Aprendizaje Incremental de Reglas en Data Streams

Francisco J. Ferrer
Dept. de Lenguajes y Sistemas
ETS Ingeniería Informática
Univ. de Sevilla
41012 Sevilla
ferrer@lsi.us.es

Jesús S. Aguilar
Dept. de Lenguajes y Sistemas
ETS Ingeniería Informática
Univ. de Sevilla
41012 Sevilla
aguilar@lsi.us.es

José C. Riquelme
Dept. de Lenguajes y Sistemas
ETS Ingeniería Informática
Univ. de Sevilla
41012 Sevilla
riquelme@lsi.us.es

Resumen

En este artículo presentamos FACIL, un algoritmo de aprendizaje incremental dirigido a la clasificación de *data streams* numéricos. Mediante un esquema de ventana múltiple y una política de generalización moderada, nuestra propuesta genera reglas de decisión cuya inconsistencia es controlada mediante ejemplos internos que indican las variaciones en las fronteras de decisión a lo largo del tiempo. Esta estrategia proporciona dos grandes ventajas: 1) evitar revisiones innecesarias del modelo cuando la función objetivo presenta cambios virtuales, con lo que se reduce notablemente el coste computacional y 2) clasificar nuevos ejemplos de test por distancia mínima mediante el vecino más cercano. Para ampliar su campo de aplicación, FACIL proporciona además dos heurísticas de olvido, una implícita basada en el vecino más cercano para modelar distribuciones estacionarias, y otra explícita basada en frecuencia - mediante un parámetro de usuario - para modelar *streams* sujetos a *concept drift* y *hidden context*.

1. Introducción

El problema de la clasificación es una tarea de la minería de datos ampliamente abordada desde distintas áreas del aprendizaje automático. Una de las principales familias de técnicas de clasificación es la formada por los sistemas de inducción de reglas *off–line*. Pertenecientes al aprendizaje por lotes o *batch learning*, el proceso de aprendizaje en tales sistemas consiste en procesar repetidas veces el conjunto de entrenamiento (*multi–pass algorithms*) 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: 1) todos los ejemplos necesarios para describir el dominio del problema están disponibles antes del proceso de aprendizaje; 2) todos los ejemplos de entrenamiento pueden ser cargados en memoria; 3) tras procesar debidamente el conjunto de entrenamiento, el aprendizaje puede darse por finalizado.

A pesar de su enorme campo de aplicación, las técnicas de aprendizaje *off–line* no pueden ser aplicadas en un gran número de problemas donde los datos proceden de entornos dinámicos y van siendo adquiridos a lo largo del tiempo, lo que obliga a procesarlos secuencialmente en sucesivos episodios, de forma *incremental*.

Junto a los efectos provocados por el orden de llegada de los ejemplos, el principal problema 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 *hidden context*. Un clásico ejemplo de la naturaleza incremental de un problema de aprendizaje es la predicción climatológica, donde las reglas a generar pueden variar radicalmente entre las distintas estaciones. Otro ejemplo es la obtención de patrones de compra entre los clientes de las

Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005, pp.261-270
ISBN: 84-9732-449-8

grandes superficies, susceptibles a variaciones permanentes en función del día de la semana, la época del año, la disponibilidad de los productos, factores económicos como la inflación, etc.

Paralelamente, en muchos dominios es de esperar que el contexto sea recurrente, debido tanto a fenómenos cíclicos - p.e. las estaciones del año para la predicción climatológica - como a fenómenos irregulares - p.e. la tasa de inflación o la atmósfera de mercado para el modelado de perfiles de compra [9]. Como consecuencia, cambios ocurridos en el contexto pueden inducir variaciones en la función objetivo, dando lugar a lo que se conoce como *concept drift* [14, 27]. Este último problema se acentúa cuando los datos presentan ruido, ya que sistemas demasiado sensibles reaccionarán rápidamente como si de un cambio se tratase, mientras que sistemas demasiado robustos se adaptarían de forma tardía ignorando incluso más de un cambio. Por tanto, un factor decisivo en el rendimiento del sistema es el balance o *trade–off* entre la robustez al ruido y la sensibilidad al cambio [27].

En general, en la literatura se distinguen dos tipos de cambio en función de la frecuencia con la que sen reciben los ejemplos que describen a la nueva función objetivo: (1) abrupto (repentino, instantáneo) y (2) gradual. Paralelamente, 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 [21, 26] (*sampling shift*), mientras que el cambio es real cuando únicamente induce un movimiento del concepto objetivo (*concept shift*). En la práctica es irrelevante el tipo del cambio ya que ambos producen un impacto en el modelo, pudiendo ocurrir ambos simultáneamente. Así por ejemplo, en la clasificación de *spam*, las reglas de usuario que determinan qué es correo no deseado suelen permanecer invariantes a lo largo de periodos de tiempo relativamente largos. Sin embargo, la frecuencia relativa de los diferentes tipos de *spam* puede cambiar de forma drástica y sucesiva durante dichos periodos. Este último cambio es virtual y puede provocar un cambio real en la definición de las reglas de usuario.

Para agravar aún más las dificultades del aprendizaje incremental, en un número creciente de aplicaciones los datos son constantemente generados con un elevado nivel de detalle desde diversas fuentes, debiendo ser procesados a gran velocidad bajo la fuerte sensibilidad al ruido que éstos presentan a causa del tráfico permanente entre dichas fuentes. Ejemplos de *data streams* o flujos continuos de datos son los registros generados en servicios web, telecomunicaciones u operaciones financieras. Ejemplos de aplicaciones de los mismos son la detección de intrusos en redes, el marketing dirigido, la identificación de fraudes en llamadas telefónicas u operaciones mediante tarjetas de crédito, donde los sistemas KDD deben operar *online* y procesar cada nuevo ejemplo en tiempo real [8]. En tales entornos, las limitaciones en tiempo y memoria impiden la aplicación de algoritmos *multi–pass* ya que los datos se reciben a una velocidad mucho mayor de la que pueden ser procesados [1, 20].

Como respuesta a una necesidad cada vez mayor, el aprendizaje en *data streams* - para los que tanto la función objetivo como la distribución de los datos es sensible al contexto y varía de forma impredecible - supone uno de los últimos retos de la minería de datos.

En este artículo describimos FACIL (***F**ast and **A**daptive **C**lassifier by **I**ncremental **L**earning*) [5], un algoritmo de aprendizaje incremental basado en reglas de decisión y dirigido especialmente a la clasificación de data streams con atributos numéricos. A pesar de la diversidad de técnicas propuestas en el aprendizaje incremental, hasta la fecha el reducido número de ellas basadas en reglas de decisión no han sido adaptadas para poder ser aplicadas a la clasificación y modelado en tiempo real de *data streams* numéricos, por lo que FACIL supone, hasta donde sabemos, la primera herramienta de aprendizaje incremental desarrollada especialmente para dominios de esta naturaleza.

2. Trabajo Relacionado

Entre todas las técnicas de decisión incrementales, la familia de algoritmos VFDT (*Very Fast Decision Tree*) parece estar llamada a convertirse, al igual que C4.5 dentro del aprendizaje por lotes, en el estándar para la clasificación de *data streams* mediante árboles [4, 6, 7, 11, 12].

Limitado inicialmente a *streams* de ejemplos con todos sus atributos simbólicos y provenientes de entornos estacionarios, VFDT [4] parte de la observación de Catlett [3] según la cual no es necesario utilizar todos los ejemplos de entrenamiento para encontrar la mejor pareja (atributo,valor) como siguiente testnodo 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 [10] 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 [12] 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 y en [13] se propone una extensión para procesar atributos numéricos mediante un esquema de discretización sin pérdida de precisión. Gama et al. [6, 7] han propuesto recientemente 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.

Por otro lado, los métodos de ensamblaje han recibido en los últimos cinco años una gran atención para el modelado y la clasificación de *data streams*. En general, las nuevas propuestas siguen el mismo esquema seguido por las técnicas de ensamblaje aplicadas en el aprendizaje por lotes, basándose en heurísticas y medidas de interés para eliminar, reactivar o añadir dinámicamente nuevos clasificadores en respuesta a las variaciones ocurridas en la consistencia del modelo con respecto a los nuevos ejemplos recibidos.

Siendo muy reducido el número de propuestas aparecidas en la literatura, entre las principales cabe destacar como pionera SEA [23], 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 bloque (*chunks*). Bajo el mismo esquema de ventana, Wang et al. [25] proponen un método basado en ponderación de clasificadores en función de la precisión obtenida por los mismos al utilizar como test los propios ejemplos de entrenamiento. Kolter & Maloof [16] proponen DWM, basado en el algoritmo de ponderación por mayoría de Littlestone [17] para ensamblar modelos de distinta edad.

3. Motivación

Tal y como observa Wang en [25], el principal inconveniente de los sistemas basdos en árboles de decisión es que pequeñas variaciones en la función objetivo pueden suponer complejas modificaciones en el árbol de salida, comprometiendo severamente el coste computacional y la eficiencia en el aprendizaje.

Por su parte, los métodos de ensamblaje basados en ponderación de ejemplos evitan revisiones innecesarias en beneficio de la eficiencia en el proceso de aprendizaje. Aun así, obtienen un rendimiento considerablemente inferior al obtenido por las técnicas basadas en reglas que emplean ventana deslizante [15]. Esto es debido a que están dirigidas a la unión de árboles de decisión, los cuales poseen una sensibilidad mucho mayor al sobreajuste que las reglas.

En cuanto a los algoritmos de aprendizaje incremental basados directamente en reglas como STAGGER [22], FLORA [27] y la familia de algoritmos AQ–PM [19], todos ellos fueron

diseñados para el aprendizaje de conceptos a partir de atributos discretos, teniendo en cuenta todos los ejemplos del conjunto de entrenamiento, por lo que hasta la fecha ninguno ha sido adaptado a entornos *data streams* cuyos ejemplos contienen atributos numéricos.

4. Reglas y Ejemplos Fronterizos

El principal distintivo de FACIL frente a las técnicas basadas en reglas relacionadas es el empleo de un esquema de múltiples ventanas, una por cada regla. Mediante esta estrategia, las reglas pueden no ser consistentes, almacenando un conjunto de ejemplos fronterizos - cercanos a otros de distinta etiqueta - que van siendo actualizados dinámicamente a medida que cambian las fronteras de decisión. Una regla se dice consistente cuando no cubre a ningún ejemplo negativo (de distinta etiqueta). Al igual que AQ11-PM [18, 19], FACIL genera un conjunto de hiperrectángulos como modelo y retiene en memoria un subconjunto de los ejemplos cubiertos por cada uno de ellos. Sin embargo, mientras que AQ11-PM retiene únicamente ejemplos positivos, FACIL asocia a cada regla un conjunto de ejemplos positivos y negativos.

Cuando un nuevo ejemplo del *stream* es recibido, AQ11-PM lo añade al conjunto de ejemplos que forman la ventana en ese momento y aplica el algoritmo de aprendizaje base AQ11 para modificar el modelo actual (el conjunto de reglas de ese momento), seleccionando para cada una de ellas ejemplos positivos pertenecientes a sus vértices, aristas o lados. Dichos ejemplos son denominados *extremos*.

A diferencia de AQ11-PM, las reglas en FACIL almacenan al menos un ejemplo positivo por cada ejemplo negativo cubierto. Tanto los ejemplos positivos como los negativos no son necesariamente extremos - pertenecen necesariamente a los vértices, aristas o lados de los hiperrectángulos - sino que buscan ser lo más cercanos posibles a otros de distinta etiqueta. Esto permite que, a diferencia de AQ11-PM, las reglas no sean constantemente reparadas cada vez que dejan de ser consistentes, reduciendo notablemente el coste computacional del proceso de modelado.

Puesto que el número de ejemplos negativos *en* que una regla puede almacenar incrementa a medida que aumenta su soporte o número de ejemplos positivos cubiertos, cada vez que *en* incrementa en una unidad, un nuevo ejemplo positivo es almacenado. Por ejemplo, sea $p = 90\,\%$ la pureza mínima o umbral de inconsistencia para cada regla. La pureza de una regla es la proporción de ejemplos positivos cubiertos con respecto al total de ejemplos cubiertos por la misma (positivos y negativos). Cuando una regla r ha cubierto 45 ejemplos positivos, el máximo número de ejemplos positivos *en* que puede cubrir r es 5. Así, tras cubrir r a 54 ejemplos positivos, *en* es 6.

Aunque esta estrategia padece los efectos provocados por el orden de llegada de los ejemplos, no compromete la eficiencia del proceso de aprendizaje y garantiza que toda regla errónea (cuya pureza sea inferior al umbral) sea reparada a partir de tantos ejemplos positivos como ejemplos negativos. Una regla es reparada cuando su pureza es inferior al umbral dado por el usuario.

5. Generalización Moderada

De aquí en adelante se utilizará la siguiente notación para describir la lógica algorítmica de FACIL. Sea m el número de atributos - continuos - del *stream* de entrada. Sea $Y = \{y_1, \ldots, y_z\}$ el conjunto de etiquetas de clase. Sea $e_i = (\overrightarrow{x_i}, y_i)$ el nuevo ejemplo recibido, donde $\overrightarrow{x_i}$ es un vector normalizado en $[0, 1]^m$ e y_i es valor discreto perteneciente a Y. Una regla de decisión r viene dada por un conjunto de m intervalos cerrados $[I_{ja}, I_{jb}]$ ($j \in \{1, \ldots, m\}$) que definen a un hiperrectángulo en el espacio de instancias. a y b denotan respectivamente los límites inferior y superior de cada intervalo.

Puesto que FACIL no utiliza una ventana global para cada todas las reglas del modelo sino que cada una de ellas gestiona su propia ventana de ejemplos, cada vez que un nuevo ejemplo e_i es recibido el modelo puede ser actualizado. Almacenando las reglas en distintos conjuntos en función de su etiqueta, el proceso

de actualización es llevado a cabo comprobando al menos una de las tres situaciones descritas a continuación, éstas, evaluadas en el orden siguiente:

1. ***Cobertura Positiva***: x_i es cubierto por una regla asociada a su misma etiqueta y_i.
2. ***Cobertura Negativa***: x_i es cubierto por una regla asociada a otra etiqueta distinta $y' \neq y_i$.
3. ***Nueva Descripción***: x_i no es cubierto por ninguna regla.

Cobertura Positiva. En primer lugar, FACIL recorre las reglas asociadas a la etiqueta y_i del nuevo ejemplo calculando, para cada regla visitada, la generalización necesaria para describirlo. Para ello, esta generalización es medida según la definición 1.

Definition 1 (Crecimiento de una regla) *Sea r una regla en $[0,1]^m$ formada por m intervalos cerrados $[I_{ja}, I_{jb}]$. Sea x un punto en $[0,1]^m$. El crecimiento $\mathcal{G}(r, x_i)$ de una regla r necesario para cubrir a un nuevo ejemplo se define como:*

$$\mathcal{G}(r, x_i) = \sum_{j=1}^{m} (g_j - r_j);$$
$$g_j = b_j - a_j;\ r_j = I_{jb} - I_{ja};$$
$$b_j = \text{máx}(x_{ij}, I_{jb});\ a_j = \text{mín}(x_{ij}, I_{ja});$$

Esta heurística da una estimación de la nueva región del espacio que sería ocupada por la regla a generalizar, favoreciendo aquella que supone el menor número de cambios en los términos de su antecedente. Es decir, se favorece a aquella regla para la cual se modifican lo mínimo sus intervalos. Mientras se visitan las reglas asociadas a la etiqueta del ejemplo, aquella que implica el menor crecimiento se marca como *candidata*, siempre y cuando el crecimiento que supone sea moderado. Decimos que un crecimiento es moderado si: $\forall j \in \{1, \ldots, m\} : \ g_j - r_j \leq \kappa$. La figura 1 ilustra un ejemplo en el que dos reglas no satisfacen esta condición siendo $\kappa = 0{,}1$. Puesto que cada ejemplo es previamente normalizado en $[0,1]^m$, el divisor para el dominio de cada atributo es omitido en la definición 1. Durante

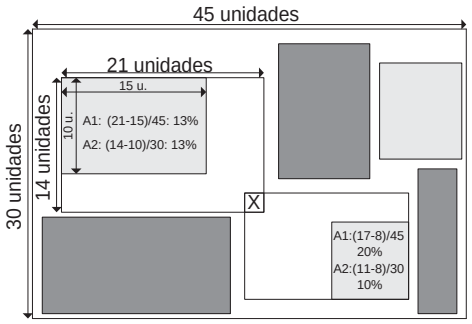

Figura 1: Dos reglas a priori candidatas para describir a un nuevo ejemplo no pueden ser consideradas como tales ya que la generalización resultante de ambas excede el máximo crecimiento permitido (κ =10 %) en el atributo A1.

este recorrido, si se encuentra una regla que cubra al nuevo ejemplo x_i - para la cual el crecimiento será 0 - el número de ejemplos positivos *ep* que ésta cubre se incrementa en una unidad y el índice del último ejemplo cubierto por la misma pasa a ser i. Si tras aumentar *ep*, el número *en* de posibles ejemplos negativos que la regla puede cubrir aumenta, entonces el nuevo ejemplo es añadido a su ventana.

Cobertura Negativa. Si x_i no es cubierto por ninguna regla asociada a y_i, FACIL recorre las reglas asociadas a las demás etiquetas $y' \neq y_i$ de forma secuencial. En caso de existir una regla *candidata* r_c para cubrir x_i, para cada nueva regla r' visitada que no cubra a x_i se calcula la intersección entre r_c y r'. Si la intersección no es vacía, r_c se rechaza de forma que la intersección no se vuelve a calcular para las demás reglas. Cuando la primera regla r' que cubre al nuevo ejemplo es encontrada, su número *en* de ejemplos negativos se incrementa en una unidad y el nuevo ejemplo x_i se añade a su ventana. En este punto, si la nueva pureza de r' es inferior al umbral de consistencia, FACIL incorpora al modelo nuevas reglas, todas ellas consistentes, a partir de los ejemplos positivos y negativos pertenecientes a la ventana de r'. Una vez ampliado el modelo, r' se marca como *dudosa* impidiendo su generalización y no siendo tenida en cuenta como po-

sible intersección en la generalización de otras reglas de distinta etiqueta. Por último, la ventana de r' es reinicializada a 0.

Nueva Descripción. Tras llevar a cabo los dos recorridos anteriores, la regla *candidata* es generalizada. Si ésta fue rechazada en el recorrido anterior y no existe ninguna regla en el modelo que cubra al nuevo ejemplo, FACIL añade al mismo una nueva regla puntual que lo describa.

5.1. Complejidad Computacional

En el peor de los casos, el nuevo ejemplo a describir exigirá la creación de una nueva regla puntual, visitando todas las reglas del modelo, por lo que la complejidad computacional será de orden $\mathcal{O}(m \cdot s \cdot \overline{e})$, siendo m el número de atributos, s el número total de reglas en el modelo y $\overline{e}$ el promedio de ejemplos asociados a cada regla.

6. Refinado del modelo

Simultáneamente al proceso de actualización, FACIL lleva a cabo un proceso de refinado a medida que visita las reglas del modelo. Antes de comprobar si una regla cubre al nuevo ejemplo recibido, ésta es eliminada si la última regla *candidata* - la última regla generalizada - asociada a su misma etiqueta la cubre en su totalidad. Si no es el caso, tras comprobar que la regla visitada no cubre al nuevo ejemplo ésta se elimina si satisface al menos una de los condiciones siguientes:

- Es una regla *dudosa* con un soporte inferior al de cualquiera de las reglas generadas a partir de los antiguos ejemplos de su ventana.
- El número de veces que dicha regla impidió a otras de distinta etiqueta ser generalizadas - por provocar una intersección no vacía - es mayor que su soporte.

7. Políticas de Olvido

De forma similar a AQ-PM, FACIL proporciona dos políticas de olvido, una implícita basada en el vecino más cercano para modelar fenómenos estacionarios, y otra explícita basada en frecuencia para modelar *streams* sensibles al contexto (*hidden context*) y sujetos a cambios en la función objetivo (*concept drift*).

La política explícita consiste simplemente en olvidar o eliminar de las ventanas asociadas a las reglas aquellos ejemplos cuya edad supera un umbral dado como parámetro de usuario.

La política implícita se lleva a cabo olvidando los ejemplos que han dejado de ser relevantes, es decir, aquellos que han dejado de pertenecer a las fronteras de decisión. Cuando un ejemplo negativo x perteneciente a una regla r tiene como vecino más cercano a otro de distinta etiqueta después de que el número de ejemplos positivos ep de r incremente en dos ocasiones desde que x fue cubierto, dicho ejemplo es eliminado. De forma análoga, un ejemplo positivo es eliminado si tiene como vecino más cercano a otro de igual etiqueta (a otro ejemplo positivo para la regla) después de que ep incremente en dos unidades desde que éste fue cubierto.

8. Clasificación de nuevos ejemplos de test

Finalmente, para clasificar a un nuevo ejemplo de test, FACIL busca una regla no *dudosa* que lo cubra. Puesto que reglas de distinta etiqueta no intersecan, si dicha regla existe y es consistente, entonces el ejemplo es clasificado directamente con la etiqueta asociada. Si la regla es inconsistente, el ejemplo se clasifica mediante el vecino más cercano a partir de los ejemplos de su ventana. En el caso en el que ninguna regla cubra al nuevo ejemplo de test, éste se clasifica con la etiqueta asociada a la regla no *dudosa* que, sin provocar una intersección con otra de distinta etiqueta, implique el menor crecimiento para describirlo.

A partir de esta información, FACIL proporciona una flexibilidad en la clasificación de nuevos ejemplos inexistente en las listas de decisión o conjuntos de reglas estándar, pudiendo aplicar, en función de la certeza sobre la región modelada a la que pertenece el ejemplo, tres criterios de etiquetado: por cobertura (induc-

ción), por votación (deducción) y por distancia mínima.

9. Evaluación Empírica

Si bien los conceptos STAGGER [22] proporcionan un *benchmark* estándar para la evaluación de algoritmos de aprendizaje incremental capaces de abordar el problema del *concept drift* en ejemplos con todos sus atributos simbólicos, actualmente los sistemas de decisión dirigidos a la clasificación de *data streams* con atributos numéricos carecen de un método experimental estándar para ser evaluados.

Siguiendo el método propuesto en [12, 25] para medir la robustez y fiabilidad de clasificadores incrementales basados en árboles de decisión y ensamblaje de modelos, FACIL es evaluado a partir de *data streams* dinámicos cuya distribución de etiquetas varía en función de un hiperplano rotante. Paralelamente y de forma similar a la metodología propuesta en [24], FACIL es también evaluado como clasificador de propósito general a partir de bases de datos reales del repositorio UCI [2] con todos sus atributos continuos. Ambas series de experimentos fueron realizadas en un PC con CPU de 1.7GHz bajo Windows XP. Los resultados obtenidos se presentan en las tablas 1–3.

9.1. Bases de datos estacionarias

En esta primera serie de experimentos los datos de entrada no presentan ruido ni están sujetos a *concept drift*, por lo que el objetivo es evaluar el rendimiento de FACIL como clasificador de propósito general. Para ello se compararon las listas de decisión de C4.5Rules con los conjuntos generados por FACIL mediante la política de olvido implícita basada en el vecino más cercano. Para cada base de datos se llevaron a cabo diez repeticiones de validación cruzada con 10 particiones reordenando de forma aleatoria en cada una de ellas todos los ejemplos de entrenamiento. Para conseguir una comparativa justa, el umbral de consistencia mínima para cada regla se fijó a partir de la precisión obtenida por C4.5Rules. De forma similar, como máximo crecimiento se fijó el mayor valor para el cual el número de reglas del modelo final no superaba el tamaño del modelo obtenido mediante C4.5Rules. La tabla 1 muestra los valores promedio de los distintos parámetros medidos (precisión, tiempo de aprendizaje complejidad del modelo) para 100 ejecuciones. Para la precisión y el número total de reglas, aquellos valores marcados con $\star$ ó $\bullet$ implican respectivamente una mejora o desmejora según una t–student con $\alpha = 0{,}05$. En cinco bases de datos (*Balance–Scale, Glass, Pima–Diabetes, Vehicle* y *Waveform*) FACIL proporciona un aumento significativo con respecto a la precisión (columna PA), siendo éste altamente notable en *Pima–Diabetes* donde el valor medio alcanza el 89,49 %, lo que supone un incremento de 15 puntos con respecto a C4.5Rules. Paralelamente, el número de reglas es también mejorado significativamente en dos bases de datos, *Balance–Scale* y *Waveform*, para las cuales se reduce respectivamente en 12 y 6 puntos. De la última fila de la tabla 1 se deduce que la propuesta de FACIL por un criterio de generalización moderada beneficia de forma global a la precisión del modelo obtenido, siendo muy reducido tanto el número medio de reglas como el de ejemplos.

9.2. Hiperplano rotante

En una segunda serie de experimentos evaluamos FACIL a partir de *data streams* dinámicos cuya distribución de etiquetas varía en función de un hiperplano rotante según el método propuesto en [12, 25] añadiendo un nivel de ruido del 5 %. Las tablas 2 y 3 muestran los resultados obtenidos tras procesar 50000 ejemplos. En ambos casos, el umbral de consistencia es fijado al 90 %. Puesto que en el aprendizaje incremental el entrenamiento y el test se llevan a cabo simultáneamente, tras cada 900 ejemplos de entrenamiento eran clasificados 100 nuevos ejemplos de test. Las resultados señalan cómo la heurística de olvido explícita proporciona un rendimiento significativamente superior frente a la heurística implícita tanto en el tiempo de aprendizaje como en la exactitud del modelo obtenido. Con respecto a la precisión, el valor medio en todos los *streams* supera el 90 % con los dos límí-

Cuadro 1: Valores promedio de precisión (PA), número de reglas (NR), ejemplos por regla (ER) y máximo crecimiento (MG) obtenidos mediante la heurística de olvido implícita.

	C4.5Rules		FACIL		
Database	PA	NR	PA	ER	MG
Balance S.	82.92±4.3	38.12•	95.68±1.3⋆	4.8	90
Breast C.	94.76±2.8	9.89	95.27±1.5	3.4	70
Glass	68.85±9.9	15.31	77.10±1.4⋆	3.2	85
Heart S.	77.81±8.7	17.82⋆	80.96±1.9	2.0	75
Ionosphere	90.71±4.9	7.45⋆	90.31±2.3	2.6	75
Iris	94.66±6.4	3.93	97.57±0.6	3.0	85
Pima D.	73.73±4.4	7.48⋆	89.49±3.4⋆	8.9	60
Sonar	74.31±8.9	7.22⋆	75.04±1.9	1.6	40
Vehicle	72.50±4.2	32.43	76.28±3.8⋆	2.3	55
Vowel	78.45±4.2	65.95⋆	77.53±3.9	2.0	100
Waveform	77.92±1.7	86.65•	84.77±7.9⋆	11.8	35
Wine	92.13±6.3	4.60⋆	95.39±0.9	2.2	80
Av.	81.56±5.6	24.73	86.28±2.6	3.9	70.8

tes de crecimiento. Con respecto al tiempo de aprendizaje, este alcanza los 3500 ejemplos por segundos para streams de 10 atributos. En general, dada la gran simplicidad de los modelos obtenidos, los resultados son excelentes.

10. Conclusiones y Trabajo Futuro

En este artículo hemos descrito y evaluado un algoritmo incremental basado en reglas que, a diferencia de las principales técnica relacionadas, utiliza un esquema de múltiples ventanas y una política de generalización moderada. Gracias a esta estrategia y a un criterio de clasificación flexible, las reglas pueden ser inconsistentes sin dañar la exactitud del modelo, con lo que se consiguen tres grandes beneficios: 1) evitar revisiones innecesarias frente a cambios virtuales; 2) reducir el coste computacional y 3) refinar el modelo simultáneamente al proceso de actualización sin comprometer la eficiencia en el aprendizaje. De forma similar a AQ-PM, FACIL proporciona una política de olvido implícita mediante la cual los ejemplos son descartados cuando dejan de pertenecer a una frontera de decisión. A partir de esta estrategia el modelo puede ser actualizado acorde con las nuevas condiciones del entorno sin la necesidad de conocer la periodicidad de los cambios. Los resultados experimentales obtenidos mediante bases de datos del almacén UCI y *data streams* dinámicos a partir de planos rotatorios demuestran el excelente rendimiento de FACIL como clasificador *multi–class* de propósito general. Nuestra futuras líneas de investigación están orientadas a la eliminación simultánea de atributos irrelevante, así como la recuperación de atributos previamente eliminados que vuelven a ser relevantes posteriormente. Actualmente estamos también evaluando medidas de crecimiento alternativas para procesar *data streams* con atributos nominales y poder comparar nuestra propuesta con otros clasificadores como CVFDT [12] and VFDTc [6].

Referencias

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the* 21^{th} *Symposium on Principles of Database SystemsProc - PODS'02*, pages 1–16.

[2] C. Blake and E.K. Merz. UCI repository of machine learning databases, 1998.

[3] J. Cattlet. *Megainduction: machine learning on very large databases.* PhD thesis,

Cuadro 2: Precisión (PA), tiempo de entrenamiento y test en segundos (LT) y número de reglas (NR) con crecimiento máximo del 50 % para *streams* dinámicos con un número de atributos entre 10 y 50 (NA).

	H. Explícita			H. Implícita		
NA	PA	LT	NR$\leq$ 50	PA	LT	NR$\leq$ 100
10	96.36	14	10	84.61	41	153
20	93.25	63	9	67.13	107	150
30	91.04	124	3	63.17	137	125
40	89.70	221	2	59.65	383	22
50	83.88	277	4	59.50	428	9

Cuadro 3: Precisión (PA), tiempo de entrenamiento y test en segundos (LT) y número de reglas (NR) con crecimiento máximo del 75 % para *streams* dinámicos con un número de atributos entre 10 y 50 (NA).

	H. Explícita			H. Implícita		
NA	PA	LT	NR$\leq$ 100	PA	LT	NR$\leq$ 50
10	98.32	20	7	87.93	35	29
20	94.57	35	12	55.61	131	27
30	89.26	53	10	53.48	124	53
40	89.19	67	7	52.25	195	65
50	87.72	77	10	51.21	246	36

Basser Department of Computer Science, University of Sydney, Australia, 1991.

[4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc. of the* 6^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'00*, pages 71–80.

[5] F.J. Ferrer, J.S. Aguilar, and J.C. Riquelme. Incremental rule learning based on example nearness from numerical data streams. In *Proc. of the* 20^{th} *ACM Symposium On Applied Computing, Data Mining Track – SAC'05)*, pages 572–576, Santa Fe, Nuevo México, 2005.

[6] J. Gama, P. Medas, and R. Rocha. Forest trees for on-line data. In *Proc. of the* 19^{th} *ACM Symposium on Applied Computing - SAC'04*, pages 632–636.

[7] J. Gama, R. Rocha, and P. Medas. Accurate decision trees for mining high-speed data streams. In *Proc. of the* 9^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'03*, pages 523–528.

[8] L. Golab and M.T. Ozsu. Issues in data stream management. *SIGMOD Record*, 32(2):5–14, 2003.

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

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

[11] G. Hulten, L. Spencer, and P. Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proc. of the* 8^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'02*.

[12] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of the* 7^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'01*, pages 97–106.

[13] R. Jin and G. Agrawal. Efficient decision tree construction on streaming data. In *Proc. of the* 9^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'03*.

[14] M.G. Kelly, D.J. Hand, and N.M. Adams. The impact of changing populations on classier performance. In *Proc. of the* 5^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'99*, pages 367–371. ACM Press, 1999.

[15] R. Klinkenberg. Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*, 8(3), 2004.

[16] Jeremy Z. Kolter and M.A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proc. of the* 3^{th} *IEEE Int. Conf. on Data Mining - ICDM'03*, pages 123–130, 2003.

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

[18] M.A. Maloof. Incremental rule learning with partial instance memory for changing concepts. In *Proc. of the* 15^{th} *IEEE Int. Joint Conf. on Neural Networks - IJCNN'03*, pages 2764–2769, 2003.

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

[20] S. Muthukrishnan. Data streams: algorithms and applications. In *Proc. of the* 14^{th} *Annual ACM SIAM Symposium on Discrete Algorithms - SODA'03. Invited Plenary Session*, pages 413–413, 2003.

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

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

[23] W. Street and Y. Kim. A streaming ensemble algorithm SEA for large-scale classification. In *Proc. of the* 7^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'01*, pages 377–382.

[24] N. Syed, H. Liu, and K. Sung. Handling concept drifts in incremental learning with support vector machines. In *Proc. of the* 5^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'99*, pages 272–276. ACM Press, 1999.

[25] H. Wang, W. Fan, P.S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of the* 9^{th} *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining - KDD'03*, pages 226–235.

[26] G. Widmer and M. Kubat. Effective learning in dynamic environments by explicit context tracking. In *Proc. of the* 6^{th} *European Conf. on Machine Learning - ECML'93*, pages 227–243. Springer-Verlag, LNCS 667, 1993.

[27] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.