HOLOS
ISSN 1807 - 1600

NUEVO ALGORITMO MULTICLASIFICADOR PARA FLUJOS DE DATOS CON CAMBIOS DE CONCEPTO

R. O. R. TASÉ[*], A. V. CABRERA, D. L. O. NARANJO, A. A. O. DÍAZ y I. F. BLANCO
Universidad de Granma, Cuba
rtase@udg.co.cu[*]

Artículo presentado en enero/2016 y aceptado en marzo/2016

DOI: 10.15628/holos.2016.3945

RESUMEN

Los algoritmos multiclasificadores se han mostrado particularmente eficientes para trabajar sobre espacios de datos grandes y complejos como los llamados flujos de datos. En estos flujos, durante la clasificación, aparecen conceptos que cambian con el tiempo, por lo que los métodos para su minería, sobre todo los que detectan y se adaptan a estos cambios, son importantes por su aplicación en áreas como: bioinformática, medicina, economía y finanzas, industria, medio ambiente, entre otras. L*a presente investigación propone un nuevo algoritmo multiclasificador que se adapta a los cambios de conceptos, tiene votación ponderada con una nueva forma para ajustar los pesos y permite variar el tipo de clasificador básico.* El algoritmo fue implementado en compatibilidad y bajo las exigencias del entorno de trabajo MOA (*Massive Online Analysis*) facilitando la *comparación con otros algoritmos conocidos y la generación de bases de datos sintéticas que simulan cambios de conceptos. Para la experimentación se generaron experiencias bajo conceptos artificiales conocidos, tales como: SEA, LED, STAGGER e Hiperplano; logrando mostrar la alta capacidad de adaptación y la estabilidad del algoritmo frente a diferentes situaciones simuladas*.

PALABRAS CLAVE: Clasificación, Aprendizaje incremental, flujos de datos, cambio de concepto, clasificadores múltiples.

ENSEMBLE ALGORITHM FOR DATA STREAMS WITH CONCEPT DRIFT

ABSTRACT

The ensemble algorithms have been particularly efficient to work on areas of large and complex data as so-called data streams. In these flows, during qualifying, they appear concepts change over time, so that its mining methods, especially those that detect and adapt to these changes are important for their application in areas such as bioinformatics, medicine, economics and finance, industry, environment, among others. This research proposes a new ensemble algorithm that adapts to concepts drift, have weighted voting with a new way to adjust the weights and can vary the type of basic classifier. The algorithm was implemented in support and under the demands of the work environment MOA (Massive Online Analysis) facilitating comparison with other known algorithms and the generation of synthetic data bases that simulate changes in concepts. For experimentation, experiences they were generated under known, such as artificial concepts: SEA, LED, STAGGER and hyperplane; managing to show high resilience and stability of the algorithm against different simulated situations.

KEYWORDS: Classification, Incremental learning, Data stream, concept drift, ensemble.

1 INTRODUCCIÓN

Debido al actual 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 los antecedentes, comportamiento y otras características de los datos, permitan predecir su comportamiento futuro (Caballero, 2007).

Para el análisis de grandes volúmenes de datos en bruto, normalmente se utilizan técnicas de clasificación como el aprendizaje automático, y dentro de este el aprendizaje inductivo, el cual parte de casos particulares (experiencias, ejemplos) y obtiene casos generales (modelos o reglas). 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 (Ferrer & Aguilar, 2005). 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).

Un flujo de datos es una secuencia muy grande, potencialmente infinita, de pares (x, y) que se van adquiriendo a lo largo del tiempo, donde x representa los atributos de los datos de entrada e y su clase correspondiente, a estos pares suele llamárseles instancias o experiencias. Estas experiencias proceden de entornos dinámicos y no se tienen previamente almacenados, lo que obliga a procesarlos secuencialmente, de forma incremental. 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. 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 (*concept drift*) (Wang *et al*., 2003).

La velocidad del cambio tiene asociada un problema de gran dificultad. En relación a esto se distinguen dos tipos, vinculado con la frecuencia con la que se reciben los ejemplos que describen a la nueva función objetivo: abruptos (repentino, instantáneo) y gradual. Algunos autores como Stanley (2003) dividen el cambio gradual en moderado y lento, dependiendo de la velocidad del mismo.

Muchos algoritmos de aprendizaje fueron usados como modelo base para manipular cambio de concepto. Entre estos se encuentran los sistemas basados en reglas (Schlimmer & Granger, 1986; Widmer & Kubat, 1993; Widmer & Kubat, 1996; Wang *et al*., 2003), árboles de decisión con su versión incremental (Harries *et al*., 1998; Hulten *et al*., 2001; Kolter & Maloof, 2003; Stanley, 2003; Wang *et al*., 2003), Nayve Bayes (Kolter & Maloof, 2003; Wang *et al*., 2003), Máquinas de Soporte Vectorial (Klinkenberg, R. & Joachims, 2000; Klinkenberg, R. , 2004), Redes de Funciones de Base Radial (Kubat & Widmer, 1994) y aprendizaje basado en instancias; (Salganicoff, 1997; Cunningham, 2003).

La presencia de ruido en los datos introduce otro gran problema 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 (Widmer & Kubat, 1996). 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 son FLORA (Widmer & Kubat, 1993), PECS (Salganicoff, 1997), SPLICE (Harries *et al*., 1998) y Local Weights and Batch Selection (Klinkenberg, R. , 2004).

El objetivo de este trabajo es proponer un nuevo algoritmo multiclasificador, para aprendizaje incremental, que sea capaz de detectar y adaptarse, lo más rápido posible, a los cambios de concepto; además, que sea capaz de trabajar, manteniendo altos niveles de precisión, con diferentes tipos de clasificadores bases, incrementales o no.

2 MATERIALES Y MÉTODOS

Para la implementación del algoritmo propuesto empleó el lenguaje Java, sobre el entono de desarrollo Netbeans IDE 7.0.1, en compatibilidad y bajo las exigencias de MOA (Massive Online Analysis) (Bifet & Gavalda, 2007) uno de los más completos entornos de trabajos para minería de datos que está relacionado con el proyecto WEKA (Waikato Environment for Knowledge Analysis) (Hall *et al*., 2009). MOA incluye una colección de algoritmos para el aprendizaje automático (Clasificación, regresión y agrupamiento) para trabajar sobre flujos de datos. Además, cuenta con herramientas para evaluar los algoritmos y generar datos (STAGGER, LED, SEA, Rotating Hyperplane, Random RBF entre otras) de forma artificial, incluyendo la posibilidad de simular cambios de conceptos.

Se realizaron varias corridas del algoritmo implementado con conjuntos de datos de características diferentes y con varios clasificadores base, buscando la existencia de diferencias significativas en su comportamiento, para evaluar su efectividad y adaptabilidad a cambios de conceptos. Para la selección de estos clasificadores se tuvieron en cuenta el modelo usado para la clasificación y la precisión ante diferentes conjuntos de datos. A continuación se mencionan los algoritmos base y los conjuntos de datos seleccionados.

2.1 Algoritmos utilizados

2.1.1 CIDIM (Control de Inducción por División Muestral).

CIDIM es un algoritmo no incremental basado en clasificación por árbol de decisión, propuesto por (Bifet & Gavalda (2007)). Refieren los autores que presenta una elevada precisión y genera árboles de tamaño reducido. Para la experimentación se utiliza una versión del mismo, implementada por los autores de este trabajo sobre el entorno de trabajo MOA.

2.1.2 Algoritmo J48:

Es no incremental, está basado en clasificación por árbol de decisión, es la versión en WEKA del algoritmo C4.5 desarrollado por Quinlan en 1993 (Hall *et al*., 2009).

2.1.3 Naive Bayes

Es un algoritmo de clasificación muy conocido por su simplicidad y bajo costo computacional. Para la experimentación se utiliza la versión incremental y adaptable de este algoritmo implementada en WEKA (Hall *et al*., 2009).

Para la comparación y el análisis, son utilizados los siguientes multiclasificadores:

2.1.4 OzaBag y OzaBoost

Son versiones incrementales de bagging y boosting para trabajar con flujos de datos, desarrolladas por Oza & Russell (2001). Para la experimentación se utilizan las versiones de estos algoritmos implementadas en MOA (Bifet & Gavalda, 2007), utilizando como clasificador base la versión incremental y adaptable del algoritmo Naive Bayes implementada en WEKA.

2.1.5 Weighted Majority Algorithm. (WMA)

Fue desarrollado por Littlestone y Warmuth en el año 1994 (Littlestone & Warmuth, 1994). Para la experimentación se utiliza la versión del multiclasificador *Weighted Majority Algorithm* implementada en MOA.

2.2 Flujos de datos empleados

Los primeros dos tipos de datos generados presentan atributos continuos a diferencia de los dos siguientes donde todos sus atributos son discretos:

2.2.1 Movimientos del Hiperplano

Los hiperplanos han sido utilizados para simular cambios de conceptos porque la orientación y posición de éste puede ser cambiada suavemente cambiando la magnitud de sus pesos. Para la experimentación con este concepto se generaron 100 000 experiencias, las cuales fueron divididas en 4 bloques de 25 000 experiencias simulando cuatro conceptos diferentes. Los cambios se consideran moderados pues se hicieron pequeños cambios en la orientación del hiperplano. Se trabajó con un 10% de ruido, introducido de forma aleatoria.

2.2.2 Concepto SEA

Así llaman en la bibliografía a la forma de generar los datos, para simular cambios de conceptos, introducida por los creadores de SEA (Kolter & Maloof, 2003).

Las experiencias cuentan con valores de tres atributos $x_i \in \Re$, tal que $0.0 \leq x_i \leq 10.0$. La clase va a tomar valor 1 siempre que se cumpla que: $x_1 + x_2 \leq b$ y $b \in \{7,8,9,9.5\}$ y va a ser 0 en caso contrario. El atributo x_3 no es relevante. Para la experimentación con este concepto se generaron 100 000 experiencias, las cuales fueron divididas en 4 bloques de 25 000 experiencias simulando cuatro conceptos diferentes. Para el primer bloque b = 7, luego 8 en el segundo, 9.5 en el tercero y 9 en el cuarto bloque. Se trabajó con un 10% de ruido, introducido de forma aleatoria.

2.2.3 LED (light emitting diode, diodo emisor de luz):

El objetivo es tratar de predecir el dígito mostrado por una pantalla-LED de siete segmentos. El generador utilizado produce para los experimentos 24 atributos binarios, de los cuales 17 son irrelevantes. Para la experimentación con este concepto se generaron 100 000 experiencias, las cuales fueron divididas en 4 bloques de 25 000 experiencias simulando cuatro conceptos diferentes. El cambio de concepto es simulado a través del intercambio de atributos relevantes. Se trabajó con un 10% de ruido, introducido de forma aleatoria.

2.2.4 STAGGER.

Los conceptos STAGGER comprenden un banco de marcas estándares para evaluar la función de aprendizaje de los clasificadores en presencia de cambios de conceptos. Cada experiencia está compuesta por valores de tres atributos:

- *Color =[verde, azul, rojo]*
- *Forma =[triángulo, círculo, rectángulo]*
- *Tamaño =[pequeño, mediano, grande]*

Para simular diferentes conceptos se utilizan diferentes funciones para obtener los valores de la clase:

- La clase tiene valor 1 siempre que la siguiente expresión sea verdadera:

$$\text{“}color = rojo \wedge tamaño = pequeño\text{”} \quad (1)$$

y 0 en caso contrario.

- La clase tomará valor 1 si la siguiente expresión es verdadera

$$\text{“}color = verde \vee forma = circulo\text{”} \quad (2)$$

y 0 en caso contrario.

- La clase es 1 cuando es verdadera la expresión

$$\text{“}tamaño = mediano \vee tamano = l\arg o\text{”} \quad (3)$$

y es 0 es caso contrario.

Para la experimentación con este concepto se generaron 100 000 experiencias, las cuales fueron divididas en 4 bloques de 25 000 experiencias simulando conceptos diferentes. Para el primer bloque se utilizó la expresión (1), luego la expresión (2) en el segundo bloque, la expresión (3) para el tercero, y por último se repite la expresión (1) en el cuarto bloque. Se trabajó con un 10% de ruido, introducido de forma aleatoria.

3 RESULTADOS Y DISCUSIÓN

3.1 Fundamentos de la nueva propuesta.

Para una mejor comprensión de la nueva propuesta, es necesario mencionar algunas deficiencias que presentan los algoritmos basados en SEA para trabajar eficientemente con flujos de datos en presencia de cambios de conceptos.

3.1.1 Algoritmos basados en SEA.

Los algoritmos MultiCIDIM-DS y MultiCIDIM-DS-CFC (del Campo Ávila, 2007) están basados en el algoritmo SEA, aunque con varias modificaciones a éste, como son: utilizar como clasificador base a CIDIM, la forma de entrada y salida de los clasificadores y la de introducción del control por filtros correctores (CFC), en la última versión.

Los artículos de Kolter y Maloof (Kolter & Maloof, 2003) y de Yue *et al*. (2007) destacan algunas deficiencias que pueden presentar los algoritmos basados en el algoritmo SEA para responder y adaptarse eficientemente a los cambios de conceptos. Estas deficiencias pueden señalarse también en la familia MultiCIDIM.

Problemas de SEA según Kolter y Maloof.

- Los clasificadores, miembros del multiclasificador, 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.
- 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.

Problemas de SEA según Yue.

- SEA requiere un grupo de instancias para detectar cambios en la cadena de datos, por lo que podría no detectar cambios de conceptos bajo ciertas circunstancias, como podrían ser los cambios de conceptos rápidos.
- El algoritmo SEA trabaja dividendo los datos en bloques por lo que no se puede adaptar al aprendizaje incremental ya que cuando una nueva instancias llega éste no puede tomar acción inmediata.

3.2 Nueva propuesta de algoritmo

La nueva propuesta tiene como objetivos lograr la adaptación a los cambios de conceptos de los algoritmos de la familia MultiCIDIM y a eliminar algunas de las deficiencias heredadas de SEA. A continuación se describen las variables usadas por el algoritmo.

Datos de estrada

- $n \in \mathrm{N}^*$: Número de experiencias. Potencialmente infinito.
- $DS = \left\{ \vec{x}, y \right\}_n^1$ Donde DS=Flujo de datos de entrada estructurados en $\vec{x}$ =Vector de atributos y y = Clase.
- $\min C \in \mathrm{N}^*$: Mínimo número de clasificadores en el multiclasicador.
- $\max C \in \mathrm{N}^*$: Máximo número de clasificadores en el multiclasicador (minC $\leq$ maxC.)
- $m \in \mathrm{N}^*$: Número de clasificadores bases en un instante de tiempo.

- $E = \{cb, w\}_m^1$ Donde E= Multiclasificador y está compuesto por cb =Clasificador base y w = peso asociado al clasificador base.
- β_1 y β_2 : Factores para ajustar los pesos asociados a los clasificadores bases. $\beta_1 > 0$; $\beta_2 > 0$; $\beta_1 + \beta_2 = 1$.
- $BloqueE \subset DS$: Bloque de experiencias para construir un clasificador.
- $BloqueC \subset DS$: Bloque de experiencias para realizar un control al multiclasificador.
- *p*: periodo de control del multiclasificador.
- θ :Umbral utilizado para eliminar clasificadores bases del multiclasificador.
- *ne*: número de experiencias de un bloque.
- $c \in \mathrm{N}^*$ **:** Número de clses.

Otras variables

- Λ : Valor de la clase predicha por el multiclasificador.
- $\vec{\lambda}$ **:** Vector de por cientos de predicción de un clasificador base por clases.
- $\vec{\sigma} \in \Re^c$ **:** Vector de sumas de pesos, según las clases predichas.

Figura 1: Datos de entrada y variables usadas por el algoritmo.

A continuación se describe el algoritmo general NuevoMC-CIDIM para la combinación de clasificadores. El seudocódigo presentado es una aproximación general, para una mejor comprensión del mismo se ha dividido en partes. Primero se expone el método general y luego algunas de las operaciones internas del mismo.

Algoritmo general
begin
1. Construir-bloqueE (*BloqueE).*
2. CrearE (minC, *BloqueE*).
3. *bloque-exp* $\leftarrow 0$
4. ***for*** *i = ne + 1…n*
   - *4.1. Adicionar-bloqueE (DS_i)*
   - *4.2. Adicionar-bloque-C (DS_i)*
   - *4.3.* ***If*** *(i* ***mod*** *ne = 0)*
     - 4.3.1. *Nuevo-clasificador = Construir-clasificador (bloque-exp)*
     - 4.3.2. Agregar-al-multiclasificador (maxC, Nuevo-Clasificador). (3)
     - 4.3.3. *bloque-exp* $\leftarrow 0$
   - 4.4. ***end if***
   - *4.5.* ***If*** *(i* ***mod*** *p = 0)*
     - 4.5.1. *Actualizar-multiclasificador*.
     - *4.5.2. bloque-C* $\leftarrow 0$
   - *4.6.* ***end if***
5. ***end for***
6. **end**

Figura 2: Algoritmo general NuevoMC-CIDIM para la combinación de clasificadores.

El algoritmo inicia con la construcción del bloque de experiencias (línea 1) y del multiclasificador (línea 2) con el mínimo permisible de clasificadores base. El algoritmo para la construcción del bloque se describe más adelante, es un procedimiento que toma cada vez, una parte o subconjunto del flujo de datos de entrada. Para mejorar la efectividad del algoritmo se crea por cada bloque de experiencias de aprendizaje un bloque de control. El método general de clasificación consiste en penalizar los clasificadores básicos cuyo peso no supere el valor de umbral dado por parámetro, estos son eliminados del multiclasificador y se irán agregando a este los de mejor precisión. Lo anteriormente planteado garantiza que al ocurrir un cambio de concepto el algoritmo sea capaz de adaptarse a este. A continuación se muestran los algoritmos auxiliares para multiclasificador.

Construir-bloqueE (*BloqueE)*
1. ***begin***
2. ***for*** i = 1…ne
   2.1. Adicionar-bloqueE (DS_i)
3. ***end for***
4. ***end***

Figura 3: Algoritmo para construir bloques de experiencias.

El algoritmo de la Figura 3 muestra los pasos para la construcción de bloques de experiencias a partir del flujo de datos, su principal función es partir el flujo de datos, que es generalmente infinito, en partes que puedan ser legibles fácilmente por el clasificador.

CrearE (*minC, BloqueE*)
1. ***begin***
2. ***for*** i = 1… minC
   2.1. Nuevo-Clasificador = Construir-clasificador (BloqueE)
   2.2. Adicionar-E (Nuevo-Clasificador)
3. **end for**
4. **end**

Figura 4: Algoritmo para crear multiclasificador.

En la Figura 4 se muestra el algoritmo para la creación del multiclasificador, procedimiento que se emplea en la línea 2 del algoritmo general. Este método crea clasificadores básicos y los adiciona al conjunto multiclasificador.

Agregar-al-multiclasificador (maxC).
1. ***begin***
2. ***if*** (E.size < maxC)
   2.1. Adicionar-E (Nuevo-Clasificador)
3. Else
   3.1. Eliminar el clasificador de menor peso.
   3.2. Adicionar-E (Nuevo-Clasificador)
4. **end if**
5. **end**

Figura 5: Algoritmo para agregar clasificadores al multiclasificador

El algoritmo de la Figura 5 permite la actualización del multiclasificador mediante la adición y eliminación de clasificadores, para ello comprueba que el tamaño del mismo (E.size) no alcance el máximo de clasificadores permitidos (maxC) en caso afirmativo admite más clasificadores y se adicionan más, en caso negativo, se elimina el de menor peso y en su lugar se construye uno nuevo.

Un elemento importante en el algoritmo propuesto lo constituye precisamente la actualización del multiclasificador (línea 4.5.1 del algoritmo general), esta actualización se realiza al cumplirse el periodo determinado por el valor p como se muestra en la línea 4.3 del algoritmo general descrito en la Figura 2. En la figura 6 se describe el algoritmo para la actualización.

Actualizar-multiclasificador ().
1. ***begin***
2. for j = 1…m
   2.1. ***If*** ($E_j.w < \theta$ ***and*** E.size > minC))
      2.1.1. $E_j.w = E_j.w * \beta_2 + \Pr ecision(E_j.cb(bloque - C)) * \beta_1$.
      2.1.2. $E_j.delete$
   2.2. ***end if***
3. ***end for***
4. ***end***

Figura 6: Algoritmo para actualizar multiclasificador.

La fórmula utilizada en el algoritmo de la Figura 6 (línea 2.2.1) para la actualización de los pesos es inspirada por estudios en disciplinas como las telecomunicaciones (Tanenbaum, 1988), fórmula de suavizado para el cálculo de una medida estable de la usabilidad de las líneas de comunicaciones. En la experimentación se utilizaron como valores para los parámetros $\beta_1 = \frac{1}{8}$ y al igual que en el trabajo de Núñez, Fidalgo y Morales (Núñez *et al*., 2007).

Clasificación de una experiencia.

Clasificación de una experiencia.
1. ***begin***
2. $\vec{\sigma}$ = 0
3. for j = 1, . . . , m
   3.1. $\vec{\lambda}$ = Clasificar ($E_j.cb, DS_i.x$)
   3.2. for k = 1…c
      3.2.1. $\sigma_k = \sigma_k + \lambda_k * E_j.w$
   3.3. ***end for***
4. ***end for***
5. Λ = Clase-Mayor-Suma-Peso ($\vec{\sigma}$)
6. ***End***

Figura 7: Algoritmo para la clasificación de una experiencia.

HOLOS
ISSN 1807 - 1600

El nuevo algoritmo tiene votación pesada para mejorar su adaptación a los cambios, renueva el sistema de agregar y eliminar clasificadores básicos y además posibilita el trabajo con clasificadores bases incrementales.

3.3 Discusión de los resultados

Para evaluar el desempeño del nuevo algoritmo ante cambios de conceptos se realizaron dos experimentos, el primero consistió en la comparación del multiclasificador usando diferentes clasificadores base, y el segundo experimento, se realizó para evaluar la efectividad del multiclasificador o método de ensamble comparándolo con otros multiclasificadores, en ambos casos, se emplearon diferentes conjuntos de datos o bases correspondientes a conceptos diferentes.

Los primeros cuatro gráficos (a, b, c, d) de la figura 8 muestran los resultados de evaluar al nuevo multiclasificador (NuevoMC) utilizando tres tipos diferentes de clasificadores bases CIDIM (NuevoMC-CIDIM), Naive Bayes (NuevoMC-NB) y J48 (NuevoMC-J48), sobre cuatro bases artificiales diferentes.

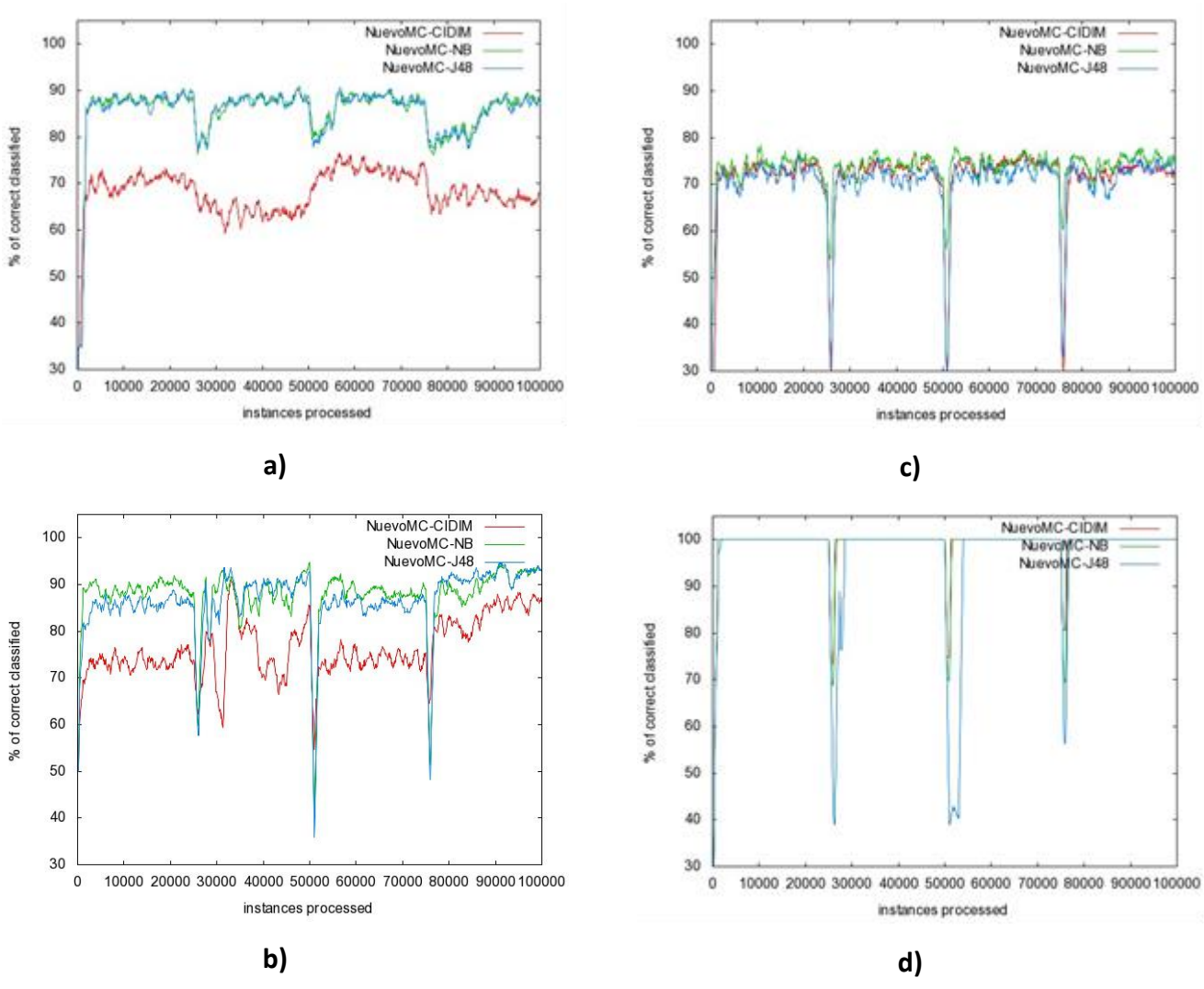


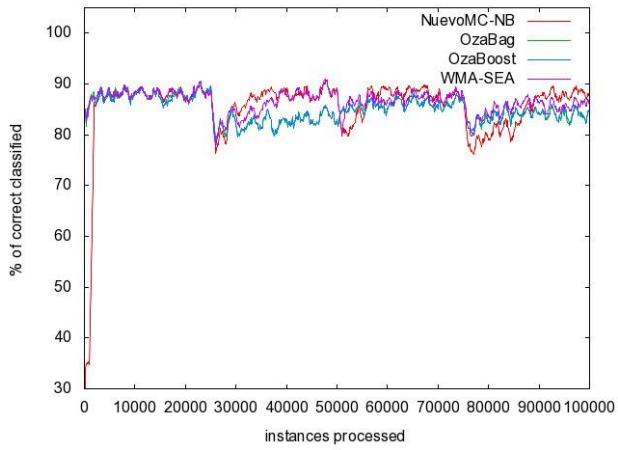
Figura 8: Aprendizaje sobre 100000 instancias a) del concepto SEA, b) del concepto Hiperplano, c) del concepto LED, d) del concepto STAGGER

HOLOS
ISSN 1807 - 1600

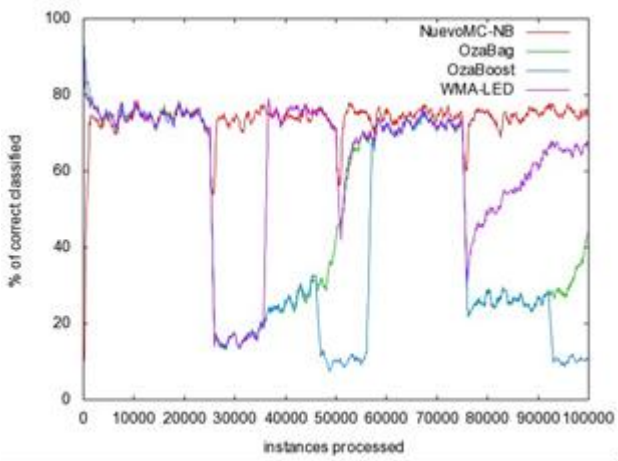
Las dos primeras bases (SEA, HiperPlano) están formadas por atributos continuos, mientras que en las dos últimas (LED, STEGGER) los atributos son nominales. Es importante resaltar que NuevoMC-CIDIM tuvo resultados significativamente inferiores a los resultados con los otros dos clasificadores bases, cuando se trabajó con bases con atributos continuos. Sin embargo, con atributos nominales, se puede observar en los gráficos a y b de la figura 1, los resultados significativamente iguales a los resultados de los otros dos clasificadores base. Además, podemos significar que sus caídas de precisión en los puntos de cambios controlados fueron menores sobre la base STAGGER.

Según muestran los resultados, el nuevo algoritmo ha tenido un comportamiento significativamente igual teniendo como clasificadores base, tanto al Naive Bayes (incremental y adaptable) como el J48 (no incremental), incluso los resultados con CIDIM (no incremental) no fueron significativamente diferentes en el caso de bases con atributos no continuos como LED y STAGGER. Por lo que no se puede establecer diferencias de comportamiento teniendo en cuenta la característica de ser incrementable o no, del clasificador base.

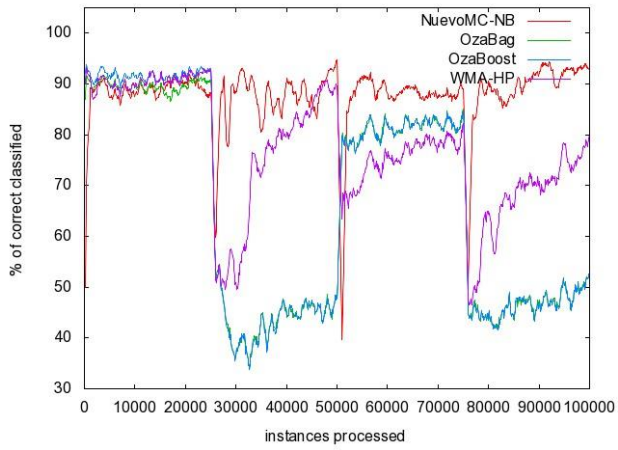
Los gráficos e, f ,g y h de la figura 9 muestran los resultados de evaluar al nuevo multiclasificador, con Naive Bayes como clasificador base (NuevoMC-NB), en comparación con los resultados de otros multiclasificadores (OzaBag, OzaBoost y WMA) implementados en la plataforma MOA.



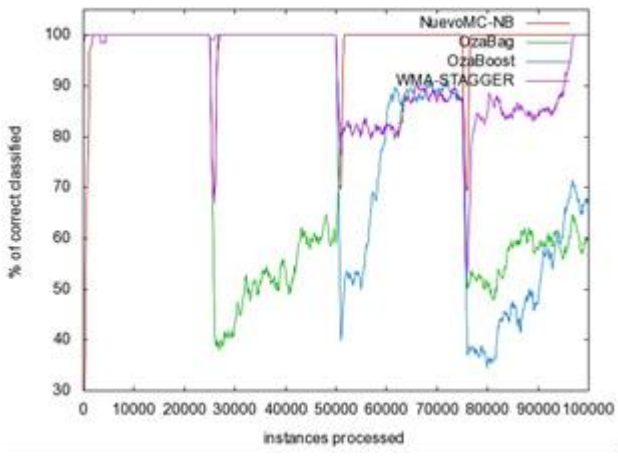
e)



g)



f)



h)

Figura. 9: Aprendizaje sobre 100 000 instancias e) del concepto SEA, f) del concepto Hiperplano, g) del concepto LED, h) del concepto STAGGER

Es importante resaltar la estabilidad que presenta el nuevo algoritmo frente a bases con características tan diferentes y la rápida recuperación después de la caída en los puntos de cambios programados.

4 CONCLUSIONES

En el presente artículo se han mostrado las características principales y los resultados obtenidos de forma experimental de un nuevo algoritmo multiclasificador para trabajar sobre flujos de datos y adaptarse a los cambios de conceptos. El nuevo algoritmo combina rasgos de la familia MultiCIDIM, del algoritmo DWM y utiliza una nueva fórmula para ajustar los pesos de los clasificadores básicos; además, utiliza votación pesada y renueva el mecanismo para agregar y eliminar clasificadores bases.

A través de un análisis comparativo con otros multiclasificadores implementados en MOA, se pudo mostrar que este algoritmo, tiene una alta capacidad de adaptación a los cambios de conceptos y muestra gran estabilidad frente a base de datos con características diferentes en cuanto a los atributos que conforman sus experiencias, los tipos de cambios que simulan y presencia de ruido.

Por otra parte, se obtuvieron bajos resultados de precisión cuando se utilizó al algoritmo CIDIM como clasificador base frente a bases de datos con atributos continuos, no siendo así frente a las bases LED y STAGGER. Además, no se pudo establecer diferencia significativa entre la utilización de algoritmos incrementales o no como clasificadores básicos pues se obtuvieron resultados significativamente igual al utilizar el J48 (no incremental) y una versión de Naive Bayes incremental y daptable.

5 REFERENCIAS

1. BIFET, A., & GAVALDA, R. (2007). *Learning from Time-Changing Data with Adaptive Windowing.* Paper presented at the SDM.
2. CABALLERO, Y. (2007). *Aplicación de la Teoría de los Conjuntos Aproximados en el Preprocesamiento de los Conjuntos de Entrenamiento para Algoritmos de Aprendizaje Automatizado.* Universidad Central "Marta Abreu" de la Villas, Santa Clara.
3. CUNNINGHAM, P. (2003). *A case-based approach to spam filtering that can track concept drift*. Paper presented at the ICCBR-2003 Workshop on Long-Lived CBR Systems.
4. DEL CAMPO ÁVILA, J. (2007). *Nuevos Enfoques en el Aprendizaje Incremental.* (Tesis Doctoral), Universidad de Málaga, Málaga.
5. FERRER, F. J., & AGUILAR, J. S. (2005). *Minería de Data Streams: Conceptos y Principales Técnicas.* Universidad de Sevilla, España.
6. HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., & WITTEN, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter, 11*(1), 10-18.
7. HARRIES, M., SAMMUT, C., & HORN, K. (1998). Extracting hidden context 1998. *Machine Learning, 32*(2), 101-126.

8. HULTEN, G., SPENCER, L., & DOMINGOS, P. (2001). *Mining time-changing data streams*. Paper presented at the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

9. KLINKENBERG, R. (2004). Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis, 8*(3), 281–300.

10. KLINKENBERG, R., & JOACHIMS, T. (2000). *Detecting concept drift with support vector machines*. Paper presented at the 17th International Conference on Machine Learning.

11. KOLTER, J., & MALOOF, M. (2003). *Dynamic weighted majority: A new ensemble method for tracking concept drift*. Paper presented at the 3rd International IEEE Conference on Data Mining.

12. KUBAT, M., & WIDMER, G. (1994). Adapting to drift in continuous domains *Technical Report ÖFAI-TR-94-27*. Vienna: Austrian Research Institute for Artificial Intelligence.

13. LITTLESTONE, N., & WARMUTH, M. K. (1994). The weighted majority algorithm. *Information and computation, 108*(2), 212-261.

14. NÚÑEZ, M., FIDALGO, R., & MORALES, R. (2007). Learning in environments with unknown dynamics: Towards more robust concept learners. *The Journal of Machine Learning Research, 8*, 2595-2628.

15. OZA, N. C., & RUSSELL, S. (2001). *Experimental comparisons of online and batch versions of bagging and boosting.* Paper presented at the Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.

16. SALGANICOFF, M. (1997). Tolerating concept and sampling shift in lazy learning using prediction error context switching. *AI Review Special Issuse on Lazy Learning, 11*(1-5), 133-155.

17. SCHLIMMER, J. C., & GRANGER, R. H. (1986). Incremental learning from noisy data. *Machine Learning, 1*(3), 317-354.

18. STANLEY, K. O. (2003). Learning concept drift with a committee of decision trees *Technical Report UT-AI-TR-03-302*. USA: Department of Computer Sciences, University of Texas at Austin.

19. TANENBAUM, A. S. (1988). *Computer networks* (2nd ed.). New Jersey, USA: Prentice-Hall.

20. WANG, H., WEI, F., PHILIP, Y., & JIAWEI, H. (2003). *Mining Concept-Drifting Data Streams Using Ensemble Classifiers.* Paper presented at the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington DC.

21. WIDMER, G., & KUBAT, M. (1993). *Effective learning in dynamic environments by explicit context tracking*. Paper presented at the 6th European Conf. on Machine Learning ECML-1993.

22. WIDMER, G., & KUBAT, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23*, 69-101.

23. YUE, S., GUOJUN, M., XU, L., & CHUNNIAN, L. (2007). *Mining concept drifts from data streams based on multi-classifiers.* Paper presented at the Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on.