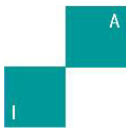
Inteligencia Artificial 21(62) (2018), 145-158
doi: 10.4114/intartif.vol21iss62pp114-158



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Learning with ensembles from non-stationary data streams

Aprendiendo con ensambles a partir de flujos de datos no estacionarios

Alberto Verdecia-Cabrera[1], Isvani Frías-Blanco[2], Luis Quintero-Domínguez[3], Yanet Rodríguez Sarabia[4]

[1]Universidad de Granma
Centro de Investigaciones de la Informática, Universidad Central "Marta Abreu" de Las Villas, Cuba
averdeciac@gmail.com

[2]LexisNexis Risk Solutions. Sao Paulo, Brazil
justisvani@gmail.com

[3]Universidad de Sancti Spíritus, Cuba
lquintero@uclv.cu

[4]Universidad Central "Marta Abreu" de Las Villas, Cuba
yrsarabia@uclv.edu.cu

Abstract Nowadays many sources generate massive data continuously, without control of the arrival order and at high speed. Internet, cell-phones, cars, and security sensors are examples of such sources. Because of the temporal dimension of the data (they are constantly arriving over time), and the dynamism of many real-world situations, the target function to be learned can change over time. This situation, known as concept drift, complicates the task of estimating this target function, because a previous learning model can become outdated, or even contradictory regarding the most recent data. There are several algorithms to manipulate concept drift, and among them are the classifier ensembles. In this article, we present a new ensemble algorithm called ADCE, able for learning from data streams with concept drift. ADCE manipulates these changes using a change detector in each base classifier. When the detector estimates a change, the classifier in which the change was estimated is replaced by a new one. ADCE combines the simplicity of the bagging algorithm to train base classifiers with methods used in batch learning to combine the output of the base classifiers. The proposed algorithm is compared empirically with several bagging family ensemble algorithms for data streams. The experimental results show that the proposed algorithm constitutes a viable option for learning from concept drifting data streams.

Resumen En la actualidad muchas fuentes generan grandes volúmenes de datos constantemente, sin control del orden de llegada y a altas velocidades. Internet, teléfonos celulares, automóviles y sensores de seguridad son ejemplos de tales fuentes. Debido a la dimensión temporal de los datos (que están llegando constantemente a lo largo del tiempo) y al dinamismo de muchas situaciones del mundo real, la función objetivo que se debe aprender puede cambiar con el tiempo. Esta situación, conocida como cambio de concepto, complica la tarea de estimar esta función objetivo, porque un modelo de aprendizaje anterior puede quedar desactualizado o incluso contradictorio con respecto a los datos más recientes. Existen varios algoritmos para manipular cambios de concepto, entre los cuales se encuentran los ensambles de clasificadores. En este artículo se presenta un nuevo algoritmo de ensamble llamado ADCE, capaz de aprender a partir de flujos de datos con cambios de concepto. ADCE manipula estos cambios utilizando un detector de cambios en cada clasificador base. Cuando el detector estima un cambio, el clasificador en que se estimó el cambio es remplazado por uno nuevo. ADCE combina la simplicidad del algoritmo

ISSN: 1988-3064(on-line)

bagging para entrenar clasificadores base con métodos utilizados en el aprendizaje por lotes para combinar la salida de los clasificadores base. El algoritmo propuesto se compara empíricamente con varios algoritmos de ensamble de la familia bagging para flujos de datos. Los resultados experimentales muestran que el algoritmo propuesto constituye una opción viable para el aprendizaje de flujos de datos con cambios de concepto.

Keywords: Data stream, classifier ensemble, concept drift.
Palabras Clave: Flujos de datos, ensambles de clasificadores, cambio de concepto.

Introducción

En la actualidad, el volumen de los datos generados por sensores, Internet, dispositivos de localización, telefonía y muchos otros, está en constante aumento. El tamaño de estos datos es potencialmente infinito debido a su constante generación, por lo que es necesario procesarlos con recursos limitados de cómputo. Para este procesamiento es factible el uso de técnicas de aprendizaje automático. Dependiendo de cómo se presenten los ejemplos de entrenamiento, el aprendizaje automático se puede clasificar en dos tipos [26]: aprendizaje por lotes y aprendizaje en línea. En el aprendizaje por lotes, los datos deben ser recolectados y almacenados antes de ser procesados, por lo que su uso no es factible para procesar datos generados constantemente en el tiempo. Sin embargo, el aprendizaje en línea permite procesar flujos de datos de manera secuencial [17].

En las tareas de clasificación, un flujo de datos es comúnmente definido 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. Debido a la dimensión temporal de los datos (estos son adquiridos en el tiempo) y la dinámica de muchas situaciones reales, la distribución de probabilidad que regula a los mismos (también llamada concepto) puede cambiar con el tiempo, un problema conocido comúnmente como cambio de concepto. Consecuentemente, los algoritmos de aprendizaje para la minería de flujos de datos deben ser actualizados con respecto a los conceptos más recientes [16].

Los métodos de ensambles de clasificadores han recibido en los últimos tiempos gran atención para el modelado y la clasificación de flujos de datos no estacionarios [5]. Los métodos de ensambles combinan las predicciones de los clasificadores base con el objetivo de mejorar la precisión obtenida por estos clasificadores de forma individual. Con el fin de manipular cambios de concepto se utilizan medidas de rendimiento para monitorizar la consistencia del ensamble en relación con los nuevos datos. Variaciones significativas en los valores de rendimiento se interpretan como un cambio de concepto y los métodos de ensamble eliminan, reactivan o añaden nuevos clasificadores base dinámicamente en respuesta a estas variaciones.

El mecanismo de adaptación a los cambios de concepto de varios algoritmos de ensambles propuestos en la literatura consiste en eliminar el peor clasificador base cuando se detecta un cambio, seguido por el entrenamiento de un nuevo clasificador base y su posterior inclusión dentro del ensamble [3, 4]. Por lo que, estos algoritmos solo manipulan cambios de concepto a nivel de ensamble y no en los clasificadores base. El peor clasificador en el ensamble puede no ser el afectado por el cambio de concepto actual. Por lo tanto, la estrategia adoptada por algoritmos de ensambles anteriores puede conducir a una adaptación ineficaz a los cambios. Además, en el aprendizaje en línea, los métodos para combinar las predicciones de los clasificadores base se han enfocado en el voto ponderado. Sin embargo la relación subyacente entre las predicciones de los clasificadores y la clase verdadera puede ser más compleja que una combinación lineal de las predicciones. Por lo que en este trabajo se utilizan otros métodos que han sido utilizados en el aprendizaje por lotes para combinar las salidas de los clasificadores base.

En este trabajo se presenta un nuevo algoritmo que combina la simplicidad del algoritmo Bagging [28] para entrenar clasificadores en flujos de datos, con métodos utilizados en el aprendizaje por lotes para combinar clasificadores y con un nuevo mecanismo de adaptación a los cambios de concepto. Para detectar los cambios de concepto en los clasificadores base se utilizó HDDM (Hoeffding Drift Detection Method) [12] como detector de cambios de concepto y estimador 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, *aviso* cuando es probable que se aproxime un cambio, y *fuera-de-control* cuando se detecta el cambio. Entonces, cuando el detector estima una cambio, se sustituye el clasificador en el que se detectó el cambio por uno nuevo.

El resto de este artículo está estructurado como sigue. Primero, en la Sección 1 se presentan las definiciones fundamentales y tipos de cambios de concepto más comunes. Luego, en la Sección 2 se describen varios algoritmos basados en ensambles de clasificadores. Posteriormente en la Sección 3 se describen los métodos utilizados en este trabajo para combinar clasificadores y en la Sección 4 se describe el algoritmo propuesto. En la Sección 5 se describe la configuración del estudio empírico para evaluar el rendimiento del algoritmo propuesto y se muestran y discuten los resultados obtenidos sobre datos sintéticos y datos reales, los cuales demuestran la validez del algoritmo propuesto. Finalmente en la Sección 6 se presentan las conclusiones de este artículo.

1. Cambio de concepto

Dentro del aprendizaje en línea [33], el problema de clasificación se define generalmente para una secuencia (posiblemente infinita) de ejemplos $S = e_1, e_2, ..., e_i...$ que se obtienen en el tiempo, normalmente uno a la vez y no necesariamente dependientes del tiempo. Cada ejemplo de entrenamiento $e_i = (\vec{a_i}, c_i)$ está formado por un vector $\vec{a_i}$ y un valor discreto c_i; donde cada vector $\vec{a_i} \in \vec{\mathcal{A}}$ tiene las mismas dimensiones y cada dimensión se llama atributo. El valor discreto c_i es llamado etiqueta y tomado de un conjunto finito de clases $\mathcal{C}$.

Concepto se refiere a la función de distribución de probabilidad del problema en un punto determinado en el tiempo [23]. Este concepto puede ser caracterizado por la distribución de probabilidad conjunta $P\left(\vec{\mathcal{A}}, \mathcal{C}\right)$ donde $\vec{\mathcal{A}}$ representa los atributos y $\mathcal{C}$ es la clase. Por tanto un cambio en la distribución del problema (también conocido como contexto) [15] implica un cambio de concepto. Gama y otros [16] distinguen dos tipos fundamentales de cambios de concepto que están presentes en muchos problemas reales:

1. *Cambios de concepto reales:* se refieren a cambios en la distribución de probabilidad a posteriori de las clases $p(\mathcal{C} \mid \mathcal{A})$. Estos cambios pueden ocurrir sin que ocurran cambios en la distribución de los datos de entrada $p(\mathcal{A})$.

2. *Cambios de concepto virtuales:* ocurren si la distribución de los datos de entrada cambia (es decir $p(\mathcal{A})$ cambia) sin afectar $p(\mathcal{C} \mid \mathcal{A})$.

Los tipos de cambio de concepto también pueden clasificarse en cuanto a su extensión (*o tasa de cambio*) [21, 18, 19, 20]: cambios locales y globales. En cambios locales, la distribución cambia solo en una región limitada del espacio de instancias. En el caso de los cambios globales, la distribución cambia sobre toda la región del espacio de instancias, es decir, para todos los valores posibles de las clases y los atributos. Otro aspecto importante a considerar es el período de transición entre conceptos consecutivos (*velocidad del cambio*), y en este caso tenemos cambios abruptos y graduales. Los cambios abruptos ocurren cuando la transición entre conceptos consecutivos es instantánea y los cambios graduales ocurren cuando el período de transición contiene cierto número de ejemplos. Otro tipo común de cambio es el recurrente, que ocurre cuando los conceptos pueden reaparecer [27].

2. Trabajos Relacionados

Los algoritmos basados en ensambles de clasificadores combinan la predicción de los clasificadores base y se pueden clasificar por la forma en que actualizan sus clasificadores base. 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 algoritmos generalmente es lenta porque tienen que esperar a que se llene el bloque para actualizar los clasificadores base. La segunda clasificación de los ensambles es por la actualización de los clasificadores de forma incremental. Estos métodos utilizan como clasificadores base clasificadores incrementales, es decir actualizan los modelos en la medida que se obtienen las instancias. Los métodos propuestos en este artículo están dentro de esta clasificación.

2.1. Ensambles que utilizan bloques de instancias

Uno de los primeros algoritmos de ensamble para el procesamiento de flujos de datos fue SEA (*Streaming Ensemble Algorithm*) [31]. SEA crea los clasificadores base a partir de pequeños subconjuntos de los datos, leídos secuencialmente en bloques de un tamaño fijo. El tamaño de esos bloques es un parámetro importante porque es responsable del equilibrio entre precisión y flexibilidad. Cada bloque se utiliza para entrenar un nuevo clasificador, el cual es comparado con los demás miembros del ensamble. Si alguno de los miembros es peor que el nuevo clasificador, éste se sustituye por el nuevo. Para evaluar los clasificadores, Street y Kim proponen utilizar la precisión de clasificación obtenida en el bloque de datos más reciente. El algoritmo cuenta con un límite máximo de clasificadores que actúa como mecanismo de adaptación. Al ser alcanzado este límite máximo obliga al algoritmo sustituir a clasificadores base anteriores siguiendo cierto criterio de reemplazo. Para combinar las predicciones de los clasificadores base utiliza voto mayoritario 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 de SEA, Wang et al. [32] propusieron el método AWE (*Accuracy Weighted Ensemble*). AWE combina las respuestas de los clasificadores base a través del voto mayoritario ponderado. La ponderación de los clasificadores base 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 base.

BWE (*Batch Weighted Ensemble*) [9] también divide el conjunto de entrenamiento en bloques de igual tamaño y utiliza el voto mayoritario ponderado para combinar la salida de los clasificadores base. 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 cuando 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.

La mayoría de los ensambles que actualizan los clasificadores base mediante bloques presentan problemas para adaptarse a los cambios de concepto, 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.

2.2. Ensambles incrementales

Bagging [6] y Boosting [11] son dos de los algoritmos más conocidos para entrenar clasificadores base. 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. A diferencia de Bagging, Boosting entrena secuencialmente un conjunto de clasificadores con instancias ponderadas en el cual el peso asociado a cada instancia depende del desempeño del clasificador anterior. El objetivo de Boosting es asignarle mayor peso a las instancias mal clasificadas.

Oza y Rusell [28] 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 base. Para manipular cambios de concepto, Bifet y otros [4] 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*) [1]. El mecanismo de adaptación utilizado se basa en la sustitución del peor de los clasificadores, cuando se estima un cambio, por un nuevo clasificador base creado más recientemente. Otro ensamble basado en Bagging es DDD (*Dealing with Diversity for Drifts*) [24]. DDD alterna entre dos estados: (1) antes de la detección del cambio y (2) después de la detección del cambio. DDD varía la diversidad del ensamble con el objetivo de adaptarse a los cambios de concepto rápidamente.

3. Métodos para combinar clasificadores

La manera en que se combinan los resultados de los clasificadores individuales es una cuestión fundamental en el estudio de los ensambles (combinaciones) de clasificadores. La idea aquí es optimizar la decisión que se va a tomar a partir de las decisiones individuales. Múltiples métodos de combinación de clasificadores han sido utilizados con éxito en el aprendizaje por lotes tradicional. En este trabajo se seleccionaron tres de los métodos más conocidos para la combinación de clasificadores base en el aprendizaje a partir de flujos de datos no estacionarios. Además, se propone el uso del método de control estadístico de procesos EWMA *(Exponentialy Weighted Moving Average)* ya que permite asignar más o menos importancia a la precisión actual y al peso anterior del clasificador. A continuación se describen los métodos de combinación empleados.

Naıve Bayes (NB)

La regla de Bayes ha tenido múltiples aplicaciones en la teoría de las probabilidades y en el aprendizaje automático. Usando la regla de Bayes, se puede extender la idea de Naive Bayes para combinar varios clasificadores [29] de la siguiente manera:

$$class\,(x) = \underset{\substack{c_j \in dom(y) \\ P(y,=c_j)>0}}{\operatorname{argmax}} \hat{P}\,(y = c_j) \times \prod_{k=1} \frac{\hat{P}_{M_k}\,(y = c_j|x)}{\hat{P}\,(y = c_j)} \tag{1}$$

donde M_k denota al clasificador k y $\hat{P}_{M_k}\,(y = c_j|x)$ denota la probabilidad de que y obtenga el valor c_j dada una instancia x.

EWMA

EWMA (*Exponentialy Weighted Moving Average*) es un método de control estadístico de procesos. La característica esencial de este método es que otorga menos peso a las observaciones cuanto más alejadas están en el tiempo. El estadístico que se representa en el gráfico es:

$$Ewma_{M_k} \rightarrow W_{M_k} = \beta x_{M_k} + (1 - \beta)\, W_{M_k} \tag{2}$$

donde x_{M_k} es una variable aleatoria , β es una constante para determinar el peso de las observaciones, $(0 < \beta < 1)$.

En este trabajo, la ecuación 2 se utiliza para determinar el peso de los clasificadores base. En este caso, W_{M_k} sería el peso de cada clasificador y x_{M_k} la precisión actual. Como cada clasificador utiliza un detector de cambio, se puede estimar en cualquier momento el error ϵ_m de cada clasificador base, por lo que la precisión sería $1 - \epsilon_m$. La constante β preestablecida representa el nivel de importancia que se les otorga al funcionamiento de los clasificadores base frente a los datos antiguos y frente a los datos actuales respectivamente. Un valor elevado de β 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.

Entropy Weighting (EW)

La idea de este método de combinación es asociarle a cada clasificador un peso que es inversamente proporcional a la entropía de su vector de clasificación [29]:

$$class\,(x) = \underset{c_i \in dom(y)}{\operatorname{argmax}} \sum_{k:c_i = \underset{c_j \in dom(y)}{\operatorname{argmax}}\, \hat{P}_{M_k}(y=c_j|x)} Ent\,(M_k, x) \tag{3}$$

donde:

$$Ent\,(M_k, x) = -\sum_{c_j \in dom(y)} \hat{P}_{M_k}\,(y = c_j|x) \times \log\left(\hat{P}_{M_k}\,(y = c_j|x)\right) \tag{4}$$

permite calcular la entropía para el clasificador M_k y la instancia x.

Dempster–Shafer (DS)

La idea de usar la teoría de la evidencia de Dempster–Shafer para combinar modelos fue propuesta por [30]. Este método utiliza la noción de asignación de probabilidades básica definida para una cierta clase c_i dada la instancia x:

$$bpa\left(c_i, x\right) = 1 - \prod_k \left(1 - \hat{P}_{M_k}\left(y = c_i | x\right)\right) \tag{5}$$

En consecuencia, la clase seleccionada es la que maximiza el valor de la función:

$$Bel\left(c_i, x\right) = \frac{1}{A} \times \frac{bpa\left(c_i, x\right)}{1 - bpa\left(c_i, x\right)} \tag{6}$$

donde A es un factor de normalización definido como:

$$A = \sum_{\forall c_i \in dom(y)} \frac{bpa\left(c_i, x\right)}{1 - bpa\left(c_i, x\right)} + 1 \tag{7}$$

Algoritmo 1: ADCE

Entrada:
ejemplos: ejemplos de entrenamiento con etiquetas de clase $y_i \in Y = 1, \ldots, L$
K: tamaño del ensamble
Comb: método de combinación de clasificadores
Salida :
$\mathfrak{C}\left(x\right)$ = combinar los clasficadores base M_k utilizando *Comb* para todo $k \in \{1, 2, \ldots, K\}$

```
1 inicio
2     Inicializar los clasficadores base M_k, detectores D_k para todo k ∈ {1, 2, ..., K}
3     para cada ejemplo de entrenamiento hacer
4         para k ← 1 hasta K hacer
5             K ← P(λ)    // P(λ) es la distribución de Poisson con media λ
6             para j ← 1 hasta K hacer
7                 Actualizar M_k y D_k con el ejemplo actual
8             si D_k estima un cambio entonces
9                 reiniciar M_k y D_k
```

4. Algoritmo propuesto

El algoritmo propuesto llamado ADCE utiliza la versión en línea del algoritmo Bagging para entrenar los clasificadores base y uno de los métodos de combinación de clasificadores descritos en la sección anterior para determinar los pesos de los clasificadores. ADCE manipula cambios de concepto de una manera simple y eficiente (ver algoritmo 1). Cada clasificador base M_k $(1 < k < K)$ utiliza un detector de cambios $\mathfrak{D}_k$ para estimar la tasa de error ϵ_k de cada clasificador. Cuando $\mathfrak{D}_k$ estima un cambio, el clasificador M_k es remplazado por uno nuevo. Para combinar la salida de los clasificadores base se puede utilizar cualquiera de los métodos mencionados anteriormente. El algoritmo solo recibe como parámetros el detector de cambios y el número de clasificadores base.

La tasa de error de los clasificadores base se monitoriza constantemente a medida que llega cada ejemplo de entrenamiento. Por lo tanto, este seguimiento también debe realizarse con recursos computacionales controlados. En los últimos años se han propuesto varios métodos en la comunidad estadística para detectar cambios en línea [25]. Sin embargo, estos suponen que los datos de entrada están regulados por una distribución de probabilidad conocida. ADCE utiliza HDDM (*Hoeffding Drift Detection Method*)

[12] como detector de cambios de concepto y estimador de error. HDDM procesa cada valor entrante con una complejidad computacional constante y proporciona garantías matemáticas para las tasas de falsos positivos y falsos negativos.

Tabla 1: Principales características de los conjuntos de datos utilizados en los experimentos.

Conjunto de Datos	Acrónimo	Eejemplos	Nominal	Numérico	Valores perdidos	Clases
LED display	LED	1,000,000	24	0	no	10
SEA	SEA	1,000,000	0	3	no	2
Radial base functions	RBF	1,000,000	0	10	no	2
Waveform	WAV	1,000,000	0	40	no	3
Agrawal	AGR	1,000,000	3	6	no	2
Stagger	STA	1,000,000	3	0	no	2
Hyperplane	HYP	1,000,000	0	10	no	2
Usenet 1	USE1	1,500	100	0	no	2
Usenet 2	USE2	1,500	100	0	no	2
Segment	SEG	2,310	0	19	no	7
Mushroom	MUS	8,124	22	0	yes	2
Spam	SPA1	4,601	1	57	mo	2
Spam corpus 2	SPA2	9,323	500	0	no	2
Nursery	NUR	12,960	8	0	no	5
EEG Eye State	EYE	14,980	0	14	no	2
Weather	WEA	18,159	0	8	no	2
Bank marketing	BAN	41,188	9	7	no	2
Electricity	ELE	45,312	1	7	yes	2
Connect-4	CON	67,557	21	0	no	3
KDD Cup 10 %	KDD	494,021	7	34	no	2
Forest Cover	COV	581,012	44	10	no	7
Poker Hand	POK	1,000,000	10	0	no	10

5. Evaluación experimental

Esta sección describe la evaluación experimental realizada al algoritmo propuesto con respecto a los algoritmos basados en Bagging más relevantes encontrados en la literatura. El algoritmo propuesto tiene cuatro variantes, una por cada método de combinación, en lo adelante ADCE+NB, ADCE+WMV, ADCE+EW y ADCE+DS. Por lo que, el estudio experimental tiene dos objetivos fundamentales. El primero es seleccionar la mejor variante del algoritmo propuesto, ya que éste utiliza diferentes métodos para combinar la salida de los clasificadores base. El segundo objetivo es comparar la mejor variante con los algoritmos más conocidos basados en Bagging. Para realizar esta comparación todos los algoritmos se evaluaron en presencia de cambios de concepto (abruptos y graduales) y frente a conjuntos de datos reales.

5.1. Configuración de los experimentos

El rendimiento de los algoritmos considerados se evaluó utilizando los conjuntos de datos más utilizados en la literatura. La Tabla 1 resume sus principales características. Los conjuntos de datos presentan ruido, atributos irrelevantes, atributos nominales y numéricos, y diferentes tipos de funciones objetivo. Como se muestra en la Tabla 1, los algoritmos se evaluaron frente a 15 conjuntos de datos de problemas reales

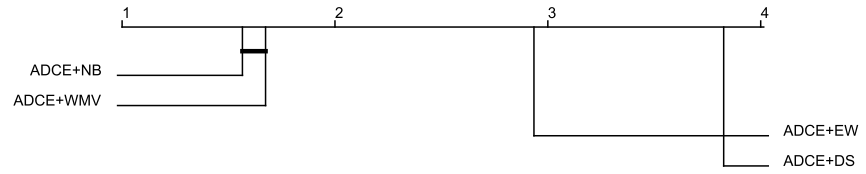


Figura 1: Ranking de los algoritmos. Los algoritmos que no son significativamente diferente están conectados.

y 7 conjuntos de datos de problemas artificiales. Los conjuntos de datos artificiales permiten evaluar los algoritmos controlando los conceptos estables, cambios abruptos, graduales; los datos artificiales también tienen la ventaja de modelar escenarios donde los algoritmos pueden demostrar su desempeño.

En los conjuntos de datos artificiales, se generaron 10 cambios de concepto para cada experimento. Los cambios se generaron cada 25,000 instancias. Los cambios graduales se simularon usando un período de transición entre conceptos consecutivos. Ese período de transición consistió en instancias de entrenamiento con diferentes funciones objetivos. El número de instancias utilizados varió entre experimentos para evaluar el comportamiento de los algoritmos con diferentes tasas de cambio. Más específicamente, se usaron 100, 500, y 1000 instancias de entrenamiento en períodos de transición.

Todos los experimentos fueron ejecutados en MOA [2], 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 de aprendizaje y métodos para generar flujos de datos artificiales con la posibilidad de incluir cambios de concepto. Para la experimentación se seleccionaron los algoritmos basados en Bagging, tales como, OzaBag (versión en línea del algoritmo bagging) [28], OzaBagAdwin [4] y LeveragingBag [3]. Los algoritmos fueron utilizados con la configuración por defecto en MOA. En el detector de cambio HDDM, el nivel de significación para el cambio fue de $0,001$ y para la alerta de $0,005$. Se utilizaron 10 clasificadores base en todos los algoritmos. Como clasificador base se eligió Naive Bayes por ser uno de los algoritmos más exitosos para aprender de los flujos de datos [22, 7, 14], además tiene un costo computacional bajo y una semántica clara.

La medida de rendimiento utilizada para evaluar los algoritmos fue la precisión de la clasificación (accuracy). Esta medida se calculó en línea, con el objetivo de medir cómo 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 sequential*) que se deriva del error predictivo secuencial [8]. 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) [10]. 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 [2].

5.2. Selección de la mejor variante del algoritmo propuesto

El algoritmo propuesto ADCE utiliza la versión en línea del algoritmo bagging para entrenar los clasificadores base. Primeramente se evaluó el algoritmo propuesto en combinación con diferentes métodos para combinar clasificadores: EWMA, NB, EW, DS.

Las combinaciones de ADCE con los diferentes métodos de combinación de clasificadores se evaluaron sobre todos los conjuntos de datos descritos en la sección anterior. Los resultados obtenidos se analizaron utilizando el test de Friedman y el procedimiento de Holm para el análisis post hoc, con un valor de significación de $0,05$. La Figura 1 muestra los rankings de los algoritmos con respecto a la precisión de la clasificación, en la cual las combinaciones entre las que no existen diferencias significativas están conectadas por una línea horizontal. En la misma se muestra que el algoritmo propuesto en combinación con NB obtiene los mejores resultados aunque sin superar significativamente a la combinación con EWMA.

5.3. Resultados frente a cambios abruptos y graduales

En este experimento los algoritmos se ejecutaron frente a cambios abruptos y graduales utilizando los conjuntos de datos artificiales descritos en la Tabla 1. Para simular cambios graduales utilizamos la función sigmoide que está implementada en MOA [2], aumentando la probabilidad de que las nuevas instancias pertenecen al nuevo concepto. Se generaron 10 cambios cada 25,000. Para evaluar el algoritmo frente a cambios graduales el período de transición entre conceptos consecutivos fue de 100, 500, 1000. Las Tabla 2 resume el rendimiento de los algoritmos sobre cambios abruptos en términos de la media y la desviación estándar para la precisión de la clasificación. Los niveles más altos de precisión están resaltados en negrita. Como se puede observar en la Tabla 2 el algoritmo propuesto obtiene los mejores resultados en cuanto a precisión.

En las Tablas 3, 4 y 5 se muestran los resultados de los algoritmos frente a cambios de conceptos graduales. Las Tablas 3, 4 y 5 muestran que el algoritmo también obtiene buenos resultados frente a cambios graduales. Adicionalmente la Figura 2 muestra la precisión de los algoritmos sobre los conjuntos de datos utilizados. Podemos ver que la precisión del algoritmo propuesto no cae drásticamente cuando ocurre el cambio de concepto. Esto se debe principalmente a la eficacia del método utilizado para adaptarse a los cambios de concepto en combinación con NB para combinar la salida de los clasificadores base.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	**83,99 ±12,49**	82,98 ±13,29	80,57 ±17,28	63,38 ±15,64
HYP	**92,98 ±02,79**	85,45 ±13,87	85,41 ±15,59	65,77 ±23,49
LED	**73,53 ±02,90**	70,16 ±08,86	71,34 ±06,46	46,87 ±20,45
RBF	71,96 ±01,54	**72,00 ±01,48**	71,98 ±01,50	71,95 ±01,51
SEA	**87,76 ±01,63**	87,69 ±01,56	87,04 ±02,03	84,42 ±03,65
STA	**99,95 ±00,28**	87,10 ±19,18	87,28 ±18,98	65,78 ±20,05
WAV	**80,52 ±01,28**	80,49 ±01,28	80,48 ±01,30	80,48 ±01,30

Tabla 2: Resultados los algoritmos frente a cambios abruptos.

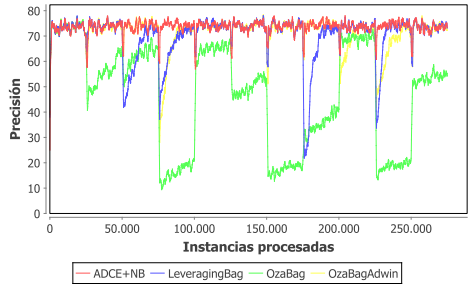
Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	**83,96 ±12,38**	81,89 ±15,24	82,20 ±14,94	63,38 ±15,63
HYP	**91,92 ±03,06**	87,12 ±14,18	90,40 ±07,44	62,12 ±20,64
LED	**73,47 ±03,06**	73,24 ±03,61	73,19 ±03,72	47,95 ±19,20
RBF	71,96 ±01,54	72,00 ±01,48	**72,02 ±01,49**	72,00 ±01,50
SEA	**87,76 ±01,64**	87,47 ±01,86	87,28 ±01,94	84,43 ±03,65
STA	**99,87 ±00,66**	91,79 ±12,37	91,48 ±12,52	65,75 ±19,99
WAV	**80,53 ±01,29**	80,50 ±01,28	80,50 ±01,30	80,50 ±01,30

Tabla 3: Resultados de los algoritmos frente a cambios graduales. El número de instancias en el período de transición fue 100.

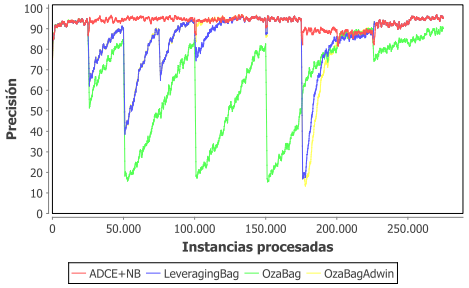
5.4. Resultados con conjuntos de datos reales

En muchos escenarios reales se ha detectado la presencia 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. Los conjuntos de datos reales seleccionados (ver Tabla 1) se han usado en diferentes estudios sobre aprendizaje a partir de flujos de datos no estacionarios [13, 27].

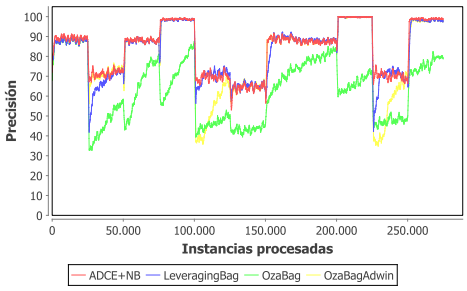
La Tabla 6 muestra el rendimiento de los algoritmos en estos conjuntos de datos reales. Los niveles más altos de precisión están resaltados en negrita. Como se puede ver en la Tabla 6 el algoritmo propuesto obtiene los mejores resultados. Por lo que reemplazar el clasificador en el que se detectó el cambio por uno nuevo es más eficiente que eliminar siempre el peor clasificador. Al eliminar el peor clasificador puede que no se elimine el clasificador donde se detectó el cambio, por lo que el ensamble se adapta más lento



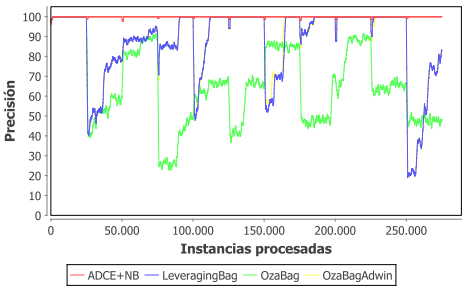
(a) HYP



(b) HYP

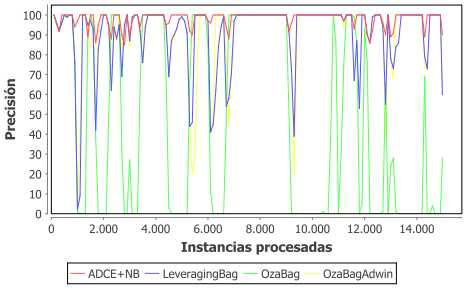


(c) SEA

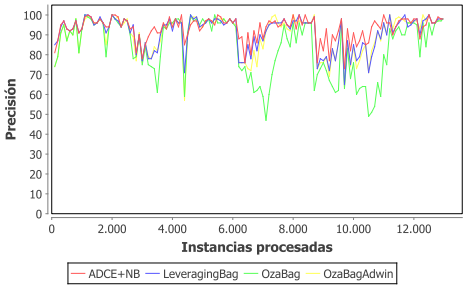


(d) STA

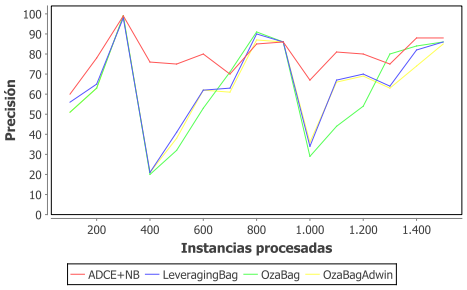
Figura 2: Precisión de los algoritmos frente a cambios abruptos. Se generaron 10 cambios cada 25,000 instancias.



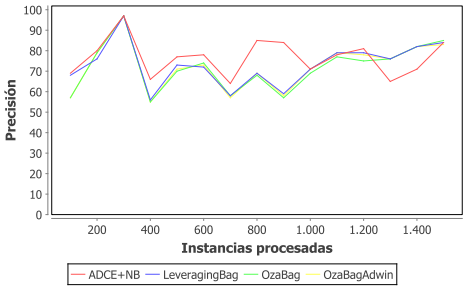
(a) EYE



(b) NUR



(c) USE1



(d) USE2

Figura 3: Precisión de los algoritmos en conjuntos de datos reales.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	**83,76 ±12,42**	83,72 ±12,35	83,72 ±12,30	63,38 ±15,57
HYP	**92,42 ±2,57**	91,72 ±3,94	91,71 ±4,02	63,21 ±18,53
LED	73,32 ±3,46	**73,50 ±2,87**	73,44 ±3,14	46,06 ±19,36
RBF	71,91 ±1,53	**71,97 ±1,48**	71,92 ±1,50	71,91 ±1,50
SEA	**87,73 ±1,64**	87,63 ±1,61	87,01 ±1,91	84,44 ±3,62
STA	**99,56 ±2,08**	99,32 ±3,19	99,11 ±4,17	65,82 ±19,95
WAV	**80,54 ±1,29**	80,50 ±1,28	80,50 ±1,30	80,50 ±1,30

Tabla 4: Resultados de los algoritmos frente a cambios graduales. El número de instancias en el período de transición fue 500.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
AGR	83,46 ±12,48	83,47 ±12,38	**83,49 ±12,35**	63,36 ±15,47
HYP	**92,75 ±03,14**	92,52 ±03,48	92,35 ±03,87	63,84 ±34,24
LED	73,20 ±03,90	73,30 ±03,60	**73,22 ±03,77**	44,66 ±19,23
RBF	**72,01 ±01,55**	71,99 ±01,47	72,00 ±01,49	71,96 ±01,51
SEA	**87,67 ±01,72**	87,41 ±01,68	87,00 ±01,80	84,46 ±03,54
STA	**99,08 ±03,70**	99,01 ±04,30	98,86 ±04,89	65,86 ±19,88
WAV	**80,54 ±01,30**	80,51 ±01,29	80,51 ±01,31	80,51 ±01,31

Tabla 5: Resultados de los algoritmos frente a cambios graduales. El número de instancias en el período de transición fue 1000.

a los cambios. La Figura 3 muestra la precisión de los algoritmos frente a los conjuntos de datos EYE, NUR, USE1 y USE2. En la Figura 3 se puede ver que la precisión del algoritmo propuesta no disminuye significativamente cuando ocurre un posible cambio de concepto.

Para comprobar diferencias significativas entre los algoritmos se utilizó el test de Friedman y el procedimiento de Holm para el análisis post hoc, con un valor de significación de $0,05$. El ranking se calculó con respecto a todos los conjuntos de datos (artificiales y reales). La Figura 4 muestra que, en los conjuntos de datos considerados, el algoritmo propuesto fue significativamente más preciso que el resto de los algoritmos competidores.

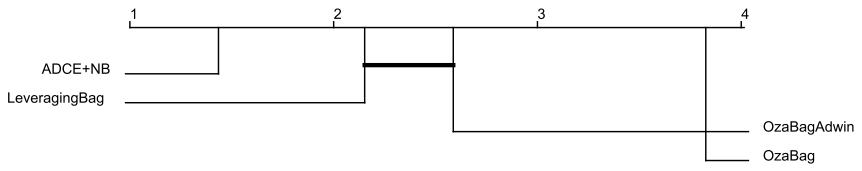


Figura 4: Ranking de los algoritmos. Los algoritmos que no son significativamente diferente están conectados. El ranking se calculó con respecto a las Tablas 2, 3, 4, 5 y 6.

Algoritmo	ADCE+NB	LeveragingBag	OzaBagAdwin	OzaBag
ADU	**83,49 ±03,80**	**83,49 ±03,84**	83,29 ±03,89	83,29 ±03,89
BAN	88,74 ±11,74	**89,94 ±11,17**	89,79 ±11,39	89,15 ±12,58
CAR	**85,56 ±10,11**	82,56 ±10,42	80,67 ±10,78	80,56 ±11,71
CON	74,83 ±12,77	**75,07 ±13,72**	74,87 ±13,99	69,17 ±17,33
COV	**87,36 ±07,70**	83,20 ±11,92	83,07 ±11,96	60,55 ±21,76
ELE	**84,75 ±06,54**	78,84 ±11,97	78,91 ±12,13	74,26 ±14,63
EYE	**98,17 ±03,71**	90,61 ±17,88	90,91 ±18,26	47,31 ±46,41
KDD	**99,80 ±01,29**	99,62 ±03,14	99,66 ±02,88	97,88 ±11,01
MUS	98,82 ±01,75	**99,28 ±01,45**	98,65 ±02,17	98,41 ±02,29
NUR	**93,34 ±05,75**	91,19 ±08,13	90,30 ±09,27	84,11 ±14,18
OUT	**63,20 ±06,69**	60,83 ±10,53	58,33 ±11,80	55,95 ±15,50
POK	**75,69 ±09,71**	73,08 ±12,58	73,48 ±12,35	59,55 ±21,95
SEG	**78,00 ±07,21**	77,83 ±06,88	77,58 ±07,19	77,58 ±07,19
SPA1	**99,38 ±04,04**	97,64 ±05,75	98,00 ±05,19	83,55 ±14,46
SPA2	**93,05 ±05,95**	91,67 ±10,08	90,53 ±10,84	90,44 ±10,97
USE1	**79,20 ±09,24**	65,67 ±20,78	64,07 ±20,50	62,87 ±23,92
USE2	**76,67 ±08,78**	73,27 ±10,40	72,33 ±11,38	71,93 ±11,43
WEA	**72,90 ±09,81**	71,20 ±11,73	72,26 ±11,06	69,95 ±11,90

Tabla 6: Resultados de los algoritmos frente a conjuntos de datos reales.

6. Conclusiones

En este artículo se presentó un nuevo algoritmo capaz de aprender de flujos de datos no estacionarios. El nuevo algoritmo combina la simplicidad de bagging para entrenar clasificadores base con los métodos de combinación de clasificadores: EWMA, NB, DS y EW. Los resultados experimentales muestran los mejores resultados utilizando NB como método de combinación. El nuevo algoritmo ADCE procesa los datos de entrada con complejidad temporal y espacial constante, y solo procesa cada ejemplo de entrenamiento una vez. Para manipular cambios de concepto, ADCE utiliza un detector de cambios en cada clasificador base para estimar su tasa de error. Cuando se estima un cambio en un clasificador base, este es remplazado por uno nuevo.

ADCE fue comparado empíricamente con varios algoritmos de ensamble de la familia bagging, utilizando Naive Bayes como clasificador base. Los experimentos incluyeron conjuntos de datos artificiales y reales. Todos los algoritmos fueron probados frente a los tipos de cambios comunes (abruptos y graduales). Los experimentos mostraron que el nuevo algoritmo es una buena opción para el aprendizaje a partir de flujos de datos con cambios de concepto.

Agradecimientos

Los autores queremos agradecer al editor y a los revisores anónimos por sus comentarios y sugerencias útiles.

Referencias

[1] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007. 2.2

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010. 5.1, 5.3

[3] 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. (document), 5.1

[4] 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. (document), 2.2, 5.1

[5] Isvani Inocencio Frias Blanco, Agustin Alejandro Ortiz Diaz, Gonzalo Ramos Jimenez, Rafael Morales Bueno, and Yaile Caballero Mota. Clasificadores y multiclasificadores con cambio de concepto basados en arboles de decision. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 14(45):32–43, 2010. (document)

[6] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2.2

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

[8] 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. 5.1

[9] Magdalena Deckert. Batch weighted ensemble for mining data streams with concept drift. In *Foundations of Intelligent Systems*, pages 290–299. Springer, 2011. 2.1

[10] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM, 2000. 5.1

[11] 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. 2.2

[12] 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. (document), 4

[13] 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. 5.4

[14] 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. 5.1

[15] João Gama, Pedro Medas, Gladys Castillo, and Pedro Pereira Rodrigues. Learning with Drift Detection. In *Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, September 29 - October 1, 2004, Proceedings*, pages 286–295, 2004. 1

[16] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, March 2014. (document), 1

[17] Haibo He, Sheng Chen, Kang Li, and Xin Xu. Incremental Learning From Stream Data. *IEEE Transactions on Neural Networks*, 22(12):1901–1914, December 2011. (document)

[18] David P Helmbold and Philip M Long. Tracking drifting concepts using random examples. In *Proceedings of the fourth annual workshop on Computational learning theory*, pages 13–23. Morgan Kaufmann Publishers Inc., 1991. 1

[19] David P. Helmbold and Philip M. Long. Tracking Drifting Concepts By Minimizing Disagreements. *Machine Learning*, 14(1):27–45, 1994. 1

[20] Elena Ikonomovska and João Gama. Learning Model Trees from Data Streams. In *Discovery Science, 11th International Conference, DS 2008, Budapest, Hungary, October 13-16, 2008. Proceedings*, pages 52–63, 2008. 1

[21] Anthony Kuh, Thomas Petsche, and Ronald L. Rivest. Learning Time-varying Concepts. In *Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3*, NIPS-3, pages 183–189, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. 1

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

[23] 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. 1

[24] 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. 2.2

[25] Douglas C Montgomery. *Introduction to statistical quality control*. John Wiley & Sons, 2007. 4

[26] Kyosuke Nishida. *Learning and detecting concept drift*. PhD thesis, School of Information Science and Technology, Hokkaido University, 2008. (document)

[27] Agustín Ortíz Díaz, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Isvani Frías Blanco, Yailé Caballero Mota, Antonio Mustelier Hechavarría, and Rafael Morales-Bueno. Fast Adapting Ensemble: A New Algorithm for Mining Data Streams with Concept Drift. *The Scientific World Journal*, 2014. 1, 5.4

[28] 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. (document), 2.2, 5.1

[29] Lior Rokach. Ensemble Methods for Classifiers. In *Data Mining and Knowledge Discovery Handbook*. 2005. DOI: 10.1007/0-387-25465-X_45. 3, 3

[30] S. Shilen. Multiple binary tree classifiers. *Pattern Recognition*, 23(7):757–763, 1990. 3

[31] 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. 2.1

[32] 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. 2.1

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