se puede ver en las tablas 5.4 y 5.5, el algoritmo FAE siempre reporta resultados entres los dos mejores valores de precisión y de tiempo de ejecución. Estos valores muestran que la nueva propuesta alcanza mejores resultados que el resto de los algoritmos sobre los conjuntos de datos con las características propuestas (cambios de conceptos, abruptos o moderados, y conceptos recurrentes).

Se considera importante hacer notar que en cada figura a partir de la mitad de los ejemplos procesados, cuando conceptos analizados previamente reaparecen (conceptos recurrentes), el algoritmo FAE prácticamente no muestra caídas en los valores de precisión en comparación con el resto de los algoritmos. Estas diferencias son más marcadas cuando trabajamos sobre conjuntos de datos con patrones LED. El algoritmo FAE incluye en su diseño mecanismos para tratar conceptos recurrentes de aquí sus buenos resultados.

De acuerdo con los resultados de los experimentos, podemos concluir que FAE es un algoritmo capaz de manipular cambios de conceptos, tanto abruptos como moderados, y además conceptos recurrentes.

CONJUNTO DE DATOS REAL.

Los conjuntos de datos reales seleccionados para los experimentos de esta sección han sido muy utilizados en varios estudios relacionados con los cambios de conceptos.

En primer lugar tenemos a la base de datos reales llamada “electricity”, también conocida como ELEC2, que fue propuesta por Harries [84] en el año 1999. Desde entonces fue muy popularizada por Gama [34] y otros reconocidos investigadores, por lo que ha sido ampliamente utilizados en varios trabajos [60, 116, 69, 115].

En ELEC2, el objetivo es predecir si el precio de la electricidad subirá o no (dos valores en el conjunto clase) teniendo en cuenta cinco atributos numéricos: (1) día de la semana, (2) período de treinta minutos del día, (3) la demanda de electricidad en New South Wales, (4) la demanda de electricidad en Victoria y (5) la cantidad de electricidad a ser transferida entre estos dos distritos de Australia.

En segundo lugar, como es ya conocido, el filtrado de correos spam es un tema muy abordado por parte de la comunidad científica internacional. Los creadores de estos correos tratan de evitar los filtros que han identificados sus correos, debido a esto, este tipo de filtros debe ser contantemente actualizados para lograr un trabajo satisfactorio [11, 83, 117]. Uno de los conjunto de datos ampliamente conocido es el llamado spam_corpus, este contiene tanto correos spam como correos genuinos [68].

Para estas dos bases de datos reales no está comprobada la presencia de cambios de conceptos ni de posibles tipos de estos. Los ejemplos son procesados siguiendo un orden temporal y simulando un flujo de datos.

Como se puede ver en la tabla 5.6 y en las dos figuras siguientes, el algoritmo FAE reporta resultados entre los dos mejores valores de precisión una vez más. Sin embargo es importante resaltar que el algoritmo OzaBagAdwin alcanzó los mejores resultados en el trabajo sobre ambas bases reales.

TABLA 5.6: BASE DE DATOS REALES: "ELECTRICITY", 45312 EJEMPLOS Y SPAM_CORPUS 9324 EJEMPLOS.

	ELECTRICITY		SPAM_CORPUS	
CLASIFICADORES	**PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)**	**TIEMPO (S)**	**PRECISIÓN FINAL (% DE BIEN CLASIFICADOS)**	**TIEMPO (S)**
FAE	**82,4**	9,34	90,97	21,31
AUE	77,7	10,62	75,33	32,56
AWE	71,13	12,89	75,95	38,11
OZABAGADWIN	**84,81**	**7,77**	91,26	27,36
HOEFFDINGTREE	78,92	**1,53**	90,39	4,1

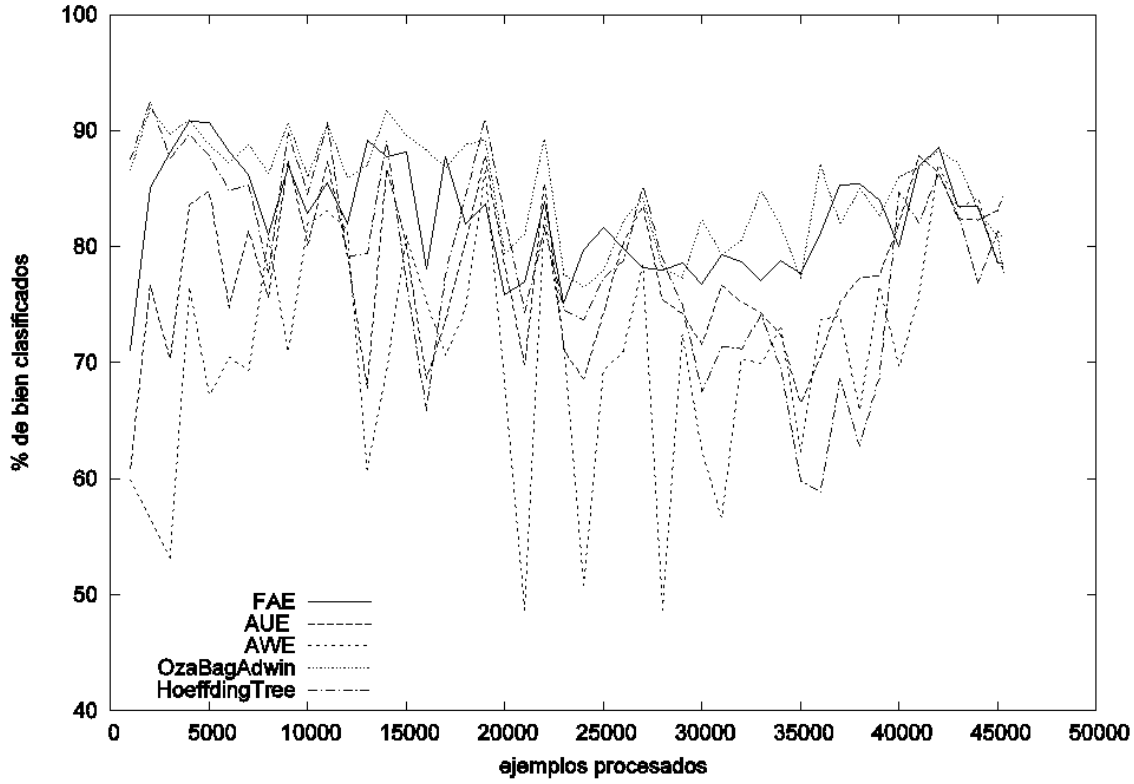


FIGURA 5.11: BASE DE DATOS REAL: "ELECTRICITY", 45312 EJEMPLOS.

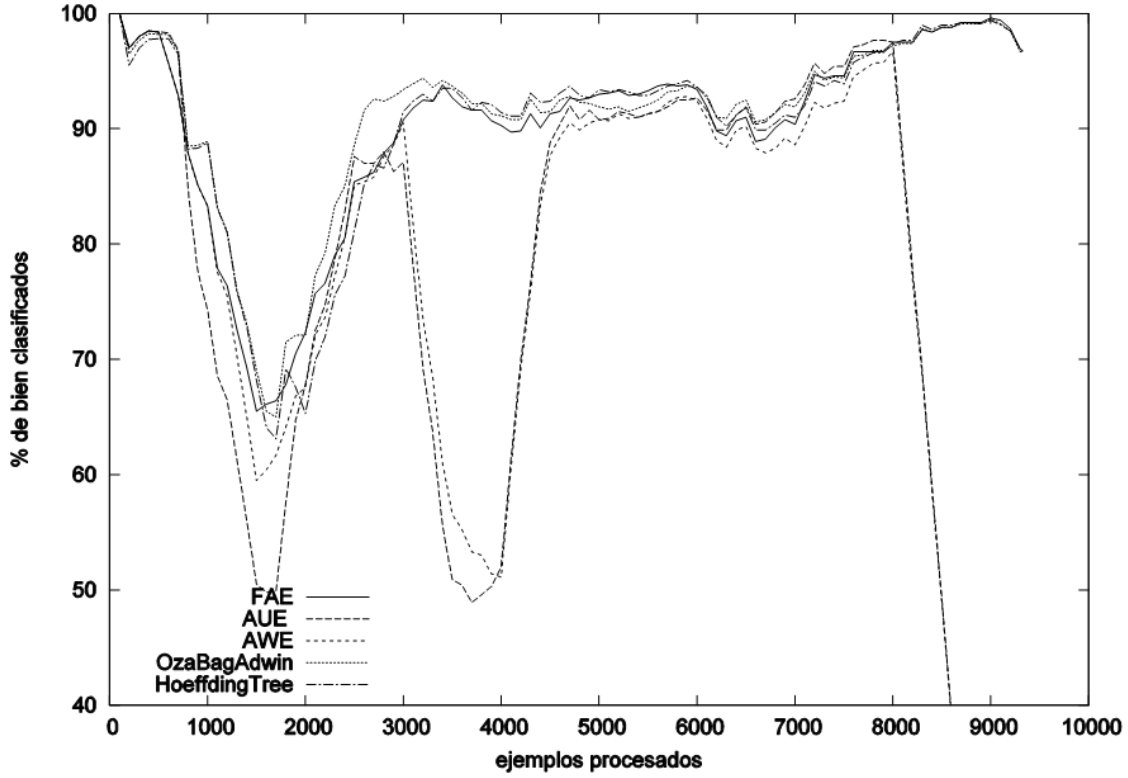


FIGURA 5.12: BASE DE DATOS REAL "SPAM_CORPUS", 9324 EJEMPLOS.

CONCLUSIONES DEL CAPÍTULO V.

A partir de la propuesta de algoritmo presentada en el capítulo anterior, y realizando las siguientes modificaciones al mismo:

- Se combina con el sistema multiclasificador un detector de cambio. Este detector se utiliza para determinar cuándo es necesario agregar un nuevo clasificador. De esta forma, no siempre se agregan nuevos clasificadores básicos, sino sólo cuando ha sido detectado un posible cambio de concepto. Este mecanismo contribuye al tratamiento de conceptos recurrentes.
- El conjunto de clasificadores básicos ahora está dividido en: clasificadores activos y clasificadores inactivos. Los clasificadores activos se utilizan para clasificar en un instante de tiempo y los inactivos constituyen un almacén de antiguos conceptos que podría reaparecer en cualquier momento.

- El umbral de eliminación ahora se utiliza para activar o desactivar clasificadores básicos según su comportamiento frente a los datos más actuales.

Se obtuvo un nuevo algoritmo multiclasificador, llamado Multiclasificador de Adaptación Rápida (FAE, del inglés *Fast Adapting ensemble*), diseñado para trabajar sobre flujos de datos y para adaptarse de forma rápida a cambios de conceptos graduales, abruptos y recurrentes.

El algoritmo FAE ha sido implementado bajo los requerimientos del entorno de trabajo MOA debido a sus características favorables para la experimentación en el contexto del aprendizaje incremental sobre flujos de datos en presencia de cambios de conceptos.

Se diseñó un sistema de experimentación donde se tuvieron en cuenta, bajo simulación controlada, los distintos tipos de cambios de conceptos, tanto graduales, abruptos como recurrentes; para esto se construyeron varios conjuntos de datos utilizando los generadores de datos artificiales LED y SEA. Además, se experimentó con dos bases de datos reales, “electricity” y “Spam_Corpus”.

El algoritmo FAE alcanzó resultados promisorios en las pruebas, en comparación con algoritmos bien conocidos implementados también en el entorno de trabajo MOA, teniendo en cuenta los parámetros: Precisión, tiempo de ejecución, comportamiento en el periodo de transición de un concepto a otro, tiempo de recuperación después de un cambio de concepto.

CONCLUSIONES GENERALES.

A lo largo de la presente investigación y de forma ordenada se han ido solucionando las tareas propuestas, la cuales guiaron el trabajo hasta los resultados finales:

Como primer paso, se realizó una descripción de los principales conceptos en las áreas de investigación de descubrimiento de conocimiento, minería de datos y aprendizaje automático; lo que permitió enmarcar la investigación en esta última, la cual estudia los algoritmos de clasificación con aprendizaje incremental diseñados para el trabajo sobre grandes flujos de datos en presencia de diferentes tipos de cambios de conceptos.

En segundo, se realizó un estudio de los principales modelos de aprendizaje que han sido adaptados para trabajar de forma incremental sobre grandes flujos de datos en presencia de cambios de conceptos. Esto permitió determinar que en los últimos años ha existido un notable aumento en la cantidad de investigaciones científicas relacionadas con los sistemas multiclasificadores vinculados a la minería de grandes flujos de datos en presencia de cambios de conceptos; sin embargo, se constató que el tratamiento de conceptos recurrentes, que emplea este tipo de modelos, no ha experimentado el mismo aumento de popularidad.

Posteriormente se realizó un análisis de las principales metodologías, herramientas de software y conjuntos de datos empleados en la evaluación y comparación de algoritmos incrementales diseñados para el trabajo sobre grandes flujos de datos en presencia de cambios de conceptos. Esto permitió distinguir varios parámetros imprescindibles en la evaluación de los algoritmos tratados: Precisión de los algoritmos, tiempo de ejecución, memoria utilizada, comportamiento de los algoritmos en el periodo de transición de un concepto a otro, tiempo de recuperación después de un cambio de concepto, entre otros.

Ya cimentadas las bases de la investigación, se tomó como punto de partida para el diseño de un nuevo algoritmo de multiclasificación a los algoritmos MultiCIDIM-

DS y MultiCIDIM-DS-CFC. Después de un estudio de sus principales características, se determinó que estos algoritmos están perfectamente adaptados para el trabajo sobre grandes flujos de datos; sin embargo, a pesar de contar con los mecanismos más frecuentes de adaptación para el trabajo incremental (mecanismos para adicionar, actualizar y eliminar sus clasificadores básicos), estos tiene ciertas dificultades para adaptarse de forma rápida a los cambios de conceptos (sobre todo a cambios abruptos) y no cuentan con mecanismos para el tratamiento de conceptos recurrentes.

De esta forma, se construyó un nuevo algoritmo multiclasificador, llamado "Multiclasificador de Adaptación Rápida" (FAE, del inglés *Fast Adapting ensemble*), diseñado para trabajar sobre flujos de datos y para adaptarse de forma rápida a cambios de conceptos graduales, abruptos y recurrentes. Entre las principales características del nuevo algoritmo están:

- Divide el flujo de datos de entrenamiento en bloques de igual tamaño para crear los clasificadores básicos.
- Cada clasificador básico tiene asociado un peso que se actualiza cada cierto número de ejemplos, sin tener que esperar a que el bloque de entrenamiento esté completo.
- Utiliza un sistema de votación por mayoría ponderado.
- Utiliza una nueva fórmula que le permite aumentar o disminuir el valor de cada peso asociado a cada clasificador básico según la precisión obtenida por estos últimos sobre los datos más actuales.
- Divide los clasificadores básicos en activo e inactivos. Utiliza los activos para clasificar en cada instante de tiempo y los inactivos como almacén de conceptos antiguos.
- Tiene diseñado un mecanismo para activar o inactivar clasificadores básicos. Los clasificadores inactivos se activan de forma muy rápida si reaparece el que concepto que ellos representan.

FAE se implementó sobre el entorno de trabajo MOA debido a las características favorables de este último para el diseño experimentos en el contexto de esta

investigación. Se diseñó un sistema de experimentación donde se tuvieron en cuenta, bajo simulación controlada, los distintos tipos de cambios de conceptos, tanto graduales, abruptos como recurrentes; para esto se construyeron varios conjuntos de datos utilizando los generadores de datos artificiales LED y SEA. Además, se experimentó con dos bases de datos reales, “electricity” y “Spam_Corpus”.

El nuevo algoritmo alcanzó resultados promisorios en las pruebas, en comparación con algoritmos bien conocidos implementados también en el entorno de trabajo MOA, teniendo en cuenta los parámetros: Precisión de los algoritmos, tiempo de ejecución, comportamiento de los algoritmos en el periodo de transición de un concepto a otro y tiempo de recuperación después de un cambio de concepto.

TRABAJOS FUTUROS.

Como posibles trabajos futuros se han valorados los siguientes aspectos:

Trabajar en la automatización de algunos parámetros de entrada para que la ejecución de algoritmo no dependa de un valor inicial asignados a estos. Por ejemplo, el parámetro utilizado como umbral de activación-desactivación en el algoritmo FAE podría reajustarse automáticamente cada vez que se actualizan los valores (peso, estado) asociados a los clasificadores básicos. Un segundo parámetro que podría no necesitar un valor inicial es la cantidad máxima de clasificadores básicos que admite el multiclasificador (max). Este parámetro se podría calcular o ajustar dependiendo de la memoria de que se dispone en el entorno de trabajo.

Un segundo aspecto a valorar sería tratar de encontrar una forma eficiente para reagrupar los clasificadores básicos por conceptos, según las compatibilidades que vayan demostrando a la hora de evaluar los datos más recientes a lo largo del tiempo.

BIBLIOGRAFÍA.

[1] Ruiz, R. Heurísticas de selección de atributos para datos de gran dimensionalidad. Departamento de Lenguajes y Sistemas Informáticos. Sevilla, Universidad de Sevilla, 2006.

[2] Caballero, Y. Aplicación de la Teoría de los Conjuntos Aproximados en el Preprocesamiento de los Conjuntos de Entrenamiento para Algoritmos de Aprendizaje Automatizado. Departamento de Ciencias de la Computación. Santa Clara, Universidad Central "Marta Abreu" de la Villas, 2007.

[3] Michalsky, R. y G. Tecuci. Machine Learning: A Multistrategy Approach. EE.UU, Morgan Kauffinan, 1994.

[4] Hernández, J., M. Ramírez y C. Ferri. Introducción a la minería de datos. Prentice Hall, 2004.

[5] Ferrer F.J., Aguilar J. S.: Minería de Data Streams: Conceptos y Principales Técnicas. Universidad de sevilla. 2005.

[6] Bifet, A. Adaptive Learning and Mining for Data Streams and Frequent Patterns. Tesis de doctorado, Universitat Politècnica de Catalunya, 2009.

[7] Schlimmer, J. y D. Fisher. A Case Study of Incremental Concept Induction. En Proc. 5th National Conf. on Artificial Intelligence, págs. 495–501, 1986.

[8] Bach, S. y M. Maloof. A Bayesian Approach to Concept Drift. En Advances in Neural Information Processing Systems 23, págs. 127–135, 2010.

[9] Klinkenberg, R. Learning Drifting Concepts: Example Selection vs. Example Weighting. Intelligent Data Analysis, 8(3):281–300, 2004.

[10] Kubat, M. y G. Widmer. Adapting to Drift in Continuous Domains. Reporte técnico ÖFAI-TR-94-27, Austrian Research Institute for Artificial Intelligence, Vienna, 1995.

[11] Cunningham, P., N. Nowlan, S. Delany, y M. Haahr. A Case-Based Approach to Spam Filtering that Can Track Concept Drift. En Proc. Workshop on Long-Lived CBR Systems (in ICCBR-2003), 2003.

[12] Ramos, G., J. del Campo y otros. Inducción en árboles de decisión con CIDIM: nuevos enfoques. Actas del III Taller Nacional de Minería de Datos y Aprendizaje (TAMIDA2005), Málaga, España, E.T.S. Ingeniería Informática. Universidad de Málaga, 2005.

[13] del Campo, J. Nuevos Enfoques en el Aprendizaje Incremental. Departamento de Lenguajes y Ciencia de la Computación. Málaga, Universidad de Málaga, 2007.

[14] Street, W. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. in 7th International Conference on Knowledge Discovery and Data Mining. New York City, NY, 2001.

[15] Rushing, J. y otros A coverage based ensemble algorithm (CBEA) for streaming data. in 16th IEEE International Conference on Tools with Artificial Intelligence. IEEE Computer Society, Washington, 2004.

[16] Fayyad, U. M., G. Piatetsky-Shapiro y otros Advances In knowledge discovery and data mining. From Capítulo: data mining to knowledge discovery: An overview. , AAAI Press. 1996

[17] Han, J. y M. Kamber. Data Mining: Concepts and Techniques. Data Mining: Concepts and Techniques., Morgan Kaufmann, 2000: 3-47.

[18] Frawley, W., G. Piatesky-Shapiro y otros. Knowledge Discovery in Databases: An Overview, AI Magazine, 1992.

[19] Witten, I. H. y E. Frank, Eds. DATA MINING Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.

[20] Dietterich, T. G. NATURE ENCYCLOPEDIA OF COGNITIVE SCIENCE, Machine learning. Macmillan. 2003.

[21] Russell, S. y P. Norvig. Inteligencia artificial: Un enfoque moderno. Mexico, Prentice Hall Hispanoamericana, 1996.

[22] Mitchell, T. Machine Learning, McGraw-Hill, 1997.

[23] Orallo, J., M. Quintana y otros. Introducción a la Minería de Datos, Pentice-Hall, 2004.

[24] Mannila, H., H. Toivonen y otros. "Efficient Algorithm for discovering Association Rules." Proceedings of the AAAI Workshop on Knowledge Discovery in Databases (KDD): 181-192, 1994.

[25] Rumelhart, D., G. Hinton y otros. Learning Internal Representations By error propagation. Parallel Distributed Processing: Explorations in the Microstructureof Cognition. M. Press. 1: 318-362, 1986.

[26] Breiman, L., J. H. Friedman y otros "Classification and Regression Trees." Wadsworth International Group, 1984.

[27] Quinlan, J. R. C4.5: PROGRAMS FOR MACHINE LEARNING. Morgan Kaufmann. 1993.

[28] Utgoff, P. (1989). Incremental induction of decision trees. Machine Learning.

[29] Ye, N. The Handbook of Data Mining. Lawrence, Lawrence Erlbaum Associates, 2003.

[30] Jin, R. y G. Agrawa. Efficient Decision Tree Construction on Streaming Data. in 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, DC, 2003.

[31] Widmer, G. y M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning, 23(1):69–101, 1996.

[32] Ferrer, F., J. Aguilar, y J. Riquelme. Incremental Rule Learning and Border Examples Selection from Numerical Data Streams. Journal of Universal Computer Science, 11(8):1426–1439, 2005.

[33] Minku, L., A. White, y X. Yao. The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift. IEEE Trans. on Knowledge and Data Engineering, 22:730–742, 2010.

[34] Gama, J., P. Medas, G. Castillo, y P. Rodrigues. Learning with Drift Detection. En Proc. 20th Brazilian Symposium on Artificial Intelligence, págs. 286–295, 2004.

[35] Wang, H., W. Fan, P. Yu, y J. Han. Mining Concept-Drifting Data Streams Using Ensemble Classifiers. in 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, DC, 2003.

[36] Fan, W. y otros. Active Mining of Data Streams. in SIAM Int'l Conf on Data Mining. 2004.

[37] Widmer, G. y M. Kubat. Effective learning in dynamic environments by explicit context tracking. in 6th European Conf. on Machine Learning ECML-1993. 227-243: Springer-Verlag, 1993.

[38] Stanley, K. Learning concept drift with a committee of decision trees. Department of Computer Sciences, at Austin, USA, University of Texas, 2003.

[39] Salganicoff, M. Tolerating concept and sampling shift in lazy learning using prediction error context switching. AI Review Special Issuse on Lazy Learning. 11(1-5): p. 133-155, 1997.

[40] Harries, M., C. Sammut, y K. Horn, Extracting hidden context. Machine Learning. 32(2): p. 101-126, 1998.

[41] Klinkenberg, R., Learning drifting concepts: example selection vs. example weighting. Intelligent Data Analysis Journal, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, 2004.

[42] Hunt, E., J. Marin y otros. Experiments in Induction, Academic Press, 1966.

[43] Domingos, P. y G. Hulten. Mining High-Speed Data Streams. En Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, págs. 71–80, 2000.

[44] Rutkowski, L., L. Pietruczuk, P. Duda, y M. Jaworski. Decision trees for mining data streams based on the mcdiarmid's bound. IEEE Tran- sactions on Knowledge and Data Engineering, 25(6):1272–1279, 2013.

[45] Ramos, G., J. del Campo, y R. Morales. Induction of Decision Trees Using an Internal Control of Induction. En Proc. 8th Int. Conf. on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems, págs. 795–803, 2005.

[46] Hulten, G., L. Spencer, y P. Domingos. Mining Time-Changing Data Streams. En Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, págs. 97–106, 2001.

[47] Natwichai, J. y X. Li. Knowledge Maintenance on Data Streams with Concept Drifting. En Proc. 1st Int. Symposium on Computational and Information Science, págs. 705–710, 2004.

[48] Bifet, A., G. Holmes, B. Pfahringer, y E. Frank. Fast perceptron decision tree learning from evolving data streams. En Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part II, PAKDD'10, págs. 299–310, Berlin, Heidelberg. Springer-Verlag, 2010.

[49] Wozniak, M. A hybrid decision tree training method using data streams. Knowledge and Information Systems, 29(2):335–347, November, 2011.

[50] Kosina, P. y J. Gama. Handling time changing data with adaptive very fast decision rules. En Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I, ECML PKDD'12, págs. 827–842, Berlin, Heidelberg. Springer- Verlag, 2012.

[51] Deckert, M. Incremental rule-based learners for handling concept drift: an overview. Foundations of Computing and Decision Sciences, 38(1):35–65, 2013.

[52] Schlimmer, J. y R. Granger. Incremental Learning from Noisy Data. Machine Learning, 1(3):317–354, 1986.

[53] Maloof, M.Incremental Rule Learning with Partial Instance Memory for Changing Concepts. En Proc. Int. Joint Conf. on Neural Networks, volumen 4, págs. 2764 – 2769 vol.4, 2003.

[54] Sculley, D. y Gabriel M. Wachman. Relaxed online svms for spam filtering. En Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07, págs. 415–422, New York, NY, USA,. ACM, 2007.

[55] Syed, N., H. Liu, S. Huan, L. Kah, y K. Sung. Handling Concept Drifts in Incremental Learning with Support Vector Machines. En Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, págs. 317–321, 1999.

[56] Dries, A. y U. Rückert. Adaptive Concept Drift Detection. Statistical Analysis and Data Mining, 2(5–6):311–327, 2009.

[57] Carpenter, G., S. Grossberg, N. Markuzon, J. Reynolds, y D. Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. Neural Networks, IEEE Transactions on, 3(5):698–713, 1992.

[58] Polikar, R., L. Udpa, S. Udpa, y V. Honavar. Learn++: An Incremental Learning Algorithm for Supervised Neural Networks. IEEE Trans. on Systems,Man, and Cybernetics, Part C: Applications and Reviews, 31(4):497 –508, 2001.

[59] Rakitianskaia, A. y A. Engelbrecht. Training feedforward neural networks with dynamic particle swarm optimisation. Swarm Intelligence, 6(3):233–270, 2012.

[60] Baena, M., J. del Campo, R. Fidalgo, A. Bifet, R. Gavaldà, y R. Morales. Early Drift Detection Method. En 4th Int. Workshop on Knowledge Discovery from Data Streams, 2006.

[61] Frías, I., J. del Campo, G. Ramos, R. Morales, A. Ortiz, y Y. Caballero, “Online and non-parametric drift detection methods based on Hoeffding’s bound,” IEEE Transactions on Knowledge and Data Engineering, 2014. DOI 10.1109/TKDE.2014.2345382.

[62] Bifet, A. Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams, volumen 207. IOS Press, 2010.

[63] Bartlett, P. y D. Helmbold. Learning Changing Problems. Reporte técnico, Australian National University, 1996.

[64] Žliobait, I. Combining Time and Space Similarity for Small Size Learning under Concept Drift. En Proc. 18th Int. Symposium on Foundations of Intelligent Systems, págs. 412–421, 2009.

[65] Kolter, J. y M.A. Maloof. Using Additive Expert Ensembles to Cope with Concept Drift. En Proc. 22nd Int. Conf. on Machine Learning, págs. 449–456, 2005.

[66] Zhao, P., R. Jin, T. Yang, y S. Hoi. Online auc maximization. En Lise Getoor y Tobias Scheffer, editores, Proceedings of the 28th International Conference on Machine Learning (ICML-11), págs. 233–240, New York, NY, USA, 2011. ACM.

[67] Angiulli, F. y F. Fassetti. Distance-based Outlier Queries in Data Streams: The Novel Task and Algorithms. Data Mining and Knowledge Discovery, 20(2):290–324, 2010.

[68] Katakis I., G. Tsoumakas, y I. Vlahavas. Tracking Recurring Contexts using Ensemble Classifiers: an Application to Email Filtering. Knowledge and Information Systems, 22(3):371–391, 2010.

[69] Brzezinski, D. Mining Data Streams with Concept Drift. Tesis de maestría, Poznan University of Technology, 2010.

[70] Ramamurthy, S. y R. Bhatnagar. Tracking Recurrent Concept Drift in Streaming Data Using Ensemble Classifiers. En Proc. 6th Int. Conf. on Machine Learning and Applications, págs. 404–409, 2007.

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

[72] Gama, J. y G. Castillo. Learning with Local Drift Detection. En Proc. 2nd Int. Conf. on Advanced Data Mining and Applications, págs. 42–55, 2006.

[73] Street, W. y Y. Kim. A Streaming Ensemble Elgorithm (SEA) for Large-Scale Classification. En Proc. 7th ACMSIGKDD Int. Conf. on Knowledge Discovery and Data Mining, págs. 377–382, 2001.

[74] Gama, J., P. Rodrigues, y G. Castillo. Evaluating Algorithms that Learn from Data Streams. En Proc. 2009 ACM Symposium on Applied Computing, págs. 1496–1500, 2009.

[75] Kolter, J. y M. Maloof. Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. in 3rd International IEEE Conference on Data Mining.. Melbourne, FL, 2003.

[76] Frank, A. y A. Asuncion. UCI Machine Learning Repository, 2010.

[77] Bifet, A., G. Holmes, B. Pfahringer, y R. Gavaldà. New Ensemble Methods for Evolving Data Streams. En Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, págs. 139–148, 2009.

[78] Narasimhamurthy, A. y Ludmila I. Kuncheva. A Framework for Generating Data to Simulate Changing Environments. En Proc. 25th IASTED Int.Multi-Conf.: artificial intelligence and applications (AIAP'07), págs. 384–389, 2007.

[79] Gama, J. y P. Kosina. Tracking Recurring Concepts with Meta-learners. En Proc. 14th Portuguese Conf. on Artificial Intelligence, págs. 423–434, 2009.

[80] Agrawal, R., T. Imielinski, y A. Swami. Database Mining: A Performance Perspective. IEEE Trans. on Knowledge and Data Engineer, 5(6):914–925, 1993.

[81] Bifet, A., G. Holmes, R. Kirkby, y B. Pfahringer. MOA: Massive Online Analysis. Journal of Machine Learning Research, 11:1601–1604, 2010.

[82] Mierswa, I., M. Wurst, R. Klinkenberg, M. Scholz, y T. Euler. YALE: Rapid Prototyping for Complex Data Mining Tasks. En Proc. 12th ACMSIGKDD Int. Conf. on Knowledge Discovery and DataMining, págs. 935–940, 2006.

[83] Delany, S., P. Cunningham, A. Tsymbal, y L. Coyle. A Case-based Technique for Tracking Concept Drift in Spam Filtering. Knowledge- Based Systems, 18(4-5):187–195, 2005.

[84] Harries, M. SPLICE-2 Comparative Evaluation: Electricity Pricing. Reporte técnico, The University of New South Wales, Sydney, Australia,1999.

[85] Geoff Hulten y Pedro Domingos. VFML – A toolkit for mining high-speed time-changing data streams. Software toolkit, 2003.

[86] Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, y I.Witten. The WEKA data mining software: an update. SIGKDD Explorations Newsletter, 11(1):10–18, 2009.

[87] Hansen, L. y P. Salamon. "Neural Network Ensembles." IEEE Transactions on Pattern Analysis and Machine Intelligence 12: 993–1001. 1990.

[88] Gama, J. Combinnin Classification Algorithms. Portugal, University of Porto, 1999.

[89] Ramos, G. Nuevos Desarrollos en Aprendizaje Inductivo. Tesis de doctorado. Universidad de Málaga, España. 2001.

[90] Deckert, M. Batch Weighted Ensemble for Mining Data Streams with Concept Drift. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Ra´s, Z.W. (eds.) ISMIS 2011. LNCS, vol. 6804, pp. 290–299. Springer, Heidelberg 2011.

[91] Gonzalves, P., Barros R. RCD: A Recurring Concept Drift Framework. Centro de Informática, Universidad Federal de Pernambuco, Ciudad Universitaria,50.740-560, Recife, Brasil, 2013.

[92] Brzezinski, D. and J. Stefanowski. Accuracy updated ensemble for data streams with concept drift. In Emilio Corchado, MarekKurzynski, and Michal

Wozniak, editors, Hybrid Artificial Intelligent Systems, volume 6679 of Lecture Notes in Computer Science, pages 155–163. Springer Berlin, Heidelberg, 2011.

[93] Littlestone, N. and M. Warmuth. The Weighted Majority algorithm. Information and Computation, 1994.

[94] Yue, S. et al. Mining Concept Drifts from Data Streams Based on Multiclassifiers. In Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology. Beijing, China. 2007.

[95] Mejri, D., Khanchel R. and Limam M. An ensemble method for concept drift in nonstationary environment. Journal of Statistical Computation and Simulation, 2012.

[96] Blum, A. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Machine Learning, 1997.

[97] Minku, L., Yao X. DDD: A New Ensemble Approach for Dealing with Concept Drift. IEEE transactions on knowledge and data engineering, vol. 24, no. 4, 2012.

[98] Breiman, L. "Bagging Predictors." Machine Learning, 1996.

[99] Freund, Y. Boosting a weak learning algorithm by majority. In 3th Annual Workshop on Computational Learning Theory. 1990.

[100] Oza, N. and Russell, S. Online bagging and boosting. In Artificial Intelligence and Statistics 2001, pages 105–112. Morgan Kaufmann, 2001.

[101] Bifet, A. and Gavalda, R. Learning from time-changing data with adaptive windowing. In SIAM International Conference on Data Mining, 2007.

[102] Kapil, K. Wankhade and Snehlata S. Dongre, A New Adaptive Ensemble Boosting Classifier for Concept Drifting Stream Data. International Journal of Modeling and Optimization, Vol. 2, No. 4, August 2012.

[103] Dongre, S., Malik L. Algorithm to handle Concept Drifting in Data Stream Mining. IJCSN International Journal of Computer Science and Network, Vol 2, Issue 1, 2013.

[104] Nishida, K., K. Yamauchi and T. Omori. ACE: Adaptive Classifiers-Ensemble System for Concept-Drifting Environments. Springer, Heidelberg. LNCS, vol. 3541, pp. 176–185, 2005.

[105] Li, P., X. Wu and X. Hu Mining Recurring Concept Drifts with Limited Labeled Streaming Data. JMLR: Workshop and Conference Proceedings. 2nd Asian Conference on Machine Learning, Tokyo, Japan, 2010.

[106] Ramos, G. Nuevos Desarrollos en Aprendizaje Inductivo. España, Universidad de Málaga, 2001.

[107] Jerez-Aragonés, J., J. A. Gómez-Ruiz, G. Ramos-Jiménez, J. Muñoz-Pérez y E. Alba-Conejo. A COMBINED NEURAL NETWORK AND DECISION TREES MODEL FOR PROGNOSIS OF BREAST CANCER RELAPSE. Artificial Intelligence in Medicine, 27 (1), páginas 45–63, 2003.

[108] Ramos, G., J. del Campo y otros. E-CIDIM: Ensemble Of CIDIM Classifiers Lecture Notes in Artificial Intelligence. 3584: 108-117, 2005.

[109] Ramos, G., J. del Campo y otros. "Incremental algorithm driven by error margins." In: Todorovski, L., Lavraˇc, N., Jantke, K.P. (eds.) DS 2006. LNCS (LNAI) 4265, 2006.

[110] Ramos, G., J. del Campo y otros. Aprendizaje Por Capas Basado En Sistemas Multiclasificadores. . XI Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA-2005): 113-122, 2005.

[111] Ramos, G., J. del Campo y otros. ML-CIDIM: Multiple Layers of Multiple Classifier System Based on CIDIM. Lecture Notes in Artificial Intelligence. 3642: 138-146, 2005.

[112] Ortega, J. Exploting Multiple Existing Models And Learning Algorithms. En Proceedings of the Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms Workshop (IMLM-1996).

[113] Tanenbaum, A. Computer Networks. Prentice-Hall Int, 314-315, second edition, 1988.

[114] Mora, Ll., I. Fortes, R. Morales-Bueno and F. Triguero. Dynamic Discretization of Continuous Values from Time Series. Book "ECML'00", 280-291, 2000.

[115] Bártolo, J. Learning Recurring Concepts from Data Stream in Ubiquitous Environments. Doctor of Philosophy Thesis. Universidad Politécnica de Madrid, 2011.

[116] Kolter, J. y M. Maloof. Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. Journal of Machine Learning Research, 8:2755–2790, 2007.

[117] S. Delany y P. Cunningham. An Analysis of Case-Based Editing in a Spam Filtering System. En Proc. 7th European Conf. in Case-Based Reasoning, págs. 128–141, 2004.

ANEXOS.

Anexo 1. Publicaciones.

1- **Ortiz A.,** Frías I., Ramos G., Morales R., del Campo J., Caballero Y. and Mustelier A. Fast Adapting Ensemble: A New Algorithm for Mining Data Streams with Concept Drift. The Scientific World Journal; Special Issue on Research and Development of Advanced Computing Technologies. http://www.hindawi.com/journals/tswj/raa/235810/. 2014.

2- I. Frías Blanco, J. del Campo Ávila, G. Ramos Jiménez, R. Morales Bueno, **A. Ortiz Díaz,** y Y. Caballero Mota, "Online and non-parametric drift detection methods based on Hoeffding's bound," IEEE Transactions on Knowledge and Data Engineering, 2014. DOI 10.1109/TKDE.2014.2345382.

3- **Ortíz A.,** Frías I., Ramos G., Morales R., del Campo J., Caballero Y. Algoritmo multiclasificador para minar flujos de datos y adaptarse a cambios de conceptos. II Conferencia Internacional de Ciencias Computacionales e Informática. XV Convención y Feria Internacional Informática 2013. Habana, Cuba 2013.

4- **Ortiz A.,** Ramos G., Morales R., del Campo J., Caballero Y., Frías I. Análisis de un algoritmo multiclasificador incremental con diferentes clasificadores bases. V Congreso Internacional de tecnologías informáticas. ITSUP, Portoviejo, Ecuador 2013. ISBN 978-9942-13-426-4.

5- Frías I., J. del Campo, G. Ramos, R. Morales, **A. Ortiz**, Y. Caballero. Aprendiendo con detección de cambios online. Revista Computación y Sistemas, Machine Learning and Pattern Recognition Special Issue. 2013.

6- **Ortiz A.,** Frías I., Ramos G., Morales R., del Campo J., Caballero Y. Propuesta de Algoritmo multiclasificador para Aprendizaje Incremental que se Adapta a los Cambios de Conceptos. I Conferencia Científica Internacional de la UNISS YAYABOCIENCIA 2011. Sancti Spíritus, Cuba, 2011.

7- Frías I., **Ortiz, A.,** Ramos G., Morales R., Caballero Y. Clasificadores y multiclasificadores basados en árboles de decisión. Revista Iberoamericana en Inteligencia artificial. ISSN: 1988-3064(on-line). 2010.

8- **Ortiz A.**, Frías I., Ramos G., Morales R., Caballero Y., Ramírez R. Comparación entre algoritmos de clasificación basados en árboles de dedición. UCIENCIA V Conferencia Científica de la Universidad de las Ciencias Informáticas. ISBN 978-959-286-011-7. 2010.

10- **Ortiz A.,** Frías I., Ramos G., Morales R., Caballero Y. Algoritmos multiclasificadores incrementales que detectan cambios de concepto. Tendencias de Soft Computing, Editorial Feijóo, UCLV. 2009; ISBN 959250525-4. 2009.

11- Frías I., **Ortiz A**., Ramos G., Morales R., Caballero Y. Clasificadores incrementales y cambio de concepto. Tendencias de Soft Computing, Editorial Feijóo, UCLV. 2009; ISBN 959250525-4. 2009

Anexo 2. Cursos del doctorado.

Esta investigación ha sido desarrollada en el marco del Doctorado Iberoamericano en Soft Computing, patrocinado por la Junta de Andalucía y basado en un convenio específico de colaboración entre las Universidades Andaluzas, la Universidad Central "Marta Abreu" de las Villas (Cuba) y la Asociación Universitaria Iberoamericana de Postgrado (AUIP). Durante la parte lectiva de este programa se recibieron los siguientes cursos de doctorado (de los cuales detallamos sus temarios):

- **Conceptos de Recuperación de Información Probabilística**. Profesores: Dr. Juan Huete, Dr. Juan Manuel Fernández Luna (U. Granada). Dr. Ramiro Pérez (U. C. Las Villas).

  1. Introducción a la recuperación de información.
  2. Indexación.
  3. Introducción a las Redes Bayesianas.
  4. Modelos de recuperación de información.
  5. Técnicas de modificación de consultas.
  6. Sistemas de recuperación de información.
  7. Recuperación de información Web.
  8. Recuperación de información estructurada.
  9. Sistemas de recomendación.
  10. Agrupamiento y clasificación documental.

- **Procesamiento Digital de Imágenes**. Profesores: Dr. Miguel García Silvente (U. Granada). Dr. Juan Lorenzo (U. C. Las Villas).

  1. Introducción a la visión por ordenador: Introducción. Filtrado. Extracción de rasgos simples (Fronteras y Esquinas). Extracción de rasgos más complejos (líneas, círculos, etc. Segmentación). Representación y descripción de formas.

2. Visión estéreo: Correspondencia entre imágenes. Rectificación. Cálculo de disparidad y profundidad.
3. Detección de movimiento: Flujo óptico.
4. Seguimiento de objetos: Filtro de partículas.
5. Reconocimiento de gestos.

- **Procesado de Datos y Aprendizaje basado en Reglas**. Profesores: Dr. Jesús Aguilar (U. Pablo de Olavide). Dra. Yanet Rodríguez (U. C. Las Villas).
  1. Introducción: Aprendizaje, Minería de Datos y KDD: Clasificación del aprendizaje. Fases del Proceso KDD. Tareas de la Minería de Datos. Enfoques de aprendizaje y modelos de conocimiento. Heurísticas de búsqueda.
  2. Preprocesado de datos: La calidad de los datos. Tratamiento de valores perdidos, outliers y ruido. Normalización y Estandarización. Discretización y Transformación. Selección de ejemplos. Selección de atributos.
  3. Aprendizaje basado en reglas: Diferencias entre modelos comprensibles y no comprensibles. Conceptos de complejidad y precisión. Medidas de complejidad y precisión. Clasificación mediante reglas de decisión. Técnicas de visualización de reglas.
- **Aprendizaje de Árboles Decisión**. Profesores: Dr. Gonzalo Ramos Jiménez (U. Málaga). Dr. Rafael Morales Bueno (U. Málaga). Dra. Yailé Caballero Mota (U. Camagüey).
  1. Introducción: Qué es un algoritmo TDIDT. El problema de partida. Árbol de decisión. Estructura general de los TDIDT. Objetivos. Clasificar ≠ Predecir.
  2. Conceptos de base: Medidas. Condición hoja. Prepoda y postpoda. Discretización. Binarización. Cotas de concentración: Chernoff y Hoeffding.
  3. Mejoras de los algoritmos TDIDT: Poda con control predictivo. Inducción con atributos desconocidos. Inducción con costes. Inducción con olvido. Técnicas de votación

4. Algunos desarrollos propios: IADEM: Inducción de Árboles de Decisión por Muestreo. CIDIM: Control de Inducción por División Muestral. MultiCIDIM
5. Experimentación.

- **Redes Neuronales Evolutivas: Aplicaciones**. Profesores: Dr. Cesar Hervás (U. Córdoba). Dra. María García (U. C. Las Villas).

1. Redes Neuronales Artificiales en Modelado de Sistemas Dinámicos.
2. Redes Neuronales de base spline, sigmoide y potencial.
3. Algoritmos Evolutivos para el modelado y entrenamiento de Redes Neuronales.
4. Algoritmos Híbridos para la optimización de modelos de Redes Neuronales de base sigmoide y potencial.
5. Aplicaciones en agronomía de precisión, microbiología, aerobiología y cinética química.

- **Soft Computing: fundamentos, hibridaciones y aplicaciones**. Profesores: Dr. David Pelta (U. Granada). Dr. Rafael Bello (U. C. Las Villas).

1. Fundamentos y componentes de Soft Computing
2. Elementos básicos de optimización y búsqueda
3. Esquemas y posibilidades de hibridación
4. Estrategias cooperativas de resolución de problemas
5. Aplicaciones de Soft Computing en bioinformatica

- **Algoritmos Genéticos y Sistemas Difusos Evolutivos**. Profesores: Dr. Jorge Casillas (U. Granada). Dr. Carlos Morell (U. C. Las Villas).

Parte I. Computación evolutiva:

1. Introducción a la computación evolutiva.
2. Algoritmos genéticos I: introducción.
3. Algoritmos genéticos II: diversidad versus convergencia.

4. Algoritmos evolutivos multiobjetivo.

Parte II. Extracción de conocimiento con algoritmos evolutivos:

5. Aprendizaje evolutivo: introducción a la extracción de conocimiento con algoritmos evolutivos.
6. Sistemas difusos evolutivos: Sistemas basados en reglas difusas (SBRD). SBRD en diferentes problemas: control, regresión, clasificación y modelos descriptivos (reglas de asociación, descubrimiento de subgrupos). Aprendizaje y ajuste de SBRD con algoritmos evolutivos. Algunas aplicaciones: control, robótica móvil, marketing,...

- **Modelos Gráficos Probabilísticos**. Profesores: Dr. Serafín Moral (U. Granada). Dr. R. Grau (U. C. Las Villas).

1. Conceptos básicos sobre probabilidad
2. Algunos conceptos sobre grafos
3. Independencia y d-separación
4. Construcción de redes Bayesianas
5. Inferencia
6. Aprendizaje
7. Toma de decisiones con diagramas de influencia
8. Temas abiertos

- **Informática Gráfica**. Profesores: Dr. Juan Carlos Torres (U. Granada). Dr. Carlos Pérez (U. C. Las Villas).

1. Introducción: Gráficos por ordenador. Hardware gráfico. Pipeline. Visualización con OpenGL. Visualización realista. Interacción.
2. Modelado geométrico: Funciones del modelo geométrico. Mallas. Curvas y superficies. Sólidos. Volúmenes.
3. Digitalización 3D: Métodos de captura. Escáner láser. Triangulación. Registrado. Fusión. Simplificación.

4. Realidad virtual: Visualización estéreo. Tecnología. Interacción. Detección de colisiones. Software. Visualización adaptativa.
5. Modelos gráficos en sistemas GIS: Representación raster. Representación vectorial. Análisis raster. GIS 3D.