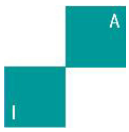
Inteligencia Artificial 45(2010), 32-43
doi: 10.4114/ia.v14i45.1080



INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

CLASIFICADORES Y MULTICLASIFICADORES CON CAMBIO DE CONCEPTO BASADOS EN ÁRBOLES DE DECISIÓN

Isvani Frías Blanco, Agustín Ortiz Díaz, Gonzalo Ramos Jiménez, Rafael Morales Bueno, Yailé Caballero Mota

Universidad de las Ciencias Informáticas
ifriasb@grm.uci.cu

Universidad de Granma, Universidad de Málaga, Universidad de Camagüey
agustin@udg.co.cu, ramos@lcc.uma.es, morales@lcc.uma.es, yailec@yahoo.com

Abstract In a large number of problems the data comes from dynamic environments and are acquired over time, which requires processing sequence. Often, changes occur that cause the model built with previous data inconsistent with current data and an update is needed it. This problem, known as concept drift complicates the task of learning to build a model. In this paper we review existing techniques for learning with concept drift focusing primarily on decision trees, considering different types of change and peculiarities of the problem.

Resumen En un gran número de problemas los datos proceden de entornos dinámicos y son adquiridos a lo largo del tiempo, lo que obliga a procesarlos de forma secuencial. Frecuentemente, se producen cambios que provocan que el modelo construido con datos anteriores sea inconsistente con los datos actuales y sea necesaria una actualización del mismo. Este problema, conocido por cambio de concepto complica la tarea del aprendizaje para la construcción de un modelo. En este trabajo se realiza una revisión de las técnicas existentes de aprendizaje con cambio de concepto centrándose fundamentalmente en árboles de decisión, considerando diferentes tipos de cambio y peculiaridades del problema.

Palabras clave: cambio de concepto, clasificadores y multiclasificadores incrementales, arboles de decision.

1 Introducción

Los procesos incrementales son aplicados en muchas áreas, no solo exclusivas a tareas de aprendizaje y minería de datos, lo que lo convierte en una poderosa herramienta en muchos problemas actuales que son presentados en formato de datastream [29, 68, 8, 53, 19, 49, 51, 71, 70, 35]. En el aprendizaje supervisado, un datastream es una secuencia muy grande de pares (X, Y) que se van adquiriendo a lo largo del tiempo, donde X representa los atributos de los datos de entrada e Y su clase correspondiente. A estos pares suele llamárseles instancias, experiencias o ejemplos; términos que en el artículo se utilizarán indistintamente.

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

ISSN: 1988-3064(on-line)

objetivo, dando lugar a lo que se conoce como cambio de concepto (concept drift) [36, 74, 40, 39, 45, 33, 41, 72, 21].

La dependencia del contexto no sólo puede inducir variaciones en la función objetivo sino que además puede cambiar la propia distribución de los atributos de los ejemplos de entrenamiento. Cuando el cambio contextual afecta únicamente a los datos de entrada éste se dice virtual (sampling shift) [73, 63], mientras que el cambio es real cuando únicamente induce un movimiento del concepto objetivo (concept shift). O sea, suponiendo que las experiencias se rigen por la función objetivo $Y = f(X)$ con una densidad de probabilidades $P(X, Y)$, en el tiempo pudiera cambiar solamente $P(X, Y)$ (cambio virtual); pudiera cambiar la función objetivo $Y = f(X)$ o pudieran cambiar ambas (cambio real) [22]. En la práctica es irrelevante el tipo del cambio ya que ambos producen un impacto en el modelo, en cuanto aumentan el error del mismo con respecto a los ejemplos actuales y lleva a la necesidad de revisarlo constantemente.

Un problema de gran dificultad está relacionado con la velocidad del cambio. En la literatura se distinguen dos tipos, relacionado con la frecuencia con la que se reciben los ejemplos que describen a la nueva función objetivo: abrupto (repentino, instantáneo) y gradual. Algunos autores, como Stanley [69], dividen el cambio gradual en moderado y lento, dependiendo de la velocidad del mismo.

De forma general, se pretende que un sistema que maneje cambio de concepto sea capaz de adaptarse rápidamente al cambio, ser robusto en la distinción entre un verdadero cambio de concepto y el ruido, así como reconocer y tratar contextos recurrentes. La distinción entre un verdadero cambio de concepto y el ruido es un problema de gran dificultad para la manipulación del cambio de concepto. Algunos algoritmos pueden ser muy susceptibles al ruido, interpretándolo erróneamente como un cambio de concepto, mientras que otros pueden ser robustos al ruido pero se ajustan al cambio muy lentamente [74]. Adicionalmente, en algunos dominios el contexto puede ser recurrente. Para adaptarse rápidamente al cambio de concepto, las descripciones de los conceptos pueden ser almacenadas para luego reexaminarlas y usarlas posteriormente. Algunos sistemas que usan esta estrategia son FLORA3 [73], PECS [63], SPLICE [30] y Local Weights and Batch Selection [38].

Muchos algoritmos de aprendizaje fueron usados como modelo base para manipular cambio de concepto. Entre estos encontramos sistemas basados en reglas [74, 72, 73, 67], árboles de decisión con su versión incremental [33, 41, 72, 30, 69], Naïve Bayes [41, 72], Máquinas de Vector Soporte [39, 38], Redes de Funciones de Base Radial [42] y aprendizaje basado en instancias [63, 1, 16].

Como es de suponer, mencionar todas las investigaciones realizadas en este sentido es una tarea prácticamente imposible. En el artículo se hace referencia a los principales trabajos relacionados con la manipulación del cambio de concepto, teniendo en cuenta principalmente su actualidad y su capacidad de aplicación a algoritmos de inducción incremental de árboles de decisión. Para ello en el apartado 2 se mencionan las principales estrategias y los sistemas más populares que manejan cambio de concepto, sin tener en cuenta el modelo para representar el conocimiento. Luego, el apartado 3 se refiere a los principales algoritmos de aprendizaje para la inducción de árboles de decisión de forma incremental, y posteriormente, se exponen las ideas principales de algunos resultados teóricos (apartado 4) y otros con un enfoque más práctico (apartado 5), teniendo en cuenta que estos detectores de cambio de concepto sean posiblemente aplicables a árboles de decisión. Por último, en el apartado 6, son mencionados sistemas y técnicas de multiclasificadores.

2 Clasificadores para el tratamiento de cambio de concepto

STAGGER [67] e IB3 [1] fueron posiblemente los primeros algoritmos diseñados expresamente para cambio de concepto, desarrollándose posteriormente otros como FLORA2 [74], AQPM [51], AQ11-PM [52], y AQ11-PM-WAH [50].

En Gama y otros [26] se distinguen dos categorías donde se ubican las estrategias para enfrentar el problema en cuestión: estrategias que adaptan el aprendizaje en intervalos de tiempo regulares sin considerar que ha ocurrido un cambio en el concepto; y estrategias que primero detectan el cambio de concepto, y luego el aprendizaje es adaptado al cambio. Dentro de la segunda categoría se distinguen las técnicas de selección de instancias; la agregación de pesos a instancias; y el aprendizaje con múltiples descriptores de concepto. Las técnicas más comunes están relacionadas con la agregación de pesos y técnicas de ventana de tamaño fijo y variable [51, 74, 40, 39].

Las técnicas basadas en selección de instancias consisten en una generalización de una ventana que se mueve sobre las instancias recientemente obtenidas y usa el concepto aprendido para predecir solamente el futuro inmediato. En la selección de instancias, el objetivo es seleccionar instancias relevantes al concepto actual. Ejemplos de algoritmos basados en ventanas son la familia FLORA [74], FRANN [42] y TWF [63]. Adicionalmente, la elección de un tamaño de ventana adecuado es un compromiso entre una adaptación rápida a cambios abruptos (tamaño pequeño), así como la detección de cambios graduales y una buena generalización en fases sin cambio de concepto (ventanas de tamaño grande). La idea básica para los administradores de tamaños de ventana es ajustar dicho tamaño para enfrentar ambos criterios. Algunos algoritmos usan un tamaño de ventana fijo mientras otros utilizan heurísticas para ajustar el tamaño, como "Adaptive Size" [38] y FLORA [74].

Por otro lado, las instancias pueden "pesarse" acorde a su edad y a su correspondencia con el concepto actual. La agregación de pesos a las instancias usa la idea de algunos algoritmos como la Máquina de Vector Soporte (SVM) para procesar instancias según su peso [38][47].

Los métodos de ensamblaje se basan en heurísticas y medidas de interés para eliminar, reactivar o añadir dinámicamente nuevos descriptores de conceptos en respuesta a las variaciones ocurridas en la consistencia del modelo con respecto a los nuevos ejemplos recibidos, como por ejemplo SEA [70] y DWM [41].

Varias investigaciones han estudiado y descrito sistemas de aprendizaje dependiendo del modelo de memoria que dictamina como tratar las instancias pasadas –completa, parcial y sin memoria de instancias [60]. Por ejemplo, en el modelo de memoria total de instancias se han desarrollado los sistemas GEM [60] e IB1 [1]; con memoria parcial de instancias los sistemas LAIR [20], IB2 [1] y DARLING [62]; mientras que sin la retención de instancias en memoria se destacan los sistemas ID4 [66], STAGGER [65], WINNOW [47], AQ-11 [54], Learn++ [58], VFDT [19] y ARCH [75].

3 Técnicas de Inducción Incremental de Árboles de Decisión sin memoria de instancias

Al igual que ocurre a la hora de diseñar algoritmos de aprendizaje tradicional, con el aprendizaje incremental también se diseñan técnicas que inducen multitud de modelos diferentes que representen el conocimiento. Añadir la característica incremental al aprendizaje no limita la diversidad de modelos que pueden ser inducidos, lo único que varía serán las técnicas para hacerlo. Así, tenemos algoritmos que inducen redes neuronales (Learn++ [58], FRANN [42]), árboles de decisión (VFDT [19], IADEM [11], reglas de decisión (FACIL [23]), etc.

El algoritmo ID4 [64] probablemente fue la primera tentativa de construir un árbol de decisión de forma incremental. Sin retener instancias en memoria, toma como entrada un árbol de decisión y una experiencia, y devuelve un nuevo árbol adaptado a esta experiencia. ID4 mantiene información sobre el número de instancias en las clases para cada valor de cada atributo que pueda servir como decisión en un nodo y calcula cuál es el mejor nodo para decidir si expandir el nuevo atributo en sus valores creando un solo nivel más (ID4). Aunque es una mejora respecto a ID3, para determinadas instancias el algoritmo da lugar a continuos descartes, no llegando a una estabilización final del árbol.

Entre todas las técnicas de aprendizaje incremental de árboles de decisión, la familia de algoritmos VFDT (Very Fast Decision Tree) parece estar llamada a convertirse, al igual que C4.5 dentro del aprendizaje no incremental, en el estándar para la clasificación de datastream mediante árboles [19, 33, 27, 28, 34].

Limitado inicialmente a streams de ejemplos con todos sus atributos simbólicos y provenientes de entornos estacionarios, VFDT [19] parte de la observación de Cattlet [12] según la cual no es necesario utilizar todos los ejemplos de entrenamiento para encontrar la mejor pareja (atributo, valor) como siguiente nodo del árbol, sino únicamente un subconjunto de aquellos no descritos previamente mediante cualquier predicado extraído de nodos anteriores. Bajo la observación anterior, VFDT utiliza la inecuación de Hoeffding [32] como cota de error para determinar el número de nuevos ejemplos necesarios que garantiza continuar la expansión del árbol de forma incremental sin dañar la precisión del mismo. VFDT garantiza además que, con el número de ejemplos necesarios, el modelo de salida generado es asintóticamente idéntico al generado por un algoritmo de aprendizaje por lotes convencional.

CVFDT [33] es una extensión de VFDT que, mediante un bosque de árboles, modela dominios no estacionarios eliminando aquellos árboles obsoletos. Ambas propuestas están limitadas a atributos simbólicos, en Jin y Agrawa [35] se propone una extensión para procesar atributos numéricos mediante un esquema de discretización sin pérdida de precisión. Gama y otros [27][28] han propuesto UFFT (Ultra Fast Forest Trees) y VFDTc, sistemas que extienden a VFDT en dos direcciones: capacidad para tratar con atributos numéricos directamente y la aplicación de Naïve –Bayes en las hojas del árbol.

El IADEM-0 [59] realiza la inducción de árboles de decisión sin memoria, solo guarda en los nodos la información relevante de las experiencias anteriores en términos de frecuencia relativa, contadores, índices, etc. Utiliza las cotas de concentración de Hoeffding [32] y Chernoff [14] para que, con una confianza determinada, los parámetros reales de la población se encuentren en un intervalo definido por las cotas. La estimación de los parámetros se calcula a través de las experiencias acumuladas. Cada parámetro relacionado con frecuencias relativas, probabilidades, y otros, tiene un intervalo de confianza. Por ende, la medida para la expansión de un nodo a través del mejor atributo, calculada a través de la entropía, (y esta a su vez por las probabilidades de los atributos con respecto a la clase de las experiencias acumuladas) también presenta un intervalo de confianza. Cuando, con la confianza requerida, se conoce el mejor atributo para la expansión (los intervalos de confianza para el cálculo de la medida no se interceptan), se expande por este atributo. El algoritmo termina cuando el árbol está completamente expandido (caso extremo) o cuando el error supremo es menor que el error deseado por el usuario.

El IADEM-2 [18] mejora la calidad de IADEM-0: este presenta tolerancia al ruido, mejora la convergencia incrementando el número de expansiones que se pueden realizar en un momento determinado, y detecta la estabilización o empeoramiento del modelo. Por otro lado, maneja atributos continuos, fija un desbalanceo máximo para expandir, reformula la idoneidad para el mejor atributo para expandir y mejora la predicción (basándose en el teorema de Bayes) utilizando la información contenida en las hojas virtuales.

4 Resultados teóricos para detectar cambios de concepto

En los resultados teóricos para el manejo de cambio de concepto usualmente se imponen un conjunto de restricciones al tipo de cambio de concepto admitido con el objetivo de demostrar la capacidad de aprendizaje en contextos cambiantes.

Por ejemplo, Kuh y otros (1991) determinan la tasa máxima de cambio (cuán frecuente puede ser un cambio) que es aceptable para un algoritmo de aprendizaje. Precisamente, su resultado principal es una cota inferior para la tasa de cambio $\lambda < (\delta/2) \cdot m(\varepsilon, \delta/2)$ donde λ significa que en promedio, un concepto es estable como mínimo por $1/\lambda$ pasos (o instancias), y estiman un tamaño fijo de ventana de $w(\varepsilon, \delta) = m(\varepsilon, \delta/2)$ donde $m(\varepsilon, \delta) = max(4/\varepsilon \cdot log(2/\delta), 8d/\varepsilon log 13/\varepsilon$.

Un trabajo similar es el propuesto por Hembold y Long (1994), donde asumen un posible permanente pero lento cambio de concepto definiendo la $Pr(f_t(X) \neq f_{t+1}(X)) \leq \Delta$, donde X representa los atributos de entrada, $\Delta \geq 0$ y una secuencia de funciones objetivo $f_t(X)$. Sus resultados incluyen un algoritmo de aprendizaje con una probabilidad ε de cometer un error si la tasa de movimiento de las funciones objetivo en un tiempo constante es como máximo $c_1\varepsilon^2/(d \cdot ln(1/\varepsilon))$, donde d es la dimensión Vapnik-Chervonenkis de los conceptos en la clase H y c_1 una constante que está en función del número de veces que el algoritmo obtuvo el mínimo de errores en experiencias anteriores. Adicionalmente, muestran que para el algoritmo de aprendizaje es suficiente examinar un número determinado de experiencias recientes (un tamaño de ventana $m = c_0 d/\varepsilon ln 1/\varepsilon$. [43]

Sin embargo, en la práctica frecuentemente no se pueden asumir este tipo de restricciones. Por otro lado, el gran tamaño de la ventana obtenido en los resultados teóricos puede no ser práctico [39]. [31]

5 Detectores de cambio de concepto basados en árboles de decisión

Abundantes y con diversas estrategias son los trabajos que utilizan un enfoque práctico, muchos de los cuales tienen embebida la estrategia para la manipulación del cambio en el algoritmo de aprendizaje, por

lo que dicha estrategia es difícilmente aplicable a otros tipos de algoritmos –por ejemplo Last [44], que utiliza redes info-fuzzy (IFN) para aprender árboles de decisión.

En este apartado se mencionan algunos de los trabajos más recientes y relevantes, teniendo en cuenta su capacidad en la aplicación de la estrategia de detección del cambio a otros algoritmos para inducir árboles de decisión de forma incremental.

Uno de los más conocidos es el sistema CVFDT [33], que usa una estrategia de monitorización para la inducción del árbol de decisión. Uno de los parámetros fijados por el usuario es el número de ejemplos que deben ocurrir para que el algoritmo recorra todos los nodos del árbol, comprobando que el atributo por el cuál se expandió dicho nodo sigue siendo el mejor. Otro de los parámetros es el número de ejemplos que serán utilizados para construir subárboles alternativos a partir de los nodos cuyos atributos ya no son los mejores. A partir de aquí los próximos T_2 ejemplos (parámetro fijado también por el usuario) son utilizados para comprobar la exactitud de los subárboles construidos para reemplazar a la rama correspondiente en el árbol original. De esta forma, el cambio de concepto es manipulado al suponer que éste ocurre cuando al llegar nuevas instancias, existe un cambio en el mejor atributo estimado para la expansión de un nodo en un momento anterior; y se reemplaza la rama subyacente a ese nodo por una más precisa en correspondencia con las instancias más recientes.

Muchos trabajos utilizan una versión de este algoritmo incorporando otra estrategia para manejar cambios de concepto. Por ejemplo, en Natwichai y Li [56] se usa una Tabla de Decisión Ambigua [15] como intermedio para la representación de conocimiento. Cada entrada en dicha tabla es una regla de decisión que se obtiene a través de CVFDT cuando éste genera un árbol alternativo debido a un posible cambio de concepto. La Tabla de Decisión Ambigua es expandida con dos columnas, la primera representa una marca de tiempo y la segunda un error, ambas correspondientes a una regla determinada (una fila). Así, cuando se quiere extraer conocimiento reciente, se induce un árbol de decisión con un algoritmo que tiene como entrada la Tabla de Decisión Ambigua y que genera el árbol con el menor error, calculado a través de las columnas que representan error y marca de tiempo.

Paralelamente, en Gama y otros [26], para un conjunto de ejemplos, se toma el error de predicción de un algoritmo de aprendizaje como una variable aleatoria correspondiente a experimentos de Bernoulli. Tomando la distribución Binomial y considerando que para un número grande de experiencias esta se aproxima a la Normal, se toma el error de predicción del algoritmo de aprendizaje p_i y su desviación estándar $s_i = \sqrt{p_i(1-p_i}$. En este sentido el método de detección del cambio mantiene dos registros en el entrenamiento del algoritmo de aprendizaje p_{min} y s_{min} que se actualizan cuando un nuevo ejemplo i provoca que $p_i + s_i$ sea menor que $p_{min} + s_{min}$. Para la detección del cambio se establece un nivel de advertencia y un nivel de cambio. En la experimentación se propone para una confianza de 0.95 el nivel de advertencia en $p_{min} + 2s_{min}$ y para una confianza de 0.99 el nivel de cambio en $p_{min} + 3s_{min}$. El enfoque propuesto por Gamma y otros (2004) tiene un buen comportamiento detectando cambios abruptos y graduales cuando el cambio no es muy lento, pero cuando el cambio es muy lento el método presenta dificultades [4].

Para mejorar la detección de un cambio de concepto gradual, en Baena y otros [4] se propone el método EDDM (Early Drift Detection Method) donde la idea básica es considerar la distancia entre dos errores de clasificación en vez de considerar solamente el número de errores. Mientras el método de aprendizaje esté aprendiendo, éste mejora la predicción y la distancia entre dos errores se incrementa.

Por otro lado, en Nuñez y otros [57] se propone un algoritmo para inducir árboles de decisión y se estima un cambio de concepto si se confirma que una medida de rendimiento tiene un descenso persistente, siendo dicha persistencia la clave para distinguir entre ruido y un verdadero cambio de concepto. De esta forma se calcula dicha medida de rendimiento $R = 7/8R_v + 1/8R_a$ a través del porcentaje de instancias bien clasificadas en el instante actual R_a y el obtenido a lo largo del tiempo R_v. Se definen variables y una función para calcular la fracción en la que la ventana de retardo debe ser dividida como mecanismo de olvido. Cada nodo usa la ventana de retardo para controlar el tiempo en que los ejemplos deben ser olvidados en dependencia de la pertenencia a un concepto anterior (descenso del rendimiento). Una reducción del tamaño de dicha ventana provoca que se olviden ejemplos anteriores al instante en que se detecto el primer descenso de la medida de rendimiento. Por otro lado, a la llegada de cada instancia se chequea la posibilidad de cambiar la estructura del árbol podando o expandiendo cada nodo según la relevancia de su respectivo atributo de expansión mediante un test χ^2.

En contraposición, Fan y otros [22] plantean que en muchos algoritmos de aprendizaje se asume que

la etiqueta de la clase de los ejemplos en un datastream se puede obtener fácilmente, mientras que la mayoría de los problemas del mundo real esta etiqueta no puede ser obtenida simplemente (por ejemplo, en la detección de fraude en tarjetas de crédito, conocer si una transacción en particular es un fraude). En este sentido, asumiendo que se ha construido un árbol de decisión con anterioridad, para la detección de un cambio de concepto sin conocer las etiquetas de las clases se proponen dos estadísticas. La primera ($PS = \sigma|Ps_{hoja} - Pd_{hoja}|/2$) tiene en cuenta como los ejemplos de entrenamiento están distribuidos en las hojas de un árbol de decisión (Ps), y se supone que si no existe un cambio de concepto, esta distribución debe mantenerse aproximadamente igual en los ejemplos sin etiqueta a clasificar (Pd). La segunda estadística $LS = L_e - L_a$, relaciona el error esperado del árbol de decisión (L_e) y la pérdida anticipada (L_a) (en el ejemplo de detección de fraude, sería el total de dinero que se espera recolectar después de clasificar n experiencias). Por supuesto, una variación en estas medidas indica un posible cambio de concepto.

Pocos trabajos ofrecen garantías rigurosas del desempeño del algoritmo propuesto para la detección del cambio. Entre estos los más destacados son los trabajos de Kifer y otros (2004); Bifet y Gavaldà (2007); y Bifet y Gavaldà (2009), cuyas principales aportes se comentan a continuación.

En Kifer y otros (2004) se realiza un tratamiento formal para la detección del cambio. Para ello se introduce una familia de medidas de distancias entre distribuciones de probabilidad, se diseña un algoritmo para el cambio de concepto y se provee de garantías de rendimiento analíticas y numéricas en la precisión para la detección del cambio. La idea de las distancias propuestas es cuantificar el mayor cambio entre dos distribuciones de probabilidad sobre un conjunto, semejante a la distancia de variación total de la teoría de probabilidades. Se detecta el cambio a través de dos ventanas de tamaño fijo, una referencia y otra actual, conteniendo todos los ejemplos. De esta forma, el algoritmo reporta un cambio si $d(VR, VA) > \alpha$, donde α es una constante,VR_{m1} es la ventana referencia de tamaño $m1$ y VA_{m2} la ventana actual que se desplaza por los últimos $n - m1 + 1$ elementos. [37]

Bifet y Gavaldà [6] proponen el algoritmo ADWIN. La idea del mismo es eliminar una parte obsoleta de una ventana (subventana) cuando se concluya que ésta es suficientemente larga y exhibe una distribución suficientemente distinta. Para esto se divide dicha ventana de tamaño n en todas las posibles de longitud n_0 y n_1 ($n = n_0+n_1$) y se eliminan los elementos menos recientes mientras la diferencia entre las medias de las mismas difieran en un valor mayor a $\sqrt{1/(2m) \cdot ln(4/\delta)}$, donde m es la media armónica de n_0 y n_1 y δ es un valor de probabilidad relacionado con el nivel de confianza. Se ofrecen garantías probabilísticas para los falsos positivos y falsos negativos. También se presenta una versión mejorada del algoritmo anterior llamada ADWIN2 que usa la idea de algunos algoritmos desarrollados para datastream [3, 55, 2, 17] para encontrar un buen punto de corte de forma rápida sin examinar todas las divisiones de la ventana. [32].

Recientemente, en Bifet y Gavaldà [7] se propone un método para desarrollar algoritmos que puedan adaptarse al cambio de concepto. Para esto se plantea un sistema con un modulo de memoria, un módulo estimador y un módulo de detección del cambio. En este sentido la memoria es el componente donde el algoritmo almacena los ejemplos que considera relevantes; el componente estimador es un algoritmo que estima la estadística deseada y el componente detector de cambios dispara una señal cuando se detecta un cambio en la distribución de los datos. Teniendo esto en cuenta se clasifican este tipo de algoritmos en cuatro clases dependiendo de la existencia de los módulos de detección y de memoria. También se define una familia de Árboles de Hoeffding con Ventanas (Hoeffding Window Tree) y se proponen y comparan tres algoritmos basados en Árboles de Hoeffding con Ventanas que difieren en el tipo de estimador usado.

6 Aprendizaje con la descripción de múltiples conceptos

El aprendizaje por multiclasificadores mantiene un conjunto de descriptores de conceptos y predice combinando estos ya sea por votación simple o por votación ponderada, o bien seleccionando la descripción más relevante.

Algunas ventajas de los multiclasificadores sobre los clasificadores simples, según Wang y otros [72] son:

- Los multiclasificadores ofrecen una mejor precisión en la predicción [73][5].
- Construir un multiclasificador es más eficiente que construir un modelo simple.

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

Una de las principales familias de técnicas de clasificación es la formada por los sistemas offline. Pertenecientes al aprendizaje por lotes (batch learning), el proceso de aprendizaje en tales sistemas consiste en procesar repetidas veces el conjunto de entrenamiento hasta obtener un modelo final que describa con la mayor exactitud posible al mayor número de ejemplos de entrenamiento posible. Para ello, es necesario aceptar como válidos tres supuestos [23]:

- Todos los ejemplos necesarios para describir el dominio del problema están disponibles antes del proceso de aprendizaje.
- Todos los ejemplos de entrenamiento pueden ser cargados en memoria.
- Tras procesar debidamente el conjunto de entrenamiento, el aprendizaje puede darse por finalizado.

6.1 Algoritmos multiclasificadores para trabajo offline

Weighted Majority [48] es un algoritmo general para pesar y combinar las decisiones de expertos, cada uno de estos expertos es un método de aprendizaje. El algoritmo comienza por crear un conjunto de expertos y dar un peso a cada uno. Cuando llega una nueva instancia, el algoritmo base recibe una predicción de cada experto, y toma una decisión combinando las predicciones y teniendo en cuenta el peso de cada uno de estos. Los expertos que no coincidieron con la mayor votación son penalizados en sus pesos. Winnow [46] es similar a Weighted Majority y llama a los expertos especialistas. En Blum [8] se evalúan otras variantes de Weighted Majority y Winnow.

Dos estrategias de propósito general, ampliamente utilizadas en el aprendizaje offline, son Bagging [9] y Boosting [24], que incrementan la exactitud de los modelos inducidos por cualquier técnica de aprendizaje. Mediante heurísticas de votación y ponderación, ambas técnicas van creando modelos intermedios en base a los cuales formar un único modelo final cuya exactitud mejore la exactitud de cualquiera de ellos.

Mediante Bagging el modelo final es compuesto a partir de las reglas más frecuentes dentro de los modelos individuales. A partir de dos parámetros de usuario p y q, se realizan k muestras con reemplazo de tamaño q a partir del conjunto de entrenamiento original. Para cada muestra se aplica un clasificador distinto de forma que cada ejemplo de test es clasificado k veces, una vez para cada modelo. Debido al reemplazo en el muestreo puede que no todos los ejemplos del conjunto original sean seleccionados y que algunos aparezcan en varias muestras.

Similar a Bagging, mediante Boosting se generan varios clasificadores y son votados de acuerdo a su tasa de error. Sin embargo, a diferencia de Bagging, no se obtienen a partir de diferentes muestras sino secuencialmente sobre el mismo conjunto de entrenamiento. Cada ejemplo del conjunto de entrenamiento tiene inicialmente asignado un peso "1" y secuencialmente cada clasificador modifica el peso de aquellos que son clasificados incorrectamente con un factor inversamente proporcional al error global del conjunto de entrenamiento. Así, conjuntos de entrenamiento con un error reducido provocarán que los pesos de los ejemplos mal clasificados aumenten en varios órdenes de magnitud. El objetivo perseguido es que tras el aprendizaje de un clasificador M_i, el siguiente M_{i+1} preste más atención a los errores de clasificación cometidos por todos sus anteriores. A su vez, M_i obtiene un número de votos en función del número de aciertos obtenidos. Al final, el último clasificador combina los modelos de cada uno de los clasificadores anteriores en función del peso (número de votos) de cada uno. [23]

Una variante significativa de Boosting es Adaboost (Adaptive boosting) [25]. En este caso, el peso de cada clasificador depende del funcionamiento de éste sobre el conjunto de entrenamiento utilizado para su construcción [5]. Este algoritmo fue diseñado para clasificadores poco robustos para los cuales una pequeña variación en los datos de entrada provoca un fuerte cambio en el modelo. Empíricamente se ha demostrado que para clasificadores con una tasa de error cercana al 50%, Adaboost no es recomendable. [23]

Otro algoritmo de clasificación por votación es Arc-4x, su nombre se deriva del termino "arcing" que fue introducido por Breiman [10] para describir a la familia de algoritmos Adaptively resample and combine [5]. Al igual que el algoritmo Adaboost (pertenece a la misma familia) es una variante del Boosting y marca con altos pesos a las instancias que son mal clasificadas. La idea es forzar a los nuevos

clasificados a enfocarse en los casos difíciles, para esto utiliza como parámetro para dar peso el número de errores del clasificador previo [13].

En Bauer y Kohavi [5] se realiza un estudio empírico, donde se establece una comparación entre los algoritmos con técnicas de votación (Bagging y Boosting) y algunas de sus variantes, utilizando conjuntos de entrenamiento artificiales y del mundo real.

Según Bauer y Kohavi [5] los algoritmos con técnicas de votación se clasifican en dos tipos; los que se adaptan a los cambios de distribución del conjunto de entrenamiento base según el funcionamiento de los clasificadores previos (como Boosting), y los que no se adaptan (como Bagging).

6.2 Algoritmos multiclasificadores para trabajo online

Contrariamente a lo que podrá parecer dada la complejidad computacional de los mismos, los métodos de ensamblaje han recibido en los últimos tiempos una gran atención para el modelado y la clasificación de datastream.

En general, y aunque dentro de un dominio incremental, las nuevas propuestas siguen el mismo esquema que las técnicas de ensamblaje aplicadas en el aprendizaje por lotes, basándose en heurísticas y medidas de interés para eliminar, reactivar o añadir dinámicamente nuevos algoritmos de aprendizaje en respuesta a las variaciones ocurridas en la consistencia del modelo con respecto a los nuevos ejemplos recibidos.

Una de las primeras propuestas es SEA [70], la cual utiliza una cantidad de memoria fija e independiente del número de ejemplos y garantiza una cota máxima para el tiempo necesario por cada ejemplo. SEA utiliza una ventana de tamaño fijo donde todos los ejemplos son consecutivos y reemplazados en bloques.

Bajo el mismo esquema de ventana, en Wang y otros [72] se propone un método basado en la ponderación de clasificadores en función de la precisión obtenida por los mismos al utilizar como test los propios ejemplos de entrenamiento. Utilizan como bases de expertos diferentes tipos de algoritmos como C4.5, RIPPER, Naïve Bayesian, entre otros.

Kolter y Maloof [41] proponen DWM (Dynamic Weighted Majority), basado en el algoritmo de ponderación por mayoría de Littlestone [48] para ensamblar modelos de distinta edad. Este algoritmo es capaz de cambiar dinámicamente los expertos menos adaptados en el conjunto base, manipulando de esta forma los cambios de conceptos. En Littlestone [48] se establece una comparación, entre dos bases de expertos diferentes para el algoritmo DWM, una con árboles de decisión y otra con Naïve Bayes, y el algoritmo de Weighted Majority implementado por Blum [8].

En Rushing y otros [61] se propone CBEA (The Coverage Based Ensemble Algorithm), este método tiene como idea conservar todos los patrones de concepto en el multiclasificador. CBEA guarda la información sobre el rango de datos usado por cada clasificador en el entrenamiento; basado en esto y en la edad decide la presencia del clasificador en el conjunto base.

Del Campo [18] propone una mejora al multiclasificador MultiCIDIM-DS que tiene características similares a SEA (antes mencionado), pero que utiliza el algoritmo CIDIM [59] para inducir los clasificadores bases, a diferencia del SEA, que utiliza el C4.5. Según los autores, existen dos diferencia fundamentales entre SEA y MultiCIDIM-DS, primero la medida de calidad del segundo de estos es más simple, y segundo, a la hora de remplazar unos de los clasificadores bases por otro de mejores características, se remueve el clasificador de peores características, cambiándolo por el nuevo. Esta nueva propuesta es llamada MultiCIDIM-DS-CFC pues utiliza filtros de corrección para la clasificación.

En Yue y otros [76] se señalan algunas deficiencias que presentan los algoritmos basados en lotes (batch based). Tomando como referencia a SEA, señala posibles problemas que pueden presentar estos algoritmos, sobre todo a la hora de sustituir un clasificador de conjunto base de clasificadores, luego que se ha alcanzado la cantidad máxima. La nueva propuesta es nombrada ICEA (Incremental Classification Ensemble Algorithm for data streams), la idea que sigue es que cada clasificador base pueda aprender incrementalmente y agregar automáticamente, y lo más pronto posible el resultado de su aprendizaje. Se obtiene como resultado, una más rápida detección del cambio de concepto, en comparación con algunos algoritmos basados en lotes.

7 Conclusiones

El aprendizaje incremental y el cambio de concepto han sido temas de creciente importancia debido a que cada vez más los datos son organizados en formato de datastream, siendo a su vez menos usual que los conceptos sean estables en el tiempo. En los últimos años han sido desarrollados numerosos algoritmos para la detección de cambio de concepto, distinguiéndose los basados en selección de instancias, la agregación de pesos a instancias y el aprendizaje mediante multiclasificadores.

Algunos de los métodos para la detección del cambio se encuentran embebidos al algoritmo de aprendizaje y no son fácilmente aplicables a otros algoritmos. Existe otro grupo de técnicas no dependientes del algoritmo de aprendizaje, cuya esencia es decidir, utilizando algún soporte estadístico, la ocurrencia o no de un cambio de concepto en dependencia de la variación de algún parámetro en el tiempo. En este último grupo se encuentran varios métodos que pueden ser aplicables al aprendizaje mediante árboles de decisión.

Por otro lado, pocos métodos para la detección de cambio de concepto ofrecen garantías rigurosas de desempeño a través de teoremas matemáticos, lo que indica que son necesarios procedimientos más robustos para la detección de distintos tipos de cambio de concepto con diferentes niveles de ruido y teniendo en cuenta los aspectos mencionados.

8 Bibliografía

References

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithm. *Machine Learning*, 6:37–66, 1991.

[2] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. 13th Annual ACM-SIAM Symposium on Discrete Algorithms, 2002.

[3] B. Babcock and et al. Models and issues in data stream system. 21st ACM Symposium on Principles of Database Systems, 2002.

[4] M. Baena and et al. Early drift detection method. Fourth International Workshop on Knowledge Discovery from Data Streams, 2006.

[5] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

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

[7] A. Bifet and R. Gavaldà. Adaptive parameter-free learning from evolving data streams. Technical Report R09-9, Universidad Politécnica de Catalunya, 2009.

[8] A. Blum. Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, 26:5–23, 1997.

[9] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[10] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–849, 1998.

[11] J.d. Campo-Ávila. Improving the performance of an incremental algorithm driven by error margins. *Intelligent Data Analysis*, 12(3):305–318, 2008.

[12] J. Catlett. *Megainduction: machine learning on very large databases.* PhD thesis, University of technology, Sydney, 1991.

[13] J.C. Chan, N. Laporte, and R.S. Defries. Texture classification of logged forests in tropical africa using machine-learning algorithms. Int. J. Remote Sensing, 2003.

[14] H. Chernoff. A measure of asymptotic efficiency for tests or a hypothesis based on the sum of obsevation. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[15] R. Colomb. Representation of propositional expert systems as partial functions. *Artificial Intelligence*, 109:187–209, 1999.

[16] P. Cunningham and et al. A case-based approach to spam filtering that can track concept drift. ICCBR-2003 Workshop on Long-Lived CBR Systems, 2003.

[17] M. Datar and et al. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 14(1):27–45, 2002.

[18] J. Del Campo. *Nuevos Enfoques en Aprendizaje Incremental*. PhD thesis, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga., 2007.

[19] P. Domingos and G. Hulten. Mining high-speed data streams. pages 71–80. International Conference on Knowledge Discovery and Data Mining, 2000.

[20] R. Elio and L. Watanabe. An incremental deductibe strategy for controlling constructive induction in learning from examples. *Machine Learning*, 7:7–14, 1991.

[21] W. Fan. Systematic data selection to mine concept-drifting data streams. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.

[22] W. Fan and et al. Active mining of data streams. SIAM Intâl Conf on Data Mining, 2004.

[23] F. Ferrer, J. Aguilar, and J. Riquelme. Incremental rule learning and border examples selection from numerical data streams. *Journal of Universal Computer Science*, 11(8):1426–1439, 2005.

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

[25] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. pages 148–156. 13th International Conference on Machine Learning, 1996.

[26] J. Gama and et al. Learning with drift detection. *Lecture Notes in Computer Science*, 3171, 2004.

[27] J. Gama, P. Medas, and R. Rocha. Forest trees for on-line data. 19th ACM Symposium on Applied Computing - SACâ04, 2004.

[28] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDDâ03, 2003.

[29] C. Giraud-Carrier. A note on the utility of incremental learning. *AI Communications*, 13:215–223, 2000.

[30] M. Harries, C. Sammut, and K. Horn. Extracting hidden context. *Machine Learning*, 32(2):101–126, 1998.

[31] D.P Hembold and P.M. Long. Tracking drifting concept by minimizing disagreements. *Machine Learning*, 14:27–45, 1994.

[32] W. Hoeffding. Probabilities inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58:13–30, 1963.

[33] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001.

[34] G. Hulten, L. Spencer, and P. Domingos. Mining complex models from arbitrarily large databases in constant time. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDDâ02, 2002.

[35] R. Jin and G. Agrawa. Efficient decision tree construction on streaming data. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[36] M.G. Kelly, D.J. Hand, and N.M. Adams. The impact of changing populations on classier performance. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDDâ99, 1999.

[37] D. Kifer, S. Ben, and J. Gehrke. Detecting change in data streams. 30th Conf. VLDB., 2004.

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

[39] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. 17th International Conference on Machine Learning, 2000.

[40] R. Klinkenberg and I. Renz. Adaptive information ltering: Learning in the presence of concept drifts. Learning for Text Categorization, 1998.

[41] J. Kolter and M. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. 3rd International IEEE Conference on Data Mining, 2003.

[42] M. Kubat and Widmer G. Adapting to drift in continuous domains. Technical Report ÖFAI-TR-94-27, Austrian Research Institute for Artificial Intelligence, Vienna, 1994.

[43] A. Kuh, T. Petshe, and R. Rivest. Learning time-varying concept. Advances in Neural Information Processing Systems, 1991.

[44] M. Last. Online classification of nonstationary data streams. *Intelligent Data Analysis Journal*, 6(2):129–147, 2002.

[45] M. Lazarescu and S. Venkatesh. Using multiple windows to track concept drift. *Intelligent Data Analysis Journal*, 8(1):29–59, 2004.

[46] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[47] N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. 4th Annual Workshop on Computational Learning Theory, 1991.

[48] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108:212–261, 1994.

[49] M. Maloof. An initial study of an adaptive hierarchical vision system. 17th Int. Conf. on Machine Learning - ICML'00, 2000.

[50] M. Maloof. Incremental rule learning with partial instance memory for changing concepts. International Joint Conference on Neural Networks, 2003.

[51] M. Maloof and R. Michalski. Selecting examples for partial memory learning. *Machine Learning*, 41:27–52, 2000.

[52] M. Maloof and R. Michalski. Incremental learning with partial instance memory. *Artificial Intelligence*, 154:95–126, 2004.

[53] M. Maybury and Wahlster. *Readings in intelligent user interfaces*. Morgan Kauffman, 1998.

[54] R. Michalski and J. Larson. Incremental generation of vl1 hypotheses: The underlying methodology and the description of program aq11. Technical Report UIUCDCS-F-83-905, Department of Computer Science, University of Illinois, Urbana, 1983., 1983.

[55] S. Muthukrishnan. Data streams: Algorithms and applications. 14th Annual ACM-SIAM Symposium on Discrete Algorithms, 2003.

[56] J. Natwichai and X. Li. Knowledge maintenance on data streams with concept drifting. CIS 2004, LNCS 3314, 2004.

[57] M. Núñez, R. Fidalgo, and R. Morales. On-line learning of decision trees in problems with unknown dynamics. *Lecture Notes in Artificial Intelligence*, 3789:443–453, 2005.

[58] R. Polikar and et al. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 31:497–508, 2001.

[59] G. Ramos and et al.. Iadem-0: Un nuevo algoritmo incremental. in tendencias de la minería de datos en españa. Tendencias de la Minería de Datos en España, 2004.

[60] R. Reinke and R. Michalski. Incremental learning of concept descriptions: a method and experimental results. *Machine intelligent*, 11, 1988.

[61] J. Rushing and et al. A coverage based ensemble algorithm (cbea) for streaming data. 16th IEEE International Conference on Tools with Artificial Intelligence, 2004.

[62] M. Salganicoff. Density-adaptive learning and forgetting. 10th International Conference on Machine Learning, 1993.

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

[64] J. Schlimmer and D. Fisher. A case study of incremental concept induction. 5th National Conference on Artificial Intelligence, 1986.

[65] J. Schlimmer and R.H. Granger. Beyond incremental processing: Tracking concept drift. 5th National Conf. on Artificial Intelligence, 1986.

[66] J.C. Schlimmer and D. Fisher. A case study of incremental concept induction. 5th National Conference on Artificial Intelligence, 1986.

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

[68] K. Srinivasan and D. Fisherneering. Machine learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21:126–137, 1995.

[69] K.O. Stanley. Learning concept drift with a committee of decision trees. Technical Report UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA, 2003.

[70] W. Street. A streaming ensemble algorithm (sea) for large-scale classification. 7th International Conference on Knowledge Discovery and Data Mining, 2001.

[71] S. Vijayakumar and S. Schaal. Fast and eficient incremental learning for highdimensional movement systems. Int. Conf. on Robotics and Automation - ICRA'00, 2000.

[72] H. Wang and et al. Mining concept-drifting data streams using ensemble classifiers. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[73] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. pages 227–243. 6th European Conf. on Machine Learning ECML-1993, 1993.

[74] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23:69–101, 1996.

[75] P. Winston. Learning structural descriptions from examples. Psychology of Computer Vision, 1975.

[76] Sun Yue, Mao Guojun, Liu Xu, and Liu Chunnian. Mining concept drifts from data streams based on multi-classifiers. volume 2. 21st International Conference on Advanced Information Networking and Applications Workshops, 2007.