Ciencias Matemáticas, Vol. 31, No. 1, Pag. 49-60, 2017
Recibido 08-2016

Estudio comparativo sobre ensambles de clasificadores en flujos de datos no estacionarios
Comparative study on classifiers ensembles in non-stationary data streams

Alberto Verdecia Cabrera[1,3*], Isvani Frías Blanco[2], Agustín Ortíz Díaz[1], Yanet Rodríguez Sarabia[3], Antonio Mustelier Hechavarría[1]

Resumen Muchas fuentes generan datos continuamente (conocidos como flujos de datos), por lo que es imposible almacenar estos grandes volúmenes de datos y es necesario procesarlos en tiempo real. Debido a que estos datos son adquiridos a lo largo del tiempo y a la dinámica de muchas situaciones reales, la distribución de probabilidades (concepto objetivo) que rige los datos puede cambiar en el tiempo, un problema comúnmente denominado cambio de concepto. Existen varios algoritmos para manipular cambios de concepto, y entre ellos se encuentran los ensambles de clasificadores incrementales y los ensambles de clasificadores basados en bloques de instancias. En la literatura revisada no existen artículos para comparar estos dos enfoques. Por lo que, en este trabajo se realiza un estudio comparativo sobre estos dos enfoques.

Abstract Many sources continuously generate data, known as data stream, creating large volumes of data. It is necessary to process them in real time but it is impossible to store them. Because these data are collected over time and in the dynamics of many real situations, the target function of the data can change over time, a problem commonly called concept drift. There are several algorithms to deal with concept drift, such as incremental ensembles and block-based ensembles. In the reviewed literature there are no articles on comparison of these two approaches. Therefore, in this work we carry out a comparative study between such approaches.

Palabras Clave
Flujos de datos — ensambles de clasificadores — cambio de concepto

[1] *Departamento de informática, Universidad de , Granma, Cuba, averdeciac@udg.co.cu, aortizd@udg.co.cu, tonym@udg.co.cu*
[2] *Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, Brazil, ifriasb@hotmail.com*
[3] *Centro de Estudios de Informática, Universidad Central "Marta Abreu" de Las Villas, Villa Clara, Cuba, yrsarabia@uclv.edu.cu*
[*] **Autor para Correspondencia**

1. Introducción

En la actualidad múltiples sistemas generan gran cantidad de información, la cual constituye una fuente importante de conocimiento [27]. En muchas situaciones el tamaño de los datos generados por sensores, telefonía y muchos otros, es potencialmente infinito debido a su constante generación. Por lo que para procesarlos se requieren técnicas de minería de flujos de datos. En tareas de clasificación, un flujo de datos se define como una secuencia muy grande (potencialmente infinita) de pares que se van adquiriendo a lo largo del tiempo. Estos pares, llamados instancias o ejemplos, están compuestos por un conjunto de atributos y una etiqueta de clase. Uno de los problemas fundamentales cuando se aprende a partir de flujos de datos no estacionarios, obtenidos en el tiempo desde diferentes fuentes, es el cambio de concepto. Por ejemplo, tales cambios pueden ocurrir debido al cambio de intereses personales, preferencias de ropas, nuevas preferencias, consumo de energía, etc. El filtrado de correos spam (basura) es otro ejemplo: los sitios o personas que se encargan de enviar correo no deseados disfrazan sus correos para tratar de evadir los filtros, por lo que los filtros anti-spam deben estar actualizados para identificar exitosamente los spam en el tiempo.

En la literatura se han utilizado muchos algoritmos como modelos base para manipular cambios de concepto. Entre estos se pueden encontrar sistemas basados en reglas [42, 46, 47], árboles de decisión [30, 31, 43, 23], Naïve Bayes [2, 1, 25], máquinas de vectores de soporte [33, 32], redes neuronales artificiales [34] y razonamiento basado en casos[15, 26]. También se han propuesto algoritmos para la detección de cambios de concepto tales como los gráficos de control (control charts) [4], DDM (*Drift Detection Method*) [28], EDDM (*Early Drift Detection Method*) [3], ECDD (*EWMA for Concept Drift Detection*), ADWIN (*Adaptive Windows*) [6] y HDDM (*Hoeffding Drift Detection Method*) [22]. Haciendo uso de los diferentes modelos bases y los detectores

de cambios de concepto se han desarrollado varios algoritmos basados en ensambles de clasificadores para el aprendizaje a partir de flujos de datos no estacionarios. Gama y otros [28] distinguen dos categorías donde se pueden ubicar las estrategias para enfrentar el problema del cambio de concepto: estrategias en las cuales el aprendizaje se adapta en intervalos de tiempo regulares sin considerar la ocurrencia de un cambio de concepto; y estrategias en las cuales se detecta primero el cambio de concepto y luego el aprendizaje se adapta al cambio. Los primeros métodos de ensambles usualmente utilizaban la primera estrategia para manipular cambio de concepto, estos solo incluían los mecanismos básicos para el aprendizaje: actualizar los clasificadores existentes, eliminar los clasificadores de bajo rendimiento, e insertar clasificadores. Aunque estos mecanismos permiten adaptarse al cambio sin necesidad de detectar este de forma explícita, la adaptación al cambio generalmente es lenta. En los últimos años se han propuesto varios métodos de ensamble que incluyen mecanismos de detección de cambios explícita (segunda estrategia), manipulando los cambios de concepto de forma más eficiente. Sin embargo, en la literatura los ensambles no se clasifican por la forma de actualizar los clasificadores bases, ni existen estudios experimentales donde se comparan estos dos tipos de ensambles. Por lo que, en este artículo se propone una nueva clasificación de los ensambles que se basa en la forma en que actualizan los clasificadores bases. De acuerdo con esta clasificación, se pueden clasificar en ensambles basados en bloques de instancias y ensambles incrementales. Además se realiza un estudio experimental donde se comparan estos enfoques frente a cambios de conceptos estables, abruptos y graduales. El resto de este artículo está estructurado de la siguiente manera: primeramente en la sección 2 se define formalmente el cambio de concepto. En la sección 3 se presentan algunos modelos individuales capaces de aprender a partir de flujos de datos no estacionarios. Luego en la sección 4 se realiza una revisión de los principales algoritmos de ensambles de clasificadores propuestos en la literatura. En la sección 5 se realiza un estudio experimental para comparar los dos enfoques estudiados en este artículo. Finalmente en la sección 6 se presentan las conclusiones de este trabajo.

2. Cambio de concepto

Dentro del aprendizaje incremental o aprendizaje en línea [47] el problema de clasificación se define generalmente por una secuencia (posiblemente infinita) de ejemplos (también conocidos como instancias) o experiencias $S = e_1, e_2, ..., e_i...$ que se obtienen en el tiempo, normalmente una a la vez y no necesariamente dependientes del tiempo. Cada ejemplo de entrenamiento $e_i = (\overrightarrow{x}, y_i)$ está formado por un vector $\overrightarrow{x}$ y un valor discreto y_i, llamado etiqueta y tomado de un conjunto $\mathscr{Y}$ denominado clase. Cada vector $\overrightarrow{x} \in \overrightarrow{\mathscr{X}}$ tiene las mismas dimensiones y a cada dimensión se le llama atributo.

Se considera *concepto* como el término que se refiere a la función de distribución de probabilidad del problema en un punto determinado en el tiempo [36]. Este concepto puede ser caracterizado por la distribución de probabilidad conjunta $P\left(\overrightarrow{\mathscr{X}}, \mathscr{Y}\right)$ donde $\overrightarrow{\mathscr{X}}$ representa los atributos y $\mathscr{Y}$ es la clase. Por tanto un cambio en la distribución del problema (también conocido como contexto [28] implica un cambio de concepto.

En la literatura existen múltiples clasificaciones para determinar la naturaleza del cambio de concepto. Consideremos que un cambio de concepto involucra dos conceptos diferentes: el concepto inicial (C_I) y el concepto final (C_F). Una de las clasificaciones más importantes se refiere a la *extensión del cambio* (también llamada tasa de cambio). Dependiendo de cuan diferentes son C_I y C_F, se puede decir que que el cambio es total o parcial. Un cambio total implica que la distribución de C_I y C_F no comparten nada, por lo que este cambio usualmente se simula con conjuntos de datos artificiales porque es difícil observarlo en conjuntos de datos reales. Por tanto cuando se aprende de conjuntos de datos reales el cambio suele ser parcial. De acuerdo con la velocidad del cambio, hay un tiempo (t_{cambio}) para cambiar de C_I y C_F, por lo que cuando $t_{cambio} = 0$ el cambio es abrupto o que es gradual cuando $t_{cambio} > 0$. Cuando el nuevo cambio que ocurrirá al finalizar el cambio actual C_F ha ocurrido previamente se dice que este cambio es recurrente.

3. Clasificadores individuales

En esta sección se presentan varios de los modelos indivuduales de aprendizaje que son utilizados para clasificación de flujos de datos no estacionarios.

3.1 Redes neuronales artificiales

Las redes neuronales artificiales han sido exitosamente utilizadas en la minería de datos. En las aplicaciones tradicionales de minería de datos las redes neuronales son entrenadas utilizando un conjunto de entrenamiento fijo. Sin embargo, la mayoría de los modelos de redes neuronales requieren varias pasadas sobre el conjunto de entrenamiento para ajustar los pesos de las neuronas, usualmente se utiliza el algoritmo backpropagation. Sin embargo en la minería de flujos de datos los ejemplos son vistos una sola vez por el modelo y es necesario mantener acostado el costo computacional para para actualizar los pesos de las neuronas con una sola pasada sobre los datos. Un ejemplo para adaptar una red para aprender a partir de flujos datos es utilizar un mecanismo de olvido, es decir cuando la red no se entrena con los mismos datos múltiples veces, esta se ajustará dinámicamente a los datos que se van obteniendo en el tiempo, por tanto pueden reaccionar de forma natural al cambio de concepto. Uno de los sistemas más conocidos capaz de manipular cambio es FRANN (*Floating Rough Approximation in Neural Network*) [34].

3.2 Naïve Bayes

Este modelo se basa en el teorema de Bayes y calcula la probabilidad condicional de cada una de las clases para cada ejemplo nuevo de entrenamiento asumiendo que los atributos son independientes dada la clase. El algoritmo Naïve

Bayes aprende eficientemente a partir de flujos de datos no estacionarios con complejidad computacional constante.

De acuerdo con lo planteado en la sección 2 cada ejemplo de entrenamiento $e_i = (\overrightarrow{x}, y_i)$ está formado por un vector $\overrightarrow{x}$ y un valor discreto y_i, llamado etiqueta y tomado de un conjunto $\mathscr{Y}$ denominado clase. el clasificador Naïve Bayes se basa en encontrar la clase más probable que describa a ese ejemplo. La clase más probable será la que cumpla:

$$y_p = argmax_{y_i \in \mathscr{Y}} P(y_i \mid \overrightarrow{x}) \tag{1}$$

es decir dada la probabilidad de los atributos que describen el ejemplo e_i, éste pertenezca a la clase y_i. Por el teorema de Bayes:

$$y_p = argmax_{y_i \in \mathscr{Y}} \frac{P(\overrightarrow{x} \mid y_i) P(y_i)}{P(\overrightarrow{x})} \tag{2}$$

$$y_p = argmax_{y_i \in \mathscr{Y}} P(\overrightarrow{x} \mid y_i) P(y_i) \tag{3}$$

Se puede estimar $P(y_i)$ contando las veces que aparece y_i en el conjunto de entrenamiento y dividiéndolo por el número total de ejemplos que forman este conjunto. Para estimar el término $P(\overrightarrow{x} \mid y_i)$ se debe recorrer todo el conjunto de entrenamiento, lo que requiere un alto costo computacional. Naïve Bayes simplifica el problema asumiendo que los atributos son independientes dada la clase. Así la probabilidad de que un ejemplo no etiquetado $\overrightarrow{x}$ pertenezca a la clase y_i está dada por:

$$P(\overrightarrow{x} \mid y_i) = \prod_j P(x_j \mid y_i) \tag{4}$$

La estimación de los parámetros del modelo (por ejemplo, $P(y_i)$ y $P(\overrightarrow{x})$) se pueden aproximar con frecuencias relativas del conjunto de entrenamiento. Estas son las estimaciones de máxima verosimilitud de las probabilidades. Las probabilidades de las clases se pueden calcular asumiendo clases equiprobables. Para estimar las probabilidades de cada uno de los atributos generalmente se utilizan la distribución de binomial para atributos discretos y la distribución normal cuando se trata con atributos continuos.

3.3 Árboles de decisión

La construcción de árboles de decisión ha sido ampliamente estudiada en la comunidad del aprendizaje automático. En la minería de flujos de datos, la desigualdad de probabilidades de Hoeffding [18, 22, 23], de Chernoff [18, 23] han sido herramientas poderosas para la inducción en línea de árboles de decisión. La cota Hoeffding indica que con probabilidad $1-\delta$, la media real de una variable aleatoria con rango R no difiere de la media estimada después de n observaciones independientes por más de:

$$\varepsilon = \sqrt{\frac{R^2 ln\left(\frac{1}{\delta}\right)}{2n}} \tag{5}$$

Utilizando esta cota, Domingos y Hulten propusieron VFDT (*Very Fast Decision Tree*) [31]. VFDT induce incrementalmente un árbol de decisión a partir de un flujo de datos, sin la necesidad de almacenar ejemplos de entrenamiento después de haber sido utilizados para actualizar el árbol. Este algoritmo es similar a los algoritmos clásicos de inducción de árboles de decisión, solo difiere en la selección del atributo para dividir. En lugar de seleccionar el mejor atributo después de ver todos los ejemplos, VFDT utiliza la cota de Hoeffding para determinar cuántas instancias deben ser muestreadas antes de poder elegir, con probabilidad $1-\delta$, qué atributo debe ser usado para la expansión de cada nodo.

4. Ensambles de clasificadores

Los algoritmos basados en ensambles de clasificadores combinan la predicción de los clasificadores bases y se pueden clasificar por la forma en que actualizan sus clasificadores bases. La primera clasificación está dada por la utilización de bloques de instancias. Los ensambles basados en bloques de instancias dividen el flujo de datos en pequeños bloques de igual tamaño y entrenan clasificadores con cada uno de estos bloques. La adaptación a los cambios de concepto de estos ensambles generalmente es lenta porque tienen que esperar a que se llene el bloque para actualizar los clasificadores bases. La segunda clasificación de los ensambles es por la actualización de los clasificadores de forma incremental. Estos métodos utilizan como clasificadores bases clasificadores incrementales, es decir actualizan los modelos en la medida que se obtienen las instancias. En las secciones 4.1 y 4.2 se describen varios algoritmos dentro de estas dos clasificaciones.

4.1 Ensambles basados en bloques de instancias

Los ensambles de clasificadores basados en bloques de instancias dividen el flujo de datos en pequeños bloques de igual tamaño para crear clasificadores con cada uno de estos bloques.

Uno de los primeros algoritmos de ensamble para el procesamiento de flujos de datos fue SEA (*Streaming Ensemble Algorithm*) [44]. SEA crea los clasificadores bases a partir de pequeños subconjuntos de los datos, leídos secuencialmente en bloques de un tamaño fijo. El algoritmo cuenta con un límite máximo de clasificadores que actúa como mecanismo de adaptación. Al ser alcanzado este límite máximo obliga al algoritmo a sustituir clasificadores bases anteriores siguiendo cierto criterio de remplazo. Para combinar las predicciones de los clasificadores base utiliza votación por mayoría no ponderada y como clasificador base utiliza el algoritmo C4.5. SEA presenta problemas para adaptarse a los cambios de conceptos abruptos; en estos resultados influye el mecanismo de votación utilizado ya que los clasificadores dejan de aprender una vez que son creados.

Bajo el mismo esquema de entrenamiento que SEA, Wang y otros [45] propusieron el método AWE (*Accuracy Weighted Ensemble*). AWE combina las respuestas de los clasificadores base a través de la votación por mayoría ponderada. La

ponderación de los clasificadores bases está en función de la precisión obtenida por los mismos, al utilizar como instancias de prueba las propias instancias del bloque de entrenamiento actual. Al igual que SEA, su rendimiento frente a cambios de conceptos graduales es aceptable, pero esto no ocurre así cuando los cambios son abruptos. Eso se debe fundamentalmente a que AWE espera al próximo bloque de entrenamiento para actualizar los pesos de los clasificadores bases.

BWE (Batch Weighted Ensemble algorithm) [17] también divide el conjunto de entrenamiento en bloque de igual tamaño y utiliza el voto mayoritario ponderado para combinar la salida de los clasificadores bases. A diferencia de los métodos anteriores, este incorpora un detector de cambios al algoritmo, BDDM (*Batch Drift Detection Method*), y utiliza un modelo de regresión para estimar cambios de concepto. El detector de cambios se utiliza básicamente para determinar cunado crear un nuevo clasificador base debido a los cambios de concepto o si el concepto permanece estable el ensamble no varía. La idea fundamental de esta propuesta es combinar la capacidad de los ensambles para adaptarse a los cambios graduales con el detector de cambios para manipular cambios abruptos.

Otra propuesta basada en bloques de instancias es AUE (*Accuracy Updated Ensemble*) [11]. AUE mantiene un conjunto de clasificadores ponderados y predice la etiqueta de clase de los ejemplos que van llegando, mediante la agregación de las predicciones de los clasificadores bases utilizando una regla de votación ponderada. Cada vez que se completa el bloque de instancias de entrenamiento, se crea un nuevo clasificador base para sustituir al peor clasificador del ensamble. Después de eliminar el peor clasificador el resto de los clasificadores se actualiza con el nuevo bloque de instancias y sus pesos son actualizados de acuerdo con su precisión. Similar a AUE lo mismos autores propusieron OAUE (*Online Accuracy Updated Ensemble*) [12]. En esta propuesta los clasificadores bases son entrenados de manera incremental y los pesos de cada clasificador base se determinan de acuerdo a su precisión estimada. OAUE mantiene el uso de bloques de instancias para cada vez que se complete el bloque crear un nuevo clasificador y sustituir el peor clasificador del ensamble.

Una propuesta más reciente es FAE (*Fast Adapting Ensemble*) [38], un ensamble diseñado para adaptarse de forma rápida a los cambios de concepto, tanto abruptos como graduales, y especializado en el tratamiento de conceptos recurrentes. 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.

La mayoría de los ensambles que actualizan los clasificadores base mediante bloques presentan problemas para adaptarse a los cambios de concepto abruptos, esto se debe a que no tienen mecanismos explícitos para detectar cambios y generalmente esperan a que se complete un bloque de instancias para actualizar los clasificadores base.

En el algoritmo 1 se muestra el esquema general seguido por los ensambles de clasificadores basados en bloques de instancias. El algoritmo recibe como parámetros, el flujo de datos, la cantidad de clasificadores bases y el tamaño del bloque de instancias. Después se llena el bloque con las instancias de entrenamiento y cuando el bloque está completo se realizan determinadas acciones para, crear nuevos clasificadores, actualizar los clasificadores, etc. Además estos algoritmos deben estar disponibles para predecir en todo momento.

Algoritmo 1: Ensambles basados en bloques de instancias

Entrada:
flujo de datos, cantidad M de clasificadores bases, tamaño N del bloque de instancias

Salida :
clase predicha por el ensamble

```
1 Bloque : bloque de instancias para crear clasificadores
2 foreach instancia de entrenamiento do
3     if Bloque no está lleno then
4         Bloque+ = nuevainstancia
5     else if Bloque está lleno then
6         realizar acciones como, crear nuevo clasificador,
          actualizar clasificadores, calcular pesos, etc.
7         Vacciar el Bloque
8 return en todo momento la clase predicha por el
  ensamble
```

4.2 Ensambles incrementales

Los ensambles incrementales utilizan como clasificadores bases algoritmos incrementales, es decir se van actualizando en la medida en la que se obtienen los datos.

Bagging [10] y Boosting [21] son dos de los algoritmos más conocidos para entrenar clasificadores bases. Bagging aplica muestreo con remplazamiento al conjunto de datos original para crear M conjuntos de entrenamientos del mismo tamaño que el original y crea M clasificadores base con estos conjuntos de entrenamiento. Estos clasificadores son creados utilizando el mismo algoritmo de aprendizaje. Para que los clasificadores bases a inducir con un mismo algoritmo cumplan el requisito de ser diferentes, se tiene que garantizar que los subconjuntos de entrenamiento generados sean también diferentes. Esto se garantiza utilizando algoritmos de induc-

ción que sean muy sensibles a pequeñas variaciones en dicho conjunto, por ejemplo, algunos algoritmos de inducción de árboles de decisión. Esta característica de inestabilidad es la que da nombre a los algoritmos conocidos como "aprendices débiles" y se ve acentuada cuando el subconjunto de entrenamiento tiene un tamaño limitado. En bagging se combinan las predicciones de los clasificadores individuales utilizando votación mayoritaria: se elige la clase que haya sido seleccionada por el mayor número de clasificadores que componen el ensamble.

El método boosting [21] asigna pesos a las instancias de entrenamiento en el conjunto de entrenamiento. El objetivo del método boosting es inducir un modelo que minimice la suma de los pesos de las experiencias que no se clasifiquen correctamente. En un principio, los pesos de las diferentes experiencias tendrán el mismo valor y los errores serán igual de importantes. Pero en cada iteración, los pesos de las experiencias que han sido incorrectamente clasificadas se aumentarán para darles más importancia. De esta forma, en cada iteración se le da más importancia a las experiencias que no han sido clasificadas correctamente en pasos anteriores. En este modelo, la combinación de los clasificadores básicos no es tan simple como con bagging. Ahora se tiene en cuenta el error estimado de cada modelo para calcular la clase de la observación que se está intentando predecir. Sumando los pesos de los modelos que eligen la misma clase, se selecciona aquella que acumule mayor valor.

Oza y Rusell [40] propusieron versiones incrementales de los algoritmos Bagging y Boosting, pero estas propuestas no tienen mecanismos para adaptarse a los cambios de concepto. Así, su adaptación al cambio depende del algoritmo de clasificación utilizado para generar los clasificadores bases. Para manipular cambios de concepto, Bifet y otros [9] propusieron un nuevo algoritmo basado en Bagging llamado OzaBagAdwin. La idea de esta variante es agregar a la versión incremental del algoritmo Bagging el detector de cambios de concepto ADWIN (*Adaptive Windowing*) [6]. El mecanismo de adaptación utilizado se basa en la sustitución del peor de los clasificadores, cuando se estima un cambio, por un nuevo clasificador básico creado más recientemente.

El algoritmo Boosting también ha sido explorado en la minería de flujos de datos [39], pero el algoritmo mantiene valores de precisión más bajos que el Bagging propuesto por Bifet y otros [9]. Otro ensamble basado en Bagging es DDD (*Diversity for Dealing with Drifts*) [37], el cual es capaz de mantener una alta diversidad de clasificadores bases y de adaptarse a los cambios de conceptos . DDD mantiene varios ensambles con diferentes niveles de diversidad. Si en los datos no se detecta la presencia de cambios de conceptos, el sistema estará compuesto por dos ensambles, uno con baja diversidad y otro con alta diversidad pero solo el ensamble con baja diversidad es utilizado para predecir. Cuando se detecta un cambio de concepto se construyen dos nuevos ensambles, uno con baja diversidad y otro con alta diversidad. Los ensambles de baja y alta diversidad después de la detección del cambio se denominan ensambles viejos de baja y alta diversidad y se mantienen porque 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 ensambles se mantienen 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 ensambles nuevamente.

Otro algoritmo de ensamble incremental es FASE (*Fast Adaptive Stacking of Ensembles*) [25]. FASE utiliza un metaclasificador para combinar la salida de los clasificadores bases. El esquema de FASE es aplicado a la versión en línea del algoritmo bagging [40] y utiliza HDDM (*Hoeffding Drift Detection Method*) [22] como detector de cambios de concepto y estimador de error. Cuando se detecta una cambio, se elimina el peor clasificador del ensamble y se agrega uno nuevo. FASE está compuesto por clasificadores adaptativos en los dos niveles (ambos clasificadores el base y el meta son adaptativos). Cada clasificador adaptativo usa HDDM, que monitoriza su tasa de error con el objetivo de emitir tres señales diferentes de cambio durante el proceso de aprendizaje. HDDM emite la señal *en-control* cuando el concepto actual permanece estable, *alerta* cuando es probable que se aproxime un cambio, y *fuera-de-control* cuando se detecta el cambio. En FASE, cada clasificador adaptativo usa un solo clasificador en los conceptos estables. Cuando el nivel de *alerta* es alcanzado, el clasificador adaptativo entrena un clasificador alternativo que reemplaza al principal si después del nivel de alerta ocurre un cambio. Los clasificadores adaptativos pueden de esta forma tener a lo sumo dos clasificadores (el clasificador principal y el alternativo), las predicciones de estos clasificadores es combinada mediante el voto ponderado.

En el algoritmo 2 se muestra el esquema general seguido por los ensambles de clasificadores incrementales. El algoritmo solo recibe como parámetros, el flujo de datos y la cantidad de clasificadores bases. Luego actualiza cada uno de los clasificadores bases con cada una de las instancias de entrenamiento.

Algoritmo 2: Ensambles incrementales

Entrada :
flujo de datos, cantidad M de clasificadores

Salida :
clase predicha por el ensamble

1 **foreach** *instancia de entrenamiento* **do**
2 aculizar los clasificadores bases con cada instancia de entrenamiento
3 **return** en todo momento la clase predicha por el ensamble

Tabla 1. Clasificación de los algoritmos según la forma de actualizar los clasificadores bases

Algoritmo	Actualización de los clasificadores	
	Bloque	Incremental
FASE		x
LeveragingBag		x
OzaBagAdwin		x
AUE	x	
AWE	x	
OAUE	x	

5. Estudio experimental

En el estudio experimental descrito en esta sección se evalúan los algoritmos frente a conjuntos de datos artificiales y reales. Los algoritmos algoritmos son evaluados sobre los diferentes tipos de cambios (abruptos y graduales).

Todos los experimentos fueron ejecutados en MOA [7], una herramienta para evaluar los algoritmos que aprenden a partir de flujos de datos. MOA provee una gran colección de herramientas de evaluación, varios algoritmos, y muchos métodos para generar flujos de datos artificiales con la posibilidad de incluir cambios de concepto.

Las medida de rendimiento utilizada para evaluar los algoritmo fue la precisión. Esta medida se calculó en línea con el objetivo de medir como evoluciona el proceso de aprendizaje en el tiempo.

En este artículo se utilizó para evaluar los algoritmos el enfoque *test-then-train*, también conocido como *prequential* (*predictive sequencial*) que se deriva del error predictivo secuencial [16]. Este enfoque consiste básicamente en calcular las medidas de interés (usualmente la precisión) para cada instancia nueva (etapa de prueba) y después utilizar el ejemplo para seguir con el entrenamiento del algoritmo (etapa de entrenamiento) [31]. Por lo tanto, en cada ejemplo nuevo, el clasificador primero se probó y luego se entrenó. Durante el proceso de aprendizaje, la precisión se calculó con respecto a una ventana deslizante de tamaño 100 [7]. La precisión de los clasificadores se calculó cada 100 ejemplos procesados por medio de la fracción entre el número de los ejemplos bien clasificados en esta ventana deslizante y el tamaño de la ventana.

5.1 Algoritmos utilizados

En este estudio experimental se configuraron los algoritmos FASE [25], LeveragingBag [8], OzaBagAdwin [9], AUE [11], OAUE [12], AWE [45] con sus parámetros por defecto en MOA.

En todos los algoritmos se utilizó como clasificador base Naïve Bayes, el cual es uno de los clasificadores más utilizado para el aprendizaje a partir de flujos de datos no estacionarios [14, 13, 35, 41, 20]. En la Tabla 1 se clasifican los algoritmos seleccionados según la forma en que actualizan sus clasificadores bases.

Tabla 2. Principales características de los generadores de flujos de datos utilizados.

Generador	Nominales	Numéricos	Clases
SEA (4 funciones objetivo)		3	2
Funciones de base radial (RBF)		50	10
Waveform (WAVE)		40	3
Hyperplane (HYP)		10	2
LED (10 % de ruido)	24		10
STAGGER (STA)	3		2
AGRAWAL (AGR, 10 funciones objetivo)	6	3	2

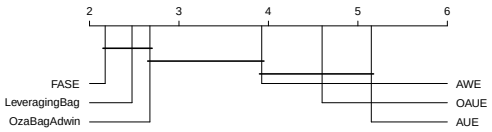
5.2 Experimentos con conjuntos de datos artificiales

Los conjuntos de datos artificiales utilizados para la experimentación están disponibles en MOA y están diseñados para simular cambios de concepto por lo que han sido ampliamente utilizados en la literatura [22, 25, 23, 24, 12, 11, 9, 5, 8]. En la Tabla 2 se presentan las principales características de los conjuntos de datos seleccionados. Por lo que de esta forma, los generadores seleccionados para los experimentos incluyen ruido, atributos irrelevantes (LED) y diferentes tipos de funciones objetivo (SEA,AGR y STA). En esta sección primeramente se estudió la estabilidad de los algoritmos frente a conceptos estables. Después para mostrar la estabilidad de los algoritmos frente a cambios de conceptos (abruptos y graduales).

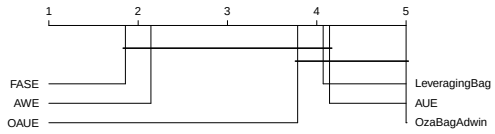
5.2.1 Experimentos con conceptos estables

En conceptos estables, los algoritmos fueron entrenados con 1,000,000 de ejemplos, y se utilizó el enfoque test-then-train para evaluar su rendimiento. La Tabla 3 muestra el rendimiento de los algoritmos en cuanto a precisión. Además en la Tabla 3 se cuenta la cantidad de veces que cada algoritmo gana (G), pierde (P) y empata (E) contra el resto. Como se puede observar el algoritmo FASE es el que más veces gana.

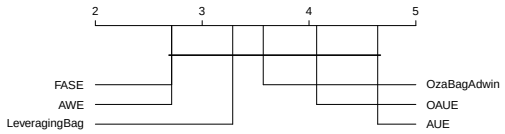
Para comprobar diferencias significativas entre los algoritmos se realizaron pruebas estadísticas siguiendo la metodología propuesta por Demšar [19], García y otros [29] para comparar varios clasificadores sobre varios conjuntos de datos. En la Figura 1a se muestra la comparación entre los algoritmos, como se puede apreciar los algoritmos incrementales (FASE, LeveragingBag, OzaBagAdwin) obtienen mejores resultados que los algoritmos que utilizan bloques de instancias (AWE, OAUE, AUE). En conceptos estables largos, teóricamente no es necesario eliminar clasificadores o actualizar el peso de las instancias de entrenamiento. Los algoritmos incrementales estudiados utilizan mecanismos de detección de cambios para estimar la tasa de error de los clasificadores bases y solo eliminan clasificadores cuando se estiman cambios de concepto, por lo que en conceptos estables la Tabla



(a) Comparación de los algoritmos frente a conceptos estables



(b) Comparación entre todos los algoritmos frente a cambios de concepto abruptos



(c) Comparación de los algoritmos frente a cambios graduales

Figura 1. Comparación entre todos los algoritmos sobre datos artificiales utilizando el test de Friedman y el procedimiento de Bergmann Hommel para el análisis post hoc. Los algoritmos que están conectados (p-value = 0.10) no existen diferencias significativas. Todos los algoritmos utilizan Naïve Bayes como clasificador base.

3 muestra que estos algoritmos son robustos al ruido y a falsas detecciones. Por el contrario los algoritmos que utilizan bloques de instancias cada vez que se completa un bloque siempre eliminan el peor clasificador, por lo que al no ocurrir cambios de concepto se produce perdida de información al eliminar clasificadores bases.

5.2.2 Cambios abruptos y graduales

En esta sección lo algoritmos fueron ejecutados frente conceptos que cambian repetidamente de acuerdo con la Tabla 2, considerando varias funciones objetivos para simular cambios abruptos y graduales . Para simular cambios graduales se utilizó la función sigmoide implementada en MOA [7], incrementando la probabilidad que a cada instante de tiempo los nuevos ejemplos pertenezcan al nuevo concepto. En las Tablas se generaron 25,000 ejemplos por concepto estable, 10 cambios de concepto, en cambios graduales (Tabla 5) se configuró el período de transición entre conceptos a 5,000. Para evaluar el rendimiento de los algoritmos, la precisión se calculó cada 100 ejemplos de entrenamiento. La precisión fue calculada a través de 100 ejemplos de prueba diferentes en cada paso de prueba (cada 100 ejemplos de entrenamiento). Las Tablas 4 y 5 resumen el rendimiento de los algoritmos en términos de promedio ($\overline{x}$) y desviación estándar (s) para la precisión. Las diferencias significativas de estas medidas con respecto a los restantes algoritmos están mostradas en negritas para la precisión. Además las Figuras 1b y 1c muestran los rankings promedio de los algoritmos frente a cambios abruptos y graduales respectivamente. Como se puede observar no existen diferencias significativas entre los dos enfoques. Por ejemplo en cambios abruptos los algoritmos AWE y OAUE obtienen buenos resultados, aunque el mejor resultado lo obtiene el algoritmo FASE, el cual manipula cambios de concepto en cada uno de los clasificadores bases. Adicionalmente en la Figura 2 se muestran las curvas de aprendizaje de los algoritmos estudiados frente a cambios abruptos. En las figuras se puede observar que los algoritmos de ensamble basados en bloques de instancias se comportan de manera similar, es decir alrededor del punto de cambio su precisión se deteriora pero se recuperan rápidamente. En el caso de los algoritmos de ensamble incrementales (LeveragingBag y OzaBagAdwin) se demoran más en adaptarse a los cambios. Esto se debe principalmente a que estos algoritmos no manipulan cambios en los clasificadores bases.

5.3 Experimentos con datos reales

En muchos escenarios reales se ha detectado la presencia de cambios de cambios de concepto. En estos escenarios no se sabe que tipos de cambios están presentes por lo que es importante realizar experimentos con datos de distintos escenarios. En este experimento se evalúan los métodos procesando los ejemplos en su orden temporal. A la llegada de cada nuevo ejemplo de entrenamiento, al clasificador primero se le realiza una prueba eliminado la etiqueta de clase del ejemplo, luego el aprendizaje continúa normalmente con el ejemplo original. La precisión es calculada con respecto a una ventana deslizante de tamaño fijo igual a 100. La Tabla 6 muestra las características principales de los conjuntos de datos reales seleccionados para este experimento. Para la precisión, la Tabla 7 muestra el promedio y la desviación estándar de la fracción entre el número de ejemplos bien clasificados y el total de ejemplos cada 100 ejemplos procesados. La Figura 3 muestra la comparación estadística entre los algoritmos estudiados. Como se puede observar el algoritmo FASE también tiene un buen rendimiento frente a conjuntos de datos reales. Por lo que el método utilizado por FASE para manipular cambios de conceptos es eficiente tanto para datos artificiales como para datos reales. En la Figura 3 se muestra claramente que los algoritmos FASE, LeveragingBag y OzaBagAdwin obtienen mejores resultados que los algoritmos AUE, AWE y OAUE. En este caso el uso de mecanismos de detección de cambios explícitos y la actualización incremental de los clasificadores bases es más eficiente que los mecanismos de adaptación de

Tabla 3. Rendimiento de los algoritmos frente a conceptos estables utilizando como clasificador base Naïve Bayes

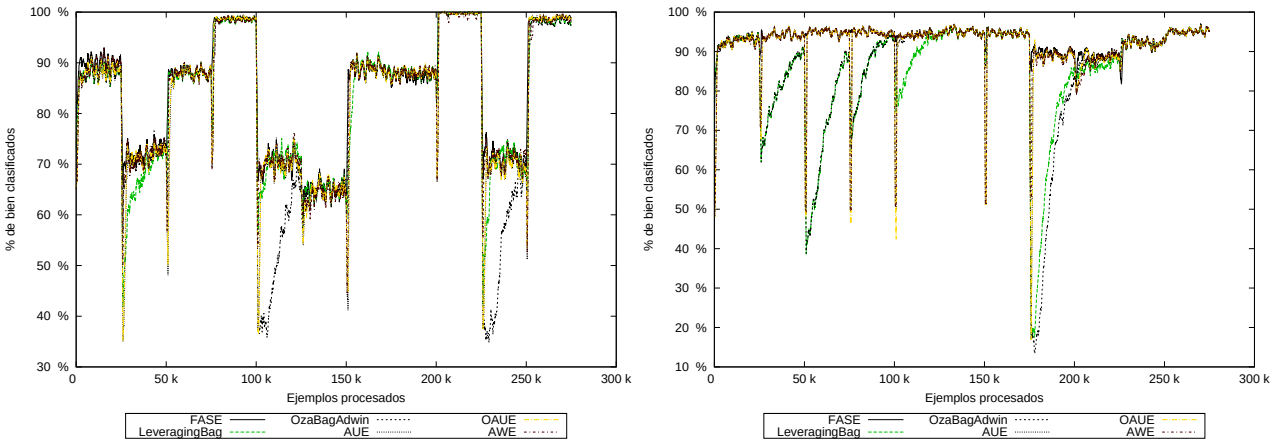
Algoritmo	LeveragingBag			OzaBagAdwin			AUE			AWE			OAUE			Promedio
	G	P	E	G	P	E	G	P	E	G	P	E	G	P	E	
FASE	11	9	0	13	7	0	19	1	0	14	6	0	18	2	0	**82,47**
LeveragingBag				8	12	0	20	0	0	13	7	0	20	0	0	82,34
OzaBagAdwin							19	1	0	13	7	0	18	2	0	82,33
AUE										9	10	1	5	14	1	82,04
AWE													10	9	1	82,28
OAUE																82,08

Tabla 4. Rendimiento de los algoritmos sobre cambios abruptos. Los cambios ocurren cada 25 000 ejemplos de entrenamiento. Se generaron 10 cambios.

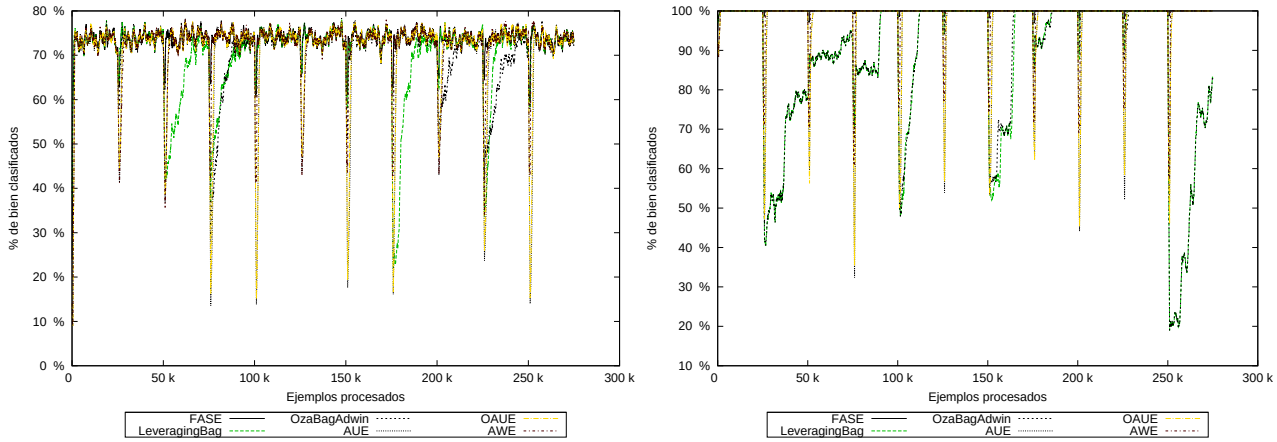
Algoritmo	Medida	AGR	HYP	LED	RBF	SEA	STA	WAVE	Promedio
FASE	$\bar{x}$	**84,37**	**93,16**	**73,56**	71,97	**87,80**	**99,93**	**80,56**	**84,48**
	s	12,14	2,53	2,64	1,56	1,54	0,37	1,29	-
LeveragingBag	$\bar{x}$	82,98	85,45	70,16	72,00	87,69	87,10	80,49	80,84
	s	13,29	13,87	8,86	1,48	1,56	19,18	1,28	-
OzaBagAdwin	$\bar{x}$	80,57	85,41	71,34	71,98	87,04	87,28	80,48	80,59
	s	17,28	15,59	6,46	1,50	2,03	18,98	1,30	-
AUE	$\bar{x}$	82,69	91,89	70,96	72,05	87,43	97,83	80,42	83,32
	s	13,88	7,09	10,31	1,90	3,37	8,40	2,83	-
AWE	$\bar{x}$	83,50	92,02	72,27	**72,55**	87,14	98,88	81,32	83,95
	s	12,65	6,34	6,87	1,98	3,47	5,21	2,88	-
OAUE	$\bar{x}$	82,87	91,85	71,27	72,00	87,53	97,88	80,43	83,40
	s	13,75	7,27	9,63	1,90	3,33	8,27	2,83	-

Tabla 5. Rendimiento de los algoritmos sobre cambios graduales. Los cambios ocurren cada 25 000 ejemplos de entrenamiento con 5 000 ejemplos en el período de transición entre conceptos consecutivos. Se generaron 10 cambios.

Algoritmo	Medida	AGR	HYP	LED	RBF	SEA	STA	WAVE	Promedio
FASE	$\bar{x}$	81,70	**90,53**	70,96	71,99	87,20	**96,28**	**80,55**	**82,74**
	s	12,67	6,19	6,73	1,55	1,89	8,29	1,30	-
LeveragingBag	$\bar{x}$	81,74	90,18	71,10	71,99	86,86	96,05	80,47	82,63
	s	12,68	6,30	6,69	1,51	2,05	8,80	1,29	-
OzaBagAdwin	$\bar{x}$	81,73	90,14	**71,20**	71,99	86,72	95,92	80,48	82,60
	s	12,81	6,37	6,47	1,51	2,06	9,08	1,31	-
AUE	$\bar{x}$	81,50	89,68	70,14	72,07	87,30	95,56	80,42	82,38
	s	13,17	7,74	9,00	1,90	3,33	9,82	2,78	-
AWE	$\bar{x}$	**81,89**	89,90	69,69	**72,39**	86,90	96,07	81,31	82,59
	s	12,77	6,71	9,51	1,94	3,42	8,74	2,82	-
OAUE	$\bar{x}$	81,64	89,73	70,29	72,05	**87,36**	95,64	80,42	82,45
	s	13,09	7,68	8,81	1,89	3,32	9,73	2,78	-



(a) Precisión de los algoritmos (generador AGR) sobre cambios abruptos. **(b)** Precisión de los algoritmos (generador HYP) sobre cambios abruptos.



(c) Precisión de los algoritmos (generador LED) sobre cambios abruptos. **(d)** Precisión de los algoritmos (generador STA) sobre cambios abruptos.

Figure 2. Precisión de los algoritmos sobre varios generadores.

Tabla 6. Principales características de los datasets reales seleccionados

Dataset	Acrónimo	Ejemplos	Nominal	Numérico	Valores perdidos	Clases
Electricity	ELE	45.312	1	7	sí	2
Forest Cover	COV	581.012	44	10	no	7
KDD Cup 10%	KDD	494.021	7	34	no	2
Nursery	NUR	12.960	8	0	no	5
Spam corpus 2	SA1	9.325	500	0	no	2
Spam 2	SA2	4.601	1	57	no	2
Usenet 1	USE1	1.500	100	0	no	2
Usenet 2	USE2	1.500	100	0	no	2
Connect-4	CON	67.557	21	0	no	3
EEG Eye State	EYE	14.980	0	14	no	2
Poker Hand	POK	1.000.000	0	11	no	10
Bank marketing	BAN	41.188	9	7	no	2

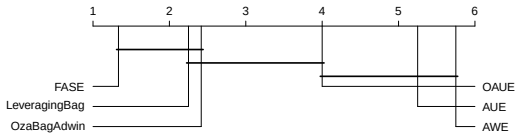


Figura 3. Comparación entre todos los algoritmos sobre datos reales utilizando el test de Friedman y el procedimiento de Bergmann Hommel para el análisis post hoc. Los algoritmos que están conectados (p-value = 0.10) no existen diferencias significativas. Todos los algoritmos utilizan Naïve Bayes como clasificador base.

los ensambles que utilizan bloques de instancias.

6. Conclusiones

La generación constante de grandes cantidades de datos en el tiempo y el cambio de concepto son dos de los problemas más complejos en la minería de flujos de datos. En este trabajo se presentaron dos enfoques para solucionar este tipo de problemas. Estos enfoques son los ensambles incrementales y los ensambles basados en bloques de instancias. Los ensambles incrementales utilizan clasificadores bases incremenatales de manera que siempre están actualizados con respecto a los datos más recientes. Mientras que los ensambles que utilizan bloques van dividiendo el flujo de datos en bloques de datos de igual tamaño y actualizan los clasificadores bases con cada uno de estos bloques. En el estudio experimental realizado se demostró que los ensambles incrementales obtienen mejores resultados frente a conceptos estables debido a que los clasificadores bases siempre están actualizados con respecto a los datos más recientes. No ocurre así con los ensambles que utilizan bloque ya que van eliminado clasificadores cada vez que se completa un nuevo bloque de instancias.

En los experimentos con cambios de concepto ambos enfoques obtienen buenos resultados, pero no se puede afirmar que con los conjuntos de datos utilizados un enfoque es mejor que otro. Sin embargo frente a conjuntos de datos reales los ensambles incrementales obtienen mejores resultados, es decir son más estables ante situaciones donde los cambios pueden ser tanto abruptos como graduales.

Referencias

[1] Stephen Bach and Mark Maloof. A bayesian approach to concept drift. In *Advances in Neural Information Processing Systems*, pages 127–135, 2010. 1

[2] Stephen H Bach, Marcus Maloof, and others. Paired learners for concept drift. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 23–32. IEEE, 2008. 1

[3] Manuel Baena-Garcia, Jose del Campo-Avila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method. 2006. 1

[4] Michèle Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. 1

[5] Albert Bifet, Eibe Frank, Geoffrey Holmes, and Bernhard Pfahringer. Accurate Ensembles for Data Streams: Combining Restricted Hoeffding Trees using Stacking. In *ACML*, pages 225–240, 2010. 5.2

[6] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007. 1, 4.2

[7] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010. 5, 5.2.2

[8] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Machine learning and knowledge discovery in databases*, pages 135–150. Springer, 2010. 5.1, 5.2

[9] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavalda. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148. ACM, 2009. 4.2, 5.1, 5.2

[10] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 4.2

Tabla 7. Rendimiento de los algoritmos en conjuntos de datos reales.

Algoritmo	Medida	BAN	CONE	COV	ELE	EYE	KDD	NUR	POK	SA1	SA2	USE1	USE2
FASE	$\bar{x}$	**89,54**	74,58	**88,16**	**85,25**	99,05	**99,83**	**93,30**	**76,56**	**99,77**	**92,03**	**76,93**	**74,00**
	s	10,90	13,18	7,46	6,40	2,58	1,03	5,68	9,56	1,45	6,35	10,65	11,25
LeveragingBag	$\bar{x}$	89,94	**75,07**	83,20	78,84	90,61	99,62	91,19	73,08	97,64	91,67	65,67	73,27
	s	11,17	13,72	11,92	11,97	17,88	3,14	8,13	12,58	5,75	10,08	20,78	10,40
OzaBagAdwin	$\bar{x}$	89,79	74,87	83,07	78,91	**90,91**	99,66	90,30	73,48	98,00	90,53	64,07	72,33
	s	11,39	13,99	11,96	12,13	18,26	2,88	9,27	12,35	5,19	10,84	20,50	11,38
AUE	$\bar{x}$	88,30	69,09	80,09	74,96	57,15	21,48	80,52	62,39	58,79	74,96	61,20	64,33
	s	14,85	18,15	14,72	15,40	42,02	40,38	19,54	21,27	46,22	32,50	22,82	6,12
AWE	$\bar{x}$	85,35	64,33	80,15	71,94	57,78	21,25	79,24	58,20	59,72	74,54	60,60	61,67
	s	19,10	19,62	15,17	16,79	41,35	40,26	17,59	22,20	47,32	32,32	22,29	8,59
OAUE	$\bar{x}$	88,50	72,96	80,86	76,98	63,87	22,99	87,25	68,50	75,30	81,96	63,67	70,80
	s	15,09	15,51	13,93	13,86	40,58	41,56	14,95	16,03	38,36	27,57	18,63	7,68

[11] D. Brzezinski and J. Stefanowski. Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94, January 2014. 4.1, 5.1, 5.2

[12] Dariusz Brzezinski and Jerzy Stefanowski. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265:50 – 67, 2014. 4.1, 5.1, 5.2

[13] Bojan Cestnik. Estimating probabilities: a crucial task in machine learning. In *ECAI*, volume 90, pages 147–149, 1990. 5.1

[14] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine learning*, 3(4):261–283, 1989. 5.1

[15] Padraig Cunningham, Niamh Nowlan, Sarah Jane Delany, and Mads Haahr. A case-based approach to spam filtering that can track concept drift. In *The ICCBR*, volume 3, pages 03–2003, 2003. 1

[16] A. P. Dawid. Present Position and Potential Developments: Some Personal Views: Statistical Theory: The Prequential Approach. *Journal of the Royal Statistical Society. Series A (General)*, 147(2):278–292, 1984. 5

[17] Magdalena Deckert. Batch weighted ensemble for mining data streams with concept drift. In *Foundations of Intelligent Systems*, pages 290–299. Springer, 2011. 4.1

[18] José Del Campo Ávila. Nuevos enfoques en aprendizaje incremental. 2007. 3.3

[19] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006. 5.2.1

[20] Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997. 5.1

[21] Yoav Freund and Robert E. Schapire. A Short Introduction to Boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999. 4.2

[22] Isvani Frias-Blanco, Jose del Campo-Avila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yaile Caballero-Mota. Online and Non-Parametric Drift Detection Methods Based on Hoeffding Bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, March 2015. 1, 3.3, 4.2, 5.2

[23] Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Andre CPLF Carvalho, Agustín Ortiz-Díaz, and Rafael Morales-Bueno. Online adaptive decision trees based on concentration inequalities. *Knowledge-Based Systems*, 104:179–194, 2016. 1, 3.3, 5.2

[24] Isvani Frías Blanco, José del Campo Ávila, Gonzalo Ramos Jiménez, Rafael Morales Bueno, Agustín Ortiz Díaz, and Yailé Caballero Mota. Aprendiendo con detección de cambio online. *Computación y Sistemas*, 18(1):169–183, 2014. 5.2

[25] Isvani Frías-Blanco, Alberto Verdecia-Cabrera, Agustín Ortiz-Díaz, and Andre Carvalho. Fast adaptive stacking of ensembles. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 929–934. ACM, 2016. 1, 4.2, 5.1, 5.2

[26] Keinosuke Fukunaga and Raymond R Hayes. Estimation of classifier performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(10):1087–1101, 1989. 1

[27] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010. 1

[28] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances*

in artificial intelligence, pages 286–295. Springer, 2004. 1, 2

[29] Salvador García, Francisco Herrera, and John Shawe-taylor. An extension on —statistical comparisons of classifiers over multiple data sets‖ for all pairwise comparisons. *Journal of Machine Learning Research*, pages 2677–2694. 5.2.1

[30] Michael Bonnell Harries, Claude Sammut, and Kim Horn. Extracting hidden context. *Machine learning*, 32(2):101–126, 1998. 1

[31] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, 2001. 1, 3.3, 5

[32] Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004. 1

[33] Ralf Klinkenberg and Thorsten Joachims. Detecting Concept Drift with Support Vector Machines. In *ICML*, pages 487–494, 2000. 1

[34] Miroslav Kubat and Gerhard Widmer. Adapting to drift in continuous domains. In *European Conference on Machine Learning*, pages 307–310. Springer, 1995. 1, 3.1

[35] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of Bayesian classifiers. In *Aaai*, volume 90, pages 223–228, 1992. 5.1

[36] Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 22(5):730–742, 2010. 2

[37] Leandro L. Minku and Xin Yao. DDD: A new ensemble approach for dealing with concept drift. *Knowledge and Data Engineering, IEEE Transactions on*, 24(4):619–633, 2012. 4.2

[38] Agustin Ortiz Diaz, Jose del Campo-Avila, Gonzalo Ramos-Jimenez, Isvani Frias Blanco, Yaile Caballero Mota, Antonio Mustelier Hechavarria, and Rafael Morales-Bueno. Fast Adapting Ensemble: A New Algorithm for Mining Data Streams with Concept Drift. *The Scientific World Journal*, 2014. 4.1

[39] Nikunj C. Oza. Online bagging and boosting. In *Systems, man and cybernetics, 2005 IEEE international conference on*, volume 3, pages 2340–2345. IEEE, 2005. 4.2

[40] Nikunj C. Oza and Stuart Russell. Online Bagging and Boosting. In Tommi Jaakkola and Thomas Richardson, editors, *Eighth International Workshop on Artificial Intelligence and Statistics*, pages 105–112, Key West, Florida. USA, January 2001. Morgan Kaufmann. 4.2

[41] Michael J Pazzani. Searching for dependencies in Bayesian classifiers. In *Learning from Data*, pages 239–248. Springer, 1996. 5.1

[42] Jeffrey C Schlimmer and Douglas Fisher. A case study of incremental concept induction. In *AAAI*, pages 496–501, 1986. 1

[43] Kenneth O Stanley. Learning concept drift with a committee of decision trees. *Informe tecnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003. 1

[44] W. Nick Street and YongSeog Kim. A Streaming Ensemble Algorithm (SEA) for Large-scale Classification. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 377–382, New York, NY, USA, 2001. ACM. 4.1

[45] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM, 2003. 4.1, 5.1

[46] Gerhard Widmer and Miroslav Kubat. Effective learning in dynamic environments by explicit context tracking. In *Machine learning: ECML-93*, pages 227–243. Springer, 1993. 1

[47] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996. 1, 2