NUEVOS MÉTODOS PARA EL APRENDIZAJE EN FLUJOS DE DATOS NO ESTACIONARIOS

Tesis Doctoral

ISVANI INOCENCIO FRÍAS BLANCO
Universidad de Granma

Directores

Dr. Gonzalo Ramos Jiménez
Dr. Rafael Morales Bueno
Universidad de Málaga

Editor: Editorial de la Universidad de Granada
Autor: Isvani Inocencio Frías Blanco
D.L.: GR 2347-2014
ISBN: 978-84-9083-391-9

A mi querida familia.

AGRADECIMIENTOS

En primer lugar quiero agradecer a mis tutores Gonzalo Ramos Jiménez, Rafael Morales Bueno y Yailé Caballero Mota, al igual que a José del Campo Ávila. Ellos han contribuido mucho con sus ideas, sugerencias y ayuda. Me han hecho evolucionar en mi carrera investigativa y sin ellos, no hay dudas que esta tesis no hubiera sido posible. A pesar de la distancia geográfica su apoyo ha sido invaluable.

Debo agradecer al DECSAI en la Universidad de Granada, al Centro de Investigaciones de Informática de la Universidad Central "Martha Abreu" de Las Villas y a la Asociación Universitaria Iberoamericana de Postgrado (AUIP), por hacer posible el programa de doctorado y por el apoyo económico. También al Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga, por recibirme durante tres meses muy provechosos de investigación.

También quisiera agradecer a mis compañeros de trabajo y amigos, en la Universidad de Granma y en la Universidad de las Ciencias Informáticas, por compartir muchos buenos momentos juntos y darme ánimo constantemente.

Más que todo agradecer a mi maravillosa familia.

RESUMEN

Esta tesis se enfoca en el desarrollo de algoritmos de aprendizaje para la minería de flujos de datos cuya función objetivo puede cambiar en el tiempo. En estos escenarios los datos llegan con una velocidad alta y constantemente. Debido a la gran cantidad de datos y a su tasa de llegada alta, estos deben procesarse con recursos computacionales limitados de tiempo y espacio. Adicionalmente, estos datos pueden estar sujetos a cambios distribucionales en el tiempo, haciendo posible que un modelo de aprendizaje inducido previamente sea obsoleto o incluso contradictorio con respecto a los datos actuales, problema conocido como cambio de concepto.

Proponemos dos acercamientos fundamentales, uno que puede ser utilizado por cualquier algoritmo de aprendizaje para adaptarse a cambios de concepto, y el segundo que explota características específicas de los modelos de árboles de decisión para adaptarse con más efectividad a los cambios. El primero, capaz de adaptarse a cambios de concepto independientemente del algoritmo de aprendizaje, tiene como componente principal a un detector de cambio en línea, que mantiene actualizados estadísticos calculados a partir de un flujo de valores reales. Este flujo de valores reales corresponde a medidas de rendimiento del algoritmo de aprendizaje en el tiempo. De esta forma, cambios significativos en los estadísticos involucrados se interpretan como cambio de concepto. La detección de estos cambios se utiliza para entrenar un modelo de aprendizaje alternativo en vista a sustituir al más antiguo. El detector de cambio propuesto, que hemos llamado HDDM (*Hoeffding-like inequalities for Drift Detection Methods*), tiene garantías probabilísticas de rendimiento en términos de cotas para la tasa de falsos positivos y falsos negativos en la detección de cambios. En la investigación hemos probamos nuestro método con varios algoritmos de aprendizaje como son el clasificador Naïve Bayes, el Perceptrón simple y mediante árboles de decisión.

En la inducción incremental de árboles de decisión, proponemos varias mejoras al algoritmo IADEM-2 (*Inducción de Árboles de DEcisión por Muestreo*) relacionadas con la convergencia del modelo, con la manipulación de atributos numéricos, con tareas de predicción y con la manipulación de cambio de concepto. A pesar de que HDDM puede ser usado directamente en combinación con IADEM-2 para manipular cambio de concepto, explotamos características específicas de los modelos de árboles de decisión para que puedan adaptarse con más efectividad a posibles cambios. Para ello se tienen en cuenta tipos de cambios comunes y los requerimientos generales para el aprendizaje en flujos de datos. De esta forma el algoritmo resultante, que hemos llamado IADEM-3, mejora significativamente la precisión en tareas de clasificación con respecto a IADEM-2 cuando se ejecuta en flujos de datos estacionarios; además de ser capaz de adaptarse a posibles cambios de concepto. Adicionalmente, los árboles generados por IADEM-3 generalmente son más pequeños que los generados por su predecesor. El algoritmo IADEM-3 también tiene un coste computacional comparable con IADEM-2, al considerar en sus mejoras métodos con complejidad computacional que no dependen del número de ejemplos de entrenamiento vistos.

ÍNDICE GENERAL

I INTRODUCCIÓN Y MARCO DE TRABAJO 1

1 INTRODUCCIÓN 3
- 1.1 Minería de datos y aprendizaje automático . . . 3
- 1.2 Minería de flujos de datos . . . 4
- 1.3 Cambio de concepto . . . 5
- 1.4 Inducción incremental de árboles de decisión . . . 6
- 1.5 Diseño de la investigación, estructura y aportes . . . 7

2 PRELIMINARES 13
- 2.1 Definiciones y tipos de cambio . . . 13
  - 2.1.1 Definiciones . . . 14
  - 2.1.2 Tipos . . . 15
- 2.2 Acercamientos teóricos . . . 16
- 2.3 Detección y aprendizaje . . . 18

3 MINERÍA DE FLUJOS DE DATOS NO ESTACIONARIOS 21
- 3.1 Métodos genéricos para manipular cambio de concepto . . . 21
  - 3.1.1 Métodos estadísticos . . . 22
  - 3.1.2 Acercamientos basados en ventanas . . . 24
  - 3.1.3 Pesado de instancias . . . 26
- 3.2 Métodos específicos acorde al modelo de aprendizaje . . . 27
  - 3.2.1 Árboles de decisión . . . 28
  - 3.2.2 Reglas de decisión . . . 31
  - 3.2.3 Máquinas de soporte vectorial . . . 33
  - 3.2.4 Redes neuronales artificiales . . . 34
  - 3.2.5 Aprendizaje basado en instancias . . . 36
  - 3.2.6 Ensamble de clasificadores . . . 38
- 3.3 Evaluación de algoritmos incrementales . . . 39
  - 3.3.1 Métricas . . . 41
  - 3.3.2 Metodología . . . 42
  - 3.3.3 Conjuntos de datos . . . 44
- 3.4 Herramientas de software de código abierto . . . 51
- 3.5 Conclusiones del capítulo . . . 52

II APRENDIZAJE EN FLUJOS DE DATOS CON CAMBIO DE CONCEPTO 55

4 NUEVOS MÉTODOS PARA LA DETECCIÓN EN LÍNEA DE CAMBIO DE CONCEPTO 57
- 4.1 Introducción . . . 57
- 4.2 Monitorizando estadísticos . . . 58
- 4.3 *A*-test: Acotando diferencia de promedios . . . 61
- 4.4 *W*-test: Acotando promedios ponderados . . . 63
  - 4.4.1 Acotando la diferencia entre promedios ponderados . . . 64
  - 4.4.2 Esquemas de pesado compatibles con *W*-test . . . 65
  - 4.4.3 Relación entre *A*-Test y *W*-Test . . . 66

4.5 El algoritmo para detectar cambios en línea . . . 67
4.6 Estudio empírico . . . 70
4.6.1 Rendimiento en flujos de bits . . . 71
4.6.2 Flujos de datos sintéticos . . . 75
4.6.3 Flujos de datos reales . . . 83
4.7 Conclusiones del capítulo . . . 86
5 IADEM-2C: MEJORANDO LA INDUCCIÓN DE ÁRBOLES DE DECISIÓN 89
5.1 Cotas de concentración de Chernoff y de Hoeffding . . . 90
5.2 La familia de algoritmos IADEM . . . 92
5.3 Mejora de la calidad de las expansiones . . . 95
5.4 Tratamiento de atributos numéricos . . . 96
5.4.1 Métodos de discretización aplicables a la inducción incremental de árboles de decisión . . . 97
5.4.2 Disminuyendo el coste computacional . . . 98
5.5 Evaluación empírica . . . 99
5.5.1 Pruebas binarias, múltiples y mixtas . . . 101
5.5.2 Métodos de discretización . . . 103
5.5.3 Conjuntos de datos reales . . . 105
5.6 Conclusiones del capítulo . . . 106
6 IADEM-3: MANIPULANDO CAMBIO DE CONCEPTO 109
6.1 Discusión de la estrategia de adaptación al cambio . . . 110
6.2 El nuevo algoritmo . . . 113
6.2.1 Detección de cambio . . . 113
6.2.2 Adaptación . . . 114
6.2.3 Método de predicción . . . 116
6.3 Estudio experimental . . . 117
6.3.1 Configuración de los algoritmos . . . 118
6.3.2 Datos sintéticos . . . 118
6.3.3 Datos del mundo real . . . 125
6.4 Conclusiones del capítulo . . . 127

III CONCLUSIONES Y ANEXOS 129
7 CONCLUSIONES Y TRABAJOS FUTUROS 131
7.1 Conclusiones . . . 131
7.2 Limitaciones y trabajos futuros . . . 132
7.2.1 Detectores de cambio en línea . . . 133
7.2.2 Inducción de árboles de decisión . . . 133
A DEMOSTRACIONES MATEMÁTICAS 135
A.1 Demostración del Corolario 3 . . . 135
A.2 Demostración del Corolario 5 . . . 135
A.3 Demostración del Ejemplo 7 . . . 136
A.4 Demostración del Ejemplo 8 . . . 136
B CURSOS DE DOCTORADO 137
C PUBLICACIONES 141

REFERENCIAS BIBLIOGRÁFICAS 143

ÍNDICE DE FIGURAS

Figura 1.1 Esquema general de las etapas del proceso de extracción de conocimiento. 4

Figura 2.1 (a) Diferentes problemas en un problema bi-dimensional. Diferentes escenarios para cambio de concepto: (b) cambio abrupto, (c) cambio gradual continuo y (d) cambio gradual discreto. 16

Figura 2.2 Categorización de las técnicas para manipular cambio de concepto atendiendo a su taxonomía. 18

Figura 2.3 Categorización de las técnicas para manipular cambio de concepto atendiendo a su especificidad. 19

Figura 3.1 Diferentes métodos de evaluación aplicados a la estimación de la precisión del algoritmo IADEM-2 aprendiendo en conceptos STAGGER. Se simula un cambio de concepto a partir del ejemplo número 10 000. Las ventanas deslizantes tienen un tamaño de 1 000 ejemplos. 44

Figura 3.2 Combinación de dos conceptos diferentes para simular un cambio gradual discreto: (a) usando una función con pendiente y (b) usando una función sigmoide. 45

Figura 4.1 Esquema para la manipulación de cambio de concepto a través de detectores de cambio en línea. 59

Figura 4.2 Conducta típica del rendimiento de un algoritmo de aprendizaje en tipos de cambio abrupto (Figura 4.2a) y gradual (Figura 4.2b). 60

Figura 4.3 Comportamiento típico de la precisión de los algoritmos de detección (con el clasificador Naïve Bayes) en cambios abruptos: 3 cambios en conceptos LED y 10 000 ejemplos por concepto. 81

Figura 4.4 Comportamiento típico de la precisión de los algoritmos de detección (con el clasificador Naïve Bayes) en cambios graduales: 3 cambios en conceptos AGRAWAL, 10 000 ejemplos por concepto y 5 000 ejemplos en el período de transición entre conceptos. 82

Figura 4.5 Precisión de los detectores de cambio propuestos en combinación con el Perceptrón en el conjunto de datos NURSERY 85

Figura 5.1 Representación de un árbol de decisión inducido por la familia IADEM [delo7]. 92

Figura 6.1 Esquema para la manipulación de cambio de concepto. . . . 112

Figura 6.2 Curvas de aprendizaje de los algoritmos en conceptos estables sobre el generador LED. 120

Figura 6.3 Curvas de aprendizaje de los algoritmos en los conceptos LED con cambios abruptos. Los cambios ocurren cada 30 000 ejemplos de entrenamiento. 124

Figura 6.4 Curvas de aprendizaje de los algoritmos en conceptos AGRAWAL con cambios graduales. Los cambios ocurren cada 30 000 ejemplos y con 3 000 ejemplos de entrenamiento en el período de transición entre conceptos. 124
Figura 6.5 Curvas de aprendizaje de los algoritmos en el conjunto de datos *Electricity Market*. 127

ÍNDICE DE TABLAS

Tabla 3.1 Métodos basados en ventanas y de pesado de instancias para detectar cambio de concepto usados en el aprendizaje incremental. 27
Tabla 3.2 Estrategias generales para la detección de cambio de concepto considerando el modelo de aprendizaje. 40
Tabla 3.3 Generadores de conjuntos de datos. Categorización de los generadores de conjuntos de datos considerando el tipo de cambio de concepto que ellos simulan y el tipo de atributos usado en el problema. El número de clases es representado entre paréntesis. El generador *wrapper* se refiere a la combinación ponderada de diferentes conceptos (con cambio abrupto) usando una función con pendiente o sigmoide. 49
Tabla 4.1 Valor medio de las medidas de rendimiento en flujos de datos, 100 bits por valor de la media estable. 71
Tabla 4.2 Valor medio de las medidas de rendimiento en flujos de datos, 1 000 bits por valor de la media estable. 72
Tabla 4.3 Valor medio de las medidas de rendimiento en flujos de datos, 100 000 bits por valor de la media estable. 73
Tabla 4.4 Promedio y desviación estándar de la precisión en experimentos con 100 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios abruptos. 76
Tabla 4.5 Promedio y desviación estándar de la precisión en experimentos con 100 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios graduales. El período de transición entre conceptos consecutivos es de 50 ejemplos. 77
Tabla 4.6 Promedio y desviación estándar de la precisión en experimentos con 10 000 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios abruptos. 78

Tabla 4.7 Promedio y desviación estándar de la precisión en experimentos con 10 000 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios graduales. El período de transición entre conceptos consecutivos es de 5 000 ejemplos. 79
Tabla 4.8 Precisión de detectores de cambio en combinación con el clasificador Naïve Bayes y Perceptrón en conjuntos de datos reales. 84
Tabla 5.1 Generadores de flujos de datos para la experimentación con distintos tipos de prueba instalada en los nodos de decisión en IADEM-2. 102
Tabla 5.2 Precisión y número de nodos de IADEM-2c comparando distintos tipos de pruebas para la expansión: salidas múltiples, salidas binarias y salidas mixtas (múltiples o binarias). Fueron 1 000 000 de ejemplos de entrenamiento, *holdout* cada 200 ejemplos con otros 200 ejemplos de prueba. 102
Tabla 5.3 Generadores de flujos de datos para el experimento de métodos de discretización aplicados a IADEM-2c. 103
Tabla 5.4 Promedio de los resultados finales sobre todos los flujos de datos para manipular atributos numéricos. Para cada flujo de datos hubieron 1 000 000 ejemplos de entrenamiento, cada 200 ejemplos de entrenamiento se evalúan los clasificadores con 200 ejemplos de prueba (*holdout*). 104
Tabla 5.5 Precisión, número de nodos y tiempo de procesamiento de IADEM-2c sobre varios conjuntos de datos reales. 107
Tabla 6.1 Principales características de los generadores de flujos de datos usados en el estudio experimental con datos sintéticos. 119
Tabla 6.2 Resultados promediados del rendimiento de los algoritmos sobre todos los flujos de datos sintéticos en conceptos estables. Los algoritmos fueron entrenados con 1 000 000 de ejemplos. 120
Tabla 6.3 Medidas de rendimiento de los algoritmos sobre cambios abruptos. Los cambios ocurren cada 30 000 ejemplos de entrenamiento. Fueron 30 cambios de concepto. 122
Tabla 6.4 Medidas de rendimiento de los algoritmos sobre cambios graduales. Los cambios graduales ocurren cada 30 000 ejemplos de entrenamiento con 3 000 ejemplos en el período de transición entre conceptos consecutivos. Fueron 30 cambios de concepto. 123
Tabla 6.5 Medidas de rendimiento de los algoritmos en conjuntos de datos del mundo real. 126

LISTA DE ALGORITMOS

1 HDDM: Método para la detección de cambio basado en la cota de Hoeffding. 68

2 Algoritmo base para la inducción en la familia IADEM. 93

3 Algoritmo para determinar la cantidad máxima de intervalos para el método de discretización. 100

4 Algoritmo para la manipulación de cambio de concepto aplicable en algoritmos de inducción de árboles de decisión. 116

Parte I

INTRODUCCIÓN Y MARCO DE TRABAJO

1

INTRODUCCIÓN

En la actualidad muchos sistemas generan continuamente grandes cantidades de datos. Nuestra vida cotidiana presenta diversos escenarios de este tipo. Internet, telefonía móvil y sensores son algunos ejemplos de estas fuentes de información [GR07]. Debido a la enorme cantidad de datos y a su obtención continua, no es posible para los humanos analizarlos ni utilizar sistemas tradicionales de procesamiento por lotes, donde todos los datos deben ser recolectados antes de su ejecución y no es necesaria una respuesta en línea. La extracción de conocimiento a partir de grandes *flujos de datos* es una de las tareas más retadoras en el área de la minería de datos y el aprendizaje automático.

Nuestro mundo es dinámico y los cambios son parte de la vida diaria. Cuando existe un cambio, por ejemplo debido a condiciones meteorológicas o actividades de adversarios, los modelos de predicción o clasificación necesitan ser capaces de adaptarse a ellos. Este es uno de los problemas fundamentales a resolver cuando se aprende a partir de datos no estacionarios recolectados en el tiempo, y es conocido como *cambio de concepto*. El aprendizaje incremental supervisado con cambio de concepto constituye el área de investigación en la cual se enmarca esta tesis. El cambio de concepto está presente en muchos problemas reales de gran auge, como son el mercado, rastro Web, detección de fraude, filtrado de información, juegos, redes de sensores, datos espaciales, biométrica y muchos otros [Žli09b, GŽB+]; de ahí su importancia y la amplia gama de investigaciones en el área.

1.1 MINERÍA DE DATOS Y APRENDIZAJE AUTOMÁTICO

La *extracción de conocimiento* es el proceso por el cual se identifican de forma no trivial patrones válidos, novedosos, potencialmente útiles y comprensibles que se encuentren en los datos [del07]. Su nombre en inglés es *Knowledge Discovery in Databases* (KDD).

Un concepto relacionado con el de extracción de conocimiento es el de *minería de datos*, que forma parte de la extracción de conocimiento (Figura 1.1) [del07], aunque algunos autores se refieren a ambos términos como el mismo concepto [Bif09]. La minería de datos es el proceso de descubrimiento de patrones a partir de los datos [Ye03]. Este proceso debe ser lo más automático posible debido a que el volumen de datos usado en él es potencialmente elevado y se suele aplicar en este un análisis inteligente de datos [HRF04]. Los datos pueden ser generados y recolectados de diversas fuentes: datos científicos, financieros, de marketing, médicos, demográficos, etc. En la actualidad también se generan grandes cantidades de datos provenientes de las tecnologías de la información y las comunicaciones: enrutadores de Internet, sensores, agentes y servidores Web son algunos ejemplos.

La minería de datos es un tópico complejo y está relacionado con múltiples disciplinas como la estadística, la recuperación de información, el reconocimiento de patrones y el aprendizaje automático.

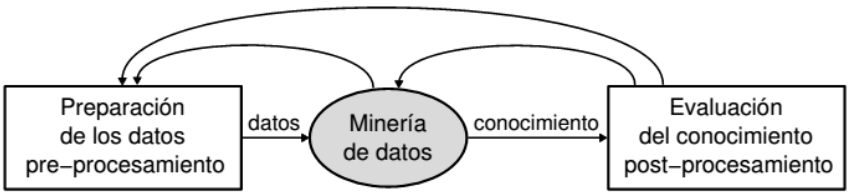


Figura 1.1: Esquema general de las etapas del proceso de extracción de conocimiento.

El *aprendizaje automático* es la disciplina encargada del estudio de los métodos usados para programar los ordenadores de forma que aprendan [HRF04], y es el objeto de estudio de la presente investigación. Desde el contexto de la minería de datos, el aprendizaje nos permite identificar regularidades en un conjunto de experiencias [HRF04] que pueden expresarse en forma de patrones (o modelos) que describan diversos conocimientos que subyacen en los datos. El uso de los modelos para describir el conocimiento puede interpretarse como una forma de compresión y puede ser utilizado con dicha finalidad, puesto que se consigue representar un conjunto de datos empleando mucho menos espacio [Yeo3].

Una consideración importante a tener en cuenta cuando se habla de aprendizaje automático es la complejidad temporal y espacial de los propios algoritmos. Así, se deberá extraer conceptos en un número de pasos razonable [Val84] y la calidad del modelo que se infiera dependerá, en gran medida, de los recursos disponibles. En este sentido surge la teoría del aprendizaje automático [Val84].

Las técnicas utilizadas en el aprendizaje automático se pueden dividir en dos grandes grupos: descriptivas y predictivas. Las primeras están en correspondencia con las tareas de detección de agrupamientos y correlaciones, y las segundas con las tareas de clasificación y regresión. Algunas de estas técnicas están enfocadas a resolver una tarea más que otra (por ejemplo, las que generan reglas de asociación son más descriptivas y las que entrenan redes neuronales son más predictivas), sin embargo, la separación no siempre es tan clara. Por ejemplo, los algoritmos que inducen árboles de decisión [BFOS84, Qui86] pueden ser fácilmente comprensibles y también tienen capacidad para ser utilizado como un sistema de predicción.

1.2 MINERÍA DE FLUJOS DE DATOS

Cada vez son más las situaciones en las que se dispone de un gran volumen de información del que intentar extraer conocimiento: medidas de rendimiento en monitorización de redes y gestión de tráfico, registros en seguimiento Web y personalización, procesos productivos, datos de sensores, registros de llamadas en telecomunicaciones, mensajes electrónicos, etc.

Uno de los principales retos de la minería de flujos de datos es la limitación de recursos computacionales (tiempo y memoria) para procesar de forma factible estos grandes conjuntos de datos. Cuando el conjunto de datos crece constantemente y de forma indefinida, no todos los datos pueden ser almacenados en memoria. En estas situaciones no se cumplen las condiciones para usar un enfoque clásico para extraer conocimiento (como el que realizan los algoritmos ID3 [Qui86] o C4.5 [Qui93]). Una de las limitaciones principales es disponer de todos los datos antes

de iniciar la inducción de los modelos. De esta forma, es común considerar las siguientes restricciones [Bif09]:

1. La cantidad de datos a procesar es potencialmente infinita. De esta forma, no es posible almacenarlos completamente. Solo una porción de estos datos puede ser procesada y almacenada, mientras que el resto debe ser descartada. Incluso, de ser posible almacenar toda la información, no sería factible procesarla en su totalidad.

2. La velocidad de llegada de estos datos es alta, por lo que deben ser procesados en tiempo real y luego descartarse.

3. La función de distribución de probabilidad que genera estos datos puede variar en el tiempo. Así, los datos pasados pueden convertirse irrelevantes o adversos con respecto a los datos actuales (cambio de concepto).

Las dos primeras restricciones limitan la cantidad de memoria y el tiempo de procesamiento con el que puede disponer un algoritmo para el procesamiento de un flujo de datos. El aprendizaje incremental es capaz de incorporar la información que aporten nuevas experiencias (que antes no estaban disponibles en el conjunto de datos) al modelo que se está induciendo y capaz de hacerlo evolucionar para que cada vez represente conceptos más complejos [Qui93]. Aunque el proceso de aprendizaje no haya concluido, como este usa el modelo anterior, siempre habrá un modelo disponible que podrá ser usado para clasificar o para visualizar el conocimiento extraído hasta ese momento.

La tercera restricción, estudiada intensamente en los últimos años, se refiere a la necesidad de adaptarse a posibles cambios en la distribución de los datos de entrada, que pueden afectar el rendimiento del modelo de aprendizaje. Por ejemplo, estos pueden estar dados debido a cambios en las preferencias de usuarios (noticias, compras, etc.). Por su importancia dentro del aprendizaje incremental y la complejidad de manipular tales cambios, a continuación describiremos sus principales características.

1.3 CAMBIO DE CONCEPTO

Debido a la dimensión temporal de los flujos de datos, es común observar en problemas reales que la distribución subyacente de estos cambie en el tiempo. Estos cambios con frecuencia impactan significativamente en el rendimiento del modelo de aprendizaje, a tal punto que pueden dejar de ser válidos. En general, los algoritmos de aprendizaje incremental deben incorporar mecanismos adicionales para manipular cambio de concepto. El algoritmo de aprendizaje debe ser capaz de capturar y mantenerse actualizado con respecto a los posibles patrones transitorios subyacentes en el flujo de datos [WFYH03]. Así, el modelo de aprendizaje debe ser constantemente revisado, aprendiendo de nueva información y olvidando experiencias que representan conceptos pasados.

Existen múltiples estrategias para manipular cambio de concepto. Estas pueden ser utilizadas por diferentes métodos y algoritmos para aprender incluso cuando los conceptos cambian. De forma general, un sistema que manipule cambio de concepto debe ser capaz de adaptarse rápidamente a los cambios, ser robusto en

la distinción entre un verdadero cambio de concepto y ruido, así como reconocer y tratar conceptos recurrentes. La distinción entre un verdadero cambio de concepto y ruido es un problema de gran dificultad. Algunos algoritmos pueden ser muy susceptibles al ruido, interpretándolo erróneamente como un cambio de concepto, mientras que otros pueden ser muy robustos al ruido pero adaptarse muy lentamente al cambio [WK96]. Para adaptarse más rápidamente a los cambios recurrentes, los modelos de aprendizaje pueden ser almacenados para posteriormente ser reexaminados y usados [HSH98, WK93, Sal97].

Existen otros tipos usuales de cambio. Los más comunes que se consideran en el diseño y evaluación de los algoritmos de aprendizaje están relacionados con la similitud entre los conceptos consecutivos y la duración del cambio.

Una estrategia ampliamente usada para la manipulación de cambio de concepto es monitorizar constantemente alguna medida de rendimiento del modelo de aprendizaje o de partes del modelo. Como consecuencia se obtienen valores de rendimiento (e.g. números reales) a lo largo del tiempo. Un deterioro de esta medida de rendimiento es interpretada como un cambio de concepto y se definen algunas acciones para actualizar el modelo o parte del mismo en respuesta al cambio.

Esta estrategia ha permitido el uso de varias herramientas estadísticas, dándole garantías teóricas a los métodos usados para dicha monitorización y detección de cambios [GMCR04, BdF+06, IGD11]. Sin embargo, la mayoría de estas herramientas para la detección de cambios en línea asumen una distribución de probabilidades conocida [BN93, Mon01], mientras que muchos datos reales no están regulados por estas distribuciones. En respuesta a esto, han surgido algunos acercamientos con garantías probabilísticas de desempeño pero con un coste computacional alto, y otros con algunos componentes heurísticos.

Muchos algoritmos de aprendizaje han sido usados como modelo base para la manipulación de cambio de concepto. Entre estos se encuentran sistemas basados en reglas [SF86, WK93, WK96], Naïve Bayes [BM10, BG07, BM08], máquinas de soporte vectorial [KJ00, Kli04], redes neuronales artificiales [KW95], aprendizaje basado en casos [CNDH03, FH89] y árboles de decisión [HSH98, HSD01, Sta03].

1.4 INDUCCIÓN INCREMENTAL DE ÁRBOLES DE DECISIÓN

La inducción de árboles de decisión ha sido usada satisfactoriamente en muchas situaciones del mundo real, su efectividad ha sido comparada con otros métodos de exploración automática de datos y con expertos humanos [Mur98]. En la literatura se han señalado varias ventajas de los métodos de clasificación basados en árboles de decisión [Mur98, GMR04, del07]: modelo altamente interpretable, método no paramétrico, construcción rápida y relativamente simple, precisión adecuada, aplicable a problemas con muchas dimensiones y con atributos nominales y numéricos, entre otras.

El procedimiento básico para inducir un árbol de decisión es expandir recursivamente los nodos, hasta que ninguno sea expansible. Surgen así dos aspectos fundamentales a la hora de definir cualquier algoritmo de inducción descendente de árboles de decisión [del07]: (1) el problema de la elección del atributo para expandir y (2) la determinación de qué condición debe cumplirse para detener la expansión por una rama concreta. Sin embargo, el punto clave en la construcción

de árboles de decisión dentro del aprendizaje incremental ha estado enfocado en la primera de estas dos cuestiones [DH00, del07, NFM07, RPDJ13, RJPD14].

En la mayoría de los algoritmos, la elección está basada en alguna medida heurística que permita obtener un árbol cercano al óptimo. Ejemplos de algunas de las medidas más usadas son la ganancia de información [Qui86], la proporción de la ganancia de información [Qui93] y el índice Gini [BFOS84]. En el aprendizaje incremental surge el problema adicional de estimar cuál es el valor real de la medida heurística, dado que el conjunto de datos de entrenamiento es posiblemente infinito y en el momento de aplicar tal medida heurística solo se cuenta con una muestra de ese conjunto.

En este nuevo problema, una herramienta poderosa ha sido la estimación por intervalos. Específicamente, en la literatura se han utilizado las desigualdades de probabilidad de Hoeffding [Hoe63], Chernoff [Che52] y McDiarmid [McD89] sin asumir alguna función de distribución de probabilidad subyacente en los datos. También se ha asumido aproximación a la curva normal para tal estimación [RJPD14].

Existen también estudios que han estado enfocados a otros aspectos de la inducción de árboles de decisión, como la aparición de ruido en los conjuntos de datos, la presencia de atributos con valores desconocidos, o incluso a la mejora de la propia precisión [del07, Kir07]. Como señalamos anteriormente, también es necesario considerar el problema de la manipulación de cambio de concepto [HSD01, NFM07, BHKP10].

Las características de los modelos basados en árboles de decisión permiten contemplar de forma explícita varios tipos de cambio importantes, como son los cambios graduales (relacionados con la duración del cambio [HSD01]) y locales (relacionados con la similitud entre conceptos consecutivos [IGD11]). Estos han demostrado ser una herramienta efectiva en el aprendizaje incremental con cambio de concepto. Una estrategia eficaz en la manipulación de cambio de concepto en estos modelos es monitorizar el rendimiento (e.g. precisión) de subárboles constantemente y reconstruir aquellos en los que se detecte un deterioro significativo del rendimiento.

1.5 DISEÑO DE LA INVESTIGACIÓN, ESTRUCTURA Y APORTES

En esta tesis nos enfocamos en el aprendizaje en flujos de datos no estacionarios. Particularmente, extendemos a la familia de algoritmos de *Inducción de Árboles de Decisión por Muestreo* (IADEM) [del07, dRGM08, RM00, Ram01, RMd04, RdCM05] para el aprendizaje incremental con cambio de concepto. Adicionalmente, abordaremos otras de las problemáticas relacionadas con la inducción de árboles de decisión, como es el problema de la elección del mejor atributo para expandir, la manipulación de atributos numéricos y el uso de técnicas para mejorar la precisión del modelo inducido.

La presente investigación está dirigida a resolver el *problema científico* de cómo avanzar en los métodos para el aprendizaje supervisado adaptativo en línea. Por tal motivo planteamos las siguientes *preguntas de investigación*:

PREGUNTA 1 ¿Cuáles son los requerimientos computacionales, indicadores de rendimiento, y tipos de cambio comunes a considerar en el aprendizaje a partir de flujos de datos no estacionarios?

PREGUNTA 2 ¿Cómo adaptar, independientemente del modelo de predicción o clasificación, el aprendizaje en flujos de datos no estacionarios?

PREGUNTA 3 ¿Si se consideran las características específicas de los modelos de aprendizaje basados en árboles de decisión, es posible adaptarse a cambios distribucionales de forma más eficaz?

PREGUNTA 4 ¿Cómo extender el algoritmo IADEM-2 al aprendizaje a partir de flujos de datos no estacionarios?

PREGUNTA 5 ¿Cómo evaluar los algoritmos desarrollados, teniendo en cuenta que los métodos clásicos de evaluación no son adecuados en el aprendizaje incremental con cambio de concepto?

El *objetivo general* de la presente investigación es aumentar la precisión de los modelos de clasificación o predicción bajo cambio de concepto, considerando las restricciones computacionales comunes en estos escenarios y los tipos de cambios más frecuentes.

Para alcanzar este objetivo seguimos metodologías de la investigación teóricas y empíricas. En este sentido analizamos trabajos científicos relacionados, formulamos y discutimos los conceptos teóricos y empíricos de la investigación, y realizamos evaluaciones empíricas a los métodos propuestos. En el análisis de las investigaciones relacionadas, identificamos las desventajas o necesidades de los acercamientos existentes así como las estrategias y métodos a seguir para enfrentar estas deficiencias. Las soluciones propuestas son validadas experimentalmente usando metodologías apropiadas de evaluación y usando un amplio rango de datos relevantes, comparando además los métodos propuestos con los acercamientos relacionados más conocidos.

Así, planteamos las siguientes *tareas de investigación*:

- Análisis de los métodos existentes para la detección de cambios distribucionales en los datos aplicables al aprendizaje en línea, así como de los métodos para la adaptación a dichos cambios centrándose en el aprendizaje mediante árboles de decisión.
- Identificación de las metodologías de evaluación de algoritmos adecuadas para el aprendizaje en línea con cambio de concepto, así como de las métricas para evaluar el rendimiento de detectores de cambio de concepto y de los modelos basados en árboles de decisión.
- Desarrollo de nuevos métodos para la detección de cambios en línea teniendo en cuenta las limitaciones de los métodos existentes y los indicadores de rendimiento más comunes a considerar en su diseño y evaluación.
- Mejora de la inducción de modelos de árboles de decisión a través del algoritmo IADEM-2 [delo7], así como su extensión para el aprendizaje a partir de flujos de datos con cambio de concepto.

- Evaluación de los métodos y algoritmos desarrollados considerando conjuntos de datos artificiales y reales, teniendo en cuenta metodologías y métricas adecuadas para el aprendizaje en flujos de datos no estacionarios.

A continuación describimos el contenido de los diferentes capítulos e indicamos los aportes fundamentales que proponemos:

- Capítulo 2. En este capítulo introducimos los principales conceptos y tipos de cambio considerados en el aprendizaje incremental con cambio de concepto. Definimos un marco de trabajo para el aprendizaje en flujos de datos no estacionarios teniendo en cuenta conceptos y requerimientos generales. Este marco de trabajo es seguido por los nuevos métodos desarrollados, descritos en capítulos posteriores.

- Capítulo 3. Realizamos un estado del arte de las estrategias, métodos y algoritmos que se han propuesto para la manipulación de cambio de concepto. Se tienen en cuenta métodos enfocados a resolver problemas similares no solo relativos al área del aprendizaje automático, evidenciando que algunos de estos aún no han sido explorados en el aprendizaje incremental supervisado. Para la manipulación de cambio de concepto, resumimos métodos y algoritmos relevantes usados por los modelos de aprendizaje más comunes como son árboles de decisión, reglas de decisión, máquinas de soporte vectorial, entre otros.

  También realizamos una revisión de las metodologías, herramientas de software y conjuntos de datos utilizados para la evaluación y comparación de algoritmos incrementales. Adicionalmente, identificamos ventajas y desventajas de métodos y algoritmos existentes, las limitaciones fundamentales existentes en el área, así como las principales tendencias actuales.

- Capítulo 4. Presentamos nuevos métodos para la manipulación de cambio de concepto independientemente del algoritmo de aprendizaje. Así, una ventaja de este tipo de acercamiento es que puede ser aplicado directamente a cualquier algoritmo de aprendizaje incremental para manipular cambio de concepto.

  La idea fundamental es monitorizar alguna medida de rendimiento de interés del modelo de aprendizaje a lo largo del tiempo (por ejemplo, la precisión en la clasificación o predicción), para disparar señales de cambio cuando se estima que esta medida ha decaído significativamente. Consideramos medidas de rendimiento que se corresponden con variables aleatorias independientes, acotadas y que toman valores en el conjunto de los números reales.

  Proveemos a los métodos con garantías probabilísticas de desempeño en términos de cotas para la tasa de falsos positivos y falsos negativos en la detección de cambios distribucionales. Estos usan como estimadores a los promedios (HDDM$_{A\text{-test}}$) y promedios ponderados (HDDM$_{W\text{-test}}$, particularmente estudiamos el estadístico EWMA [BN93, Mono1]) para la detección en línea de cambios significativos en la media poblacional, considerando tipos de cambio abruptos y graduales. Además, estos métodos tienen complejidad computacional constante en tiempo y espacio, no asumen alguna restricción

relacionada con la función de distribución de probabilidades que genera estos valores reales, y solo reciben como parámetros los niveles de confianza requeridos para la detección de tales cambios; características relevantes en el aprendizaje a partir de flujos de datos no estacionarios.

Realizamos evaluaciones empíricas a través de conjuntos de datos ampliamente usados en el área combinando los métodos propuestos con el clasificador Naïve Bayes y el Perceptrón simple, con el objetivo de comparar el rendimiento de los algoritmos propuestos con respecto a otros algoritmos relevantes existentes.

- Capítulo 5. Mejoramos al algoritmo IADEM-2 en la convergencia del modelo y en la manipulación de atributos numéricos. Llamamos al algoritmo resultante IADEM-2c.

  Para mejorar la convergencia del modelo y así aumentar la precisión en la predicción en etapas más tempranas de la inducción, consideramos tipos adicionales de pruebas a instalar en los nodos de decisión. En este sentido comparamos las pruebas originales del algoritmo IADEM-2, pruebas que dan lugar a árboles binarios, así como pruebas mixtas que contemplan las dos anteriores. Lo que resulta en una mejora significativa de la precisión.

  Por otro lado, aplicamos a IADEM-2 y comparamos empíricamente algunos de los métodos de discretización más relevantes utilizados en el aprendizaje incremental. También proponemos una extensión a estos métodos al aplicarlos a la inducción incremental de árboles de decisión, con el objetivo de disminuir el coste temporal de los mismos. En algunos de estos métodos, esta extensión no solo disminuye el coste computacional, sino que contribuye a la inducción de árboles más pequeños y al aumento de la precisión.

- Capítulo 6. Le incorporamos a IADEM-2c, el algoritmo desarrollado en el capítulo anterior, mecanismos para la manipulación de cambio de concepto. Al algoritmo resultante de estas extensiones lo llamamos IADEM-3.

  Si bien los detectores de cambio propuestos en el Capítulo 4 pueden ser aplicados directamente a IADEM-2c para dicha manipulación, estudiamos mecanismos adicionales que explotan características específicas de los modelos de árboles de decisión, para de esta forma adaptarse al cambio con más eficacia. Estos mecanismos usan a los detectores de cambio propuestos para monitorizar el desempeño de partes del modelo, y reconstruir estas partes cuando se estime un deterioro significativo. Para ello discutimos los principales aspectos del mecanismo propuesto, tenemos en cuenta diferentes tipos de cambio, así como ventajas y desventajas con respecto a acercamientos anteriores.

  Específicamente, se crea un subárbol alternativo con raíz en un nodo de decisión cuando se estima que el respectivo subárbol principal (con raíz en ese nodo de decisión) ha aumentado significativamente su tasa de error. Un árbol alternativo reemplaza al principal cuando se estima que este intercambio aumenta significativamente la precisión del modelo.

  Al considerar subárboles alternativos, el modelo de aprendizaje resultante también puede ser visto como un árbol de opciones [PHK07, IGZD11], ya que

una determinada experiencia puede tomar varios caminos y así varias predicciones teniendo en cuenta el árbol principal y los alternativos. En IADEM-3 estas predicciones son combinadas para proveer una más exacta.

Finalmente, comparamos al algoritmo resultante con otros relevantes en el área.

2

PRELIMINARES

Los flujos de datos constituyen una fuente importante de conocimiento [GR07]. Algunos de estos flujos pueden ser almacenados como un conjunto de datos completo, donde todos los datos son salvados, mientras que otros no pueden ser almacenados y solo se extrae la información más relevante para ser usada posteriormente. En cualquier caso, es usual tener datos que crecen incrementalmente incorporándoles nuevos datos. Los procesos incrementales, caracterizados por refinamientos sucesivos o modificaciones hechas a un producto existente, son aplicados a muchas áreas, y el aprendizaje automático también puede tomar ventajas de sus propiedades. En nuestro contexto, el acercamiento incremental usado para el aprendizaje se convierte en una herramienta poderosa, especialmente para tratar con problemas presentados cuando se aprende de flujos de datos [Žli09b, GŽB+].

Uno de los problemas fundamentales a resolver cuando se aprende de datos no estacionarios, colectados en el tiempo o desde diferentes fuentes, ya sea un conjunto de datos completos o un flujo de datos sin límite, es el cambio de concepto. Por ejemplo, tales cambios pueden ocurrir debido a cambios en las preferencias de usuarios (debido a un cambio de tiempo), preferencias de noticias, gasto de energía, etc. El filtrado de correo spam es otro ejemplo común: los creadores de estos correos tratan de eludir a los filtros que distinguen sus correos, los filtros de correo spam deben ser actualizados para identificarlos satisfactoriamente en el tiempo [DC04, DCTC05].

Existen muchas razones que pueden causar la presencia de estos cambios en los conceptos objetivos. Por ejemplo, la función objetivo puede depender de características (ocultas) que no están presentes en los atributos de los datos de entrada [HSD01], o la función objetivo puede depender del contexto (e.g. condiciones ambientales, ubicaciones geográficas, etc.) que pueden cambiar la distribución de probabilidades de los atributos en el conjunto de entrenamiento [WK93].

En este capítulo formalizamos el problema del aprendizaje con cambio de concepto, definimos los principales conceptos y discutimos sus peculiaridades fundamentales.

2.1 DEFINICIONES Y TIPOS DE CAMBIO

En los últimos años muchos trabajos han investigado el problema del cambio de concepto. Los acercamientos teóricos han considerado variedad de suposiciones relacionadas con las características del cambio, donde comúnmente son impuestas más restricciones teóricas para los flujos de datos, pero con fuertes garantías matemáticas de desempeño. Otros acercamientos consideran menos restricciones teóricas e incorporan más componentes heurísticos a los algoritmos de detección y adaptación al cambio. El cambio de concepto está presente en el aprendizaje supervisado, no supervisado y semi-supervisado y ha sido abordado desde diferentes puntos de vista, como estadísticos, de procesamiento de señales o desde el área

del aprendizaje automático, tratando de dar soluciones con diferentes grados de garantías matemáticas.

Como es de suponer, existe gran variedad de definiciones de cambio de concepto. Algunos trabajos investigativos han tratado así de unificar notaciones, definiciones y tipos comunes de cambio [MRA+12]. En esta sección mostramos las definiciones más comunes. Adicionalmente, algunos acercamientos teóricos y estadísticos asumen modelos alternativos (Sección 2.2). Sin embargo, la mayoría de las investigaciones concuerdan en relación a los diferentes tipos de cambios mencionados a continuación.

2.1.1 *Definiciones*

Dentro del aprendizaje en línea o incremental [WK96], el problema de clasificación es generalmente definido para una secuencia (posiblemente infinita) de ejemplos (también conocidos por instancias o experiencias) $S = (\vec{a}_1, c_1); (\vec{a}_2, c_2); \ldots; (\vec{a}_i, c_i); \ldots$ que se obtienen en el tiempo, normalmente uno a la vez. Cada ejemplo de entrenamiento $(\vec{a}_i, c_i)$ es formado por un vector $\vec{a}_i$ y un valor discreto c_i, llamado etiqueta y tomado de un conjunto finito $\mathcal{C}$ llamado clase. Cada vector $\vec{a}_i \in \overrightarrow{\mathcal{A}}$ tiene la misma dimensión, cada dimensión es llamada atributo y cada componente a_{ij} es un valor de atributo (numérico o nominal). Es supuesto que existe una función $c_i = f(\vec{a}_i)$ y el objetivo es obtener un modelo a partir de S que aproxime a f como $\hat{f}$ para clasificar o predecir la etiqueta de ejemplos no etiquetados (también conocidos como observaciones), tal que $\hat{f}$ maximice la precisión en la predicción [FAR05]. Algunas veces también se asume que los ejemplos se obtienen en lotes de igual tamaño.

Consideraremos *concepto* como el término que se refiere a la función de distribución de probabilidades completa del problema en un punto determinado en el tiempo [MWY10]. Este concepto puede ser caracterizado por la distribución conjunta $P(\overrightarrow{\mathcal{A}}, \mathcal{C})$. De esta forma, un cambio en la función de distribución de probabilidades del problema (también conocido como contexto [GMCR04]) trae consigo un cambio de concepto.

Así, suponiendo que los ejemplos están sujetos a la función objetivo $c_i = f(\vec{a}_i)$ con la función de distribución de probabilidades $P(\overrightarrow{\mathcal{A}}, \mathcal{C})$, usualmente son considerados tres tipos fundamentales de cambios: cambios en el muestreo [WK93, Sal97] donde solo cambia $P(\overrightarrow{\mathcal{A}}, \mathcal{C})$; cambio condicional [GFH07] donde solamente $P(\mathcal{C} | \overrightarrow{\mathcal{A}})$ (o la función objetivo $c_i = f(\vec{a}_i)$) podría cambiar; y cambio real [FHWY04] donde ambas podrían cambiar (también conocido como cambio dual [GFH07]).

Muchos autores consideran el cambio de muestreo como un *cambio virtual* [FAR05, WK93, Sal97] mientras que el cambio es real cuando este induce un movimiento en la función objetivo. Sin embargo, en la práctica estas categorías de cambio generalmente no son relevantes ya que todas impactan de forma similar en el modelo de aprendizaje. Como los errores producidos por el modelo se incrementan con respecto a los ejemplos actuales, es necesario revisar constantemente el modelo.

Con esta idea, algunos autores definen cambio de concepto más formalmente con el objetivo de establecer un método (o una familia de estos) para manipularlo. Por ejemplo, Basseville y Nikiforov [BN93] discuten el problema de detección de cambio desde un punto de vista de la matemática estadística. Ellos definen

$(Y_k)_{1 \leq k \leq n}$ como una secuencia de variables aleatorias con función de probabilidad de densidades condicional $P_\theta(Y_k|y_{k-1}, \ldots, y_1)$. Antes del punto de cambio desconocido t_0, el parámetro θ es constante e igual a θ_0. Después del cambio, el parámetro es igual a θ_1. El problema es detectar la ocurrencia de un cambio tan pronto como sea posible con una tasa fija de falsas detecciones antes de t_0. Un acercamiento similar es propuesto por Dries and Rückert [DR09]. Ellos pretenden determinar si los últimos n_1 ejemplos han sido muestreados por una distribución diferente que los n_2 precedentes. Como hemos señalado, otros trabajos teóricos han asumido modelos adicionales.

2.1.2 *Tipos*

Existen múltiples categorizaciones para determinar la naturaleza de un cambio de concepto. Las propuestas que presentamos usan diferentes criterios y tratan de ser lo más extensivas posible, considerando una amplia variedad de puntos de vistas presentados en trabajos investigativos previos. Es claro que en ocasiones ideas similares han tomado nombres distintos, por lo que algunos términos han estado relacionados o incluso se refieren a la misma idea. Por simplicidad, consideraremos que un cambio de concepto tiene asociado dos conceptos diferentes: concepto inicial (C_I) y concepto final (C_F). No obstante, las definiciones que se muestran a continuación pueden ser aplicables también a situaciones donde están presentes más de dos conceptos.

Una de las categorizaciones más importantes se refiere a la *extensión del cambio* (también llamada *tasa de cambio*). Dependiendo de cuán diferentes sean los conceptos C_I y C_F, podemos decir que el cambio es *total* o *parcial*. Un cambio total implica que las distribuciones de C_I y C_F son totalmente distintas. Este tipo de cambio usualmente es simulado con flujos de datos sintéticos porque es difícil de observar este tipo de cambio en flujos de datos reales. De esta forma, cuando se aprende de flujos de datos reales, el cambio frecuentemente es parcial con diferentes grados de semejanza. La extensión del cambio es un punto importante diferenciando problemas y algoritmos. Muchas investigaciones han estudiado por separado este tipo de cambio [KPR90, HL91, HL94, SK07].

Teniendo en cuenta la velocidad del cambio, dígase el tiempo usado para cambiar desde C_I hasta C_F (t_{cambio}), podemos definir *cambio abrupto* ($t_{cambio} = 0$) y *cambio gradual* ($t_{cambio} > 0$). Cuando el cambio es gradual podemos considerar dos tipos adicionales de cambio. Si el cambio es hecho de una forma progresiva, por medio de conceptos intermedios que son diferentes levemente (y diferentes de C_I y C_F), tenemos un cambio gradual *continuo*. Por otro lado, si ambos conceptos existen (C_I y C_F) y su presencia alterna, variando su frecuencia durante la transición (con mayor presencia de C_I al inicio y de C_F al final), tenemos un cambio *discreto*. La velocidad del cambio puede ser descrita independientemente del tipo de cambio gradual y pueden ser usados diferentes métodos atendiendo a dicha velocidad [GMCR04, BdF+06].

Otro aspecto considerado en el desarrollo de algunos algoritmos, aunque no tan relevante como los anteriormente planteados, es los conceptos recurrentes [WK93, RB07, GMS11]. Cuando el nuevo concepto que se presenta al final del cambio actual (C_I) ha sido visto previamente, tenemos un cambio *recurrente*. De otra forma,

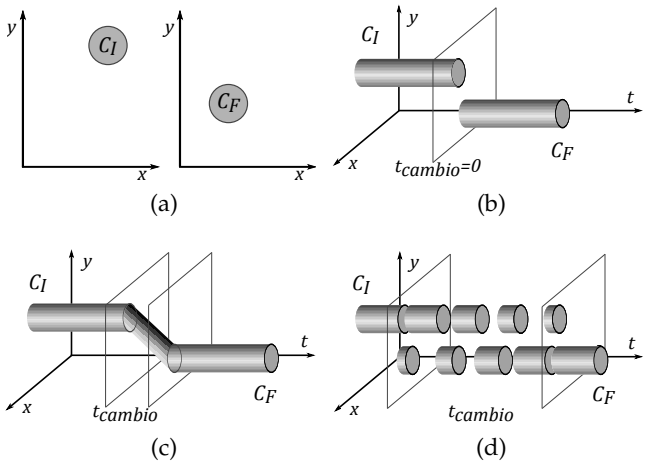


Figura 2.1: (a) Diferentes problemas en un problema bi-dimensional. Diferentes escenarios para cambio de concepto: (b) cambio abrupto, (c) cambio gradual continuo y (d) cambio gradual discreto.

el cambio es *no recurrente*. Cuando hay algún patrón temporal en la recurrencia, podemos hablar de cambio *cíclico*. De lo contrario, si no existe tal patrón el cambio es *acíclico*.

Para resumir las categorizaciones más importantes, la Figura 2.1 presenta algunos ejemplos. En ella, C_I y C_F representan conceptos diferentes en un problema con dos dimensiones (ver Figura 2.1a), y se añade un tercer eje para representar el tiempo (t). Dicha Figura muestra ejemplos de diferentes escenarios: cambio abrupto (Figura 2.1b), cambio gradual continuo (Figura 2.1c) y cambio gradual discreto (Figura 2.1d).

Los tipos de cambio considerados en este apartado tratan de ser los más extensivos posible pero, como anteriormente planteamos, en la literatura diferentes nombres se han usado para referirse a definiciones semejantes. Por ejemplo, Stanley [Sta03] reconoce dos tipos de cambio relacionados con la frecuencia con la cual se reciben los ejemplos que describen el nuevo concepto objetivo: repentino y gradual. Stanley también divide el cambio gradual en moderado y lento. Žliobaitė [Žli09b] propone otros términos equivalentes, cuyo *cambio incremental* corresponde a nuestro cambio discreto gradual, y su *cambio gradual* corresponde a nuestro cambio gradual continuo. Kelly y otros [KHA99] también proponen tipos similares de cambio y estudian empíricamente el efecto de tales tipos de cambio en el rendimiento de algunos clasificadores.

2.2 ACERCAMIENTOS TEÓRICOS

Semejante a varios estudios en el campo de la teoría del aprendizaje computacional, a algunos algoritmos de aprendizaje se les han establecido garantías rigurosas de rendimiento asumiendo que este se ejecutan bajo una estructura de cambio específica. En este sentido, varias investigaciones se han enfocado en definir estructuras de cambio más reales, en estudiar diferentes algoritmos de aprendizaje

y en establecer intervalos de confianza más ajustados para algunas medidas de rendimiento.

Por ejemplo, Kuh y otros [KPR90] consideran una secuencia de conceptos c_i y la tarea de estimar un concepto c_t al tiempo t por un algoritmo de aprendizaje propuesto. El algoritmo produce la hipótesis actual $\hat{c}_t$ usando los $w(\epsilon, \delta)$ ejemplos más recientes (una ventana deslizante). Para t suficientemente grande, con t' seleccionado uniforme y aleatoriamente entre 1 y t, en esencia asumen una estructura del cambio por medio de $P\left(d\left(c_{t'}, \hat{c}_{t'}\right) \leq \epsilon\right) \geq 1 - \delta$, donde $\epsilon, \delta > 0$, y $d(c_{t'}, \hat{c}_{t'})$ es la probabilidad que $c_{t'}$ y $\hat{c}_{t'}$ difieran en un ejemplo aleatoriamente seleccionado. Ellos estiman un tamaño de ventana y la tasa máxima de cambio (cuán frecuente puede ocurrir un cambio) que es aceptable para el algoritmo de aprendizaje en dependencia de este tamaño de ventana fijo.

Probablemente, una estructura más práctica de cambio fue estudiada más tarde por Helmbold y Long [HL94]. Ellos asumen un posible cambio permanente y lento definiendo $P\left(f_t\left(X\right) \neq f_{t+1}\left(X\right)\right) \leq \Delta$, donde X representa los datos de entrada, $\Delta > 0$ una cota para la tasa de cambio y una secuencia de funciones objetivo $f_t(X)$. Sus resultados incluyen dos algoritmos de propósito general que aprenden a partir de una ventana deslizante de tamaño fijo $m = O\left(\left(d/\epsilon\right) \log 1/\epsilon\right)$ con una probabilidad ϵ de cometer un error, siendo d la dimensión de Vapnik-Chervonenkis. El primer algoritmo elige la hipótesis que minimiza las inconsistencias a partir de los ejemplos almacenados en la ventana. Este requiere un método que soporte eficientemente esta hipótesis si el concepto fuera estable. Ellos demuestran que este algoritmo tolera una tasa de cambio de $\Delta < c_1\epsilon^2 / \left(d \ln 1/\epsilon\right)$. El segundo algoritmo es una versión aleatoria del primero, ligeramente menos tolerante al cambio pero con menos coste computacional. Esta cota para la tasa de cambio permitida fue mejorada por Bartlett y Helmbold [BH96] a $\Omega\left(\epsilon^2 / \left(d + \ln 1/\epsilon\right)\right)$.

Otros resultados teóricos se han basado en dos modelos adicionales [Bar92]. En el primero (caso agnóstico) se asume que los ejemplos son generados independientemente y de forma aleatoria a partir de una secuencia de distribuciones conjuntas sobre $X \times \{0, 1\}$ con la restricción de que pares consecutivos de distribuciones tienen a lo sumo γ distancia de variación total ($0 < \gamma \leq 1$). En el segundo modelo (caso realizable) existe una función fija f que mapea cada $\overrightarrow{a}_t$ a c_t; nuevamente dos distribuciones consecutivas sobre el dominio puede cambiar a lo sumo en γ de distancia de variación total. Por ejemplo, Barve y Long [BL96] muestran una cota $\gamma = O\left(\epsilon^3 / \left(d \ln 1/\epsilon\right)\right)$ en el caso agnóstico y $\gamma = O\left(\epsilon^2 / \left(d \ln 1/\epsilon\right)\right)$ en el caso realizable. Más tarde, estas cotas fueron mejoradas por Long [Lon99] mostrando que $\gamma = O(\epsilon^3/d)$ es suficiente en el caso agnóstico y $\gamma = O(\epsilon^2/d)$ para el caso realizable.

Además, otros estudios han propuesto tipos de cambio de concepto más específicos, como los propuestos por Blum y Chalasani [BC92], Bartlett y otros [BBDK96], así como Freund y Mansour [FM97].

Blum y Chalasani [BC92] estudian una situación similar a la ofrecida por Helmbold y Long [HL91], en la cual un concepto puede cambiar en el tiempo, pero el "concepto activo" puede alternar solo en un número pequeño de conceptos. La propuesta de Blum y Chalasani [BC92] también permite que los cambios ocurran más rápido y sean más repentinos.

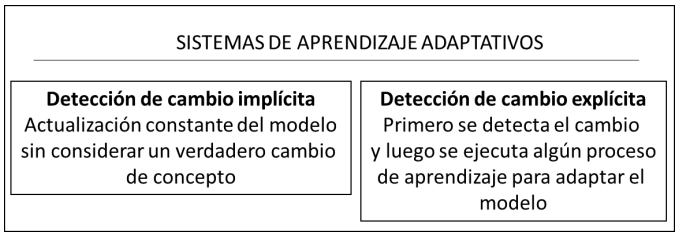


Figura 2.2: Categorización de las técnicas para manipular cambio de concepto atendiendo a su taxonomía.

Por otro lado, Barlett y otros [BBDK96] estiman una secuencia de funciones objetivo y ofrecen cotas para la tasa de cambio permitida en tres situaciones de cambio: cambios arbitrarios pero infrecuentes del concepto objetivo, cambios que corresponden a pasos lentos en un grafo cuyos nodos son funciones, y cambios a conceptos de tamaño pequeño medidos por las inconsistencia entre funciones objetivos consecutivas.

Finalmente, Freund y Mansour [FM97] asumen un modelo semejante al sugerido por Bartlett y otros [BBDK96] pero restringido al caso especial en que el cambio en los conceptos y en la distribución es una función lineal en el tiempo.

Aunque las garantías matemáticas de rendimiento para el algoritmo de aprendizaje son muy importantes, y la manipulación de cambio de concepto es imposible si no existen restricciones acerca de los tipos de cambio en cuestión [HL91, HL94], muchos datos reales presentan patrones que no concuerdan con alguno de estos modelos concretos. Adicionalmente, los tamaños de ventana que se han obtenido en algunos estudios teóricos son muy grandes, lo que los convierte en poco prácticos [WK96, KJ00].

2.3 DETECCIÓN Y APRENDIZAJE

En el área de la minería de datos y el aprendizaje automático, existen muchas estrategias para detectar cambio de concepto y estas pueden ser usadas por diferentes algoritmos para incorporar la habilidad de aprender incluso cuando los conceptos cambian. Varios investigadores han tratado de dar una estructura a estos algoritmos. A continuación mencionamos algunas de las más importantes.

Una posible categorización de las técnicas puede hacerse atendiendo a su taxonomía (Figura 2.2). Gama y otros [GMCR04] distinguen dos categorías: estrategias que adaptan el aprendizaje en intervalos regulares de tiempo sin considerar que ha ocurrido un cambio (detección de cambio implícita); y estrategias que primero detectan el cambio y luego ejecutan algún proceso de aprendizaje para adaptar el modelo (detección de cambio explícita). La segunda categoría regularmente trae consigo métodos más complejos y se distinguen dos principales técnicas: selección de instancias y pesado de instancias.

Las técnicas basadas en selección de instancias consisten en la generalización de una ventana que almacena las instancias obtenidas más recientemente y usa el concepto aprendido para predecir el futuro inmediato. En la selección de instancias, el objetivo es seleccionar las instancias más relevantes para el concepto actual. Algunos ejemplos de estos algoritmos con la familia FLORA [WK96], FRANN [KW95]

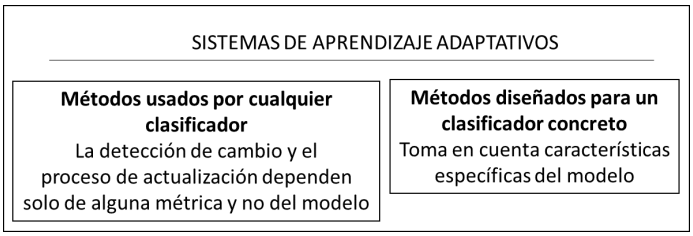


Figura 2.3: Categorización de las técnicas para manipular cambio de concepto atendiendo a su especificidad.

y TWF [Sal97]. Adicionalmente, la selección de un tamaño de ventana apropiado es un compromiso entre una rápida adaptación a cambios abruptos (tamaños de ventana pequeños), 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 ventana es ajustar este tamaño para enfrentar estas situaciones. Algunos algoritmos usan un tamaño de ventana fijo mientras que otros usan heurísticas para ajustar el tamaño, como *Adaptive Size* [Kli04] y FLORA [WK96].

Las instancias también pueden pesarse acorde a su edad y su correspondencia con el concepto actual. La agregación de pesos a instancias usa la idea de algunos algoritmos, como las máquinas de soporte vectorial para procesar instancias acorde a su peso.

Gama y otros [GŽB+] proponen otro esquema genérico para un sistema de aprendizaje adaptativo con cuatro módulos: memoria, aprendizaje, estimación de pérdida y detección de cambio. El módulo de memoria define cómo y cuáles datos deben ser presentados al algoritmo de aprendizaje (módulo de aprendizaje). El módulo de estimación de pérdida controla el rendimiento del algoritmo de aprendizaje y envía información al módulo de detección de cambio para la actualización del modelo si es necesario. Los autores proponer ver a los sistemas de aprendizaje adaptativos como estos cuatro módulos, los cuales pueden permutarse e intercambiarse unos con otros. Sin embargo, algunos algoritmos basados en pesado de instancias no están acorde a este esquema genérico [Rüp01, SW07, GGC08, GUCG11]. Por otro lado, muchos algoritmos tienen una relación fuerte entre los módulos de memoria y aprendizaje, por lo que puede ser difícil verlos como módulos separados.

También puede considerarse la especificidad del método [Kun08] para categorizarlos (Figura 2.3). De esta forma, podemos distinguir entre métodos que pueden ser usados por muchos clasificadores, porque la detección de cambio y el proceso de actualización depende solamente de alguna métrica observable (precisión, etc.) y no del modelo; y métodos específicamente diseñados para un clasificador concreto.

Precisamente, en el siguiente capítulo se sigue esta categorización para presentar diferentes métodos de detección y aprendizaje acorde a su especificidad, moviéndonos desde estrategias generales hasta las más específicas.

3

MINERÍA DE FLUJOS DE DATOS NO ESTACIONARIOS

En algunos modelos de aprendizaje, debido a sus características específicas, se ha aceptado una estrategia general para la manipulación de cambio de concepto. Debido a las peculiaridades de otros modelos, no existe tal estrategia y han surgido métodos muy específicos. En cualquier caso, para el desarrollo o mejora de algoritmos es esencial el entendimiento de los trabajos investigativos existentes, así como la identificación de sus desventajas y limitaciones, que es el objetivo que persigue el presente capítulo. El problema del cambio de concepto ha tomado diferentes nombres en diferentes áreas, lo que ha conllevado a esfuerzos redundantes para solucionar problemas similares.

En este capítulo se resumen los métodos existentes más relevantes para manipular cambio de concepto (generales y específicos para un modelo de aprendizaje), y se presentan métodos y herramientas de evaluación que pueden ser usados para la experimentación con algoritmos que se ejecutan en flujos de datos no estacionarios.

3.1 MÉTODOS GENÉRICOS PARA MANIPULAR CAMBIO DE CONCEPTO

Una estrategia ampliamente usada para la manipulación de cambio de concepto monitoriza constantemente alguna medida de rendimiento del modelo de aprendizaje en el tiempo. Si esta medida se deteriora significativamente, se asume un cambio de concepto y algunas acciones son definidas para actualizar el modelo de aprendizaje acorde a los ejemplos más recientes. En esta estrategia, los detectores de cambio de concepto desempeñan un papel fundamental, ellos se incorporan en el algoritmo de aprendizaje, ya sea para monitorizar la consistencia de partes del modelo o la del modelo completo.

Sobre estos detectores de cambio generalmente son impuestas restricción similares a las de los algoritmos de aprendizaje, estas restricciones están relacionadas fundamentalmente con la complejidad temporal y espacial de los mismos. En general, dadas las restricciones de complejidad espacial y temporal, así como la dificultad de estimar cambios distribucionales arbitrarios, estos detectores generalmente operan sobre un flujo de datos de valores reales correspondientes a alguna medida de rendimiento del modelo de aprendizaje en el tiempo. De esta forma, el problema de detectar cambio de concepto se reduce a estimar cambios distribucionales en un flujo de valores reales.

Mientras algunos autores asumen que la complejidad temporal y espacial no debe depender del número de valores vistos [BN93, Mon01, GMCR04, BdF+06, RTA11] con el objetivo de garantizar un límite teórico para el procesamiento de un número grande (posiblemente infinito) de ejemplos, otros no consideran estas restricciones [KBG04, BG07] aunque sí toman en cuenta la eficiencia computacional de los detectores de cambio propuestos. Por ejemplo, algunos métodos [BN93, GMCR04] tienen un desempeño pobre cuando los conceptos se mantienen constantes por un largo período de tiempo o cuando existen cambios graduales len-

tos; otros acercamientos tratan de resolver este problema considerando estructuras de datos que no son constantes en complejidad temporal y espacial [KBG04, BG07]. Otra discrepancia esta relacionada con los indicadores a tener en cuenta en el diseño y evaluación de detectores de cambio en línea [BN93, GMCR04, KBG04, BG07].

La estrategia de monitorizar la consistencia del algoritmo de aprendizaje a través de detectores de cambio para ejecutar algún mecanismo de aprendizaje a señales de cambio ha sido ampliamente usada. Sin embargo, pocos trabajos [BG09b] le han dado garantías matemáticas a la estrategia en sí, considerando por ejemplo diferentes algoritmos de aprendizaje y tipos comunes de cambio de concepto, para asegurar que el algoritmo de aprendizaje se adapta efectivamente a estos cambios, para garantizar la estabilidad del modelo cuando los conceptos permanecen constantes, etc. Tales acercamientos son más comunes en el campo de la teoría del aprendizaje computacional, donde se asume una estructura subyacente del cambio. Adicionalmente, pocos estudios han investigado cómo el clasificador pasado puede ser usado después de la detección de un cambio de concepto [Žli08].

La detección en línea de cambios distribucionales arbitrarios en largas secuencias de datos ha sido estudiada activamente tanto en las comunidades estadísticas como de aprendizaje automático. En los últimos años, los acercamientos no paramétricos han recibido mayor atención porque ellos asumen menos restricciones teóricas en los datos y así son más flexibles en escenarios del mundo real. Sin embargo, muchos acercamientos existentes no paramétricos no pueden acotar la cantidad de recursos computacionales (tiempo y memoria) necesarios para tal detección. Estas garantías son muy importantes en el procesamiento en línea de enormes cantidades de datos. Inicialmente, algunos acercamientos con componentes heurísticos trataron de solucionar estos problemas, en la actualidad estos han evolucionado a métodos más robustos con garantías matemáticas rigurosas de rendimiento [GMCR04, KBG04, BdF+06, BG07].

3.1.1 *Métodos estadísticos*

Muchos acercamientos estadísticos comparan muestras generadas a partir de dos funciones de distribución de probabilidades, haciendo una prueba estadística bajo la hipótesis nula de que ambas muestras son generadas acorde a la misma función [DR09]. Este es comúnmente conocido como el problema de la homogeneidad. Como ya hemos discutido, los algoritmos de aprendizaje diseñados para detectar cambio usualmente incluyen alguna métrica relacionada con el rendimiento del modelo de aprendizaje inducido, monitorizando esta métrica para disparar señales de alerta cuando el desempeño del modelo ha disminuido significativamente. Quizás por el coste computacional de la inducción de dicho modelo de aprendizaje, el uso de un modelo alternativo para monitorizar el comportamiento de dicha métrica no es común, haciendo posible el uso de varios acercamientos estadísticos existentes. Aún así, considerando la conducta de esta métrica como una señal, existen herramientas del área del procesamiento de señales en línea para detectar cambios con menos coste.

Las pruebas paramétricas son bien conocidas pero tienen algunas limitaciones: restricciones fuertes para datos reales, necesidad de recolectar todos los datos antes de su ejecución, etc. Por otro lado, muchas pruebas no paramétricas populares son

computacionalmente costosas y no incrementales, por lo que la mayoría no son adecuadas para aplicarse directamente a la detección de cambio en flujos de datos.

Afortunadamente, se han propuesto algunos métodos para detectar cambio en escenarios de flujos de datos. Dada una secuencia $\langle x_1, x_2, \ldots \rangle$, donde cada elemento x_i es generado por alguna función de distribución de probabilidades P_i, algunos autores [KBG04] llaman punto de cambio al tiempo $i+1$ si existe un cambio en estas distribuciones ($P_i \neq P_{i+1}$). En este sentido, algunos trabajos extienden pruebas no paramétricas [CAW98], otros están basados en análisis de vecinos más cercanos [Taj02], distancias entre estimación de densidades [AHT94, DKVY06, Kun13], estadísticos basados en discrepancia media máxima para núcleos universales [GBR+06] y detección de cambios abruptos en línea [BN93].

Cuando la función de distribución de probabilidades está completamente especificada antes y después del punto de corte, el problema ha sido resuelto bajo varios criterios [BN93]. Algunos esquemas populares son el control de gráficos de Shewart, el control de gráficos de promedios, el procedimiento CUSUM (*Cumulative Sum*) y el procedimiento Shiryayev-Roberts. Otra área bien conocida envuelve el estadístico EWMA (*Exponentially Weighted Moving Average*) [BHKP10, BN93, Mon01].

Otros métodos modelan los datos de entrada incrementalmente, aprendiendo una función de densidad de probabilidades. Así, se monitorizan cambio entre pares (o secuencias) de estas funciones de densidad de probabilidades. El logaritmo de la proporción de probabilidades es usado frecuentemente en la detección de cambio [Kun13]. Por ejemplo, $s(X) = \log\left(P_{\theta_1}(X) / P_{\theta_2}(X)\right)$ es el logaritmo de la proporción de probabilidades, donde X es una variable aleatoria y $P_\theta(X)$ es una función de densidad de probabilidades con parámetro escalar θ. Un cambio en el parámetro θ se refleja en un cambio de signo en el valor medio de $s(X)$.

Los acercamientos más explorados que modelan los datos de entrada se basan en modelos paramétricos predefinidos, como modelos de densidad de probabilidades [KBG04, DKVY06], modelos auto-regresivos [YT02] y modelos de estado-espacio [KYM07].

Kawahara y Sugiyama [KS09] evitan el problema de la estimación de densidad no paramétrico calculando directamente la proporción de densidad de probabilidades. La tarea es detectar si existe un punto de cambio entre par de intervalos de tiempo consecutivos, llamados los intervalos de prueba y referencia. Si $P_{prueba}(X)$ y $P_{ref}(Y)$ son funciones de densidad de probabilidades de las muestras de prueba y referencia respectivamente, basado en el logaritmo de la proporción de probabilidades $S = \sum_{i=1}^{n_{prueba}} \ln\left(P_{prueba}(X_i)/P_{ref}(X_i)\right)$, se estima un cambio si $S > \mu$, donde $\mu > 0$ es un umbral predeterminado. La estimación de la proporción $w(X) = P_{prueba}(X)/P_{ref}(X_i)$ es modelada por un núcleo Gausiano no paramétrico mediante un algoritmo incremental propuesto.

Kifer y otros [KBG04] usan funciones de distancia entre distribuciones a través de pruebas no paramétricas. De esta forma no es necesario asumir la forma de la distribución de probabilidades. Ellos definen tres funciones de distancia y mantienen conjuntos formados por dos ventanas (anterior y posterior) de diferentes tamaños y almacenando secuencias de flujos de datos de entrada. Se acepta un cambio en la distribución de probabilidades si la función de distancia entre algún par de ventanas es mayor que un determinado umbral. En este sentido, definen

una generalización de la prueba de Kolmogorov-Smirnoff teniendo en cuenta estas funciones de distancia.

Otros acercamientos alternativos son los bayesianos, que surgen cuando se tiene información relacionada con la distribución del punto de cambio. Se han propuesto muchos métodos bayesianos para detectar puntos de cambio, pero la mayoría de estos no son directamente aplicables al aprendizaje supervisado. Por otro lado, pocos de estos métodos han sido aplicados al aprendizaje en flujos de datos [BN93, BM10, AM07, PM07].

La mayoría de los acercamientos estadísticos mencionados proveen garantías matemáticas de desempeño. Sin embargo, de forma general, más garantías de rendimiento implica tener más conocimiento acerca de la función de distribución de probabilidades de la secuencia de entrada y de esta forma, mayores restricciones para datos reales. Muchos acercamientos no paramétricos populares que asumen menos conocimiento relacionado con los datos de entrada no satisfacen requerimientos comunes necesarios para el aprendizaje incremental (de simple pasada por los datos, complejidad temporal constante por ejemplo procesado, etc. [BN93, Mono1, DGIM02, RTA11]). En otros casos, los métodos requieren el ajuste de varios parámetros de entrada [KS09], lo que puede ser una tarea difícil. En este contexto, son necesarias nuevas investigaciones para solucionar estas limitaciones.

3.1.2 *Acercamientos basados en ventanas*

Muchos algoritmos usan un espacio de memoria para almacenar ejemplos con el objetivo de ejecutar algún proceso de aprendizaje con estos ejemplos almacenados. Este espacio de memoria es llamado usualmente ventana (o ventana de tiempo) y existen múltiples acercamientos e ideas para usarlas. Las ventanas de tiempo [GR07] son una estructura de datos ampliamente usada para manipular cambio de concepto (e.g. ventanas de marcas [GKS01, Foro6, CKN08], ventanas deslizantes [GMCR04, BdF+06, BG07] y ventanas etiquetadas [HK06, BG07, BPRH13]). Los ejemplos dentro de estas ventanas de tiempo tienen asociado una marca de tiempo que define su edad. Los algoritmos tratan de mantener la información más relevante en estas ventanas acorte al concepto actual.

Klinkenberg y Joachims [KR98] monitorizan los valores de tres indicadores de rendimiento del modelo de aprendizaje en el tiempo para manipular cambio de concepto. El promedio $\mu_M^{Indicador}$ y la desviación estándar $s_M^{Indicador}$ es computado en cada marca de tiempo para cada uno de los indicadores basado en los últimos M lotes. Si algún valor del indicador $\mu_t^{Indicador}$ está por debajo del intervalo de confianza $\mu_M^{Indicador} - \alpha \cdot s_M^{Indicador}$, el tamaño de ventana es reducido en una proporción determinada $\gamma\,\%$ (α y γ son parámetros predefinidos por el usuario). Adicionalmente, si $\mu_t^{Indicador} < \beta \cdot \mu_{t-1}^{Indicador}$ (β es otro parámetro predefinido) se estima un cambio de concepto abrupto y el tamaño de la ventana es reducido al tamaño de un lote. Si no se detecta cambio de concepto, todos los ejemplos vistos con almacenados en la ventana para tener un conjunto de entrenamiento de tamaño máximo. Sin embargo, como señalamos anteriormente, configurar estos parámetros para obtener un comportamiento adecuado del método en la mayoría de los casos es una tarea difícil [ŽBG+12a].

En este sentido, Gama y otros [GMCR04] toman en el algoritmo DDM (*Drift Detection Method*), para un grupo de ejemplos, el error en la predicción de un algoritmo de aprendizaje como una variable aleatoria correspondiente a experimentos de Bernoulli. Ellos asumen la distribución binomial y consideran que para un número grande de ejemplos esta se aproxima a la normal. Así calculan el error en la predicción p_i y su desviación estándar $s_i = \sqrt{p_i(1-p_i)/i}$. El método para la detección de cambio mantiene almacenadas dos variables en el entrenamiento del algoritmo de aprendizaje, p_{min} y s_{min}, que son actualizadas cuando un nuevo ejemplo i causa que $p_i + s_i < p_{min} + s_{min}$. Para la detección de cambio son establecidos un umbral de alerta (ϵ_A) y otro de cambio (ϵ_C), que son alcanzados si la suma $p_i + s_i$ los sobrepasa. En los experimentos propuestos, para una confianza aproximada de 0,95, configuran el nivel de alerta a $\epsilon_A = p_{min} + 2s_{min}$ y para una confianza aproximada de 0,99 el nivel de cambio a $\epsilon_C = p_{min} + 3s_{min}$. Este método tiene un buen desempeño en cambios abrupto y en cambios graduales cuando el último no es muy lento. Para mejorar la detección de cambios graduales muy lentos, Baena y otros [BdF+06] proponen EDDM (*Early Drift Detection Method*) donde la idea básica es considerar la distancia entre dos errores de clasificación en vez de solamente considerar el número de errores. Cuando el modelo está aprendiendo en ausencia de cambio, la distancia entre errores debe incrementarse, pero cuando un cambio ocurre la distancia disminuye.

Otro uso interesante de las ventanas es el algoritmo ADWIN (*Adjustive Window*) [BG07]. Su idea es eliminar una parte obsoleta de la ventana cuando se concluye que esta es suficientemente larga y exhibe una distribución suficientemente distinta. Por esto la ventana de tamaño n es dividida en todas las posibles de longitud n_0 y n_1 ($n = n_0 + n_1$). Los elementos menos recientes almacenados en la ventana son eliminados cuando las medias de las subventanas (de longitudes n_0 y n_1) difieren en un valor mayor que $\sqrt{1/(2m)\,(\ln 4/\delta)}$, donde m es la media armónica entre n_0 y n_1; y δ es un valor de probabilidad relacionado con el nivel de confianza. ADWIN tiene garantías probabilísticas de rendimiento en términos de la tasa de falsos positivos y falsos negativos. ADWIN2 es una versión mejorada que usa la idea de algunos algoritmos desarrollados para flujos de datos [BDM02, BBD+02, DGIM02, Muto5] de encontrar un punto de cambio rápidamente sin considerar todas las divisiones posibles de la ventana. Más recientemente, los mismos autores propusieron el diseño de un sistema con un módulo de memoria, un módulo de estimación y un módulo de detección de cambio [BG09b]. En este sentido, la memoria es el componente donde el algoritmo almacena los datos que considera relevantes, el estimador es el algoritmo que estima el estadístico deseado y el detector de cambio dispara señales de cambio cuando se estima un cambio en la distribución de los datos. Esta arquitectura puede ser fácilmente adaptable a muchos otros sistemas y ampliamente configurable.

Žliobaitė y Kuncheva [ŽK09] asumen dos clases Gausianas y derivan un tamaño de ventana óptimo después del punto de cambio. El método no elimina inmediatamente el clasificador antiguo porque él puede tener más precisión en la clasificación que el nuevo clasificador entrenado con pocos ejemplos. Las autoras derivan un tamaño de ventana $N^* = f(C)/(E_2^\infty(C_1) - E_2^\infty(C_2))$ donde $f(C)$ es una función que depende del tipo de clasificador. Variantes de $f(C)$ para diferentes clasificadores es tabulada por Fukunaga and Hayes [FH89]. Así, $E^N(C)$ es el error del

clasificador C entrenado con N ejemplos y $E^{\infty}(C) = \lim_{N\to\infty} E^{N}(C)$; el subíndice de E_2^{∞} corresponde al nuevo concepto. El valor del error $E_i^{\infty}(C)$ puede ser derivado para distribuciones específicas y clasificadores [Rau01].

3.1.3 *Pesado de instancias*

Aunque las ventanas de tiempo constituyen una herramienta poderosa para el aprendizaje en línea, ellas también tienen algunas limitaciones [LV04]. Por ejemplo, no hay un solo tamaño de ventana para lidiar con cambios abruptos y graduales de forma eficaz. Incluso con ventanas de tamaño dinámico, es difícil manipular cambios de concepto graduales (donde no están bien definidos los puntos de cambio). Como hemos indicado, otra estrategia difundida asigna pesos a los ejemplos [BN93, CE02, LV04, Kli04, NFM05, NFM07, Mon01] o a partes de la hipótesis [WFYH03, KTV08] acorde a su edad o utilidad actual. El pesado de ejemplos sigue la idea de algunos algoritmos como las máquinas de soporte vectorial para procesar ejemplos acorde a su peso [Lit91, Kli04].

En el dominio del filtrado de información, Klinkenberg [Kli04] indica que los usuarios pueden cambiar su interés en un tópico específicos de manera lenta. Para tratar este tipo de cambio sugiere decrementar la importancia de los ejemplos por medio de un esquema de pesado. En este sentido, el pesado global selecciona los pesos de los ejemplos x_t basado en su respectiva edad y usando una función de edad exponencial $w_\eta(x_t) = \exp(-\eta t)$, donde el parámetro$\eta$ funciona como el factor de olvido ($\eta \geq 0$). En el pesado local de ejemplos los pesos óptimos son estimados por medio del rendimiento del algoritmo de aprendizaje en nuevos lotes de datos.

Otra solución difundida usa promedios ponderados, donde los datos recientes reciben más peso (ya que ellos tienen mayor probabilidad de ocurrencia) que los datos previos. Por ejemplo, Seldin y otros [SLC+12] presentan algunas desigualdades para controlar promedios ponderados en dominios con cambio de concepto y con aplicaciones en el aprendizaje con reforzamiento y otros dominios de aprendizaje interactivo. Un área conocida envuelve al estadístico EWMA, que es un estimador óptimo cuando la media sigue un modelo de promedios integrables de primer orden [BJ90] y cuando la media está sujeta a determinados cambios aleatorios [CE02]. EWMA es tan simple de aplicar como CUSUM, puede ser usada para estimar el valor actual de la media poblacional, y tiene un rendimiento comparable con el procedimiento CUSUM [LSB+90]. Los pesos en el estimador EWMA $\hat{X}_t$ decrecen exponencialmente como en una serie geométrica en el tiempo: $\hat{X}_t = (1-\lambda)\hat{X}_{t-1} + \lambda X_t$, donde λ actúa como factor de olvido ($0 < \lambda \leq 1$), X_t es el valor actual de la secuencia de variables aleatorias y $\hat{X}_0$ puede ser tomado como el promedio de datos anteriores. Usualmente, si este estadístico excede un límite de control específico se estima un cambio distribucional. Sin embargo, la mayoría de los estudios asumen que las variables aleatorias X_t están reguladas por una función de distribución de probabilidades conocida [Mon01].

Nuñez y otros [NFM07] implementan una variante del procedimiento EWMA para detectar cambio de concepto en la inducción de árboles de decisión. Específicamente, ellos consideran un cambio de concepto si una medida de rendimiento

Nombre	Año	No paramétrico	Complejidad computacional constante	Garantías matemáticas de desempeño
Klinkengerg y Joachims [KR98]	1998	X	X	
DDM [GMCR04]	2004	X	X	
EDDM [BdF+06]	2006	X	X	
ADWIN [BG07]	2007	X		X
Žliobaitė y Kuncheva [ŽK09]	2009			X
ECDD [RATH12]	2012	X	X	

Tabla 3.1: Métodos basados en ventanas y de pesado de instancias para detectar cambio de concepto usados en el aprendizaje incremental.

tiene un descenso persistente, siendo esta persistencia la clave para distinguir entre ruido y un verdadero cambio de concepto.

Otro método más reciente, ECDD (*EWMA Concept Drift Detector*) [RATH12], usa el estadístico EWMA para monitorizar la tasa de errores en la clasificación de un clasificador en línea. El límite de control es calculado a través de un acercamiento Monte Carlo y un método de regresión para ajustar un polinomio bajo disímiles condiciones. Los polinomios son hallados teniendo en cuenta el parámetro λ y la tasa de falsos positivos deseada. Estos polinomios deben ser almacenados en una tabla para detectar cambios de concepto en el aprendizaje en línea.

Cohen y Strauss [CS06] formalizan el problema de mantener sumas y estadísticos en un flujo de datos. Ellos examinan algunas familias de funciones de decaimiento y exploran requerimientos de almacenamiento principalmente para escenarios exponenciales ($\exp(-\eta t)$), polinomiales ($1/t^{\alpha}$) y de ventanas deslizantes. También proponen propiedades deseables para una familia de funciones de decaimiento, entre ellas una clase suficientemente rica de tasas de pesado ya que aplicaciones particulares pueden depender de escalas de tiempo y correlaciones entre valores.

Aunque la mayoría de los acercamientos descritos anteriormente pueden aplicarse a cualquier clasificador en línea para manipular cambio de concepto, las características particulares de cada modelo de aprendizaje comúnmente se tienen en cuenta para que la adaptación al cambio sea más eficaz. En la siguiente sección se muestran los principales métodos existentes para tal adaptación teniendo en cuenta la especificidad del modelo.

3.2 MÉTODOS ESPECÍFICOS ACORDE AL MODELO DE APRENDIZAJE

Los requerimientos comunes para los sistemas de aprendizaje incremental incluyen procesar ejemplos uno a la vez e inspeccionarlo solo una vez, usar una cantidad limitada de memoria, ejecutarse con una cantidad limitada de tiempo y

estar listos para predecir en cualquier momento [BHKP10]. También es deseable que el modelo de aprendizaje inducido por un algoritmo incremental tenga similitud con el modelo inducido por un algoritmo de aprendizaje por lotes [DH00], principalmente debido al poder de generalización de estos últimos. Algoritmos no paramétricos, que significa tener tan pocos parámetros de entrada como sea posible son también muy ventajosos, debido a que la alta velocidad a la que pueden recibirse los datos generalmente no permite el ajuste de tales parámetros en línea.

Para manipular cambio de concepto, algunos algoritmos de aprendizaje usan una o más instancias de un detector de cambio de concepto dado, pero a diferencia de estos detectores, estos métodos son específicos al modelo de aprendizaje y son difícilmente generalizables a otros modelos. Sin embargo, al tener en cuenta características específicas del modelo de aprendizaje estos pueden adaptarse al cambio más efectivamente. Además, algunas investigaciones han observado que modelos particulares de aprendizaje pueden ser más adecuados para varios tipos de aplicaciones.

Por ejemplo, para el filtrado de correo spam se han utilizado técnicas de Naïve Bayes [APK+00], *boosting* [FHT00], máquinas de soporte vectorial [DWV99] y reglas de clasificación [Joa97]; pero los acercamientos basados en casos funcionan mejor que las técnicas mencionadas en estos dominios [DC04]. Cunningham y otros [CNDH03] sugieren que el razonamiento basado en casos ofrece ventajas en el filtrado de correos spam debido a que en estos dominios existen conceptos disjuntos, mientras que por ejemplo, el algoritmo Naïve Bayes que es popular para la clasificación de textos, trata de aprender una descripción de concepto unificado. El tratamiento de cambio de concepto en correos spam es difícil ya que los creadores de estos correos cambian activamente la naturaleza de sus mensajes para eludir a dichos filtros [FID+07].

En cualquier caso, el algoritmo diseñado para manipular cambio de concepto debe incorporar mecanismos para olvidar ejemplos pasados que no son significativos para el concepto actual. Muchas ideas previas pueden ser usadas en una gran variedad de algoritmos, por ejemplo los ensamble de clasificadores como SEA (*Streaming Ensemble Algorithm*) [SK01] y DWM (*Dynamic Weighted Majority*) [KM07]. Ellos están basados en heurísticas y medidas de interés para eliminar, reactivar o adicionar nuevos descriptores dinámicamente en respuesta a variaciones en la consistencia del modelo con respecto a nuevos ejemplos recibidos. Debido al potencial de la heterogeneidad es sencillo combinar diferentes acercamientos en el mismo ensamble.

En los apartados siguientes mostramos cómo algunos modelos de aprendizaje comunes son capaces de manipular cambio de concepto teniendo en cuenta características específicas del modelo.

3.2.1 *Árboles de decisión*

Debido a conocidas características, la construcción de árboles de decisión ha sido ampliamente estudiada en la comunidad del aprendizaje automático [Mur98]. En la minería de flujos de datos, la desigualdad de probabilidades de Hoeffding [DH00, HSD01, del07, dRGM08, RM00, Ram01, RMd04, RdCM05], de Chernoff [del07, dRGM08, RM00, Ram01, RMd04, RdCM05] y recientemente la de McDiar-

mid [RPDJ13, RJPD14] han sido herramientas poderosas para la inducción en línea de árboles de decisión.

Una familia popular de algoritmos para la manipulación de cambio de concepto esta basada en VFDT (*Very Fast Decision Tree*) [DH00], incluso aplicando estas ideas a otros modelos de aprendizaje como reglas de decisión [KG12]. Sin embargo, Rutkowski y otros [RPDJ13] demostraron recientemente que la desigualdad de Hoeffding no es una herramienta adecuada para resolver problemas de calcular un intervalo de confianza para cualquier medida heurística (ganancia de información, etc.), por lo que los métodos y algoritmos desarrollados en la literatura que usan la cota de Hoeffding deben ser revisados. Un uso correcto de las cotas de Hoeffding y Chernoff es presentado en la familia IADEM de algoritmos [del07, dRGM08, RM00, Ram01, RMd04, RdCM05].

Una estrategia común para la manipulación de cambio de concepto sobre la inducción incremental de árboles de decisión es llevada a cabo en dos pasos generales: (1) estimar en cada nodo de decisión si el subárbol con raíz en ese nodo es consistente con los datos actuales, (2) si se estima una inconsistencia en cualquier nodo de decisión se ejecutan algunas acciones para reconstruir el subárbol correspondiente. Con esta estrategia se tienen en cuenta cambios locales, ya que las inconsistencias del modelo se monitorizan en subconjuntos del espacio de instancias (en los nodos de decisión), permitiendo además reaccionar más eficazmente a cambios de concepto [HSD01, NFM07, CCW08, IGD11]. Los cambios graduales y lentos también han sido considerados [HSD01] monitorizando subárboles en nodos de decisión pertenecientes ya sea al árbol principal o a un subárbol alternativo creado anteriormente debido a alguna inconsistencia.

Para monitorizar la consistencia de subárboles se han propuesto algunos métodos: verificar que la prueba de división instalada previamente en un nodo de decisión todavía es la mejor opción [HSD01]; confirmar que cada nodo visitado brinda alguna significación estadística al concepto [GFR06, NFM05, NFM07]; poda basada en errores de clasificación [GFR06]; monitorizar la tasa del error en cada nodo de decisión, para disparar señales de cambio cuando ocurran incrementos significativos [GMR05, BHKP10, IGD11]. Cuando se monitoriza el error, Ikonomovska y otros [IGD11] hacen dos distinciones: cuando el error es calculado usando la predicción del nodo actual (*top-down*); y métodos donde el error se calcula usando la predicción de las hojas del árbol (*bottom-up*). Por ejemplo, UFFT (*Ultra Fast Forest of Trees*) usa una estrategia *top-down* con un clasificador Naïve Bayes instalado en cada nodo, y una estrategia *bottom-up* es presentada en AdaptiveHoeffdingTree [BHKP10].

Cuando se monitoriza la tasa del error regularmente se obtiene un flujo de valores reales [NFM05, NFM07] o un flujo de bits [GMCR04, GMR05]. Como se ha discutido anteriormente, cambios significativos en un estadístico calculado a partir de estos valores (e.g. el promedio) puede interpretarse como un cambio de concepto. Así, varios métodos estadísticos pueden ser aplicados para tal monitorización en línea (ver Sección 3.1).

Nuñez y otros [NFM07] monitorizan la tasa de error por medio de una medida de rendimiento en los nodos hojas. Cada nodo hoja mantiene una ventana deslizante que contiene las instancias más relevantes. Ellos diseñan una heurística para disminuir el tamaño de la ventana instalada en un nodo hoja cuando se estima

una degradación persistente de la medida de rendimiento (estado de degradación). También definen una heurística para olvidar ejemplos pasados cuando los nodos hojas están en un estado de mejora y así evitar acumulación excesiva de ejemplos. La medida de rendimiento ($perf$) es calculada por medio de un acercamiento de pesado exponencial, a partir de la precisión instantánea actual (ia) y la calculada en el paso previo $perf(n) = \frac{7}{8}perf(n-1) + \frac{1}{8}ia$. Si un determinado nodo de decisión no pasa una prueba χ^2 (interpretado como que la prueba de división instalada en ese nodo no contribuye más al concepto actual), los nodos hojas que se encuentran en estado de degradación y que son descendentes de ese nodo de decisión son forzadas a olvidar ejemplos viejos, un nodo hoja reemplaza el subárbol en cuestión y se intenta expandir esta nueva hoja.

A una inconsistencia previa, algunos algoritmos basados en árboles de decisión usan una memoria a corto plazo para entrenar un subárbol alternativo que podrá reemplazar a su correspondiente subárbol original si esta inconsistencia persiste [GMR05, NFM07]. Otros acercamientos son más conservativos, ellos construyen un subárbol alternativo con raíz en un nodo inconsistente que solo reemplazará el subárbol original si prueba ser más preciso [HSD01, BHKP10]. Usar una memoria a corto plazo es más simple y más eficiente computacionalmente, pero desafortunadamente tiene dos desventajas fundamentales. Primero, si la persistencia estimada no se corresponde con un cambio de concepto (e.g. una detección falsa), podar un subárbol grande puede deteriorar significativamente el modelo de aprendizaje. Segundo, incluso cuando la detección del cambio se corresponde con un cambio real, el nuevo subárbol entrenado con pocos ejemplos puede ser menos preciso que el subárbol antiguo entrenado con muchos ejemplos. Sin embargo, para usar la segunda estrategia y ser más conservativo es necesario resolver dos cuestiones adicionales: (1) cuál subárbol (el alternativo o el original) es más coherente (e.g. preciso) con respecto al concepto actual y (2) cuándo eliminar un subárbol alternativo que ya no es efectivo.

Se han propuesto varios métodos para resolver estas cuestiones, pero muchos de estos tienen algunos inconvenientes. En CVFDT (*Concept-adapting Very Fast Decision Tree*) [HSD01] los subárboles alternativos cambian entre los modos entrenamiento y prueba por medio de parámetros ajustados por el usuario. En la fase de prueba, CVFDT elimina subárboles alternativos cuya precisión no aumenta en el tiempo. Ikonomovska y otros [IGD11] monitorizan la diferencia de errores entre el subárbol alternativo y el original usando el estadístico Q [GSR09] en un intervalo de evaluación fijado por el usuario. Ellos también monitorizan este estadístico para eliminar subárboles alternativos si no mejoran en el tiempo. AdaptiveHoeffdingTree [BHKP10] compara los errores cometidos por los subárboles (ya sea para promover o para eliminar subárboles alternativos) usando un intervalo de confianza calculado a través de la desigualdad de Hoeffding [BG07].

Cheng y otros [CCW08] proponen otro acercamiento interesante. Diseñan un algoritmo aplicable en situaciones donde se requiere más sensibilidad en la detección de cambio de concepto (e.g. detectando transacciones de crédito fraudulentas). En este sentido, el algoritmo tiene contadores globales para cada clase en cada valor de atributo. Los contadores se actualizan en dos bloques que contienen un determinado número de ejemplos. Una prueba χ^2 compara la distribución de clases en los valores de los atributos para estimar cambio de concepto. Si se estima una

variación en esta distribución, en dependencia de cómo están distribuidos en el árbol los valores y los atributos en los cuales se ha estimado dicho cambio, se crean subárboles a partir del nuevo bloque de ejemplos que reemplazará o expandirá partes del árbol original.

Otro método diferente es propuesto por Natwichai y Li [NL04]. Ellos usan una tabla de decisión ambigua [Col99] como intermediaria para la representación del conocimiento. Cada fila en dicha tabla es una regla de decisión obtenida desde CVFDT cuando este genera un subárbol alternativo en respuesta 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 el error en la predicción, ambas pertenecientes a una determinada regla (una fila). Para extraer conocimiento reciente, se induce un árbol de decisión desde dicha tabla generando el árbol con menor error, este error calculado a partir de las columnas que representan la marca de tiempo y el error en la precisión.

Adicionalmente, algunas técnicas de hibridación recientes que envuelven árboles de decisión han probado ser una herramienta poderosa en la tarea del aprendizaje incremental [BHPF10, Woz11]. La extensión de estas técnicas al aprendizaje en flujos de datos no estacionarios es un área de investigación prometedora.

3.2.2 *Reglas de decisión*

Aunque los árboles de decisión pueden manipular cambios locales y graduales, incluso una variación pequeña en la función objetivo puede causar cambios considerables en el modelo y de esta forma comprometer severamente la eficiencia en el aprendizaje [WFYH03]. Las reglas de decisión pueden tener una adaptación más rápida al cambio de concepto ya que se pueden eliminar reglas individuales sin la necesidad de reconstruir otras partes del modelo. Sin embargo, las reglas de decisión han recibido poca atención en la minería de flujo de datos [GK11, KG12, Dec13]. Un área reciente de investigación está relacionada con sistemas difusos basados en reglas [Bou11, BV13].

La mayoría de los sistemas basados en reglas que manipulan cambio de concepto adaptan el aprendizaje continuamente sin considerar que ha ocurrido un cambio de concepto. En estos casos la adaptación es probablemente menos efectiva que cuando se usa la detección de cambio explícita. Entre los algoritmos basados en reglas de decisión más populares que manipulan cambio de concepto encontramos a STAGGER [SG86], la familia FLORA [WK96] y la familia de algoritmos AQ-PM [Mal03, MM04].

Por ejemplo, STAGGER [SG86] induce formulas booleanas (caracterizaciones) que se mueven desde caracterizaciones más generales (e.g. disyunción de pares atributo-valor) a más particulares (e.g. conjunciones de pares atributo-valor). Sobre estas caracterizaciones son definidos tres operadores de búsqueda (para crear una fórmula más general, una más particular, o negar una fórmula existente), los cuales son ejecutados cuando el modelo comete un error en la clasificación. Las caracterizaciones compiten por ser parte de una frontera de búsqueda acorde a su precisión, ya que solo es retenida una frontera con las mejores caracterizaciones. Como las formulas inefectivas son eliminadas, STAGGER puede aprender en dominios cambiantes en el tiempo.

Otro ejemplo es AP11-PM [MM04], que selecciona ejemplos que se encuentran en la frontera de las reglas de asociación sobre una ventana de tamaño fijo; y FLORA2 [WK96] que usa una ventana de tamaño variable basada en una heurística que decide cuántos ejemplos deben ser eliminados, ya que todos los ejemplos nuevos son adicionados a la ventana.

Mientras que los sistemas anteriores basados en reglas solo manipulan atributos numéricos, FACIL (*Fast and Adaptive Classifier by Incremental Learning*) [FAR05] aprende reglas de decisión incrementalmente a partir de ejemplos en un flujo de datos que pueden tener atributos numéricos. Se mantiene una ventana por cada regla y solo es permitido que una regla almacene al menos un ejemplo positivo por ejemplo negativo cubierto. Cuando una nueva instancia x_i se obtiene, el sistema chequea si alguna regla asociada con su misma clase puede cubrir esta instancia, si esta instancia es cubierta por una regla que tiene asociada un valor de clase diferente, o si x_i no es cubierta. Para cubrir un ejemplo una regla puede tener un crecimiento moderado. Los autores usan una heurística para insertar ejemplos positivos y negativos con respecto a una regla, dentro de su respectiva ventana. Cuando la proporción entre ejemplos positivos y negativos excede un límite determinado, se construyen nuevas reglas consistentes a partir de los ejemplos almacenados en las respectivas ventanas; como resultado las reglas viejas son probablemente eliminadas. Parecido a AQ-PM, FACIL también posee un mecanismo de olvido que es activado cuando los ejemplos son más antiguos que un umbral fijado por el usuario (explícito) o elimina ejemplos que no son relevantes debido a que no se encuentran en ninguna frontera de descripción de conceptos (implícito).

Aunque las heurísticas anteriormente mencionadas son intuitivas y tienen un buen rendimiento en varios dominios de aplicación, ellas también requieren ajustar varios parámetros de entrada. Este ajuste naturalmente puede ser una tarea difícil, particularmente si existe poco conocimiento del problema en cuestión. Adicionalmente, ellas carecen de una base teórica apropiada. Por otro lado, algunos de estos algoritmos [SG86, WK96] asumen una forma restringida de modelos de aprendizaje, lo que puede limitar su aplicabilidad en situaciones del mundo real.

Más recientemente, Gama y Kosina [GK11] propusieron el sistema VFDR (*Very Fast Decision Rules*). Similar a VFDT [DH00], VFDR expande una regla por el literal que minimiza la entropía de las etiquetas de clase de los ejemplos cubiertos por la regla. La desigualdad de Hoeffding determina el número de ejemplos después de los cuales la regla puede ser expandida. Mientra que VFDR se adapta al cambio solamente de manera implícita cuando infiere nuevas reglas y especializa las existentes, AVFDR [KG12] manipula cambio de concepto por medio de DDM [GMCR04] en cada regla de decisión. Una señal de alerta disparada por DDM detiene el aprendizaje en la regla correspondiente, si la regla alcanza el estado de cambio estimado por DDM la misma es eliminada del conjunto de reglas. Cuando DDM estima que el concepto actual es estable el proceso de aprendizaje continúa normalmente. Sin embargo, como Rutkowski y otros [RPDJ13] señalan, los algoritmos de aprendizaje incremental que usan la cota de Hoeffding deben ser revisados.

Un problema diferente de cambio de concepto es estudiado por Cheng y otros [CCW09]. Ellos pretenden descubrir las reglas que causan tal cambio de concepto. Este acercamiento es útil cuando en la toma de decisiones (por ejemplo, en el diag-

nóstico médico) se está interesado en las reglas (causas) de un cambio de concepto. Para minar las reglas del cambio de concepto, los algoritmos propuestos integran ejemplos antiguos y nuevos en tiempos diferentes. Así, las reglas son construidas siguiendo una inducción tradicional de un modelo de árboles de decisión. Como los autores señalan, este interesante problema nunca había sido estudiado anteriormente.

3.2.3 *Máquinas de soporte vectorial*

Las máquinas de soporte vectorial [Vap95] construyen el hiperplano óptimo que separa a los datos de entrada a través de minimización del riesgo estructural. Las máquinas de soporte vectorial son un método robusto y popular en muchas áreas del aprendizaje. Por ejemplo, ellas tienen un rendimiento muy alto para la clasificación de textos debido a las propiedades de estos [Joa98]: alta dimensión del espacio de entrada, pocos atributos relevantes, poca densidad de los vectores de los documentos, separabilidad lineal de muchos problemas de clasificación de textos. Las máquinas de soporte vectorial también han sido favorecidas para la detección de correos spam basada en el contenido [SW07].

Sin embargo, el entrenamiento de las máquinas de soporte vectorial es con frecuencia computacionalmente costoso, y hasta ahora presenta algunos retos que no han sido resueltos en el aprendizaje en línea [KSW04, SW07]. Wen y Lu [WL07] indican que varios tipos de sistemas artificiales de aprendizaje aprenden por un procedimiento de optimización global y no se adaptan al aprendizaje incremental de forma natural, ya que el modelo de computación carece de la habilidad de adquirir nuevos conocimientos o no puede retener el conocimiento anteriormente aprendido. Algunas máquinas de soporte vectorial ofrecen resultados aproximados con el objetivo de reducir el coste computacional [SLH+99, Rüp01, SW07]. Otras investigaciones intentan reducir el coste computacional pero no garantizan una cota para la complejidad temporal y espacial [CP01]. Por otro lado, algunos métodos de entrenamiento son lineales en el número de ejemplos pero son cuadráticos en el número de atributos [Joa06]. Una máquina de soporte vectorial rápida fue propuesta por Joachims [Joa06], la cual entrena en $O(ns)$ de complejidad temporal, donde n es el número de ejemplos y s el número de atributos que no tienen componentes nulas.

Muchos acercamientos incrementales que conciernen máquinas de soporte vectorial se adaptan a cambios de concepto de forma implícita, ellos dan más importancia a los datos más recientes como mecanismo de olvido [Rüp01, KSW04, SW07, GGC08, GUCG11]. Tales algoritmos generalmente incorporan un parámetro al problema de optimización para regular la importancia relativa entre ejemplos antiguos (o hipótesis antiguas) y nuevos. Sin embargo, estos parámetros pueden restringir la cantidad de cambio con la que el algoritmo de aprendizaje puede lidiar [GGC08, GUCG11]. Como hemos discutido, el ajuste de estos parámetros puede ser una tarea difícil.

Por ejemplo, Syed [SLH+99] entrena una máquina de soporte vectorial tomando los vectores de soporte calculados a partir de un lote de instancias. Cuando un nuevo lote de instancias es obtenido, los vectores de soporte son actualizados usando los anteriormente calculados y los nuevos a partir de este nuevo lote. Más tarde,

Rüping [Rüp01] argumenta que si el lote de ejemplos es una buena muestra, se puede esperar que la función de decisión resultante sea aproximadamente similar a la función de decisión final. De esta forma, un vector de soporte en el conjunto final es probable que haya sido un vector de soporte en una iteración previa. El problema con este acercamiento es la suposición que un lote es una muestra adecuada de los datos. Así, el autor propone una mejora que consiste en la penalización del error cuando se usan vectores de soporte viejos (que representan conjuntos de aprendizaje antiguos). De esta forma usar modelos viejos tiene más coste que usar nuevos ejemplos. Esta modificación al problema de las máquinas de soporte vectorial puede ser visto como entrenar una máquina de soporte vectorial con respecto a una nueva función objetivo.

Un acercamiento basado en ventanas directamente relacionado con máquinas de soporte vectorial es propuesto por Klinkenberg y Joachims [KJ00]. Ellos manipulan cambio de concepto manteniendo una ventana sobre los datos de entrenamiento que incluye los últimos n ejemplos, asumiendo que la cantidad de cambio se incrementa en el tiempo. La idea clave es seleccionar el tamaño de ventana tal que el error estimado de generalización en los nuevos ejemplos se minimice usando una forma especial de estimadores ($\xi\alpha$-estimates) [Joa00]. Asumiendo que los ejemplos llegan en lotes de igual tamaño, el método esencialmente trata varios tamaños de ventana, entrenado una máquina de soporte vectorial para cada conjunto de entrenamiento. El algoritmo selecciona el tamaño de ventana que minimice la tasa de error estimada por esta forma especial de estimadores.

Un método que manipula cambio de concepto puramente con máquinas de soporte vectorial es descrito en el trabajo de Dries y Rückert [DR09]. Su idea es evaluar una función de decisión fija f en dos secuencias (lotes) de datos y usar alguna prueba estadística en las secuencias resultantes de la evaluación, calculando un valor de probabilidad bajo la hipótesis nula que estas secuencias corresponden a la misma distribución. Para reducir el sesgo introducido al seleccionar a f , ellos seleccionan una clase restringida de funciones lineales $f : x \to w^T x$, donde w es un vector de pesos con $\sum_j |w_j| = 1$. De esta forma ellos proponen tres pruebas estadísticas para calcular valores de probabilidad a partir de estas secuencias.

3.2.4 *Redes neuronales artificiales*

Las redes neuronales artificiales son un poderoso modelo computacional que puede representar cualquier relación no lineal entre los datos de entrada y el espacio objetivo. Las redes neuronales se han usado para resolver muchos problemas del mundo real incluyendo áreas del aprendizaje supervisado [Zha00]. Sin embargo, la mayoría de los modelos de redes neuronales requieren varias pasadas sobre los datos vistos, y su extensión a escenarios del aprendizaje incremental no es una tarea obvia [CGM+92]. Por ejemplo, algunos de los acercamientos más conocidos para manipular cambio de concepto básicamente usan un algoritmo de aprendizaje por lotes basados en una ventana deslizante [KW95], en ensamble de clasificadores [PUUH01] o en búsquedas heurísticas [Yao99, LCG10, RE12] con un alto coste computacional. Muchos de estos enfoques necesitan también fijar varios parámetros de entrada.

Uno de los sistemas más conocidos capaz de manipular cambio es FRANN (*Floating Rough Approximation in Neural Network*) [KW95]. Dados p ejemplos con n atributos, FRANN realiza un mapeo no lineal del espacio de instancias $\mathbb{R}^n$ a $\mathbb{R}^m$ de m neuronas ocultas de funciones de base radial. Usando una filosofía de ventana, el sistema toma cada uno de los p ejemplos en la ventana y trata de crear una neurona oculta quitando al ejemplo seleccionado. Se selecciona la neurona que provea mayor precisión sobre la ventana de ejemplos. El proceso termina cuando se han incluido m neuronas ocultas; el parámetro m es configurado a $k * 100$ por ciento del tamaño de la ventana ($0 < k \leq 1$; típicamente $k = 1/2, 1/3, 1/4$, etc. [KW95]). Cuando en cada paso son adicionados un determinado número de ejemplos a la ventana, la precisión en la clasificación se calcula en los últimos M ejemplos. Siendo t_1 y t_2 umbrales definidos por el usuario, e el número de errores en la clasificación y l el tamaño de ventana actual, si $l * e \leq t_1$ entonces se mantiene el tamaño de la ventana, si $l * e > t_1$ el tamaño de la ventana es reducido en un 20 %, de otra forma la ventana no olvida ejemplos antiguos.

Mientras que en FRANN se usa una búsqueda de escalador de colina con un operador (adicionar una neurona), muchos algoritmos usan algoritmos evolutivos para adaptar el modelo de aprendizaje a ambientes dinámicos [Yao99, RE12] y para evitar quedar atrapado en mínimos locales de la superficie del error [BHGH96] por medio de algoritmos de propagación hacia atrás. Sin embargo, es necesario precaución en el aprendizaje en flujos de datos ya que los acercamientos evolutivos son frecuentemente costosos computacionalmente.

Leite y otros [LCG10] proponen un marco de trabajo de redes neuronales difusas, particularmente redes granulares evolutivas. El aprendizaje en este caso contiene dos estados, primero la información de los gránulos (intervalos o más general conjuntos difusos) son construidos en la base de la representación numérica original. Luego, el aprendizaje en las redes neuronales está basado en la información de los gránulos y no en los datos originales. El modelo difuso aprende a partir del flujo de datos actualizando y creando gránulos en línea. La manipulación de cambio se realiza podando gránulos inactivos. Los pesos en la capa de decisión δ_i codifican la cantidad de datos asignados al gránulo γ_i. Comenzando por todos los pesos $\delta_i = 1$, si durante la evolución el gránulo γ_i no es activado en un determinado número de pasos entonces $\delta_i\,(new) = \zeta\delta_i\,(old)$. Si γ_i se activa entonces δ_i se incrementa mediante $\delta_i\,(new) = \delta_i\,(old) + \zeta\,(1 - \delta_i\,(old))$. Un umbral ϑ para δ_i y $\zeta \in [0, 1]$ deben ser fijados con anterioridad.

Rutkowski [Rut04] diseña redes neuronales probabilísticas para el aprendizaje en ambientes cambiantes en el tiempo. En particular, dada una secuencia de variables aleatorias independientes con densidad de probabilidades $f_n(x) = f(x - n^t)$, dichas redes neuronales son capaces de estimar densidades de probabilidades que varían en el tiempo a pesar que f y el parámetro t ($0 < t < 1$) sean desconocidos. La estimación de la función de densidad de probabilidades se realiza mediante dos técnicas no paramétricas: núcleos de Parzen y series ortogonales.

Como hemos visto, algunas investigaciones consideran solo algunos tipos de cambio de concepto (como el trabajo antes mencionado de Rutkowski [Rut04]), por ejemplo restringiendo la secuencia de densidad de probabilidades por la que se rigen los datos de entrada o restringiendo la secuencia de funciones objetivos. Los algoritmos de aprendizaje derivados de estas circunstancias generalmente es-

tán acompañados de garantías matemáticas de rendimiento. A continuación se muestra otro ejemplo.

Biehl y Schwarze [BS93] estudian el aprendizaje mediante un perceptrón con una capa. El perceptrón es entrenado para aprender una función objetivo $S_B(\xi) = \text{sign}(\sum_{j=1}^{N} B_j \xi_j)$ donde $B \in \mathbb{R}^N$ es normalizado a 1 y $\xi \in \mathbb{R}^N$ son vectores de entrada. Ellos consideran algoritmos semejantes a los de Hebb, los cuales usan cada ejemplo solo una vez. La función objetivo cambia sujeta aleatoriamente a $B^{\mu+1}B^{\mu} = 1 - \eta/N$ y $B^{\mu+1}B^{\mu+1} = 1$, el parámetro de cambio η determina el ángulo entre dos vectores consecutivos. El error de generalización de un perceptrón simple se determina por el ángulo entre el vector de parámetros en la red (J) y el vector B, y está dado por la expresión $\epsilon_g = \cos^{-1}(\rho)/\pi$ donde $\rho = JB/\sqrt{JJ}$. En este sentido, ellos investigan el rendimiento de varios algoritmos semejantes a los algoritmos de Hebb.

3.2.5 *Aprendizaje basado en instancias*

Contrario a los algoritmos de aprendizaje que inducen un modelo computacional a partir de los datos de entrada y usan este modelo para la ulterior predicción, el aprendizaje basado en instancias almacena las propias instancias. Los algoritmos basados en instancias también son llamados métodos de aprendizaje perezosos porque posponen el procesamiento de los datos hasta que se requiera una predicción [Aha97]. Con mayor frecuencia la predicción es llevada a cabo por medio del principio de vecinos más cercanos [Das90], asumiendo que instancias similares tienen clasificaciones semejantes [AKA91]. Los algoritmos basados en instancias son inherentemente incrementales y la adaptación es simple comparada con la mayoría de los algoritmos de aprendizaje basados en modelos, donde la actualización de este modelo de aprendizaje es por lo general más compleja y requiere de cantidades adicionales de información.

Sin embargo, los algoritmos basados en instancias tienen un coste computacional alto en las tareas de predicción. Si las predicciones deben ser hechas frecuentemente bajo restricciones de tiempo ajustadas, y la actualización del modelo debido a nuevos ejemplos ocurre con poca frecuencia, los métodos basados en modelos son una mejor opción [BH07].

Posiblemente el primer algoritmo de aprendizaje basado en instancias capaz de manipular cambio de concepto en el aprendizaje supervisado fue IB3 [AKA91]. Luego surgió la familia de algoritmo IB*n* (*Instance Based learning algorithm*) como IB-C1 [Aha91] que usa ideas de IB3 y STAGGER [SG86].

Existen tres componentes fundamentales que describen un algoritmo basado en instancias [AKA91]: función de similitud, función de clasificación y actualizador de la descripción de conceptos. En la manipulación de cambio de concepto generalmente interviene el actualizador de la descripción de conceptos, en el que se definen mecanismos para incorporar incrementalmente nuevas instancias (casos) a la base de casos y para identificar y eliminar casos que dejan de ser significativos para el concepto actual. Por ejemplo, IB3 adiciona una nueva instancia a la descripción de conceptos si esta es clasificada erróneamente. IB3 mantiene almacenadas las clasificaciones realizadas por cada instancia. Las instancias almacenadas cuya precisión es significativamente menor que la frecuencia observada de su clase

correspondiente (asumiendo una distribución de probabilidades binomial) es eliminada de la descripción de conceptos. Aunque IB3 es capaz de manipular cambio de concepto, su adaptación es relativamente lenta [Sal97].

Un acercamiento de selección de instancias que sigue esta estrategia fue propuesto más recientemente por Delany y otros [DCTC05]. Ellos actualizan la base de casos con una técnica llamada edición basada en competencia [DC04] para eliminar ruido y casos redundantes (aplicada en el dominio de filtrado de correos spam). Esta técnica inicialmente construye un modelo de competencia de la base de casos, identificando para cada caso sus propiedades de competencia, que incluye aquellos casos que contribuyen a la clasificación correcta y aquellos que contribuyen a un fallo en la clasificación.

En general, varios acercamientos construyen un modelo de competencia para determinar cuáles instancias mantener en la base de casos, como consecuencia estos acercamientos también pueden adaptarse al cambio de concepto. Por ejemplo, eliminando aquellos casos c, donde el número de otros casos que clasifican correctamente a c es mayor que el número de casos que c puede clasificar correctamente (eliminando casos lejos de las fronteras de clase [BM02]); eliminando casos c si como mínimo la cantidad de casos asociados a él pueden ser clasificados correctamente sin c (eliminando casos ruidosos y casos en el centro de aglomeraciones de casos [WM97]).

Otros acercamientos conocidos propuestos por Salganicoff [Sal97] basados en pesado de instancias son LWF (*Locally Weighted Forgettinh*) y PECS (*Prediction Error Context Switching*).

A la llegada de cada nueva instancia x_0, LWF reduce el peso de los k vecinos más cercanos x_i $(1 \leq i \leq k)$ acorde a la función de similitud Δ con respecto a x_0 por el factor $\tau + (1-\tau)\,\Delta(x_i; x_0)^2 / \Delta(x_k; x_0)^2$, donde x_k es el vecino más distante. Se elimina una instancia si su peso está por debajo de un umbral θ. El parámetro k se define de forma adaptativa por medio de $k = \lceil \beta |D| \rceil$, donde $|D|$ es el tamaño de la base de casos actual. Salganicoff [Sal97] tambien considera al algoritmo TWF (*Time Weighted Forgetting*) que pesa instancias acorde a su edad: en cada punto de tiempo t, el ejemplo observado al tiempo $t-k$ es pesado por w^k, donde $w \in (0,1)$ es una constante.

En vez de eliminar instancias, PECS solo las desactiva de forma que estas instancias pueden ser reactivadas más tarde. Esta estrategia puede evitar algunas desventajas [Sal97] de LWF pero los requerimientos de almacenamiento excluyen a PECS del procesamiento de flujo de datos [BH07].

Aunque los métodos mencionados se adaptan al cambio en ambientes dinámicos, estos no detectan el cambio explícitamente. Manipular cambio de concepto por medio de detectores de cambio de concepto es un método prominente. Por ejemplo, en este caso encontramos el acercamiento propuesto por Beringer y Hüllermeier [BH07]. En este trabajo los autores proponen el algoritmo IBL-DS que manipula cambios abruptos usando una prueba estadística semejante a la propuesta de Gama y otros [GMCR04]. Si se detecta un cambio, un número grande de ejemplos es eliminado instantáneamente de la base de casos. El mantenimiento de la base de casos también es llevada a cabo insertando cada instancia que llega. Luego se chequea si en la vecindad de esta instancia deben ser eliminados algunos ejemplos, ya sea porque estos se han convertido redundantes o porque son datos ruidosos.

Para hacer esto, el algoritmo crea dos conjuntos, uno de ellos (S) contiene los k vecinos más cercanos y el otro (T) los $k(k+1)$ vecinos más cercanos de la nueva instancia. Si la clase estimada por S es la más frecuente de las k instancias más recientes de T, aquellas instancias en S que tienen una etiqueta de clase diferente y que no están en las k instancias más recientes del conjunto T son eliminadas.

3.2.6 *Ensamble de clasificadores*

Los métodos de ensamble (sistemas de múltiples clasificadores) han recibido en los últimos tiempos una gran atención para el modelado y la clasificación de flujos de datos. En general, y aunque dentro de un dominio incremental, las nuevas propuestas siguen el mismo esquema que las técnicas de ensamble aplicadas en el aprendizaje por lotes. Estas se basan 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. Existen una gran variedad de acercamientos, ya que la idea de combinar muchos clasificadores provee de una buena arquitectura para manipular los problemas que surgen cuando se aprende con cambio de concepto.

SEA [SK01] fue una de las primeras propuestas, este usa una cantidad de memoria fija independiente del número de ejemplos y asegura un tiempo máximo para el procesamiento de cada instancia. SEA usa una ventana de tamaño fijo donde todos los ejemplos son reemplazados en bloques consecutivos. Bajo el mismo esquema de ventana, Wang y otros [WFYH03] proponen un método basado en el pesado de clasificadores en términos de su precisión. Como clasificadores bases se han usado diferentes tipos de algoritmos: C4.5, RIPPER, Naïve Bayes, etc. Otro método también inspirado por SEA es MultiCIDIM-DS [del07] que usa el algoritmo CIDIM (*Control de Inducción por División Muestreal*) [RdCM05] para inducir clasificadores base. La principal diferencia entre estos es la definición de una función más simple que mide la calidad de los clasificadores y que usa dicha función para eliminar clasificadores con baja calidad.

El algoritmo SEA aprende a partir de lotes de ejemplos de los datos originales, pero este método de procesamiento tiene algunos problemas. Por ejemplo, el proceso de aprendizaje no puede ser aplicado con pocos ejemplos en el lote y es necesario completar un lote antes del aprendizaje. Sun y otros [SMLL07] introducen algunas ideas para superar estos problemas en el algoritmo ICEA (*Incremental Classification Ensemble Algorithm*). Este es diseñado para que cada clasificador base puede aprender incrementalmente y de forma automática. Como resultado, se obtiene una detección de cambios de concepto más rápida comparable con algoritmos basados en lotes.

Kolter y Maloof [KM07] proponen DWM, basado en el algoritmo de pesado por mayoría [LW94] para ensamblar clasificadores de diferentes edades. Este algoritmo es capaz de cambiar dinámicamente los expertos menos adecuados en el ensamble, siendo capaz de esta forma de manipular cambios de concepto. Littlestone [LW94] realiza una comparación entre dos expertos diferentes para el algoritmo DWM, una con árboles de decisión y con el clasificador Naïve Bayes, así como otra con el algoritmo de pesado por mayoría de Blum [Blu97]. La idea de usar un pesado dinámico para destacar el clasificador básico más relevante constituye un

acercamiento que atrae mucha atención. Por ejemplo, Grossi y Turini [GT12] proponen dos variantes de ensamble (SE y FE) que combinan lo que ellos denominan modelos activos. Otro algoritmo que usa un voto por mayoría de pesos dinámicos es Learn++.NSE [EP11]. Este aprende incrementalmente a través de la creación de un nuevo clasificador base por cada nuevo lote de datos recibidos. El mismo es diseñado para trabajar con diferentes tipos de cambio: tasas de cambio constantes o variables, adición o eliminación de clases de concepto, cambios cíclicos, etc.

Otros métodos de aprendizaje conocidos basados en ensamble usados cuando los conceptos son estables son *bagging* y *boosting*, aunque estos también han sido adaptados para aprender con la presencia de cambio de concepto. Así Oza [OR01] propone algoritmos de bagging y boosting en línea. Scholz y Klinkenberg [SK07] también adaptan el algoritmo de boosting para aprender con cambio de concepto y cuantifican la extensión del cambio usando la información de los clasificadores base.

Bonissone y otros [BCG+10] desarrollan otra adaptación a bagging, boosting y a los llamados bosques aleatorios (*random forest*). En este caso, ellos usan árboles de decisión difusos como clasificadores base. La justificación de la mezcla de árboles de decisión con la lógica difusa está dada por el hecho de que los árboles de decisión son interpretables, eficientes y capaces de trabajar con grandes conjuntos de datos; pero son muy inestables cuando se introducen pequeñas cantidades de ruido en los datos de entrenamiento, lo cual es mejorado a través del uso de la lógica difusa. Esta integración preserva las ventajas de los dos componentes: el tratamiento de incertidumbre con variables lingüísticas de la lógica difusa que son comprensibles, y la popularidad y aplicación sencilla de árboles de decisión.

3.3 EVALUACIÓN DE ALGORITMOS INCREMENTALES

Un aspecto muy importante en el campo de la minería de datos es la evaluación de los métodos y algoritmos con el objetivo de validar su rendimiento. Al mismo tiempo, los procesos de evaluación son útiles para evaluar la aplicabilidad de estos métodos, para detectar puntos en los que pueden ser mejorados o incluso para comparar diferentes alternativas. Por las mismas razones, estas metodologías también son necesarias cuando el proceso de aprendizaje se realiza en presencia de cambio de concepto. Debido a las propiedades particulares de este contexto y la relativa novedad de esta área, las estrategias tradicionales no son válidas [Bif10] y son necesarias varias modificaciones y mejoras.

Aunque algunos métodos incluyen garantías matemáticas y cotas cuando son presentados [HL94, BH96], estas usualmente se refieren al rendimiento en el peor de los casos [FS97, BG07, Žli09a] o incluso calculados en relación con el rendimiento actual del clasificador base [KM05]. Adicionalmente, las suposiciones usadas para calcular esas cotas son en muchos casos muy restrictivas o no reales, por lo que es difícil determinar si están presentes en el problema en estudio (o conjunto de datos), el que puede incluir ruido y sesgo en el muestreo. Debido a estos inconvenientes, es difícil encontrar conclusiones significativas relacionadas con el rendimiento exclusivamente en términos de cotas. Las técnicas de evaluación ofrecen una aproximación diferente que puede ser aplicada a una amplia variedad de casos, independientemente de la naturaleza especial del problema.

	1985-1995	1996-2000	2001-2005	2006-presente
SVM		Klinkenberg *et al.* [KJ00]	Rüping [Rüp01]	Dries *et at.* [DR09]
Reglas de Decisión	STAGGER [SG86]	FLORA [WK96]	AQ11-PM [MM04] FACIL [FAR05]	CVFDR [KG12]
Árboles de Decisión			CVFDT [HSD01] UFFT [GMR04] Natwichai *et al.* [NL04]	OnlineTree2 [NFM07] Cheng *et al.* [CCW08]
Redes neuronales	Biehl *et al.* [BS93] FRANN [KW95]	Blum [Blu97] Yao [Yao99]	Rutkowski [Rut04]	Leite *et al.* [LCG10]
Basado en casos	IB3 [AKA91] IB-C1 [Aha91]	LWF [Sal97] PECS [Sal97]	ECUE [DC04, DCTC05]	ECUE [DB07] IBL-DS [BH07]
Ensamble			SEA [SK01] Wang *et al.* [WFYH03] CBEA [RGCL04] Online Bagging [OR01] Online Boosting [OR01]	DWM [KM07] MultiCIDIM-DS [del07] ICEA [SMLL07] KBS-Stream [SK07] Learn++.NSE [EP11] Grossi and Turini [GT12]

Tabla 3.2: Estrategias generales para la detección de cambio de concepto considerando el modelo de aprendizaje.

En esta sección presentamos las metodologías de evaluación usadas hasta el momento en el aprendizaje incremental con cambio de concepto, analizando la evolución de los primeros acercamientos y proponiendo algunas características que pueden ser de beneficio.

3.3.1 *Métricas*

Para analizar el rendimiento de algoritmos que llevan a cago tareas de clasificación en un contexto donde los conceptos pueden cambiar, las principales dimensiones a evaluar son aquellas relacionadas con el poder de generalización del modelo inducido y los requerimientos de cómputo para en los que el algoritmo debe ejecutarse (tiempo y memoria) [Bif10].

La primera dimensión, poder de generalización, usualmente se mide por medio de la precisión (*accuracy*) [Har99] (o error [GMCR04]) producido en un escenario de predicción, aunque la precisión y la rememoración (*recall*) están tomando mayor presencia [Kli04]. Incluso algunas medidas basadas en las anteriores, como el área bajo la curva de precisión-rememoración (AUC-PR) [DR09], el área bajo la curva de la característica operativa receptora (AUC) [ZJYH11], o la medida *F-score* [AF10], se han usado para evaluar el rendimiento. Actualmente es común encontrar estudios experimentales que analicen precisión y rememoración [KTV10].

Por otro lado, el coste de actualizar el modelo y los requerimientos computacionales del método, medidos en tiempo y espacio, son más importantes en este contexto ya que el rendimiento del sistema es un aspecto esencial, y el tiempo y el espacio están limitados debido a la gran cantidad de datos y a su tasa de llegada. La relevancia de estos aspectos se considera en los primeros estudios [SF86], donde los requerimientos de memoria son medidos usando el número de elementos necesitados para definir el concepto. Obviamente, la intensidad del proceso de evaluación se ha incrementado y en la actualidad son estudiados más detalles. Desde el punto de vista de los requerimientos de memoria, es usual calcular la memoria como el número de componentes que forman el modelo y que describen el concepto aprendido (número de reglas en el modelo [FAR05], clasificador base en un ensamble[KTV10], etc.) aunque esta puede no ser la mejor aproximación ya que diferentes componentes pueden usar distintas cantidades de memoria. En base a esta fuente de heterogeneidad, algunos trabajos recientes ejecutan los algoritmos bajo un límite máximo de memoria, detectando así cuando un algoritmo excede la cota máxima de memoria [JA03, DH00] o incluso mejor, es posible monitorizar el tamaño real de la memoria usada, expresada de una forma absoluta (MB) [Brz10] o relativa al tamaño máximo permitido [AF10].

Relacionado con el tiempo de ejecución, se pueden monitorizar diferentes niveles especificación. De esta forma, se pueden considerar resúmenes donde solo se obtiene el tiempo global (fase de entrenamiento [WFYH03], mezclando las fases de entrenamiento y prueba [KTV10]) o descripciones detalladas, donde se puede apreciar como se incrementa el tiempo acumulado cuando nuevas instancias son procesadas por el modelo [Brz10]. Adicionalmente, se pueden encontrar trabajos donde se calcula el tiempo de ejecución necesario para entrenar el modelo con un solo ejemplo [AF10].

Con el objetivo de complementar las métricas anteriores, frecuentemente se incluyen medidas adicionales. Algunas de estas son estudiadas generalmente en el contexto del aprendizaje automático. Así, aunque pudieran existir pequeñas diferencias por la presencia de cambio de concepto, se pueden considerar métricas que prueban la resistencia al ruido (ya sea fijo [SK01, WFYH03] o variable [JA03, BH99]), la robustez en presencia de atributos irrelevantes [KW95], la sensibilidad en el orden en que se presentan los ejemplos de entrenamiento [SF86], la complejidad del modelo inducido [FAR05] o más usualmente la combinación de algunos de estos [WK96, MWY10, KW95].

Algunas otras características relacionadas con la detección del cambio lógicamente también son relevantes. Específicamente, se puede estudiar el rendimiento de los métodos bajo diferentes suposiciones para los cuales los algoritmos son específicamente diseñados. En este grupo, se pueden considerar métricas que: *(a)* cuentan el número de ejemplos necesarios para estabilizar el proceso de aprendizaje [SF86], o para recobrarse de una fase de empeoramiento del rendimiento [SLH+99]; *(b)* evaluar la respuesta de los métodos atendiendo a diferentes tipos de cambio (abrupto o gradual) con diferentes niveles de la extensión del cambio [WK93, WFYH03]; *(c)* probar las ventajas de adaptar el modelo cuando los conceptos son recurrentes [RB07]; y *(d)* monitorizar el tiempo necesario para detectar cambio, complementado con las medidas de falsos positivos y falsos negativos en dicha detección, con el objetivo de evaluar la sensibilidad y especificidad de los detectores de cambio[BdF+06, Bif10, Gam10].

3.3.2 *Metodología*

En general, todas las métricas expuestas previamente pueden ser calculadas usando distintos procedimientos. En este punto, presentamos los más extendidos a partir de las primeras ideas usadas hasta los mecanismos más avanzados propuestos recientemente.

Una de las formas de calcular las métricas es solo considerar un punto de medición, al final del proceso de aprendizaje. Este acercamiento es más frecuente cuando se aprende de conjuntos de datos reales donde la información puede ser insuficiente [GC06]. Sin embargo, como puede ser deducido, este acercamiento no muestra cómo el proceso de aprendizaje evoluciona en el tiempo, que es un aspecto más relevante que la puntual (y final) conducta. De esta forma, es mucho más informativo presentar curvas de aprendizaje expresando el rendimiento de diferentes métricas en línea [KW95, BH99] a medida que los ejemplos son procesados. Un punto interesante es cómo las métricas individuales que forman la curva de aprendizaje son calculadas con una perspectiva secuencial, estos es, qué configuración usar a cada paso de aprendizaje para calcular los valores deseados (precisión, memoria, etc.). En este sentido existen dos acercamientos fundamentales [Bif10, GC06], uno que separa ejemplos de entrenamiento y de prueba (*holdout*) y otro que utiliza el mismo conjunto para ambas tareas (*test-then-train*).

El primer acercamiento (*holdout*) fija, en un paso preliminar, uno o más conjuntos de prueba que contienen ejemplos que solamente serán usados para evaluar el algoritmo de aprendizaje. Es posible usar solo un conjunto de prueba [SLH+99], pero esta configuración no parece ser la más apropiada porque con un solo conjun-

to de prueba no podemos saber cuál es el concepto evaluado. Es más conveniente utilizar diferentes conjuntos de prueba que serán usados en diferentes períodos [WK96, SK01, TPCP08]. Si los conceptos y los períodos de cambio son desconocidos (como es usual en conjunto de datos reales), la selección del conjunto de prueba no puede ser guiada precisamente, pero las especificaciones del proceso serán mayores que cuando sólo se usa un conjunto de entrenamiento. Por otro lado, si el concepto y los períodos de cambio son conocidos (lo que es posible en los conjuntos de datos artificiales), la creación de conjuntos de prueba es inmediata, y también es conocido cuáles conjuntos de prueba en cada período corresponde al concepto actual.

El segundo acercamiento (*test-then-train*), también conocido como *prequential* (formado de *pre*dictive se*quential*), se deriva a partir del error predictivo secuencial [Daw84]. Este consiste básicamente en calcular las medidas de interés (usualmente la precisión) a la llegada de cada ejemplo (paso de prueba); luego, el ejemplo es utilizado por el método de aprendizaje para continuar con su entrenamiento (paso de entrenamiento) [HSD01]. Esta metodología está basada en la suma acumulada de los valores de una función dada. De esta forma el valor del error predictivo secuencial se convierte pesimista a medida que más ejemplos son vistos, debido a que no hay mecanismo de olvido y el rendimiento actual es diluido en las medidas pasadas de rendimiento. La primera variación sugerida para mejorar esta deficiencia fue el uso de ventanas deslizantes. Así, el error predictivo secuencial se calcula considerando solo los últimos ejemplos [GRC09], olvidando contextos más antiguos y ofreciendo una estimación que refleja mejor el estado actual. Una desventaja del uso de las ventanas deslizantes es la determinación del tamaño de ventana, entonces surge otra aproximación para mantener el mecanismo de olvido: factores de desvanecimiento [GSR09] o uso del estadístico EWMA. Con este método, una función de decremento de pesos se aplica y la importancia (o peso) de los valores vistos anteriormente decrece, dando más importancia a los valores más recientes. Adicionalmente, el uso de este método es más eficiente ya que no requiere almacenar ningún ejemplo (ver Figura 3.1).

Considerando algunas métricas, se ha mostrado [GSR09] que la precisión predictiva secuencial (o el error) calculado con mecanismos de olvido (con ventanas deslizantes o funciones de decremento de pesos) converge a la medida calculada con un acercamiento *holdout*, y que el tiempo para la detección de cambio es menor cuando se usan funciones de decremento de pesos en la prueba Page-Hinkley, manteniendo la robustez cuando no existen cambios.

El método predictivo secuencial considera un paso de evaluación siempre que se recibe un ejemplo, y acumula el resultado (con o sin mecanismo de olvido) teniendo en cuenta el último ejemplo. Existe una variación del método predictivo secuencial que usa bloques de ejemplos para realizar la prueba [Brz10]. En orden de conectar estas dos alternativas, se puede ver el método predictivo secuencial como un caso particular de la variación mencionada, donde los bloques son limitados a un solo ejemplo. Esta generalización del método predictivo secuencial no debe ser confundida con el acercamiento *holdout*, donde los conjuntos de prueba son usados en diferentes intervalos [FAR05]. Esto es debido al hecho que en el último acercamiento los conjuntos de prueba son fijados con anterioridad y más importante, estos solo son usados posteriormente en la fase de entrenamiento.

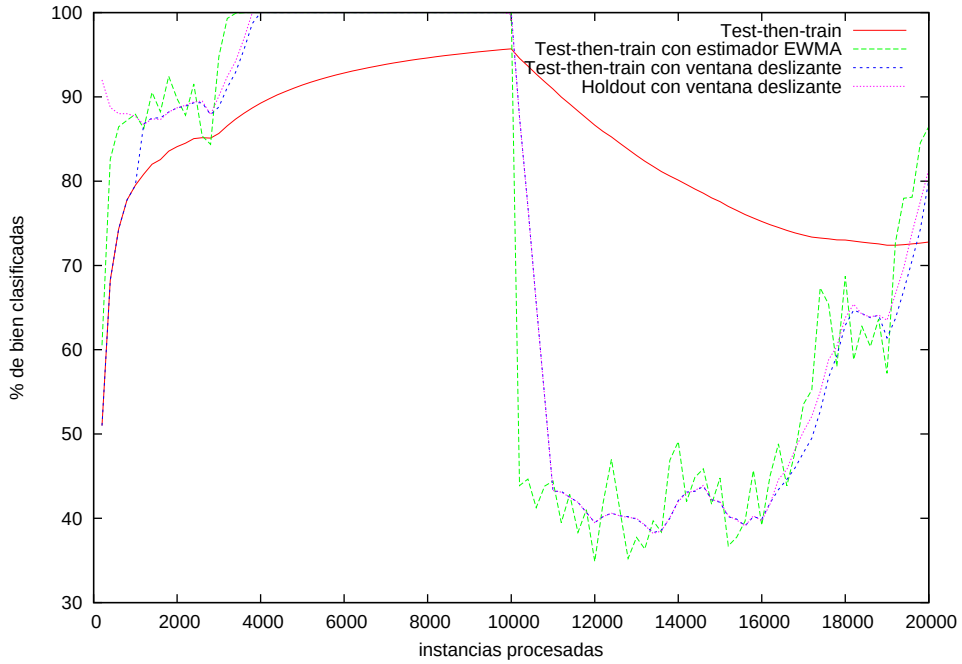


Figura 3.1: Diferentes métodos de evaluación aplicados a la estimación de la precisión del algoritmo IADEM-2 aprendiendo en conceptos STAGGER. Se simula un cambio de concepto a partir del ejemplo número 10 000. Las ventanas deslizantes tienen un tamaño de 1 000 ejemplos.

Otro punto interesante que diferencia las metodologías de evaluación en este contexto con respecto a las metodologías tradicionales es el soporte estadístico que puede ser garantizado. En el aprendizaje tradicional por lotes, donde los conjuntos de datos son finitos y se asume que los ejemplos son independientes, muchas técnicas (validación cruzada, *bootstrapping*, etc.) pueden ser usadas para obtener algún soporte estadístico de los resultados. Pero en la presencia de cambio de concepto, donde los conjuntos de datos pueden ser además infinitos, el contexto puede cambiar y los ejemplos pueden presentar correlaciones con respecto al tiempo, por lo que no es posible el uso de estas técnicas anteriores para obtener el mismo resultado [Gam10]. Algunos diseños de experimentos usan acercamientos inspirados por repetición [KW95, WK96] y validación cruzada [SLH+99] pero, configurados en su forma original, no es posible realizar alguna prueba estadística que pueda asegurar las diferencias significativas deseadas. De esta forma, nuevas estrategias de evaluación deben ser desarrolladas.

3.3.3 *Conjuntos de datos*

El desarrollo de las metodologías previas para evaluar las métricas deseadas, solo es posible si son aplicadas a conjuntos de datos. Consideraremos dos principales grupos de conjuntos de datos, aquellos que son generados artificialmente (ya sea basados en un problema real o no) y aquellos conjuntos de datos estrictamente del mundo real. Cada uno de ellos tiene sus propias ventajas. Los conjuntos de datos artificiales permiten probar los métodos bajo situaciones controladas, simulando todos los aspectos que los algoritmos deben superar en el futuro (requerimientos

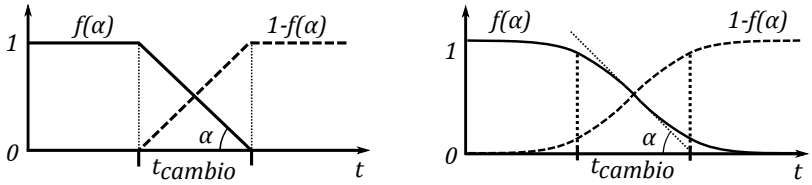


Figura 3.2: Combinación de dos conceptos diferentes para simular un cambio gradual discreto: (a) usando una función con pendiente y (b) usando una función sigmoide.

de tiempo y espacio, resistencia al ruido, etc.) y enfocándose en las características relacionadas con el cambio de concepto (el tiempo necesario para detectar cambios, su conducta al enfrentar diferentes tipos de cambio, etc.). Los conjuntos de datos reales permiten extender el proceso de aprendizaje a situaciones reales para las cuales los algoritmos son diseñados.

Aunque la mayoría de los conjuntos de datos para monitorizar el rendimiento de los algoritmos de aprendizaje son dinámicas, es usual empezar el proceso de aprendizaje con algún conjunto de datos grande y estático donde no se incluya cambio de concepto [FAR05, KM07]. El principal objetivo es asegurarse que el algoritmo de aprendizaje es estable en la ausencia de cambio, aparte de la comparación que se puede establecer con algunos algoritmos tradicionales. Una de las fuentes de conjuntos de datos más usada para esta tarea es el repositorio de la UCI (*University of California Irvine*) [FA10].

Regresando a los conjuntos de datos dinámicos, a continuación presentamos una lista de algunos de los más usados para ejecutar el proceso de evaluación.

3.3.3.1 *Conjuntos de datos artificiales*

Los conjuntos de datos que son generados de forma artificial tienen el beneficio de modelar diferentes escenario donde los algoritmos pueden demostrar su rendimiento. Podemos distinguir dos tipos de conjuntos de datos, aquellos que son orientados a probar cambios abruptos y aquellos que introducen cambios graduales. Adicionalmente pueden ser añadidas otras características como ruido artificial, atributos irrelevantes, etc.

Relacionado con estos conjuntos de datos donde se introducen cambios abruptos, una idea común que soporta sus operaciones es la generación de conceptos distintos activos en diferentes períodos. El cambio entre conceptos es inmediato (cambio abrupto) aunque la extensión del cambio puede simular un cambio gradual de alguna forma. Así, si dos conceptos consecutivos son muy similares, se podría considerar que la velocidad del cambio es lenta. De cualquier forma, consideraremos que la activación de un nuevo concepto y la desactivación del concepto previo produce un cambio abrupto. Existen muchos conjuntos de datos que operan de esta forma.

Uno de los más populares es el generador de conceptos STAGGER, propuesto por Schlimmer y Granger [SF86], el cual es usado con frecuencia con la misma configuración [KW95, GMCR04, KM07]. Este generador crea ejemplos definidos

por atributos nominales (forma, color y tamaño) y una clase binaria, usando tres funciones booleanas donde cada una de ellas corresponde a un concepto diferente. Estos conceptos son fijados y tienen un particular solapamiento, pero la idea puede ser extendida para obtener otras funciones que determinen más conceptos [MWY10].

Tomando a los conceptos STAGGER como un punto de partida, es fácil generalizar el proceso de definir conceptos diferentes y más complejos. Así, Widmer y Kubat [WK96] incluyen modificaciones simples para tratar con conceptos recurrentes o ruido, pero ellos proponen también un generador basado en funciones booleanas que combina más de tres atributos nominales.

En este sentido, LED [BHPG09] puede ser considerado como un generador que también usa funciones booleanas, ya que este modela la combinación de atributos (segmentos individuales en una pantalla digital de siete segmentos) que tienen que ser activados en forma de dígito (diez etiquetas de clase). Este generador es una variante de la versión original propuesta por Breiman y otros [BFOS84] donde el patrón de activación de segmentos individuales es estable (sin cambio de concepto). Bifet y otros [BHPG09] definen diferentes patrones de activación para segmentos individuales (para ser usadas en diferentes períodos) por medio del intercambio de atributos que forman parte de los siete segmentos de la pantalla por otros atributos que son irrelevantes. Este generador es usado con mucha frecuencia [GK09, Brz10].

Los generadores vistos hasta el momento solo incluyen atributos nominales, pero como es de esperar, también es posible incluir atributos numéricos. Las funciones descritas por Agrawal y otros [AIS93] para determinar cual préstamo debe ser aprobado pueden ser usadas para generar diferentes conceptos. Cada función modela un concepto diferente y se puede simular nuevamente cambios abruptos considerando una de ellas en diferentes períodos de tiempo [BHPG09]. Una aproximación similar es presentada por Kubat y Widmer [KW95] en el generador mixto (MIXED), donde son usados dos atributos booleanos y dos numéricos para describir el problema.

Como hemos mostrado, las funciones booleanas pueden ser usadas fácilmente y adaptadas para generar problemas artificiales con atributos nominales. Cuando los atributos son nominales exclusivamente, pueden ser usados otros tipos de funciones. Aparte de la definición del generador mixto, Kubat y Widmer [KW95] describen otro tipo de funciones como SINE, CIRCLES y GAUSSIAN [KW95], que usan funciones matemáticas y estadísticas para determinar los conceptos. Todos estos generadores producen problemas con dos valores de clases (positivos o negativos) y dos formas diferentes de simular cambios abruptos: invirtiendo la clasificación, donde los ejemplos positivos se convierten en negativos y viceversa (SINE y GAUSSIAN); o definiendo conceptos diferentes, incluso si ellos se solapan (CIRCLES) y el cambio pueda parecer gradual. Siguiendo esta idea, se pueden combinar mas funciones generadoras [Hoo5].

Aunque estos generadores son relativamente simples, permiten probar los algoritmos desde otra perspectiva que propicia su uso [GMCR04, BdF+06]. Otro generador ampliamente usado es SEA, presentado por Street y Kim [SK01]. El valor de la clase binaria que es asignada a los ejemplos generados depende de una función que chequea si la suma de dos atributos numéricos excede un umbral determinado. Pueden ser simulados varios conceptos cambiando el nivel del umbral. Esta idea

es fácilmente reproducida [GRC09] y puede ser extendida para la evaluación del rendimiento de los algoritmos en presencia de ruido [KM07, GC06].

Similar al generador LED que fue modificado para incluir cambio de concepto. El generador de formas de ondas (*waveform*) [BFOS84] también ha sido adaptado [BHPG09]. En este caso, el generador resultante usa atributos numéricos y diferencia entre tres clases, cada una de las cuales es generada a partir de una combinación de dos o tres funciones base con formas de ondulación. La forma de simular cambio es similar al método usado con el generador LED, este intercambia el papel que desempeñan los atributos en la descripción del problema: algunos atributos relevantes son intercambiados por atributos irrelevantes y viceversa.

Todos los generadores presentados hasta el momento simulan tipos de cambio abruptos. El cambio puede ser suavizado si la extensión del cambio es pequeña, pero existen mejores formas de simular cambios graduales. El acercamiento más usado y adaptado es el propuesto por Widmer y Kubat [WK96]. En vez de producir cambio entre dos conceptos inmediatamente, este acercamiento combina los conceptos por medio de su ponderación durante un período de tiempo. Así, antes de que el cambio comience, el concepto actual tiene una presencia total (su peso es 1), mientras que el concepto final no tiene presencia (su peso es 0). Existe una etapa de transición entre conceptos (t_{cambio}) que termina con la configuración opuesta (el concepto antiguo no tiene presencia y solo está presente el concepto final). El método más extendido propone un cambio uniforme caracterizado por la pendiente de una función (α), mientras la presencia de un concepto previo decrece, la presencia del segundo concepto aumenta ($1 - \alpha$). La Figura 3.2a muestra este escenario. Se puede observar cómo la presencia de ambos conceptos siempre representa el 100 % del conjunto de datos resultante. Esta propuesta es muy conveniente cuando se necesita la simulación de cambios graduales, e incluso más cuando el cambio deseado debe ser muy lento [BdF+06]. Para suavizar el extremo de la función previa se propone la función sigmoide [BHPG09] (ver Figura 3.2b). Note que la combinación de ambos conceptos se mantiene igual al 100 %.

Adicionalmente se han definido algunas notaciones y propiedades para describir cuán diferentes son los conceptos cuando se combinan en fases de cambio usando la función sigmoide [BHPG09]. Una notación equivalente puede extrapolarse para la función con pendiente. El caso usual considera solamente dos conceptos, pero otras propuestas recientes [NK07] sugieren el incremento de este número para combinar más de dos conceptos. Otro acercamiento para simular diferentes escenarios en la transición de conceptos (desde estocásticos hasta deterministas) es hecha por Yang y otros [YWZ06]. Ellos definen una matriz de transición basada en la distribución Zipfian, donde la frecuencia de ocurrencia del elemento n-ésimo es $Zipf(N) = 1/n^z$ ($z \geq 0$ es un parámetro que controla la oblicuidad de la distribución). Después de transformar el problema por medio de cadenas de Markov y mecanismos de ordenamiento, y tomando en consideración la matriz de transición definida con la función Zipfian, ellos generan secuencias de ejemplos combinando los conceptos.

En adición a estas propuestas para simular cambios de concepto graduales desde conceptos separados (con mayor o menor solapamiento), existen otros generadores que logran la misma conducta. Ellos están definidos para problemas con atributos numéricos. El generador más conocido es el hiperplano rotante usado por Hulten

y otros [HSD01]. Este es definido para d atributos (dimensiones), donde cada uno de ellos (x_i) tiene un vector de pesos asociado (w_i) con el que determina su importancia en la fórmula ($\sum_{i=1}^{d} w_i x_i = w_0$). El cambio gradual puede ser simulado fácilmente variando los pesos de una forma leve. Este generador ha recibido mucho interés y los elementos que determinan el tipo de cambio (número de atributos, extensión del cambio, etc.) fueron rápidamente parametrizados [WFYH03, FAR05].

Otro generador que puede crear conjuntos de datos con cambio gradual está basado en funciones de base radial (RBF) [BHPG09]. El mismo genera un número diferente de centroides que son usados para definir funciones de base radial. Cada centroide tiene una clase asociada a él, entonces el número máximo de clases en este problema está limitado por el número de centroides (podría ser menor si algunos centroides tienen asociada la misma etiqueta de clase). La generación de ejemplos lo guían hiperesferas distribuidas normalmente alrededor de cada centroide. El cambio de concepto es simulado moviendo los centroides de una forma suave. Otro generador propuesto está definido sobre el generador estático NDC[1] [Mus98] propuesto por Ho y Wechsler [Ho05]. Este puede ser considerado como un caso particular del generador RBF porque usa un hiperplano para asignar dos etiquetas a los centroides, aunque el cambio de concepto puede ser simulado de una forma diferente.

Hasta ahora hemos mostrado los generadores más importantes, destacando sus detalles más importantes y explicando sus principales características y alcance. En la Tabla 3.3 ofrecemos un resumen, agrupando estos generadores por sus características más relevantes.

Los generadores de conjuntos de datos previos pueden ser adaptados para adicionarles nuevas características, lo que es importante para diferentes métodos y algoritmos. Algunos de ellos están diseñados para tratar con diferentes grados de ruido, o para asumir varios niveles de cambio, por lo que es importante comprobar cómo ellos son afectados cuando ocurren tales situaciones. Por ejemplo, existen alternativas para extender los conjuntos de datos con atributos irrelevantes. Así, la asignación de valores aleatorios a los nuevos atributos que no influyen en el concepto es una opción para su extensión [KW95, WK96]. Otra opción es asignar pesos a cada atributo y configurar a cero el peso de los atributos irrelevantes [HSD01]. Si el objetivo es adicionar ruido, se puede cambiar la etiqueta de clase de los ejemplos de forma aleatoria e independientemente [WK96, HSD01, GC06], o seleccionar aleatoriamente la etiqueta de clase si existiera una mezcla de conceptos [KW95]. La extensión del cambio puede variar también, dependiendo del acercamiento seleccionado. Podemos encontrar definiciones de conceptos que cambia a su opuesto, como el caso de SINE y GAUSSIAN, de conceptos que son fijos y se solapan parcialmente, así como STAGGER o SEA, cuya extensión del cambio puede ser calibrada. Teniendo en cuenta la calibración, pueden ser consideradas las siguientes acciones: selección guiada de conceptos (funciones booleanas), variación de los pesos de los atributos (hiperplano rotante) o determinación del número de atributos relevantes que pueden intercambiarse con atributos irrelevantes (LED o *waveform*).

Como se ha señalado anteriormente, una de las principales ventajas de este tipo de conjunto de datos es la posibilidad de crear cualquier cantidad de datos, con

1 http://research.cs.wisc.edu/dmi/svm/ndc

		Cambio abrupto	Cambio gradual
Atributos	Nominales	STAGGER (2) Conceptos booleanos (2) LED (10)	Wrapper
	Numéricos	SEA (2) SINE (2) CIRCLE (2) GAUSSIAN (2) Waveform (3)	Hiperplano rotante (2) RBF Aleatorio (configurable) NCD_drift (2)
	Combinados	MIXED (2) Función de préstamo (2)	Wrapper

Tabla 3.3: Generadores de conjuntos de datos. Categorización de los generadores de conjuntos de datos considerando el tipo de cambio de concepto que ellos simulan y el tipo de atributos usado en el problema. El número de clases es representado entre paréntesis. El generador *wrapper* se refiere a la combinación ponderada de diferentes conceptos (con cambio abrupto) usando una función con pendiente o sigmoide.

variables grados de configuración. Para facilitar esta tarea, algunos autores ofrecer su propia implementación de estos generadores. Por ejemplo, MOA[2] [BHKP10] es uno de los marcos de trabajo más completos que soporta muchos conjuntos de datos (STAGGER, LED, funciones de préstamo, SEA, *waveform*, hiperplano rotante, y RBF aleatorio). Narasimhamurthy y Kuncheva [NK07] proveen herramientas[3] que ofrecen otros generadores (STAGGER, GAUSSIAN, hiperplano rotante). Minku y otros [MWY10] también facilitan implementaciones alternativas para generar conjuntos de datos con cambio de concepto[4] (STAGGER, conceptos booleanos, SEA, SINE, CIRCLES, hiperplano rotante).

3.3.3.2 *Conjuntos de datos reales*

El aprendizaje a partir de flujo de datos está directamente relacionado con el cambio de concepto porque esta es una característica inherente que aparece usualmente cuando los datos son adquiridos a lo largo del tiempo. Se han propuesto algunas categorizaciones para los escenarios de flujos de datos [Žlio9b, Muto5] y muchas de estas categorías dan una buena perspectiva, aunque no todas ellas se ajustan al contexto de la clasificación. Por ejemplo, la monitorización generalmente emplea técnicas del aprendizaje no supervisado y la toma de decisiones usualmente sufre retardos en el etiquetado de las clases, problemas que no pueden ser tratados directamente como tareas de clasificación.

Para tener una idea aproximada de problemas reales donde es importante el cambio de concepto, se puede observar cuáles conjuntos de datos reales ha sido

2 http://moa.cs.waikato.ac.nz
3 http://www.bangor.ac.uk/~mas00a/EPSRC_simulation_framework/changing_environments_stage1a.htm
4 http://www.cs.bham.ac.uk/~minkull/opensource

usado con el propósito de la evaluación. A pesar que estos conjuntos de datos pueden ser usados indistintamente con respecto al modelo de aprendizaje, en específicos campos de aplicación, algunos modelos de aprendizajes tienen ventajas con respecto a otros [Joa98, CNDH03, DC04, SW07].

Internet es uno de los escenarios más motivadores para las tareas de aprendizaje incremental con la presencia de cambios de concepto. En este contexto, múltiples fuentes generan enormes cantidades de datos con una tasa de llegada alta: datos de correos, flujos de clics, peticiones de usuarios, registros web, servidores web, descargas punto a punto, etc. Aquí podemos diferenciar entre dos categorías fundamentales: aquellas relacionadas con sistemas de comunicación (capas de red, capas de transporte, etc.) y las relacionadas con el contenido que se intercambia. Desde un punto de vista de la administración, es importante la extracción de conocimientos interesantes que posibiliten la predicción de futuras situaciones (saturación de la red, intrusión en la red, etc.). Con respecto al contenido, una tarea muy común es el filtrado de correos spam.

El objetivo final del diseño de algoritmos y métodos que puedan aprender cuando los conceptos cambian es aplicarlos a situaciones reales. Lógicamente, en este punto la variedad de conjuntos de datos se convierte mucho más extensiva. Desde el punto de vista de aplicar procesos de evaluación a diferentes algoritmos, la principal diferencia de estos conjuntos de datos es la capacidad de reproducirlas. En estos problemas reales son comunes los datos privados (rastro web [DH00], datos de fraude de créditos [WFYH03], etc.), ya que estos no pueden publicarse para el uso abierto debido a cuestiones de derechos de autor, limitaciones técnicas, etc. Pero en otros casos, aunque la reproducción exacta no es posible, se pueden obtener conjuntos de datos similares desde la misma fuente de datos y siguiendo un proceso específico (subconjunto de documentos seleccionados a partir de un conjunto de datos[Kli04], simulación de juegos de video[5] [LDM08], etc.) .

Lo más importante en nuestro caso es la identificación de conjuntos de datos reales que puedan ser considerados como puntos de referencia, porque ellos han sido usados ampliamente en diferentes trabajos investigativos. Como sugerimos anteriormente, el repositorio de la UCI[6] [FA10] es una de las fuentes más comunes y algunos de los conjuntos de datos que allí se encuentran han sido estudiados desde la perspectiva del cambio de concepto (*Adult* [FHY04], *Poker hand* [BG09a], *Ozone level detection* [Žli09a, Brz10], etc.).

Otra fuente de conjunto de datos es facilitada por competiciones organizadas con el soporte de conferencias bien conocidas. La copa KDD, organizada por la *ACM Special Interest Group on Knowledge Discovery and Data Mining*, es especialmente destacable. Desde 1997 han propuesto tareas a resolver[7], muchas de las cuales pueden ser manipuladas desde un punto de vista del aprendizaje con la presencia de cambio de concepto [AHWY03, Brz10]. Otra competición anual, *PAKDD Data Mining Competition*, organizada por la conferencia *Pacific-Asia Knowledge Discovery and Data Mining* ofrece problemas y conjuntos de datos en diferentes sitios siempre

5 http://create.msdn.com/en-US/education/catalog/sample/racing_game
6 http://archive.ics.uci.edu/ml/index.html
7 http://www.sigkdd.org/kddcup/index.php

que esta tiene lugar[8]. *Data Expo*[9] ofrece también grades conjuntos de datos donde es usual encontrar cambios de concepto [IGD11].

Un área muy interesante que produce enormes cantidades de datos, en la cual los algoritmos son evaluados, es la relacionada con correos spam. Así, Katakis y otros [KTV10] tratan de simular y detectar cambio de concepto (abrupto o gradual) adaptando conjuntos de datos originales [KTV10] y haciendo disponible los conjuntos de datos finales usados[10]. Delany y otros [DC04, DCTC05, DB07] también ofrecen los conjuntos de datos generados[11], conocidos como *ECUE Spam Datasets*.

Es común que los investigadores ofrezcan los conjuntos de datos que usan. Así, Gama y otros [GMCR04] han popularizado el conjunto de datos *Electricity*[12] propuesto por Harries [Har99], el cual ha sido extensamente usado [BdF+06, KM07, Brz10]. Žliobaitė [Žli10] también usa diferentes conjuntos de datos basados en juegos (ajedrez) y en respuestas dadas a un cuestionario de prueba. Estos son construidos con datos recolectados durante períodos de tiempo de cuatro o cinco años, en los que se espera algún cambio de concepto[13]. Otros investigadores que publican algunos conjuntos de datos en sus propios sitios son Zhu[14] (conjuntos de datos enfocados en sensores y suministro de energía [ZZTG10]) y Polikar[15] (conjunto de datos climáticos).

3.4 HERRAMIENTAS DE SOFTWARE DE CÓDIGO ABIERTO

En general, es útil tener la implementación de los algoritmos y métodos en el área del aprendizaje automático. Desde un punto de vista práctico y enfocándose en aquellos que pueden aprender en la presencia de cambio de concepto, esta disponibilidad permite a los investigadores desarrollar nuevas ideas, evaluar el rendimiento de distintas alternativas, etc. En este contexto los algoritmos pueden ser ofrecidos en un escenario independiente y aislado, como ocurre con la herramienta VFML (*Very Fast Machine Learning*)[16] [HD03], el cual incluye el algoritmo CVFDT; o el software ADWIN[17] [Bif10], que implementa una algoritmo basado en ventanas deslizantes para detectar cambio. La integración de estas herramientas con otras puede ser una tarea más o menos simple, pero tomar marcos de trabajo más generales como punto de partida ofrece ventajas adicionales.

El sistema popular Weka[18] (*Waikato Environment for Knowledge Analysis*) [HFH+09] contiene una colección de herramientas de visualización y algoritmos para el análisis de datos y el modelado predictivo. En la actualidad este es usado en muchas áreas, fundamentalmente con objetivos educacionales e investigativos, pero el mismo ha sido adoptado por algunas compañías, como Pentaho®. Weka soporta pre-procesado, agrupamiento, clasificación, regresión, visualización y selección de

8 http://fit.mmu.edu.my/pakdd2012/dmcomp.html
9 http://stat-computing.org/dataexpo
10 http://mlkd.csd.auth.gr/concept_drift.html
11 http://www.comp.dit.ie/sjdelany/Dataset.htm
12 http://www.liaad.up.pt/~jgama/ales/elec.tar
13 https://sites.google.com/site/zliobaite/resources-1
14 http://www.cse.fau.edu/~xqzhu/stream.html
15 http://users.rowan.edu/~polikar/research/nse
16 http://www.cs.washington.edu/dm/vfml
17 http://adaptive-mining.sourceforge.net/?page_id=20
18 http://www.cs.waikato.ac.nz/ml/weka

atributos. Los algoritmos que están más cercanos al contexto del cambio de concepto son aquellos que implementan la interfaz *updateableclassifier*, ya que ella induce modelos de clasificación incremental que pueden aprender usando una instancia a la vez.

Entre los software que están más específicamente orientados al aprendizaje en presencia de cambio de concepto, podemos encontrar al marco de trabajo MOA (*Massive Online Analysis*)[19] [BHKP10]. Este está relacionado con el proyecto WEKA. MOA incluye una colección de algoritmos de aprendizaje automático (clasificación, regresión y agrupamiento) orientados a la minería de flujos de datos, donde el cambio de concepto es un aspecto relevante. Este dispone de una gran variedad de algoritmos inherentes al aprendizaje incremental, métodos para detectar cambio de concepto, herramientas para la evaluación y muchos generadores de conjuntos de datos artificiales con la posibilidad de incluir varios tipos de cambio de concepto.

Otro sistema que considera parcialmente cambio de concepto es RapidMiner[20] (previamente YALE) [MWK+06]. Este sistema ofrece una gran variedad de métodos y permite una rápida fase de prototipado, reduciendo el costo de nuevas aplicaciones. Adicionalmente, posee una amplia funcionalidad para la evaluación y optimización de procesos. Para manipular cambio de concepto, RapidMiner necesita un software adicional (*plugin*) que aunque no está disponible para la versión actual, se puede obtener para una versión anterior. Así, este sistema es especialmente notorio porque sus desarrolladores planifican soportar minería de flujos de datos y aprendizaje en línea en versiones futuras.

3.5 CONCLUSIONES DEL CAPÍTULO

Muchas áreas relativas al aprendizaje supervisado han estudiado recientemente el problema del cambio de concepto, como el aprendizaje activo [ŽBPH13], la detección de novedades [MGK+11] y el aprendizaje con distribuciones de clase sesgadas [GFHY07]. Otras investigaciones han iniciado el estudio de la influencia del cambio de concepto en otras fases del proceso de minería de datos, como el pre-procesamiento [ŽG14].

La mayoría de los investigadores coinciden en los requerimientos para un sistema de aprendizaje incremental: procesar cada ejemplo por separado e inspeccionarlo solo una vez, usar una cantidad limitada de tiempo y memoria y estar listo para predecir en cualquier momento. Sin embargo, la terminología no está bien establecida, y se han desarrollado independientemente similares estrategias bajo diferentes nombres y en contextos diferentes [MRA+12]. Los algoritmos deben tener además tan pocos parámetros a definir por los usuarios como sea posible, ya que debido a la velocidad de los datos y a la gran variedad de sus características, este ajuste generalmente no es práctico [ŽBG+12b].

Los acercamientos heurísticos surgieron inicialmente para satisfacer las restricciones de computación mencionadas, estos han evolucionado continuamente a métodos más robustos con garantías matemáticas de rendimiento. Aun así, varios algoritmos de aprendizaje incremental no satisfacen todos estos requerimientos comunes. Los métodos no paramétricos han sido favorecidos en los últimos años

19 http://moa.cs.waikato.ac.nz
20 http://rapid-i.com/

porque imponen menos restricciones teóricas a los datos reales, además de tener pocos parámetros ajustables por el usuario [KBG04].

En los últimos años se han desarrollado innumerables algoritmos para la manipulación de cambio de concepto. Algunos de ellos pueden ser usados por muchos clasificadores, porque la detección de cambio y el proceso de actualización depende solo de alguna métrica observable y no del modelo. Sin embargo, los indicadores de rendimiento de estos algoritmos de detección de cambio difieren en muchas investigaciones. Otros métodos son específicos para un clasificador concreto y difícilmente pueden ser generalizados para otros, estas técnicas por lo general se adaptan al cambio de forma más eficaz ya que toman ventaja de características específicas del modelo. Adicionalmente, algunas investigaciones han señalado que modelos de aprendizaje particulares pueden ser más convenientes para varios tipos de aplicaciones. Los detectores de cambio y los algoritmos de aprendizaje tienden al uso de herramientas estadísticas bien establecidas para dar garantías matemáticas de su rendimiento.

En general, la estrategia de monitorizar sobre el tiempo la consistencia del algoritmo de aprendizaje a través de detectores de cambio, para ejecutar algún mecanismo de adaptación cuando se estime un cambio significativo ha sido ampliamente usada. Sin embargo, pocos acercamientos han propuesto garantías matemáticas a la estrategia en sí. Tal acercamiento es más común en el contexto del aprendizaje PAC, donde frecuentemente se asume una estructura subyacente del cambio. Pocos estudios han estudiado además cómo el clasificador pasado puede ser usado después de la detección del cambio.

Un ejemplo del dinamismo y de avances recientes en el área es el trabajo propuesto por Rutkowski y otros [RPDJ13], los que sugieren que los algoritmos que usan la cota de Hoeffding deben revisarse ya que esta cota no es una herramienta adecuada para la estimación de intervalos de confianza para cualquier medida heurísta (e.g. usada para la expansión de un árbol de decisión). Algunos ejemplos de algoritmos populares que usan esta cota son la familia de algoritmos VFDT y VFDR. Sin embargo, las cotas propuestas por Rutkowski y otros [RPDJ13] son muy conservativas y por lo general los algoritmos resultantes necesitan grandes cantidades de datos para alcanzar niveles de precisión adecuados. En este sentido, también se propone un método paramétrico asumiendo la distribución normal [RJPD14]. Otras desigualdades de probabilidad quizás puedan adaptarse a nuevos problemas identificados en la minería de flujos de datos, donde todos los ejemplos no están independientemente distribuidos [MvdBW07, BRŽ+13].

Otra limitación en el área es la metodología de evaluación, porque es deseable incluir soporte estadístico y la mayoría de las propuestas de este tipo están diseñadas para ausencia de cambio de concepto.

La Tabla 3.2 muestra cómo el problema del cambio de concepto ha sido considerado gradualmente en diferentes algoritmos de aprendizaje. En particular, los métodos de ensamble de clasificadores han ganado fuerza. Las reglas de decisión han sido menos estudiadas en los últimos años sin una justificación bien fundada, por lo que puede ser otra área de investigación prominente.

Parte II

APRENDIZAJE EN FLUJOS DE DATOS CON CAMBIO DE CONCEPTO

4

NUEVOS MÉTODOS PARA LA DETECCIÓN EN LÍNEA DE CAMBIO DE CONCEPTO

Los algoritmos de aprendizaje incrementales y en línea cada día son más relevantes en la minería de datos por la necesidad creciente de procesar flujos de datos, siendo una característica inherente a estos flujos de datos el cambio de concepto. Para manipular cambio de concepto independientemente del algoritmo de aprendizaje, en este capítulo proponemos nuevos métodos para monitorizar medidas de rendimiento obtenidas durante el proceso de aprendizaje. Para monitorizar este rendimiento, aplicamos desigualdades de probabilidad que asumen solo variables aleatorias independientes, univariadas y acotadas. Así, establecemos garantías teóricas en la detección de tales cambios distribucionales. Se consideran además restricciones de cómputo comunes para la detección de cambios en línea así como tipos comunes de cambio de concepto. En este sentido son propuestos dos principales acercamientos, el primero involucra a la media aritmética como estimador y es más adecuado en la detección de cambios abruptos. El segundo sigue una idea intuitiva ampliamente usada para para lidiar con cambios graduales a través de promedios ponderados. La simplicidad de los métodos propuestos, su complejidad computacional y sus garantías teóricas de rendimiento los hacen muy ventajosos para la detección y manipulación de cambios de concepto.

4.1 INTRODUCCIÓN

Como hemos señalado en capítulos anteriores, una estrategia ampliamente extendida para manipular cambio de concepto monitoriza constantemente alguna medida de rendimiento del modelo de aprendizaje (por ejemplo, la precisión). Debido a la complejidad de detectar cambios distribucionales en línea, la mayoría de los acercamientos existentes monitorizan cambios en un estadístico conveniente, tal como la media o la mediana [RTA11]. De esta forma, el problema de detección de cambio de concepto se reduce a la estimación de cambios significativos en un estadístico calculado a partir de la secuencia de valores que miden una característica de desempeño.

Como el modelo de aprendizaje es monitorizado constantemente en el tiempo, esta secuencia de valores generalmente es grande y se obtiene a una tasa de llegada alta, por lo que es común imponer restricciones computacionales para su procesamiento [DGIM02, RTA11]. La complejidad computacional requerida para procesar cada valor de rendimiento debe ser constante y los métodos deben ser capaces de operar en una simple pasada por los valores vistos, o sea, cada valor de rendimiento debe ser procesado una vez y luego descartarse. Estos detectores de cambio deben ser capaces de manipular tipos comunes de cambio que prevalecen en muchos problemas reales [GŽB+]. Bajo estas condiciones, muchos acercamientos estadísticos tradicionales que asumen un tamaño fijo de los datos de entrada para estimar cambios distribucionales no son adecuados. Algunos de los esque-

mas paramétricos más estudiados para detectar cambios en línea son el control de gráficos de Shewhart, el control de gráficos EWMA y el procedimiento CUSUM de Page [Mono1]. Sin embargo, en muchas situaciones los datos no están regulados por estas distribuciones de probabilidad conocidas y los esquemas no paramétricos son más adecuados.

Otros acercamientos existentes no cumplen las restricciones computacionales mencionadas [KBG04, BG07]. Por ejemplo, ADWIN2 [BG07] mantiene una ventana de tamaño W con complejidad espacial y temporal de $O(\log W)$, donde W es el número de valores de rendimiento vistos después de la última detección de cambio. Otros métodos relacionados como DDM [GMCR04] , EDDM [BdF+06] y ECDD [RATH12] no están equipados de garantías matemáticas de rendimiento; mientras que DDM [GMCR04] y ECDD [RATH12] asumen valores de medición regulados acorde a una distribución de Bernoulli, por lo que están restringidos a flujos de bits. ECDD, que usa un estimador EWMA para la detección de cambio, solo se enfoca en la tasa de falsos positivos.

El aporte fundamental descrito en este capítulo es la obtención de una familia de métodos para la monitorización constante en el tiempo de la media estimada a partir de una secuencia de valores reales (correspondiente a alguna medida de rendimiento) con el objetivo de detectar cambios significativos. De esta forma, extendemos algunos métodos para detectar cambios en flujos de datos eliminando cualquier suposición relacionada con la distribución de probabilidades que genera estos valores. En cambio, se asume que estos valores están dados acorde a variables aleatorias independientes y acotadas. Adicionalmente, los métodos propuestos tienen complejidad temporal y espacial constante y son capaces de detectar cambio realizando una sola pasada por los valores vistos. Se ofrecen garantías rigurosas de su desempeño en forma de cotas para la tasa de falsos positivos y falsos negativos.

Aunque este capítulo se enfoca en la detección de cambio de concepto asumiendo que todas las instancias están etiquetadas, los detectores de cambio propuestos pueden ser aplicados a escenarios más reales donde las instancias se obtienen con una tasa alta de llegada pero las etiquetas son difíciles de obtener [ŽBPH13].

4.2 MONITORIZANDO ESTADÍSTICOS

Como es habitual en la minería de flujos de datos, asumimos una secuencia muy larga (posiblemente infinita) $S = (\vec{a}_1, c_1); (\vec{a}_2, c_2); \ldots$ de ejemplos $(\vec{a}_i, c_i)$, que se obtienen en el tiempo donde $\vec{a}_i \in \overrightarrow{\mathcal{A}}$ es el vector de atributos y $c_i \in \mathcal{C}$ es su etiqueta de clase correspondiente (ver Sección 2.1).

Los detectores de cambio que se proponen en este capítulo se pueden aplicar a un esquema bien conocido para la manipulación de cambio de concepto con dos módulos [GMCR04, BdF+06, BG07]: el detector de cambio y el algoritmo de aprendizaje (Figura 4.1). Los detectores de cambio pueden alternar entre dos estados: *en control* cuando se estima que el concepto es estable y *fuera de control* cuando se estima un cambio de concepto. Cuando cada ejemplo es adquirido el algoritmo de aprendizaje realiza una predicción $\hat{c}_i \in \mathcal{C}$ basada en el vector de atributos $\vec{a}_i$. Luego, el detector de cambio es actualiza a través de una función de pérdida $\ell(\hat{c}_i, c_i)$ definida como $\ell : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ (por ejemplo, la función de pérdida 0-1 con

$\ell(\hat{c}_i, c_i) = 1$ si $\hat{c}_i \neq c_i$ y $\ell(\hat{c}_i, c_i) = 0$ en otro caso). Después, el ejemplo en cuestión se le provee al algoritmo de aprendizaje para continuar con su entrenamiento.

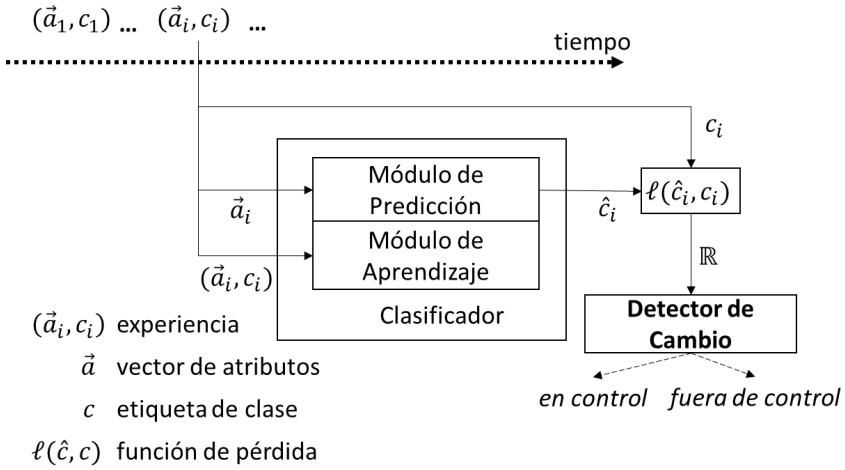


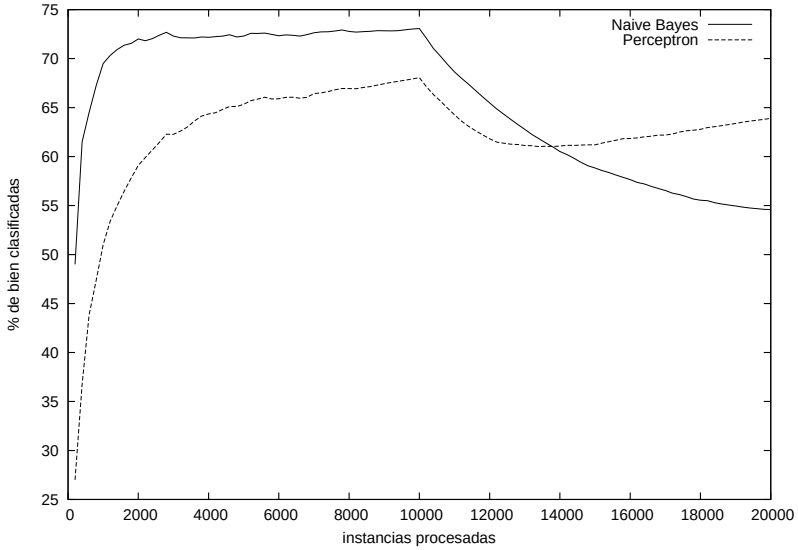
Figura 4.1: Esquema para la manipulación de cambio de concepto a través de detectores de cambio en línea.

De esta forma, el detector de cambio recibe como entrada un flujo de valores reales (e.g. obtenidos de la función de pérdida) y da como salida la información del estado actual estimado por él. La familia de métodos propuestos se aplica en este esquema si $\ell(\hat{c}_i, c_i)$ se corresponde con variables aleatorias independientes y acotadas.

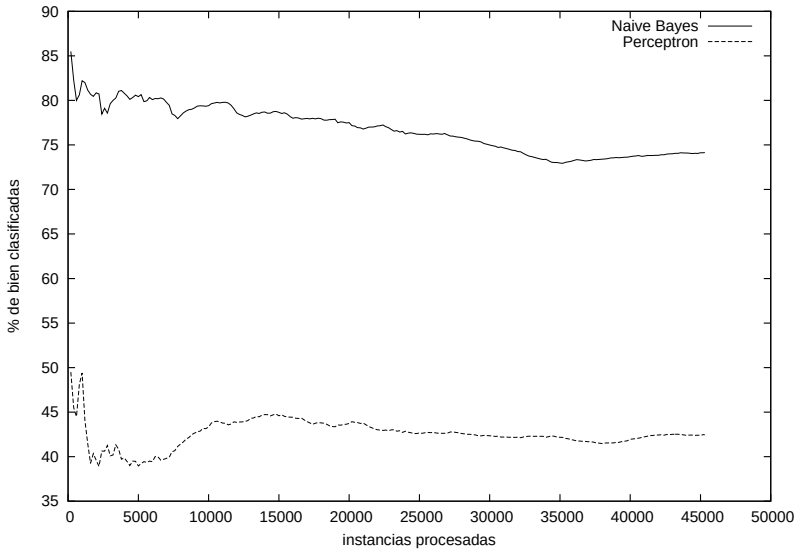
Siguiendo el modelo de aprendizaje PAC [Mit97], si la distribución de los ejemplos es estacionaria, la tasa de error del algoritmo de aprendizaje decrecerá cuando el número de ejemplos aumenta [GMCR04, BdF+06]. En este sentido, un aumento significativo en la media (e.g. en la tasa de error del algoritmo de aprendizaje) estimada al monitorizar $\ell(\hat{c}_i, c_i)$ en el tiempo, puede de hecho significar un cambio de concepto.

La Figura 4.2 representa el deterioro típico de la precisión de un algoritmo de aprendizaje en tipos de cambio abruptos y graduales. Esta precisión, en este caso correspondiente a dos algoritmos incrementales bien conocidos (Naïve Bayes y Perceptrón), es calculada con una configuración *test-then-train* [GC06, Bif10]: el algoritmo de aprendizaje primero es probado con cada ejemplo para luego continuar con su aprendizaje. Sobre los datos correspondientes a una pantalla LED de siete segmentos [BHPG09], se simula un cambio de concepto a partir del ejemplo número 10 000 intercambiando algunos atributos relevantes e irrelevantes en la función objetivo. El deterioro permanente y lento de la precisión en presencia de cambios graduales está representado en la Figura 4.2b por medio del conjunto de datos reales ELEC2 [Har99].

Como hemos señalado, una idea intuitiva ampliamente extendida para detectar cambio de concepto es monitorizar algún estadístico conveniente $\hat{X}$ en el tiempo, calculado a partir de un flujo $x_1, x_2, ..., x_t, x_{t+1}, ..., x_n$ de valores de rendimiento. En este capítulo estudiamos los promedios y los promedios ponderados, que tienen la forma $\hat{X} = \sum_{i=1}^{n} v_i x_i$ (por ejemplo, en los promedios $\forall i$, $v_i = 1/n$ y $\hat{X} = \overline{X}$); adicionalmente, asumimos valores de rendimiento dados acorde a n variables alea-



(a) Cambio abrupto: Datos de pantalla LED de 7 segmentos.



(b) Cambios graduales: Datos de predicción de electricidad.

Figura 4.2: Conducta típica del rendimiento de un algoritmo de aprendizaje en tipos de cambio abrupto (Figura 4.2a) y gradual (Figura 4.2b).

torias $X_1, X_2, ..., X_t, X_{t+1}, ..., X_n$ independientes, acotadas, y reales. El problema en cuestión es estimar si $\hat{X}$ cambia significativamente en el tiempo. En cambios abuptos (esto es $E(X_i) = \mu_0$ para $0 < i \leq t$ y $E(X_j) = \mu_1 \neq \mu_0$ para $t < j \leq n$), el problema es la estimación de si existe tal punto de cambio $t+1$ en la secuencia. Desafortunadamente, asumir cambios abruptos es frecuentemente irreal.

Usualmente los cambios son graduales, donde no está definido un punto de cambio sino un período de transición entre conceptos consecutivos. Los cambios graduales lentos son los más difíciles de detectar. En este caso, la media poblacional de los valores de rendimiento (por ejemplo, correspondientes a la precisión) por lo general varía lenta y continuamente.

4.3 A-TEST: ACOTANDO DIFERENCIA DE PROMEDIOS

Existen métodos aplicados al aprendizaje en flujos de datos que calculan intervalos de confianza para diferentes parámetros (e.g. tasa de error) considerando distribuciones conocidas. Por ejemplo, se ha asumido la distribución normal asumiendo que el tamaño de la muestra es lo suficientemente grande (n) para aproximar una función de distribución de probabilidad desconocida a la distribución normal [BG07].

Otros métodos no asumen ninguna función de distribución de probabilidad y usan varias desigualdades de probabilidad. Por ejemplo, la familia de algoritmos IADEM [del07, dRGM08, RM00, Ram01, RMd04, RdCM05] induce árboles de decisión sin memoria de instancias a través de las cotas de Hoeffding y de Chernoff; esta familia solo almacena en los nodos la información relevante en términos de frecuencias relativas. Así, dada una confianza deseada $(1 - \delta)$, usa las cotas de concentración para estimar cuál es el error máximo (ε) y calcula cuál es el intervalo de confianza ($[E(\overline{X}) - \varepsilon, E(\overline{X}) + \varepsilon]$) para todos los parámetros y medidas. De esta forma, la inducción del árbol considera información enriquecida para ejecutar diferentes acciones (expansión de un nodo, selección del atributo más apropiado para la expansión, etc.).

Las cotas de concentración han sido usadas con anterioridad en escenarios de flujos de datos, y una de las más extendidas es la propuesta por Hoeffding [Hoe63].

Teorema 1 (Desigualdad de Hoeffding). *Sean $X_1, X_2, \ldots, X_n$ variables aleatorias independientes tal que $X_i \in [a_i, b_i]$, donde $i \in \{1, \ldots, n\}$. Sea $\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$. Entonces para cualquier $\varepsilon > 0$,*

$$\Pr\left\{ \overline{X} - E\left[\overline{X} \right] \geq \varepsilon \right\} \leq e^{-2n^2\varepsilon^2 / \sum_{i=1}^{n} (b_i - a_i)^2}$$

Para este teorema y considerando el promedio $\overline{X}$, podemos estimar el error ε_δ, dado un nivel de confianza de a lo sumo δ:

$$\varepsilon_\delta = \sqrt{\frac{1}{2n} \ln \frac{1}{\delta}}$$

Como podemos observar, la desigualdad de Hoeffding asume solo variables aleatorias independientes y acotadas, pero no es asumida ninguna función de probabilidad. El estadístico en cuestión ($\overline{X}$) y la cota de error (ε_δ) pueden ser calculados en $O(1)$ de complejidad temporal y espacial, lo que hace el teorema aplicable al

aprendizaje en flujos de datos [delo7, dRGMo8, RMoo, Ramo1, RMdo4, RdCMo5]. Particularmente, un corolario propuesto por Hoeffding [Hoe63, página 16] puede ser aplicado a la detección de cambios significativos en la media poblacional de flujo de valores.

Corolario 2. *Si* $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ *son variables aleatorias independientes con valores en el intervalo* $[a, b]$, *y si* $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\overline{Y} = \frac{1}{m}\sum_{i=1}^{m} Y_i$, *entonces para* $\varepsilon > 0$*:*

$$\Pr\left\{\overline{X} - \overline{Y} - \left(E\left[\overline{X}\right] - E\left[\overline{Y}\right]\right) \geq \varepsilon\right\} \leq e^{\frac{-2\varepsilon^2}{(n^{-1}+m^{-1})(b-a)^2}} \tag{4.1}$$

Análogamente al Teorema 1, pero considerado más bien la diferencia entre promedios, podemos estimar el error ε_α, dado un nivel de significación de a lo sumo α:

$$\varepsilon_\alpha = (b-a)\sqrt{\frac{n^{-1}+m^{-1}}{2}\ln\frac{1}{\alpha}} \tag{4.2}$$

De esta forma, el Corolario 2 puede ser aplicado a la detección de cambios distribucionales en línea, ya que este puede ser usado con complejidad temporal y espacial constante. A partir del Corolario 2, podemos obtener una prueba estadística acotando la probabilidad de los errores de tipo I y II. Llamaremos a esta prueba A-test (A porque concierne a promedios, *Averages*). Sea la hipótesis nula $H_0 : E\left[\overline{X}\right] \leq E\left[\overline{Y}\right]$ contra la alternativa $H_1 : E\left[\overline{X}\right] > E\left[\overline{Y}\right]$ y sea $\overline{X} - \overline{Y} \geq \varepsilon_\alpha$ la regla para rechazar H_0. El corolario siguiente acota la probabilidad de los errores de tipo I y II para esta prueba estadística.

Corolario 3 (A-test)**.** *Bajo las condiciones del Corolario 2, para* ε_α *definido en la ecuación (4.2):*

1. *si* $E\left[\overline{X}\right] \leq E\left[\overline{Y}\right]$, *entonces* $\Pr\left\{\overline{X} - \overline{Y} \geq \varepsilon_\alpha\right\} \leq \alpha$ *(cota para el error de tipo I),*
2. *si* $E\left[\overline{X}\right] \geq E\left[\overline{Y}\right] + \varsigma$, *y* $\varsigma > \varepsilon_\alpha$ *entonces*

$$\Pr\left\{\overline{X} - \overline{Y} < \varepsilon_\alpha\right\} \leq e^{\frac{-2(\varsigma-\varepsilon_\alpha)^2}{(n^{-1}+m^{-1})(b-a)^2}}$$

(cota para el error de tipo II).

De una forma obvia podemos derivar una prueba similar para la hipótesis nula $H_0 : E\left[\overline{X}\right] \geq E\left[\overline{Y}\right]$ contra la alternativa $H_1 : E\left[\overline{X}\right] < E\left[\overline{Y}\right]$, siendo $\overline{Y} - \overline{X} \geq \varepsilon_\alpha$ la regla para rechazar H_0. Así, también puede derivarse una prueba de dos colas por medio de la hipótesis nula $H_0 : E\left[\overline{X}\right] = E\left[\overline{Y}\right]$ contra la alternativa $H_1 : E\left[\overline{X}\right] \neq E\left[\overline{Y}\right]$, siendo $\left|\overline{X} - \overline{Y}\right| \geq \varepsilon'_\alpha$ la regla para rechazar H_0, donde

$$\varepsilon'_\alpha = (b-a)\sqrt{\frac{n^{-1}+m^{-1}}{2}\ln\frac{2}{\alpha}} \tag{4.3}$$

En este sentido, dada una secuencia de variables aleatorias $X_1, \ldots, X_n, Y_1, \ldots, Y_m$, podemos detectar cambios en la media poblacional monitorizando la diferencia entre promedios a partir de la prueba A-test.

4.4 W-TEST: ACOTANDO PROMEDIOS PONDERADOS

En esta Sección derivamos una prueba estadística más general para promedios ponderados, aunque de igual forma eficiente y simple. En este caso, los valores reales recientes tendrán más peso que los más antiguos, asumiendo que ellos tienen mayor probabilidad de ocurrencia.

McDiarmid [McD89] generalizó la desigualdad de Hoeffding para variables aleatorias independientes, a continuación mostramos un resultado interesante.

Teorema 4 (Desigualdad de McDiarmid). *Sean $X_1, X_2, \ldots, X_n$ variables aleatorias independientes tal que $X_i \in A_i \in [a_i, b_i]$, donde $i \in \{1, \ldots, n\}$. Sea $g : A_1 \times \ldots \times A_n \to \mathbb{R}$ una función mensurable que satisface la condición de diferencias acotadas independientes, i.e., existe un vector $\mathbf{d} = (d_1, \ldots, d_n)$ tal que $\left| g\left(\overrightarrow{X}\right) - g(\overrightarrow{X}') \right| \leq d_i$ para todos los vectores $\overrightarrow{X}$ y $\overrightarrow{X}'$ que difieren solo en la coordenada i-ésima. Sea $\overline{Y}$ una variable aleatoria definida como $\overline{Y} = g(X_1, X_2, \ldots, X_n)$. Entonces para cualquier $\varepsilon > 0$,*

$$Pr\left\{\overline{Y} - E\left[\overline{Y}\right] \geq \varepsilon\right\} \leq e^{-2\varepsilon^2 / \sum_{i=1}^{n} d_i^2} \tag{4.4}$$

Teniendo en cuenta el Teorema 4, podemos definir a $g(\overrightarrow{X})$ como una función que dé más peso a los ejemplos recibidos más recientemente. Proponemos monitorizar una suma ponderada que se actualiza cada vez que se obtiene un valor. Sin embargo, como en el contexto de los flujos de datos todos los ejemplos no están disponibles antes de su procesamiento, sino que llegan continuamente en el tiempo, debemos definir a $g(\overrightarrow{X})$ de forma eficiente en términos de complejidad temporal y espacial. En este sentido, examinaremos algunas funciones de pesos cuyas sumas ponderadas respectivas pueden ser actualizadas con complejidad computacional constante para cada nuevo valor.

Si asumimos que $X_1, X_2, \ldots, X_n$ son variables aleatorias independientes cuyos valores se encuentran en el intervalo $\forall i\ A_i \in [a, b]$, adicionalmente tomamos a $\overrightarrow{X}$ como un vector $\overrightarrow{X} = (X_1, X_2, \ldots, X_n)$ y consideramos los pesos $v_i \in (0, 1]$ que pueden ser generados por una función de decremento de pesos; entonces podemos definir el promedio ponderado como

$$g(\overrightarrow{X}) = \hat{X}_n = \sum_{i=1}^{n} v_i X_i \tag{4.5}$$

que satisface la condición de diferencias acotadas independientes (necesaria en el Teorema 4): como $X_i \in [a, b]$ y $v_i \in (0, 1]$ ($i \in \{1, \ldots, n\}$), entonces $d_i = (b - a)v_i$.

En general, si en un esquema de pesado dado, los pesos v_i no están restringidos en el intervalo $(0, 1]$ (e.g. $0 < v_i < \infty$), ellos pueden ser normalizados como sigue.

Considerando $E[X_i] = \mu$ para todo $i \in \{1, \ldots, n\}$, podemos presentar a $E[g(\overrightarrow{X})]$ como

$$E\left[\sum_{i=1}^{n} v_i X_i\right] = \sum_{i=1}^{n} v_i E[X_i] = \mu \sum_{i=1}^{n} v_i$$

Para monitorizar cambios en el valor del promedio convenientemente de una forma simple, podemos hacer $\sum_{i=1}^{n} v_i = 1$ y consecuentemente $E[g(\overrightarrow{X})] = \mu$. Así, reescribimos la ecuación (4.5) en una forma más general:

$$\hat{X}_n = \frac{1}{\mathcal{W}_n} \sum_{i=1}^{n} w_i X_i \tag{4.6}$$

donde $\mathcal{W}_n = \sum_{i=1}^{n} w_i$ y $v_i = w_i / \mathcal{W}_n$. Adicionalmente, escrita en la siguiente forma recurrente, la ecuación (4.6) puede ser calculada en $O(1)$ de complejidad temporal y espacial a cada valor procesado (note que $\mathcal{W}_n = \mathcal{W}_{n-1} + w_n$; $\hat{X}_1 = X_1$):

$$\hat{X}_n = \frac{\mathcal{W}_{n-1}}{\mathcal{W}_n} \hat{X}_{n-1} + \frac{w_n}{\mathcal{W}_n} X_n.$$

Adicionalmente, podemos calcular también a $\mathcal{D}_n = \sum_{i=1}^{n} d_i^2$ en la ecuación (4.4) con $O(1)$ de complejidad temporal y espacial a cada valor procesado ($\mathcal{D}_1 = (b-a)^2$):

$$\begin{aligned} \mathcal{D}_n &= \sum_{i=1}^{n} d_i^2 = \left(\frac{b-a}{\mathcal{W}_n}\right)^2 \sum_{i=1}^{n} w_i^2 \\ \mathcal{D}_n &= \left(\frac{\mathcal{W}_{n-1}}{\mathcal{W}_n}\right)^2 \mathcal{D}_{n-1} + (b-a)^2 \left(\frac{w_n}{\mathcal{W}_n}\right)^2. \end{aligned}$$

Lógicamente, se puede obtener un intervalo de confianza a partir del Teorema 4 para promedios ponderados $\hat{X}_n$ con respecto a su valor esperado.

4.4.1 *Acotando la diferencia entre promedios ponderados*

Siguiendo la idea mostrada en la Sección 4.3, podemos obtener una prueba estadística más general con garantías de rendimiento para los errores de tipo I y II acotando análogamente la diferencia entre dos promedios ponderados. Para esto necesitamos el corolario que se enuncia a continuación.

Corolario 5. *Si $X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m$ son variables aleatorias independientes acotadas en el intervalo $[a, b]$, y si $\hat{X}_n = \sum_{i=1}^{n} v_i X_i$, $\hat{Y}_m = \sum_{i=1}^{m} v'_i Y_i$, tal que $\hat{X}_n$ y $\hat{Y}_m$ satisfacen la condición de diferencias acotadas independientes, entonces para ε>0:*

$$\Pr\left\{\hat{X}_n - \hat{Y}_m - \left(E\left[\hat{X}_n\right] - E\left[\hat{Y}_m\right]\right) \geq \varepsilon\right\} \leq e^{-\frac{2\varepsilon^2}{\mathcal{D}_{n,m}}} \tag{4.7}$$

donde $\mathcal{D}_{n,m} = (b-a)^2 \left[\sum_{i=1}^{n} (v_i)^2 + \sum_{i=1}^{m} (v'_i)^2\right]$.

Consideraremos ahora la hipótesis nula $H_0 : E[\hat{X}_n] \leq E[\hat{Y}_m]$ contra la alternativa $H_1 : E[\hat{X}_n] > E[\hat{Y}_m]$, siendo $\hat{X}_n - \hat{Y}_m > \hat{\varepsilon}_\alpha$ la regla para rechazar H_0, donde α es el nivel de significación y ε_α es la cota de error definida como

$$\hat{\varepsilon}_\alpha = \sqrt{\frac{\mathcal{D}_{n,m}}{2} \ln \frac{1}{\alpha}} \tag{4.8}$$

Obviamente, la ecuación (4.7) es una generalización de la ecuación (4.1) y la última prueba estadística es también una generalización de la prueba A-test presentada en la Sección 4.3. Similarmente, se pueden garantizar cotas para la probabilidad de los errores de tipo I y II. Llamaremos W-test a esta prueba estadística (W porque concierne a promedios ponderados, *Weighted averages*).

Corolario 6 (W-test)**.** *Bajo las condiciones del Corolario 5, para $\hat{\varepsilon}_\alpha$ definido en la ecuación (4.8):*

1. *si* $E\left[\hat{X}_n\right] \leq E\left[\hat{Y}_m\right]$, *entonces* $\Pr\left\{\hat{X}_n - \hat{Y}_m \geq \hat{\varepsilon}_\alpha\right\} \leq \alpha$ *(cota para el error de tipo I),*
2. *si* $E\left[\hat{X}_n\right] \geq E\left[\hat{Y}_m\right] + \varsigma$, *y* $\varsigma > \varepsilon_\alpha$ *entonces*

$$\Pr\left\{\hat{X}_n - \hat{Y}_m < \hat{\varepsilon}_\alpha\right\} \leq e^{\frac{-2(\varsigma-\varepsilon_\alpha)^2}{D_{n,m}}}$$

(cota para el error de tipo II).

Demostración. La demostración es análoga a la del Corolario 3. □

4.4.2 *Esquemas de pesado compatibles con W-test*

A continuación mostraremos un ejemplo motivador donde se obtiene un intervalo de confianza a partir del Teorema 4 para el estadístico EWMA $\hat{X}_n = (1-\lambda)\hat{X}_{n-1} + \lambda X_t$ con respecto a su valor esperado. Para unificar notaciones, equivalentemente definimos $\hat{X}_n$ para $n > 0$ y tomamos $\hat{X}_1 = X_1$.

Ejemplo 7. Cuando $n \to \infty$, si $X_1, X_2, \ldots, X_n$ son variables aleatorias acotadas en el intervalo $[a, b]$, y si $\hat{X}_n$ es el estadístico EWMA, entonces

$$\Pr\left\{\hat{X}_n - E\left[\hat{X}_n\right] \geq \varepsilon\right\} \leq e^{-\frac{2\varepsilon^2(2-\lambda)}{\lambda(b-a)^2}} \tag{4.9}$$

Es fácil demostrar que en el estadística EWMA (ver Ejemplo 7)

$$\lim_{n,m\to\infty} \hat{\varepsilon}_\alpha = (b-a)\sqrt{\frac{\lambda}{\lambda+2}\ln\frac{1}{\alpha}} \tag{4.10}$$

En el área de control de procesos estadísticos, podemos pensar análogamente en $\hat{\varepsilon}_\alpha$ como un límite de control. Para n grande y para el estadístico EWMA, si las variables aleatorias siguen una distribución normal (en este caso denotaremos a $\hat{\varepsilon}_\alpha$ como $\hat{\varepsilon}_\mathcal{N}$), $\hat{\varepsilon}_\mathcal{N} = L\sigma\sqrt{\lambda/(\lambda+2)}$ es un límite de control clásico [Mono1], donde σ es la desviación estándar y L es un factor multiplicativo que define el rendimiento relacionado con los errores de tipo I y II. De esta forma, podemos mostrar la relación asintótica que existe entre α en la prueba W-test y L teniendo en cuenta la ecuación (4.10), precisamente $\alpha = \exp^{-1}(L^2\sigma^2/(b-a))$.

Por ejemplo, asumiendo normalidad, un valor típico $L = 3$ (límites de control de tres veces sigma [Mono1]) hace la probabilidad del error de tipo I aproximadamente igual a 0,001. Sin embargo, considerando variables aleatorias acotadas en el intervalo $[0, 1]$ y el peor caso de la varianza $\sigma^2 = 1/4$; podemos comprobar fácilmente que para n grande, ajustar sobre esta configuración $L = 3$ es similar a configurar $\alpha = 0{,}11$ (i.e. la cota para la probabilidad del error de tipo I) en la prueba W-test.

Por otro lado, otro esquema que ha sido usado con anterioridad es pesar valores pasados acorde a una función exponencial de edad, esto es decir que $w_i = \beta^{n-i}$ $(0 < \beta \leq 1)$. Por ejemplo, $\beta = exp(-\eta)$ [Klio4, CSo6]. Llamaremos el estadístico correspondiente *envejecimiento exponencial*.

Ejemplo 8. Cuando $n \to \infty$, si $X_1, X_2, \ldots, X_n$ son variables aleatorias acotadas en el intervalo $[a, b]$, y si $\hat{X}_n$ es el envejecimiento exponencial; entonces

$$\Pr\left\{\hat{X}_n - E\left[\hat{X}_n\right] \geq \varepsilon\right\} \leq e^{-\frac{2\varepsilon^2(1+\beta)}{(1-\beta)(b-a)^2}}$$

Considere ahora la configuración del Ejemplo 7 y un promedio $\overline{X}$ calculado a partir de las m variables aleatorias más recientes. Teniendo en cuenta la ecuación (4.9) y la desigualdad de Hoeffding, podemos decir que $\Pr(\overline{X} - E[\overline{X}] > \varepsilon)$ y $\Pr(\hat{X}_n - E[\hat{X}_n] > \varepsilon)$ están acotados idénticamente seleccionando a λ en el estadístico EWMA $\hat{X}_n$ como $\lambda = 2/(m+1)$. Análogamente, para el envejecimiento exponencial considerado en el Ejemplo 8 podemos fijar $\beta = (m-1)/(m+1)$. El estadístico EWMA y el envejecimiento exponencial también están relacionados asintóticamente por medio de $\lambda = 1 - \beta$.

Los esquemas de pesado polinomial han sido menos estudiados en la literatura. De cualquier forma, la formulación de promedios ponderados definida anteriormente también es conveniente para algunos esquemas de este tipo. Una solución elegante es definir $w_i = i^p$. Por ejemplo, para valores convenientemente pequeños de $p \in \mathbb{Z}^+$, las sumas $\mathcal{W}_n$ y $\mathcal{D}_n$ pueden ser representadas en una expresión cerrada usando la fórmula de Faulhaber. Así, sería innecesario mantener actualizadas estas sumas ya que para n grande, incluso para valores pequeños de p, estas sumas pueden crecer rápidamente.

4.4.3 *Relación entre A-Test y W-Test*

Comparemos ahora las pruebas A-test y W-test suponiendo que conocemos dónde se encuentra exactamente el punto de cambio. Considere un escenario con un cambio abrupto en la variable aleatoria n-ésima (esto es, todas las variables aleatorias X_i están idénticamente distribuidas, existe un cambio en la media poblacional justo después de obtener a X_n y luego nuevamente todas las variables aleatorias Y_j están distribuidas idénticamente). Informalmente, podemos esperar que $\hat{X}_n \approx \overline{X}$ y que $\hat{Y}_m \approx \overline{Y}$ ya que en este caso $E[\hat{X}_n] = E[\overline{X}]$ y $E[\hat{Y}_m] = E[\overline{Y}]$. Sin embargo, $\varepsilon_\alpha \leq \hat{\varepsilon}_\alpha$ (ε_α fue definido en la ecuación 4.2) ya que $\mathcal{D}_{n,m}$ tiene un valor mínimo si todas las variables $v_i = 1/n$ y todas las variables $v'_i = 1/m$. Consecuentemente, en presencia de cambios abruptos, la prueba A-test debe detectar los cambios más rápidamente que la prueba W-test que concierne a promedios ponderados.

Por otro lado, frecuentemente en los cambios graduales se da el caso en que no todos los valores de entrada tienen la misma importancia, sino que su importancia puede decrecer con el tiempo. El pesado puede ser visto como cambiar la distribución de probabilidades $P(X_i)$ con el objetivo de otorgar mayor peso a los valores con mayor probabilidad de ocurrencia [Klio4]. En esta configuración, puede ser más apropiado usar estimadores que estén relacionados con un esquema de pesado. Sin embargo, surge un problema adicional relacionado con la estimación de los pesos. Como se ha discutido en capítulos anteriores, algunos estudios investigativos han fijado algún parámetro con una evaluación empírica extensa [NFM07], otros pesos han sido adaptados con respecto al rendimiento actual del algoritmo de aprendizaje o a la edad de los valores de entrada [Klio4], y otros pesos han sido calculados para ser óptimos asumiendo una estructura subyacente del cam-

bio [CE02]. En cualquier caso la prueba W-test puede ser aplicada directamente a muchos de estos esquemas de pesado, principalmente debido a la formulación general de los promedios ponderados $\hat{X}_n$ y $\hat{Y}_m$, así como a la desigualdad de probabilidades dada en el Corolario 5.

Finalmente, como ambas pruebas (A-test y W-test) pueden ser ejecutadas con complejidad temporal y espacial constante, solo queda estimar un punto de cambio relevante en dicha secuencia de valores con el objetivo de detectar cambios en línea. Nuevamente es usado un método simple pero efectivo.

4.5 EL ALGORITMO PARA DETECTAR CAMBIOS EN LÍNEA

Un método muy extendido para detectar la ocurrencia de un cambio es monitorizar la evolución de algunos indicadores o medidas. Se pueden considerar muchas opciones para esta monitorización, y un acercamiento que encaja muy bien con el uso de estimaciones de intervalos está basado en la idea del control de procesos estadísticos [BN93, Mono1, GMCR04]. En este acercamiento son definidos dos niveles de cambio, alerta y cambio, sobre la base de la función de densidad de probabilidad de una distribución normal. De esta forma, el método trata de detectar cuándo un estadístico (p) calculado a partir de los últimos valores vistos está significativamente lejos del valor esperado (μ), específicamente fija el nivel de alerta al 95 % ($\Pr\{\mu - 2\sigma \leq p \leq \mu + 2\sigma\} \approx 0{,}95$, donde σ es la desviación estándar) y el nivel de cambio al 99 % ($\Pr\{\mu - 3\sigma \leq p \leq \mu + 3\sigma\} \approx 0{,}99$).

Podemos adaptar esta idea relajando la suposición de normalidad, y sustituyendo la estimación del intervalo que usa la desviación estándar (σ) por otra que fija el nivel de significación deseado (α) y estima el intervalo de confianza (por medio de ε_α o $\hat{\varepsilon}_\alpha$). En nuestro caso, podemos definir dos valores diferentes de confianza, uno correspondiente al nivel de alerta (α_W) y otro correspondiente al nivel de cambio (α_D). En el Algoritmo 1 se presenta un método simple para diferenciar entre tres estados separados: $ESTABLE$, cuando parece no haber cambio; $ALERTA$, cuando se estima que un posible cambio de concepto puede aparecer; y $CAMBIO$ cuando el cambio se identifica claramente. La información dada por este método en la variable $ESTADO$ puede ser usada de muchas formas, y nuestro acercamiento no limita las acciones a ejecutar cuando se estima un estado de alerta o de cambio. Sin embargo, uno de los usos más directos es el siguiente: *a)* si se alcanza el nivel de alerta puede llegar a ocurrir un posible cambio, por lo que los ejemplos vistos a partir de ese momento se almacenan para entrenar a un clasificador alternativo; *b)* cuando se dispara una señal de cambio, un hipotético clasificador alternativo puede reemplazar al más antiguo para adaptar el aprendizaje acorde a los ejemplos almacenados.

Considere una secuencia de variables aleatorias $X_1, X_2, \ldots, X_{corte}, X_{corte+1}, \ldots, X_n$ y el problema de detectar un incremento significativo en el valor de la media de esta secuencia (por ejemplo, un incremento en la tasa de error de un algoritmo de aprendizaje). La primera tarea es estimar un punto de cambio relevante en esta secuencia, que llamaremos *corte*, con el objetivo de aplicar luego la prueba A-test o W-test sobre las muestras $X_1, X_2, \ldots, X_{cut}$ y $X_{cut+1}, \ldots, X_n$. Considere además la expresión $\hat{X}_i + \varepsilon_{\hat{X}_i}$ ($1 \leq i \leq n$), donde $\hat{X}_i$ puede ser lo mismo un promedio que un promedio ponderado, y $\varepsilon_{\hat{X}_i}$ es su correspondiente cota de error calculada por el

entrada:
$x_1, x_2, ...$: flujo de valores potencialmente infinito donde $\forall i,\ x_i \in [a, b]$
$\alpha_W \in (0, 1]$: nivel de significación para el nivel de alerta
$\alpha_D \in (0, 1]$: nivel de significación para el nivel de cambio
salida :
$ESTADO \in \{ESTABLE, ALERTA, CAMBIO\}$

/ Declaración de variables, a n valores reales vistos: */*
$\hat{X}_{corte}$: estadístico calculado a partir de $x_1, x_2, ..., x_{corte}$
$\hat{Y}_{n-corte}$: estadístico calculado a partir de $x_{corte+1}, ..., x_n$
$\hat{Z}_n$: estadístico calculado a partir de $x_1, x_2, ..., x_n$
$\varepsilon_{\hat{X}_{corte}}$, $\varepsilon_{\hat{Y}_{n-corte}}$ y $\varepsilon_{\hat{Z}_n}$: cota de error en correspondencia con el estadístico usado
inicializar() */* variables inicializadas */*
para todo x_i *perteneciente al flujo de valores reales* $x_1, x_2, ..., x_i, ...$ **hacer**
  / actualizar estadísticos e intervalos de confianza */*
  actualizar $\hat{Y}_{i-corte}$, $\hat{Z}_i$, $\varepsilon_{\hat{Y}_{i-corte}}$, y $\varepsilon_{\hat{Z}_i}$ teniendo un nuevo valor real
  / actualizar punto de corte */*
  si $\hat{Z}_i + \varepsilon_{\hat{Z}_i} \leq \hat{X}_i + \varepsilon_{\hat{X}_i}$ **entonces**
    $\hat{X}_{corte} = \hat{Z}_i$ y $\varepsilon_{\hat{X}_{corte}} = \varepsilon_{\hat{Z}_i}$
    inicializar $\hat{Y}_{i-corte}$ y $\varepsilon_{\hat{Y}_{i-corte}}$
  fin
  / determinar el estado actual del flujo de datos */*
  si $H_0 : E[\hat{X}_{corte}] \geq E[\hat{Y}_{i-corte}]$ *es rechazada con nivel de significación* α_D
  entonces
    $ESTADO \leftarrow CAMBIO$
    inicializar()
  fin
  sino si $H_0 : E[\hat{X}_{corte}] \geq E[\hat{Y}_{i-corte}]$ *es rechazada con nivel de significación* α_W
  entonces
    $ESTADO \leftarrow ALERTA$
  fin
  sino
    $ESTADO \leftarrow ESTABLE$
  fin
fin

Algoritmo 1: HDDM: Método para la detección de cambio basado en la cota de Hoeffding.

Teorema 1 o por el Teorema 4. Cuando no existe cambio en la media poblacional, $\hat{X}_i$ se debe mantener aproximadamente constante y consecuentemente $\hat{X}_i + \varepsilon_{\hat{X}_i}$ debe disminuir su valor (en el caso de A-test) o mantenerse aproximadamente constante (para W-test). Por otro lado, cuando ocurre un incremento en la media poblacional, el valor del estimador de la media $\hat{X}_i$ y como consecuencia $\hat{X}_i + \varepsilon_{\hat{X}_i}$ deben incrementarse. Siguiendo esta idea, un punto de cambio en esta secuencia se puede estimar a través del valor mínimo de la expresión $\hat{X}_i + \varepsilon_{\hat{X}_i}$ $(1 \leq i \leq n)$ [Mono1, GMCR04, BdF+06].

De esta forma podemos aplicar la respectiva prueba estadística (A-test o W-test) para estimar el estado actual ($ESTABLE$, $ALERTA$ o $CAMBIO$) del detector de cambio a partir de α_W y α_D sobre las muestras $X_1, X_2, \ldots, X_{cut}$ y $X_{cut+1}, \ldots, X_n$. En este caso, si la hipótesis nula es rechazada con nivel de confianza $1 - \alpha_W$, el estado actual es puesto a $ALERTA$. De forma similar, si la hipótesis nula es rechazada con nivel de confianza $1 - \alpha_D$, el detector de cambio alcanza el nivel de $CAMBIO$ y todos los contadores son reiniciados. Hemos llamado a este método HDDM porque es similar a DDM pero usando un acercamiento basado en desigualdades de Hoeffding para realizar la prueba estadística.

Para ejecutar esta prueba estadística, el Algoritmo 1 mantiene tres contadores ($\hat{X}_{corte}$, $\hat{Y}_{n-corte}$ y $\hat{Z}_n$). Para la prueba A-test, este método puede ser mejorado levemente aplicando el siguiente corolario (equivalente al Corolario 2) derivado de la desigualdad de Hoeffding. De esta forma, solo son necesarios dos contadores (e.g. $\hat{X}_{corte}$ y $\hat{Z}_n$).

Corolario 9. *Si $X_1, \ldots, X_n$, $X_{n+1}, \ldots, X_{n+m}$ son variables aleatorias independientes con valores en el intervalo $[a, b]$, y si $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, $\overline{Z} = \frac{1}{n+m}\sum_{i=1}^{n+m} X_i$, entonces para $\varepsilon > 0$:*

$$\Pr\left\{\overline{X} - \overline{Z} - \left(E\left[\overline{X}\right] - E\left[\overline{Z}\right]\right) \geq \varepsilon\right\} \leq e^{-\frac{2\epsilon^2 n(n+m)}{m(b-a)^2}} \tag{4.11}$$

En este caso, la regla para rechazar la hipótesis nula $H_0 : E\left[\overline{X}\right] \leq E\left[\overline{Z}\right]$ contra la alternativa $H_1 : E\left[\overline{X}\right] > E\left[\overline{Z}\right]$ sería $\overline{X} - \overline{Z} \geq \varepsilon_\alpha$, donde

$$\varepsilon_\alpha = (b-a)\sqrt{\frac{m}{2n(n+m)}\ln\frac{1}{\alpha}}$$

De forma general, ambas pruebas (A-test y W-test) pueden ser aplicadas a otras estrategias de ventana con el objetivo de estimar varios puntos de cambio [DGIM02, BG07]. En este caso, una corrección de Bonferroni para α puede corregir varias pruebas manteniendo acotada la probabilidad para las detecciones falsas [BG07].

Aparte del coste computacional de ADWIN2, las cotas probabilísticas obtenidas en el presente capítulo pueden ser extendidas fácilmente a la estructura de datos usada por ADWIN2, siendo estas cotas más ajustadas que las propuestas por Bifet y Gavaldà [BG07] en un factor constante y bajo las mismas suposiciones teóricas. Por ejemplo, a partir de la ecuación (4.3) y de forma análoga al método usado por Bifet y Gavaldà [BG07], para la estructura de datos usada por ADWIN es fácil comprobar que en este caso ($n > 0$, $m > 0$)

$$\varepsilon_\alpha^{ADWIN} = (b-a)\sqrt{\frac{n^{-1} + m^{-1}}{2}\ln\frac{2(n+m-1)}{\alpha}}$$

Adicionalmente, el método propuesto es más simple que el algoritmo ECDD para detectar cambios por medio del estadístico EWMA (por ejemplo, no es necesario una tabla para almacenar información adicional ni se requieren métodos matemáticos adicionales para estimar intervalos de confianza).

En lo que resta del capítulo estudiaremos empíricamente algunas configuraciones particulares de estos detectores de cambio en línea tomando en cuenta diferentes tipos de cambio y algoritmos de aprendizaje.

4.6 ESTUDIO EMPÍRICO

Varios autores han propuesto medidas de rendimiento para ser consideradas en el diseño y evaluación de algoritmos para la detección de cambios [Mono1, BG07, DKP11, GSaR13]. En esta sección nos enfocaremos en evaluar la tasa de falsos positivos (probabilidad de una detección falsa), tasa de falsos negativos (probabilidad de que no se detecte un cambio dado) y retardo en la detección de cambio, ya que estas medidas son las más consideradas en escenarios de aprendizaje incremental [DKP11].

El método predictivo secuencial es una metodología general para la evaluación de algoritmos de aprendizaje incremental (ver Sección 3.3). Gama y otros [GSaR13] proponen un marco de trabajo para evaluar la calidad de algoritmos de aprendizaje en línea. Ellos defienden el uso del error predictivo secuencial estimado sobre una ventana deslizante para calcular el rendimiento del algoritmo que aprende a partir de flujos de datos en ambientes no estacionarios. Al mismo tiempo, presentan estrategias que se aplican a problemas tanto del aprendizaje supervisado como del no supervisado, dado que una función adecuada de pérdida es definida. Al mismo tiempo estudian propiedades y pruebas estadísticas para evaluar y comparar el rendimiento de los algoritmos.

De esta forma, para evaluar la precisión en flujos de datos artificiales, periódicamente se prueba (cada 100 ejemplos de entrenamiento) el algoritmo de aprendizaje con otros 100 ejemplos usados solo para dicha prueba, tomando ventaja de los generadores de flujos de datos ya que ellos pueden producir un posible número infinito de ejemplos. Obviamente, este método no es adecudo para conjunto de datos reales, por lo que en este caso usamos un acercamiento *test-then-train* [BHKP10, GSaR13]. En este estudio experimental no nos enfocamos en medidas de tiempo y espacio porque todos los algoritmos propuestos tienen complejidad computacional constante. Ellos no almacenan ejemplos sino que la información necesaria es mantenida en un número fijo de contadores.

Los métodos considerados en este estudio experimental son HDDM en combinación con las pruebas A-test y W-test, así como otros que siguen estrategias y características similares. Concretamente usamos como estadístico la media aritmética (por medio de la prueba A-test) y el estadístico EWMA (aplicando la prueba W-test), ambos configurados con $\alpha_W = 0{,}005$ y $\alpha_D = 0{,}001$. También hemos considerado el algoritmo DDM [GMCR04] porque este usa un acercamiento similar y muestra un buen rendimiento; el mismo detecta cambios abruptos y cambios graduales cuando el cambio no es muy lento, pero su rendimiento no es el mismo con cambios graduales lentos [BdF+06]. También incluimos a ADWIN2 [BG07], aunque este no procesa los valores de rendimiento con complejidad constante en

tiempo y espacio, presenta un buen comportamiento y garantías probabilísticas de desempeño; adicionalmente, ADWIN2 ha sido usado como detector de cambio en muchos algoritmos de aprendizaje. Por último incluimos a ECDD [RATH12], el cual es un detector de cambio de concepto basado en el estadístico EWMA.

Previamente hemos mostrado la relación que existe entre la prueba *W*-test usando el estadístico EWMA y el límite de control ampliamente estudiado en la literatura asumiendo que las variables aleatorias están distribuidas normalmente. De la misma forma, establecimos la relación entre el estadístico EWMA y otro esquema de pesado común que denominamos envejecimiento exponencial. También mostramos la relación asintótica entre EWMA y un promedio moviéndose sobre los últimos m valores reales (una ventana deslizante de tamaño fijo). Por esto en este estudio empírico no incluimos algunos de estos algoritmos básicos.

4.6.1 *Rendimiento en flujos de bits*

		100 bits entre cambios			
		[0,1; 0,3]	[0,3; 0,5]	[0,5; 0,7]	[0,7; 0,9]
$\text{HDDM}_{W\text{-test(0.025)}}$	FP	0	0	0	0
	FN	0,67	0,49	0,33	0,09
	RET	16,76	30,22	36,04	40,24
$\text{HDDM}_{W\text{-test(0.050)}}$	FP	0	0	0	0
	FN	0,62	0,16	0	0
	RET	16,05	28,98	19,80	14,68
$\text{HDDM}_{W\text{-test(0.150)}}$	FP	2,22E-05	2,22E-05	1,11E-05	0
	FN	0,75	0,351	0,023	0
	RET	9,99	20,04	16,96	10,71
$\text{HDDM}_{A\text{-test}}$	FP	0	3,33E-05	1,11E-05	1,11E-05
	FN	0,69	0,138	0	0,001
	RET	15,27	28,67	14,05	8,28
DDM	FP	2,33E-04	0,001	0,001	0,003
	FN	0,56	0,088	0,037	0,036
	RET	18,68	24,20	12,86	7,94
ADWIN2	FP	0,004	0,006	0,008	0,023
	FN	0,70	0,67	0,25	0
	RET	12,27	14,91	46,66	32,30
ECDD	FP	0,04	0,05	0,05	0,06
	FN	0,6	0,49	0,53	0,20
	RET	6,70	11,81	11,64	26,12

Tabla 4.1: Valor medio de las medidas de rendimiento en flujos de datos, 100 bits por valor de la media estable.

		1 000 bits entre cambios			
		[0,1; 0,3]	[0,3; 0,5]	[0,5; 0,7]	[0,7; 0,9]
$HDDM_{W\text{-test}(0.025)}$	FP	1,48E-04	1,87E-04	9,78E-05	1,44E-05
	FN	0,088	0,018	0,004	0,002
	RET	105,62	34,52	19,65	15,09
$HDDM_{W\text{-test}(0.050)}$	FP	3,11E-04	3,89E-04	1,86E-04	2,33E-05
	FN	0,149	0,023	0,002	0
	RET	110,68	29,64	14,86	11,71
$HDDM_{W\text{-test}(0.150)}$	FP	2,62E-04	3,24E-04	1,37E-04	7,78E-06
	FN	0,411	0,076	0,003	0
	RET	110,44	56,28	16,95	11,03
$HDDM_{A\text{-test}}$	FP	4,33E-05	4,00E-05	3,22E-05	1,22E-05
	FN	0,022	0,001	0	0
	RET	157,78	40,38	22,02	13,24
DDM	FP	3,08E-04	4,53E-04	6,20E-04	0,001
	FN	0,104	0,003	0,001	0,004
	RET	208,67	109,95	57,34	26,80
ADWIN2	FP	0,004	0,007	0,008	0,009
	FN	0,05	0	0	0
	RET	209,59	56,38	28,26	16,10
ECDD	FP	0,04	0,05	0,05	0,06
	FN	0,6	0,45	0,47	0,16
	RET	23,69	95,61	114,11	260,76

Tabla 4.2: Valor medio de las medidas de rendimiento en flujos de datos, 1 000 bits por valor de la media estable.

En este experimento generamos un flujo de bits acorde a una distribución de Bernoulli con parámetro μ haciendo 30 cambios separados por 100, 1 000 y 100 000 bits (i.e. longitud de conceptos estables, ver Tablas 4.1, 4.2 y 4.3). Los algoritmos se ejecutan 30 veces para cada una de estas situaciones. Así, asumimos que en una ejecución un detector de cambio debe disparar 30 señales de cambio y no más de una señal por media estable. De esta forma podemos estimar la tasa de falsos positivos y falsos negativos. También distinguimos tipos de cambio controlando la diferencia aleatoria entre valores de medias: el valor absoluto de la diferencia entre la media anterior y la nueva está restringido en un intervalo $[a, b]$. Esto es reflejado en las Tablas 4.1, 4.2 y 4.3 separando el rendimiento de cada algoritmo en diferentes columnas. Para cada algoritmo estimamos la tasa de falsos positivos (FP), la tasa de falsos negativos (FN) y el promedio de retardo en la detección de un cambio (RET, que mide por ejemplo, cuán rápido un algoritmo puede detectar un cambio). Las Tablas 4.1, 4.2 y 4.3 muestran el promedio de dichas medidas sobre estas 30 ejecuciones. Debido a la dificultad de estimar estas medidas de rendimiento en

		100 000 bits entre cambios			
		[0,1; 0,3]	[0,3; 0,5]	[0,5; 0,7]	[0,7; 0,9]
$HDDM_{W\text{-test}(0.025)}$	FP	3,24E-04	3,19E-04	1,78E-04	2,86E-05
	FN	0,009	0,012	0,004	0,001
	RET	569,49	112,66	17,37	13,45
$HDDM_{W\text{-test}(0.050)}$	FP	5,17E-04	5,16E-04	2,70E-04	3,50E-05
	FN	0,042	0,011	0,004	0,001
	RET	1071,21	64,43	25,91	11,09
$HDDM_{W\text{-test}(0.150)}$	FP	4,37E-04	4,61E-04	1,73E-04	6,58E-06
	FN	0,183	0,051	0,004	0
	RET	2392,23	186,09	16,25	10,97
$HDDM_{A\text{-test}}$	FP	1,02E-05	9,89E-06	6,32E-06	1,86E-06
	FN	0	0	0	0
	RET	549,02	224,03	145,07	97,26
DDM	FP	5,79E-06	9,70E-06	1,23E-05	1,96E-05
	FN	0,097	0,001	0	0
	RET	13047,91	9413,26	4248,15	2024,57
ADWIN2	FP	8,40E-05	8,22E-05	8,83E-05	9,32E-05
	FN	0	0	0	0
	RET	198,34	53,27	26,66	15,56
ECDD	FP	0,038	0,0497	0,049	0,060
	FN	0,602	0,420	0,463	0,157
	RET	2009,02	9289,00	11177,65	25832,48

Tabla 4.3: Valor medio de las medidas de rendimiento en flujos de datos, 100 000 bits por valor de la media estable.

cambios graduales (por ejemplo, no están bien definidos los puntos de corte ni la cantidad de cambios) este tipo de cambio no es considerado en los flujos de bits.

En este experimento estudiamos diferentes configuraciones del parámetro λ en el estadístico EWMA usando W-test en HDDM (esta combinación es representada como $HDDM_{W\text{-test}(\lambda)}$), la prueba A-test (representado como $HDDM_{A\text{-test}}$), DDM, ADWIN2 con la cota que envuelve aproximación a la curva normal para un tamaño de la muestra grande, y ECDD. Sobre $HDDM_{W\text{-test}(\lambda)}$, $HDDM_{A\text{-test}}$ y ADWIN2 se fija el nivel de significación a $\alpha = 0{,}001$. En ECDD, se fija $\lambda = 0{,}2$ y $ARL_0 = 100$ (que controla la tasa de falsos positivos).[1]

Como se asumen variables aleatorias acotadas, valores de λ cercanos a 1 en $HDDM_{W\text{-test}(\lambda)}$ no son útiles. Podemos comprobar esta observación en la ecuación (4.10). Por ejemplo, como $X_i \in [a, b]$, como mínimo $\lim_{n,m\to\infty} \hat{\varepsilon}_\alpha \leq b - a$, lo que conlleva a $\lambda \leq 2/(\ln(1/\alpha) - 1)$. Como en todos los algoritmos el nivel de significación fue configurado a $\alpha = 0{,}001$, en $HDDM_{W\text{-test}(\lambda)}$ los valores útiles son $\lambda \leq 0{,}33$.

1 Una discusión de esta configuración es ofrecida por Ross y otros [RATH12]

A pesar de esto, empíricamente observamos que las configuraciones para $\lambda \leq 0{,}25$ generalmente funcionan mejor que las configuraciones para $\lambda > 0{,}25$. Por esta razón estudiamos algunas configuraciones del parámetro para $\lambda \leq 0{,}25$.

Similarmente, podemos explicar la tasa de falsos negativos defectuosa del algoritmo HDDM$_{W\text{-test}(\lambda)}$ (principalmente en las Tablas 4.1 y 4.2) en cambios con poca escala (e.g. $|\mu_{old} - \mu_{new}| \in [0{,}1, 0{,}3]$). HDDM$_{W\text{-test}(\lambda)}$ no detecta tales cambios debido a que el intervalo de confianza para este estadístico no converge a cero como el intervalo de confianza de los otros detectores. Sin embargo, HDDM$_{W\text{-test}(\lambda)}$ muestra una tasa baja de falsos positivos para todas las configuraciones del parámetro λ, y muy buen desempeño cuando los cambios en la escala no son tan pequeños.

Acorde a experimentos adicionales, notamos que en el estadístico EWMA las configuraciones para $\lambda \leq 0{,}1$ tienen un buen comportamiento en muchas situaciones de cambio; las variantes para $0{,}1 < \lambda \leq 0{,}2$ no tienen un desempeño tan bueno como el anterior caso. En general, las configuraciones para $\lambda \leq 0{,}2$ tienen un mejor desempeño que las variantes para $\lambda > 0{,}2$.

Como se puede observar en las Tablas 4.1, 4.2 y 4.3, todos los detectores de cambio propuestos mantuvieron la tasa de falsos positivos y falsos negativos por debajo del valor predicho por la teoría. Cuando los valores de la media varían muy frecuentemente (100 bits por concepto), la detección de cambios con poca escala es difícil debido a que los intervalos de confianza no están suficientemente ajustados. Sin embargo, cuando los conceptos se mantienen constantes por un período de tiempo más largo, estas tasas son de hecho mucho más pequeñas que α.

Note que los tipos de cambios abruptos considerados en este experimento favorecen a la prueba A-test con respecto a W-test (ver Subsección 4.4.3). Las Tablas 4.1, 4.2 y 4.3 muestran que HDDM$_{A\text{-test}}$ se desempeña con frecuencia mucho mejor que DDM en las tres medidas de rendimiento, esta diferencia es más notable en el retardo de la detección del cambio cuando los conceptos estables son más duraderos. Note también que el algoritmo ECDD mantiene aproximadamente constante las tasas de falsos positivos y falsos negativos, lo que indica la robustez del método.

Adicionalmente, HDDM$_{A\text{-test}}$ también tiene un desempeño mejor que ADWIN2 cuando los conceptos estables no son tan largos (Tablas 4.1 y 4.2). En ADWIN2, donde se estiman varios puntos de cambio, el retardo en la detección del cambio no es tan susceptible a la longitud de los conceptos estables como cuando se estima un solo punto de cambio. Aparte del coste computacional mayor, para mantener acotada la probabilidad de falsos positivos, el término logarítmico extra $\ln(n)$, donde n es el número de puntos de cambio considerados por ADWIN2 en un momento determinado, puede hacer la cota mucho más conservativa que en HDDM$_{A\text{-test}}$ cuando n es grande. De esta forma, cuando los detectores que consideran un solo punto de cambio (e.g. HDDM$_{A\text{-test}}$) son capaces de estimar precisamente dónde está ubicado este punto (como en las configuraciones de las Tablas 4.1 y 4.2), ellos pueden detectar el correspondiente cambio de concepto más rápidamente debido a que el intervalo de confianza calculado es más ajustado.

Sin embargo, cuando los conceptos permanecen constantes por un período de tiempo más largo (por ejemplo, en la configuración de la Tabla 4.3), ADWIN2 tienen un mejor desempeño que todos los restantes algoritmos (especialmente considerando la medida RET). Aun así, su coste computacional es mucho mayor. Como HDDM$_{A\text{-test}}$ y DDM usan el promedio para estimar los puntos de cambio, los valo-

res más antiguos tienen la misma importancia (peso) que los valores más recientes. Entonces, cuando los conceptos se mantienen constantes por un período de tiempo largo, existe una diferencia notable entre el punto de cambio estimado y el real, conllevando a un retardo considerable en la detección de cambios. En este sentido, un esquema de pesado usado solamente para estimar dichos puntos de cambio puede mejorar el desempeño de HDDM$_{A\text{-test}}$ cuando los conceptos estables son muy largos.

De forma general, para valores pequeños de λ, la longitud de los conceptos estables no es tan determinante en el desempeño de HDDM$_{W\text{-test}(\lambda)}$ como lo es en los detectores HDDM$_{A\text{-test}}$ y DDM. Así, la diferencia en el retardo de la detección de estos cambios entre ADWIN2 y HDDM$_{W\text{-test}(\lambda)}$ es similar para cambios con escalas no tan pequeñas. Al mismo tiempo, esta conducta implica que un nivel de alerta adecuado en los detectores HDDM$_{A\text{-test}}$ y DDM es más efectivo que en los algoritmos ADWIN2, HDDM$_{W\text{-test}(\lambda)}$ con λ cercano a cero, y ECDD.

4.6.2 *Flujos de datos sintéticos*

Este experimento es implementado sobre MOA [BHKP10], un marco de trabajo para el aprendizaje incremental. Este tiene una colección de herramientas de evaluación, una gran variedad de algoritmos inherentes al aprendizaje incremental y muchos métodos para generar flujos de datos artificiales con la posibilidad de incluir cambio de concepto.

Los detectores se evalúan con dos algoritmos de aprendizaje incremental diferentes [BHKP10]: un clasificador Naïve Bayes (NB) y un Perceptrón (P). De esta forma, a la llegada de un nuevo ejemplo, este primero es clasificado tal que su error pueda ser monitorizado. A una señal de alerta, se entrena un clasificador alternativo hasta que vuelva nuevamente una estimación de concepto estable. En el caso que la señal de alerta siga de una señal de cambio, el clasificador alternativo reemplaza al principal. Es posible usar una estrategia más sofisticada para la manipulación de cambio de concepto en situaciones similares (por ejemplo, ver el acercamiento de Bach y Maloof [BM08]), sin embargo, nuevamente seleccionamos la opción más simple.

Como hemos indicado en la sección anterior, en el algoritmo HDDM$_{W\text{-test}(\lambda)}$, todas las configuraciones para $\lambda \leq 0{,}1$ (así como para $0{,}1 < \lambda \leq 0{,}2$) tienen un comportamiento similar. Por esta razón en este experimento solo estudiamos dos variantes para el algoritmo HDDM$_{W\text{-test}(\lambda)}$: $\lambda = 0{,}05$ y $\lambda = 0{,}15$. Obviamente, no garantizamos que alguna de estas configuraciones es óptima, por lo que puede ser posible mejorar la precisión de todos los métodos. Sin embargo, como tal configuración no tiene utilidad en la práctica usamos las anteriores, las cuales tienen un buen desempeño en muchas circunstancias. Considerando los detectores de cambio (HDDM$_{W\text{-test}(0.05)}$, HDDM$_{W\text{-test}(0.15)}$, HDDM$_{A\text{-test}}$, DDM, ADWIN2 y ECDD) y los clasificadores incrementales (Naïve Bayes y Perceptrón) hemos experimentado con doce algoritmos combinando detectores y algoritmos de aprendizaje. Adicionalmente, también hemos incluido los algoritmos de aprendizaje incremental sin ningún método para la detección de cambio.

Precisamente, los algoritmos son ejecutados sobre los siguientes generadores de cambio:

			LED	STA	AGR	HYP	RBF	RTG
$HDDM_{W\text{-test}(0.05)}$	NB	$\overline{x}$	**63,35**	**98,76**	77,36	90,44	74,10	**68,75**
		s	**7,77**	**0,22**	4,46	1,02	1,53	**4,04**
	P	$\overline{x}$	41,07	85,05	58,94	83,65	65,48	55,77
		s	2,42	2,80	3,57	6,16	3,71	3,27
$HDDM_{W\text{-test}(0.15)}$	NB	$\overline{x}$	58,85	94,44	**80,19**	90,47	74,10	71,12
		s	8,51	2,13	**2,62**	1,25	1,53	2,52
	P	$\overline{x}$	41,35	84,49	59,80	84,12	68,18	54,33
		s	1,34	2,59	6,49	6,43	4,65	3,12
$HDDM_{A\text{-test}}$	NB	$\overline{x}$	56,80	98,38	68,98	90,52	74,10	64,54
		s	10,39	0,69	6,95	1,11	1,53	7,18
	P	$\overline{x}$	40,39	84,91	63,25	83,71	68,21	54,00
		s	2,70	2,81	3,52	6,37	4,63	3,74
DDM	NB	$\overline{x}$	54,49	**98,73**	67,76	**91,05**	74,10	63,16
		s	10,54	**0,20**	7,69	**2,12**	1,53	6,25
	P	$\overline{x}$	40,30	85,58	60,27	85,43	68,30	52,01
		s	2,20	2,81	5,00	6,30	4,58	2,74
ADWIN2	NB	$\overline{x}$	50,93	77,99	67,16	87,26	74,10	63,16
		s	12,53	9,07	6,54	3,42	1,53	6,25
	P	$\overline{x}$	40,19	85,66	53,13	84,01	68,17	52,81
		s	1,52	2,88	5,58	6,04	4,62	3,00
ECDD	NB	$\overline{x}$	48,85	98,54	74,67	85,31	58,62	65,32
		s	13,91	0,36	4,30	7,01	3,47	3,99
	P	$\overline{x}$	40,49	82,50	53,25	80,47	55,29	54,29
		s	1,84	3,52	8,39	6,88	4,09	1,58
Sin detector	NB	$\overline{x}$	48,85	72,93	66,92	68,93	74,10	63,16
		s	13,91	5,94	6,78	9,20	1,53	6,25
	P	$\overline{x}$	40,56	85,66	58,98	85,48	68,18	54,88
		s	1,88	2,88	4,00	6,09	4,58	1,62

Tabla 4.4: Promedio y desviación estándar de la precisión en experimentos con 100 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios abruptos.

			LED	STA	AGR	HYP	RBF	RTG
$HDDM_{W\text{-test}(0.05)}$	NB	$\overline{x}$	**63,39**	85,46	**73,76**	**87,10**	75,02	66,91
		s	**5,34**	6,05	**4,36**	**3,08**	1,73	3,27
	P	$\overline{x}$	40,09	77,03	55,31	76,29	68,76	54,61
		s	1,74	3,23	5,59	8,06	3,90	2,04
$HDDM_{W\text{-test}(0.15)}$	NB	$\overline{x}$	61,41	82,48	72,89	85,29	75,05	**67,96**
		s	6,57	3,27	4,83	5,56	1,68	**2,53**
	P	$\overline{x}$	40,23	78,26	59,25	78,28	69,09	54,55
		s	1,62	3,65	4,71	7,79	3,66	1,56
$HDDM_{A\text{-test}}$	NB	$\overline{x}$	55,31	**87,62**	69,58	85,21	**75,17**	61,28
		s	8,64	**4,51**	5,61	5,51	**1,47**	4,15
	P	$\overline{x}$	40,13	77,35	61,76	76,16	69,07	51,38
		s	1,63	3,42	3,48	8,80	3,75	2,12
DDM	NB	$\overline{x}$	47,89	86,75	67,59	83,07	**75,17**	61,28
		s	12,49	5,00	6,20	3,56	**1,47**	4,15
	P	$\overline{x}$	40,45	78,32	59,38	78,82	69,28	50,86
		s	1,47	3,61	3,53	7,94	3,52	3,16
ADWIN2	NB	$\overline{x}$	47,89	75,51	67,08	81,36	**75,17**	61,28
		s	12,49	4,86	5,51	6,20	**1,47**	4,15
	P	$\overline{x}$	39,94	77,70	58,30	79,77	69,12	50,63
		s	1,81	4,13	3,43	7,61	3,68	3,61
ECDD	NB	$\overline{x}$	47,89	80,49	64,76	72,88	64,07	60,43
		s	12,49	8,29	6,06	9,49	3,03	2,16
	P	$\overline{x}$	40,01	69,03	48,07	66,23	52,54	48,41
		s	1,80	5,44	7,35	9,92	1,86	1,31
Sin detector	NB	$\overline{x}$	47,89	72,42	67,08	64,38	**75,17**	61,28
		s	12,49	5,25	5,51	12,43	**1,47**	4,15
	P	$\overline{x}$	40,49	77,70	59,98	79,80	69,14	51,00
		s	1,56	4,13	3,77	7,48	3,65	1,78

Tabla 4.5: Promedio y desviación estándar de la precisión en experimentos con 100 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios graduales. El período de transición entre conceptos consecutivos es de 50 ejemplos.

			LED	STA	AGR	HYP	RBF	RTG
$HDDM_{W\text{-test}(0.05)}$	NB	$\overline{x}$	**76,31**	99,98	**82,54**	91,93	74,30	**69,57**
		s	**1,92**	0,06	**11,62**	3,01	2,19	**3,47**
	P	$\overline{x}$	71,22	89,84	61,18	93,24	71,43	56,73
		s	5,80	12,31	21,71	2,82	3,39	4,75
$HDDM_{W\text{-test}(0.15)}$	NB	$\overline{x}$	75,87	99,89	82,21	91,17	74,69	69,21
		s	1,93	0,77	11,83	5,04	2,11	3,55
	P	$\overline{x}$	71,88	90,09	61,36	93,22	72,47	56,72
		s	6,20	12,00	21,87	2,86	3,64	4,69
$HDDM_{A\text{-test}}$	NB	$\overline{x}$	75,61	**99,99**	81,79	92,30	74,63	66,77
		s	1,81	**0,05**	12,43	2,82	2,10	4,17
	P	$\overline{x}$	75,07	90,10	59,02	93,05	73,82	57,13
		s	4,14	11,58	22,46	3,05	2,68	5,13
DDM	NB	$\overline{x}$	74,46	**99,99**	63,77	92,31	**74,82**	64,70
		s	4,62	**0,05**	15,45	3,13	**1,87**	5,22
	P	$\overline{x}$	75,12	88,78	59,98	93,02	74,34	56,30
		s	4,60	11,11	22,03	3,03	2,49	5,69
ADWIN2	NB	$\overline{x}$	75,27	98,59	81,70	**92,50**	74,70	66,10
		s	1,86	7,88	12,57	**2,74**	1,93	4,80
	P	$\overline{x}$	75,05	90,07	65,95	92,97	73,99	57,33
		s	4,78	11,56	19,14	3,27	2,96	5,22
ECDD	NB	$\overline{x}$	70,98	99,96	74,08	89,27	63,58	63,94
		s	5,03	0,51	15,47	7,43	2,92	3,45
	P	$\overline{x}$	60,25	85,15	47,83	85,71	55,02	52,30
		s	4,31	18,24	23,42	12,90	2,51	3,05
Sin detector	NB	$\overline{x}$	39,86	68,70	61,25	57,69	74,76	53,15
		s	20,20	16,05	14,79	18,41	1,91	4,06
	P	$\overline{x}$	47,04	59,91	59,46	88,59	74,36	52,01
		s	18,22	27,42	22,36	18,42	2,45	7,26

Tabla 4.6: Promedio y desviación estándar de la precisión en experimentos con 10 000 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios abruptos.

			LED	STA	AGR	HYP	RBF	RTG
$\text{HDDM}_{W\text{-test(0.05)}}$	NB	$\overline{x}$	**69,86**	**90,73**	**78,15**	**87,14**	74,63	**68,15**
		s	**6,84**	**10,41**	**11,11**	**9,80**	2,24	**3,70**
	P	$\overline{x}$	64,25	83,39	57,55	87,05	71,39	56,21
		s	9,54	13,14	20,11	10,53	3,35	4,43
$\text{HDDM}_{W\text{-test(0.15)}}$	NB	$\overline{x}$	**69,84**	90,13	77,80	86,56	74,71	67,87
		s	**6,66**	11,33	11,32	10,16	2,05	3,29
	P	$\overline{x}$	64,74	83,76	57,35	87,37	72,35	55,88
		s	10,03	12,79	20,48	10,24	3,60	4,39
$\text{HDDM}_{A\text{-test}}$	NB	$\overline{x}$	68,28	90,06	76,16	87,01	74,91	64,08
		s	7,67	10,64	11,96	10,01	1,99	4,71
	P	$\overline{x}$	67,07	83,76	60,98	87,04	73,86	54,84
		s	10,14	12,72	18,28	10,37	2,47	4,63
DDM	NB	$\overline{x}$	67,33	89,75	72,79	85,83	**75,06**	60,76
		s	7,81	10,75	14,40	11,17	**1,86**	5,87
	P	$\overline{x}$	68,97	83,89	58,98	87,34	74,38	53,65
		s	9,24	12,64	19,44	10,17	2,44	5,78
ADWIN2	NB	$\overline{x}$	66,68	90,35	77,08	**87,15**	75,01	63,43
		s	8,27	10,43	12,13	**9,48**	1,91	5,05
	P	$\overline{x}$	67,28	83,40	62,84	86,78	74,04	56,04
		s	11,15	12,78	17,22	10,78	2,85	5,00
ECDD	NB	$\overline{x}$	67,54	88,68	67,72	82,66	63,66	63,70
		s	6,58	12,61	13,82	12,15	3,13	3,26
	P	$\overline{x}$	58,62	76,29	47,34	79,15	55,15	52,31
		s	5,41	17,36	21,01	15,57	2,42	3,05
Sin detector	NB	$\overline{x}$	39,92	68,55	61,27	58,43	74,94	53,11
		s	16,47	14,19	13,32	17,48	1,90	3,70
	P	$\overline{x}$	48,03	82,84	59,45	87,32	74,41	50,57
		s	14,81	14,36	19,57	11,49	2,43	4,56

Tabla 4.7: Promedio y desviación estándar de la precisión en experimentos con 10 000 ejemplos por concepto y 30 cambios de concepto en flujos de datos artificiales. Se evalúan diferentes detectores de cambio en combinación con algoritmos de aprendizaje en cambios graduales. El período de transición entre conceptos consecutivos es de 5 000 ejemplos.

- LED: El objetivo es predecir el dígito mostrado en una pantalla LED de siete segmentos, donde cada atributo tiene un 10 % de posibilidad de ser cambiado (10 % de ruido). La configuración particular del generador usado para el experimento produce veinticuatro atributos, diecisiete de los cuales son irrelevantes. El cambio de concepto es simulado intercambiando atributos relevantes e irrelevantes.

- STA: Genera conceptos STAGGER introducidos por Schlimmer y Granger [SG86]. Los conceptos son funciones booleanas de tres atributos que codifican a objetos: *tamaño* (*pequeño*, *medio*, y *grande*), *forma* (*círculo*, *triángulo*, y *rectángulo*), y *color* (*rojo*, *azul*, y *verde*). Nuevamente es posible simular cambio de concepto eligiendo entre tres funciones de clasificación distintas: $(tamaño = pequeño) \wedge (color = rojo)$, $(color = verde) \vee (forma = círcuo)$, o $(tamaño = medio) \vee (tamaño = grande)$.

- AGRA: Presenta diez funciones objetivo predefinidas [AIS93]. El generador produce un flujo de datos que contiene nueve atributos, seis numéricos y tres nominales. Las funciones objetivo generan etiquetas de clase binarias a partir de los atributos. El cambio de concepto es simulado cambiando entre funciones objetivo.

- HYP: Representa un hiperplano en un espacio con diez dimensiones [HSD01]. Cada coordenada x_i (dimensión) toma valores en el intervalo $[0; 1]$. Normalizamos la suma de este hiperplano tal que $\sum_{i=1}^{d} w_i = 1$ ($w_i \in [0, 1]$). Precisamente, ejemplos para los cuales $\sum_{i=1}^{d} w_i x_i \geq 0{,}5$ son etiquetados positivos y los que cumple que $\sum_{i=1}^{d} w_i x_i < 0{,}5$ son etiquetados negativos. El cambio es simulado cambiando aleatoriamente los pesos w_i.

- RBF: Genera un número diferente de centroides que son usados para definir funciones de base radial [BHPG09]. Cada centroide tiene una clase asociada a él. La generación de ejemplos lo guían hiperesferas distribuidas normalmente alrededor de cada centroide. En este caso el experimento se configuró con cincuenta centroides, diez atributos y dos clases. El cambio de concepto es simulado moviendo como mínimo diez centroides.

- RTG: Construye un árbol de decisión eligiendo aleatoriamente el atributo a expandir y asignándole una etiqueta de clase aleatoria a cada nodo hoja [DH00]. Cuando el árbol ya está construido, los nuevos ejemplos se generan asignando valores aleatorios distribuidos uniformemente, la clase de estos se determina a partir del árbol. En la configuración particular de este experimento, fijamos diez atributos nominales y la misma cantidad de numéricos. Las hojas empiezan a partir de profundidad tres y tienen una probabilidad de 0,15 de alcanzar una profundidad superior hasta una profundidad máxima del árbol de cinco. Cambiamos aleatoriamente la estructura del árbol (aunque con la misma configuración descrita) para generar cambio de concepto.

Medimos el rendimiento de los algoritmos bajo tipos de cambio graduales y abruptos. Para simular cambios graduales usamos una función sigmoide incrementando la probabilidad que a cada marca de tiempo los ejemplos pertenezcan al nuevo

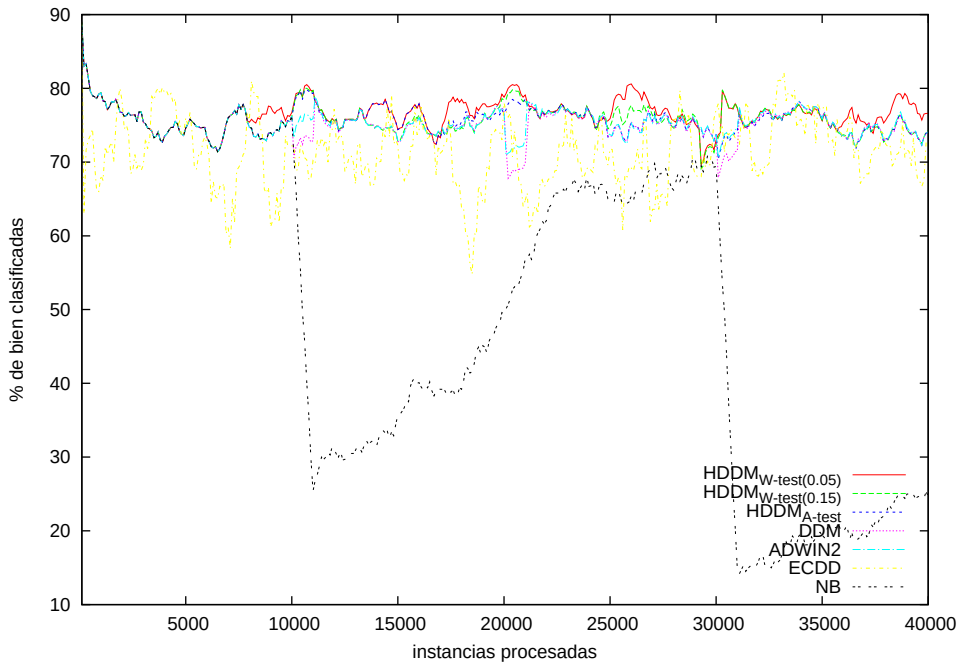


Figura 4.3: Comportamiento típico de la precisión de los algoritmos de detección (con el clasificador Naïve Bayes) en cambios abruptos: 3 cambios en conceptos LED y 10 000 ejemplos por concepto.

concepto [BHKP10]. En estos cambios graduales, configuramos el período de transición entre conceptos a 50 y 5 000 (para 100 y 10 000 ejemplos por concepto respectivamente). Para cada flujo de datos se simularon 30 cambios. Cada algoritmo fue evaluado cada 100 ejemplos con otros 100 ejemplos usados solo para la prueba. Las Tablas 4.4, 4.5, 4.6 y 4.7 muestran la media y la desviación estándar de estas medidas de precisión en una ejecución para cada una de estas configuraciones. Naturalmente, para evaluar el rendimiento de los detectores de cambio no comparamos precisión entre clasificadores diferentes. Las diferencias significativas con respecto al resto de las precisiones de los restantes métodos es mostrada en negritas para el clasificador Naïve Bayes y subrayada para el Perceptrón.

Como esperábamos, el clasificador sin detector de cambio raramente mejora la precisión de cualquiera de los métodos para detectar cambio. El clasificador Naïve Bayes casi siempre es más preciso que el Perceptrón, lo que indica que la frontera de clasificación óptima en la mayoría de los casos es no lineal. Adicionalmente, el clasificador Naïve Bayes por lo general aprende los conceptos más rápidamente que el Perceptrón.

Los conceptos que permanecen constantes por un período pequeño de tiempo son difíciles de manipular, los clasificadores no tienen mucha información para aprender el concepto actual y de esta forma el error de clasificación no se ve tan afectado como consecuencia de un posible cambio. Adicionalmente, en este punto los intervalos de confianza pueden no estar lo suficientemente ajustados para detectar cambios. Si tales cambios no son detectados, los clasificadores aprenden un concepto unificado y su precisión por lo general se deteriora significativamente. En este caso la habilidad del clasificador de aprender rápidamente los conceptos es más influyente. Las Tablas 4.4 y 4.5 reflejan esta situación cuando los cambios

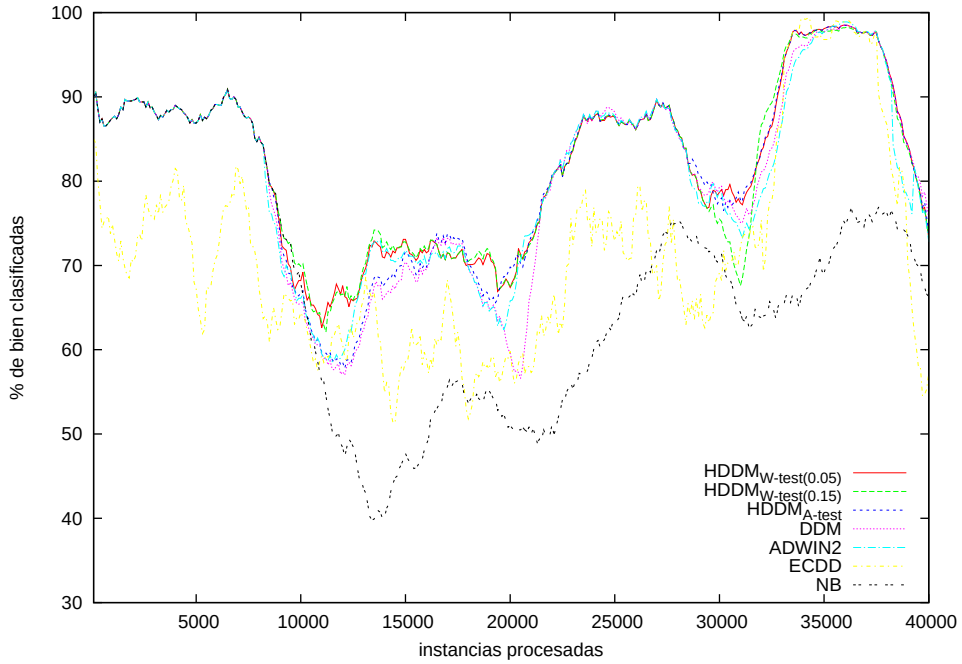


Figura 4.4: Comportamiento típico de la precisión de los algoritmos de detección (con el clasificador Naïve Bayes) en cambios graduales: 3 cambios en conceptos AGRAWAL, 10 000 ejemplos por concepto y 5 000 ejemplos en el período de transición entre conceptos.

ocurren cada 100 instancias. Los detectores de cambio basados en EWMA, que dan más peso a los valores más recientes, tienen un desempeño mejor comparado con los restantes detectores en estas situaciones (particularmente $\text{HDDM}_{W\text{-test}(\lambda)}$, cuyo intervalo de confianza converge rápidamente a su límite, ver Ejemplo 7). Es importante detectar cambios tan pronto como sea posible cuando los conceptos son cortos porque hay un tiempo corto para aprender. El clasificador Naïve Bayes usualmente aprende más rápido estos conceptos. En general, los detectores de cambio basados en EWMA en combinación con el clasificador Naïve Bayes tienen un mejor desempeño que el resto de los métodos por estas razones. Esta diferencia no es tan apreciable cuando se aprende con el Perceptrón (el cual no aprende tan rápidamente). Incluso así, en estas configuraciones $\text{HDDM}_{W\text{-test}(\lambda)}$ nunca detectó todos los cambios (30 cambios) cuando los conceptos permanecían constantes por 100 ejemplos. Notamos además que ECDD tuvo una tasa alta de falsos positivos durante todos los experimentos.

Los algoritmos propuestos también tienen un buen comportamiento en cambios graduales (Tablas 4.5 y 4.7). Note que los flujos de datos con cambios graduales en 100 ejemplos por concepto son más difíciles de manipular. Aun así, todos los detectores de cambio, a pesar de la estrategia simple seleccionada para alternar clasificadores, logran mejorar el desempeño del clasificador sin detector de cambio en muchas ocasiones (principalmente teniendo en cuenta el clasificador Naïve Bayes). Sin embargo, en 100 ejemplos por concepto, los métodos no se desempeñan tan bien en combinación con el Perceptrón, por lo que parece necesaria una estrategia alternativa.

Cuando los conceptos se mantienen constantes por un período de tiempo más largo (10 000 ejemplos por concepto), los detectores de cambio propuestos también tienen un buen desempeño, en especial $\text{HDDM}_{W\text{-test}(0.05)}$. En correspondencia con la teoría y con el experimento que tuvo en cuenta flujos de bits, los métodos propuestos tuvieron una tasa de falsos positivos y falsos negativos pequeña, así como un retardo en la detección del cambio competitivo. También observamos que los algoritmos $\text{HDDM}_{A\text{-test}}$ y DDM detectaron aproximadamente la misma cantidad de cambios durante todos los experimentos, por lo que estimamos que la diferencia en la precisión estuvo dada por la rapidez en la detección de los cambios.

Los falsos positivos no afectan la precisión tanto cuando el clasificador puede aprender el concepto subyacente rápido. Incluso, la precisión puede mejorar con detecciones falsas cuando el clasificador puede aprovechar estas detecciones para aprender pequeñas fluctuaciones en los datos. Sin embargo, en casos como el del Perceptrón, las señales de cambio tienen el coste adicional relacionado con un aprendizaje lento. Por ejemplo, cuando los conceptos permanecen estables por un período de tiempo largo, los detectores de cambio que monitorizan promedios generalmente tienen una precisión más alta que los detectores basados en EWMA en combinación con el Perceptrón, porque los detectores basados en EWMA son más susceptibles al ruido.

En las Figuras 4.3 y 4.4 se muestra el comportamiento típico de la precisión de los algoritmos en presencia de cambio. En estas podemos observar cómo a la llegada de un cambio la precisión se deteriora significativamente con el clasificador Naïve Bayes sin detector de cambio. Aunque la precisión de este en conceptos estables se recupera gradualmente al tratar de aprender un concepto unificado, lo hace muy lentamente. Sin embargo, al aprender con detectores de cambio este deterioro prácticamente no se visualiza en dichas figuras, debido a la rápida detección del cambio de los correspondientes detectores. También podemos observar que en etapas de estabilidad del concepto, los detectores de cambio propuestos mantienen también la precisión estable, lo que implica la robustez de los métodos en la tasa de falsos positivos. Finalmente, es válido destacar que de forma general, una vez que el clasificador con detector de cambio se estabiliza luego de un cambio de concepto, independientemente del tipo de detector el algoritmo de aprendizaje resultante alcanza niveles de precisión similares, por lo que las diferencias significativas en la precisión está determinada fundamentalmente por la detección más rápida en estas etapas de transición entre conceptos.

4.6.3 *Flujos de datos reales*

Los conjuntos de datos reales seleccionados en este experimento han sido usados en diferentes estudios relacionados con cambio de concepto, para estos conjunto de datos, no es posible hacer conjeturas fuertes relacionadas con la presencia o tipo de cambio de concepto. Sin embargo, el beneficio de evaluar los métodos con datos reales explica su presencia. En todos los casos, evaluamos los métodos procesando los ejemplos en línea en su orden temporal. A cada llegada de un nuevo ejemplo el clasificador primero es probado y luego se continúa con su aprendizaje. La Tabla 4.8 muestra el promedio y desviación estándar de la fracción entre el número de ejemplos correctamente clasificados y el total de ejemplos clasificados

			ELEC	SPAM	USE1	USE2	COVE	NURS
$HDDM_{W\text{-test}(0.05)}$	NB	$\overline{x}$	84,47	**91,51**	75,07	70,93	86,23	91,71
		s	6,56	**7,35**	11,51	13,32	8,07	7,58
	P	$\overline{x}$	45,01	97,24	75,07	74,93	0,79	83,65
		s	15,07	3,07	9,95	9,04	5,35	11,68
$HDDM_{A\text{-test}}$	NB	$\overline{x}$	85,09	90,67	75,20	71,00	87,44	**92,51**
		s	6,32	9,26	11,20	12,84	7,96	**6,48**
	P	$\overline{x}$	46,65	97,23	76,93	74,93	1,68	82,62
		s	15,53	3,05	9,15	8,82	9,15	12,00
DDM	NB	$\overline{x}$	82,70	89,50	73,73	**72,93**	88,03	91,72
		s	8,69	13,82	12,26	**11,68**	8,35	7,09
	P	$\overline{x}$	43,45	97,47	74,40	74,93	32,39	83,48
		s	14,47	2,97	11,46	9,12	33,76	10,97
ADWIN2	NB	$\overline{x}$	81,01	91,37	70,27	72,13	83,28	91,90
		s	9,44	7,15	15,79	11,15	10,85	6,93
	P	$\overline{x}$	43,08	97,23	71,80	74,87	4,15	83,91
		s	14,28	3,34	11,54	8,44	15,27	10,98
ECDD	NB	$\overline{x}$	**87,08**	88,03	**76,53**	66,53	**90,52**	84,72
		s	**4,56**	15,20	**8,63**	15,24	**6,97**	7,17
	P	$\overline{x}$	42,44	95,00	71,73	72,93	36,45	65,74
		s	14,00	5,43	10,55	9,97	29,47	17,27
Sin detector	NB	$\overline{x}$	74,17	90,63	63,33	72,13	60,53	83,35
		s	14,67	10,87	22,84	11,15	21,76	14,88
	P	$\overline{x}$	42,44	97,30	73,20	74,73	48,75	75,78
		s	14,00	3,25	11,97	8,23	32,12	19,11

Tabla 4.8: Precisión de detectores de cambio en combinación con el clasificador Naïve Bayes y Perceptrón en conjuntos de datos reales.

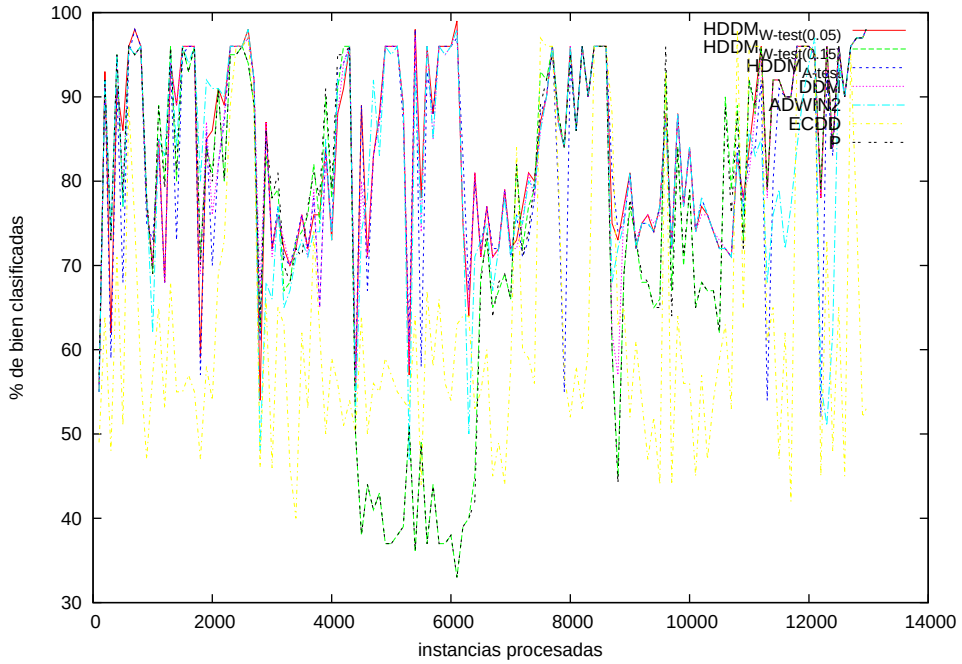


Figura 4.5: Precisión de los detectores de cambio propuestos en combinación con el Perceptrón en el conjunto de datos NURSERY

cada 100 ejemplos procesados; la precisión es calculada con respecto a una ventana deslizando de tamaño 100 [BHKP10, GSaR13]. Experimentamos con los siguientes conjuntos de datos:

- ELEC2 (ELEC):[2] La tarea es predecir si el precio de la electricidad subirá o bajará basado en cinco atributos numéricos: día de la semana, período de treinta minutos del día, la demanda de la electricidad en *New South Wales*, la demanda en *Victoria*, y la cantidad de electricidad a ser transferida entre estos dos distritos de Australia.

- USENET1 (USE1), USENET2 (USE2), SPAM: Los primeros dos conjuntos de datos (USENET1, USENET2) están basados en veinte colecciones de grupos de discusión [FA10]. Ellos simulan un flujo de mensajes de diferentes grupos de discusión que son presentados secuencialmente a un usuario, que los etiqueta como interesantes o los deshecha, acorde a su interés personal. El tercer conjunto de datos (comúnmente conocido como *spam_corpus*2) contiene correos spam y legítimos [KTV08].

- COVERTYPE (COVE): Contiene el tipo de cubierta de un bosque para celdas de 30×30 metros obtenido de los datos de *US Forest Service Region 2 Resource Information System* [FA10].

- NURSERY (NURS): El conjunto de datos fue derivado de un modelo de decisión jerárquico originalmente desarrollado para aplicaciones de asignación de rangos a escuelas de niños. La decisión final depende de tres sub-

2 http://moa.cms.waikato.ac.nz/datasets/

problemas: ocupación de los padres y escuela de los niños, estructura de la familia y estado financiero, y gráfico social y de salud de la familia.

En este experimento omitimos a $\text{HDDM}_{W\text{-test}(0.15)}$ ya que su rendimiento fue superado en la mayoría de los casos por $\text{HDDM}_{W\text{-test}(0.05)}$. También en los datos reales la detección de cambio casi siempre mejora significativamente la precisión de los algoritmos de aprendizaje sin detección de cambio. El Perceptrón tuvo un rendimiento pobre en COVERTYPE, lo que indica alta no linealidad en las fronteras de clasificación de estos conceptos.

ECDD tuvo muy buen rendimiento, principalmente en combinación con el clasificador Naïve Bayes. Como señalan Ross y otros [RATH12], este desempeño con un bajo ARL_0 denota que los cambios en los datos están ocurriendo muy frecuentemente. Esto también demuestra la habilidad del clasificador Naïve Bayes de aprender conceptos rápidamente. Como hemos señalado, cuando el clasificador base no aprende el concepto tan rápidamente, disparar tales señales de cambio tiene el coste adicional de un aprendizaje más lento. Por ejemplo, DDM y $\text{HDDM}_{A\text{-test}}$, los cuales tienen una tasa de falsos positivos baja, muestran un buen desempeño en combinación con el Perceptrón.

La Tabla 4.8 refleja que todos los métodos propuestos también muestran un rendimiento competitivo en comparación con estos algoritmos populares. Al contrario de los experimentos con datos sintéticos, esta vez $\text{HDDM}_{A\text{-test}}$ tuvo con frecuencia mejor precisión que $\text{HDDM}_{W\text{-test}(\lambda)}$. Estimamos que esta conducta está dada porque en estos datos reales muchos conceptos consecutivos son muy similares, causando que los cambios en la tasa de error tengan una escala pequeña. Como explicamos previamente, estos cambios son difíciles de detectar por $\text{HDDM}_{W\text{-test}(\lambda)}$ con el estadístico EWMA. Por ejemplo, $\text{HDDM}_{A\text{-test}}$ usualmente estima menos cambios que $\text{HDDM}_{W\text{-test}(\lambda)}$, pero en estos conjuntos de datos reales generalmente ocurrió lo contrario.

La Figura 4.5 muestra la precisión de los detectores en combinación con el Perceptrón en el conjunto de datos NURSERY. Como se puede apreciar, los puntos de cambio no están bien definidos como en los experimentos con datos artificiales. En estos casos es más difícil mejorar la precisión del algoritmo base ya que no es seguro si un clasificador alternativo entrenado con menos ejemplos mejorará la precisión del clasificador con más ejemplos de entrenamiento. Como hemos señalado, en estos casos, una estrategia alternativa para la manipulación de cambio de concepto probablemente mejore el rendimiento de los algoritmos de aprendizaje.

4.7 CONCLUSIONES DEL CAPÍTULO

En este capítulo hemos presentado una familia de métodos para detectar cambio de concepto en dominios de aprendizaje en línea, monitorizando el rendimiento de un determinado algoritmo de aprendizaje a través de promedios y promedios ponderados. Se ofrecieron garantías teóricas de rendimiento para los métodos propuestos usando desigualdades de probabilidad que no asumen conocimiento relacionado con la función de distribución de probabilidad que rige a estos valores de rendimiento. En los promedios ponderados hemos estudiado al estadístico EWMA, que es un estimador óptimo de la media para algunos procesos estadísticos y

un estimador efectivo para varios otros. Sin embargo, debido a la generalidad del método propuesto, este se puede aplicar a muchos otros esquemas de pesado.

Los métodos propuestos no dependen del algoritmo de aprendizaje y consecuentemente ellos pueden ser aplicados a cualquier clasificador para la manipulación de cambio de concepto. Otro aspecto relevante en el procesamiento de flujos de datos es su complejidad temporal y espacial constante. Todos los métodos propuestos son capaces de mejorar la precisión de algoritmos de aprendizaje cuando se modelan problemas no estacionarios.

También iniciamos un estudio de cómo los detectores propuestos responden a diversas características del cambio. En particular probamos nuestros detectores de cambio en flujos de bits, así como en conjunto de datos sintéticos y reales. En flujos de bits encontramos empíricamente las tasas de falsos positivos, falsos negativos y el retardo de la detección de cambios, valores que estuvieron en correspondencia con lo predicho en la teoría. En conjuntos de datos sintéticos y reales probamos los algoritmos monitorizando el error del clasificador Naïve Bayes y Perceptrón. Una señal de alerta se usó para entrenar a un clasificador alternativo que remplazaría al original si dicha señal es seguida de una señal de cambio.

Una observación importante es que cuando el clasificador aprende rápidamente los conceptos subyacentes esta estrategia simple funciona muy bien. Sin embargo, cuando existe el coste adicional de un aprendizaje lento (como es el caso del Perceptrón o como puede pasar con árboles de decisión) esta estrategia no funciona tan bien, por lo que es necesario un nuevo enfoque. Adicionalmente, estos detectores pueden ser aplicados a algunos algoritmos de aprendizaje con otras estrategias que tengan en cuenta características específicas del modelo, para así manipular cambio de concepto más efectivamente. Por ejemplo, en algunos algoritmos de aprendizaje es posible monitorizar la consistencia de partes del modelo de aprendizaje para cambiar la estructura de estas partes sin cambiar el modelo completo, como es el caso de los árboles de decisión y reglas de decisión, permitiendo una adaptación al cambio más rápida.

Los resultados de la evaluación empírica sobre estos conjuntos de datos mostraron a los métodos propuestos para la detección de cambio prominentes.

5

IADEM-2C: MEJORANDO LA INDUCCIÓN DE ÁRBOLES DE DECISIÓN

Los algoritmos y modelos basados en árboles de decisión han incentivado gran cantidad de estudios en múltiples disciplinas como la estadística, la teoría de la decisión, el procesamiento de señales, y el aprendizaje automático [Mur98]. Sin embargo, la mayoría de estos estudios han estado enfocados en el aprendizaje por lotes, mientras que en los últimos años se han incrementado las investigaciones en el área del procesamiento incremental de flujos de datos [DH00, Ram01, del07, RPDJ13, RJPD14].

Los árboles de decisión son un modelo de aprendizaje popular debido a sus características: modelo altamente interpretable, construcción rápida y simple, alta precisión, adecuado para problemas con muchas dimensiones, entre otras. Esencialmente, un clasificador basado en árboles de decisión expande nodos hoja recursivamente convirtiéndolos en nodos de decisión, hasta que ninguna hoja pueda expandirse. Una prueba de división se instala en cada nodo de decisión, la cual divide al conjunto de entrenamiento en subconjuntos. Una de las pruebas de división más efectiva envuelve un solo atributo, y dicha partición depende de los valores del atributo asignados a los arcos que conectan al correspondiente nodo de decisión con sus hijos. Así, un aspecto clave en la inducción incremental de árboles de decisión es la elección del mejor atributo para la expansión, con el objetivo de obtener un árbol cercano al óptimo. Esta elección está basada frecuentemente en alguna medida heurística.

Para resolver este problema, algunos algoritmos usan cotas de concentración para determinar un número apropiado de ejemplos de entrenamiento necesarios para tal elección. Podemos encontrar varios algoritmos que usan diferentes cotas de concentración: la cota de Hoeffding [DH00, RM00, Ram01, del07, dRGM08], de Chernoff [RM00, Ram01, del07, dRGM08] o de McDiarmid [RPDJ13]. Como hemos señalado, un algoritmo popular es VFDT [DH00], pero su justificación matemática no es correcta [RPDJ13]. Rutkowski y otros [RPDJ13] también proponen una familia de algoritmos basados en la cota de McDiarmid, pero la estimación por intervalos es muy conservativa para las medidas heurísticas consideradas (ganancia de información e índice Gini) y estos algoritmos requieren, en general, grandes cantidades de datos para alcanzar niveles adecuados de precisión.

La familia de algoritmos de inducción de árboles de decisión IADEM [del07, dRGM08, RM00, Ram01, RMd04, RdCM05] olvida todas las experiencias usando una representación probabilística [SM81], lo que permite su aplicación en condiciones de flujos de datos. Esta familia usa las cotas de concentración de Hoeffding [Hoe63] y de Chernoff [Che52] tal que con una determinada confianza, los parámetros reales de la población estén dentro dentro del intervalo de confianza calculado. IADEM-2 [del07, dRGM08] mejora la calidad de algoritmos previos de la familia IADEM. Estas mejoras incluyen más tolerancia al ruido, mayor rapidez en la con-

vergencia del modelo, reformulación de la idoneidad del mejor atributo para la expansión, y aumento de la precisión usando el clasificador Naïve Bayes.

Aunque IADEM-2 presenta características interesantes, también tiene algunas limitaciones. Por ejemplo, IADEM-2 usa un árbol binario exhaustivo [GRM03] para manipular atributos numéricos, el cual almacena cada valor visto. De esta forma, su coste computacional es alto cuando los atributos numéricos pueden tomar un número infinito de valores diferentes. Por otro lado, algunos métodos de discretización pueden manipular atributos numéricos en la inducción incremental de árboles de decisión [BHKP10], pero su aplicación directa puede tener un coste computacional alto. IADEM-2 tampoco contempla pruebas de división binarias en los nodos de decisión, por lo que no es capaz de inducir árboles binarios. Quinlan [Qui86] indica que en las pruebas múltiples, donde la prueba de división tiene tantas salidas como el número de valores del atributo nominal que dicha prueba contempla, los atributos nominales con más valores son favorecidos por la ganancia de información, incluso cuando un atributo dado no es tan relevante para el problema subyacente. Adicionalmente, las pruebas de división binarias son más generales que las pruebas múltiples y de esta forma pueden modelar problemas más complejos. Este capítulo está dirigido a resolver estas limitaciones. Al algoritmo resultante lo nombramos IADEM-2c.

Para ello, primeramente mostramos las características fundamentales de las cotas de Hoeffding y Chernoff, así como del algoritmo IADEM-2. Luego presentamos la solución propuesta y evaluamos al algoritmo resultante en conjuntos de datos artificiales y reales.

5.1 COTAS DE CONCENTRACIÓN DE CHERNOFF Y DE HOEFFDING

En la estimación estadística se determina el valor aproximado de un variable a partir de resultados muestrales. Las cotas de concentración constituyen uno de los enfoques para calcular el intervalo en el que debe encontrarse el valor real de dicha variable con una determinada confianza $(1-\delta)$.

Sea X una variable aleatoria y sea x un valor real, las probabilidades $Pr[X < x]$ y $Pr[X > x]$ son conocidas como las colas izquierda y derecha respectivamente. Las cotas de concentración ofrecen cotas superiores para las colas izquierda y derecha de la variable aleatoria $X - E[X]$. En general, las cotas de concentración sirven para probar que la diferencia entre un valor estimado y el valor real de una variable se encuentra acotada en un intervalo. Las desigualdades de Chernoff y de Hoeffding (ver Teorema 1) han sido usadas con efectividad para dichas estimaciones en ambientes de flujos de datos.

Teorema 10 (Teorema de Chernoff)**.** *Sean $X_1, X_2, ..., X_n$ variables aleatorias independientes que se corresponden con experimentos de Bernoulli. Cada variable aleatoria X_i tiene una probabilidad de éxito igual a p_i y una probabilidad de fracaso igual a $q_i = 1 - p_i$. Sea $X_s = \sum_{i=1}^{n} X_i$ una variable aleatoria cuyo valor esperado es $\mu_s = \sum_{i=1}^{n} p_i$. Entonces para cualquier $\beta > 0$,*

$$Pr[X_s > (1+\beta)\mu_s] \leq \left(\frac{e^{\beta}}{(1+\beta)^{1+\beta}}\right)^{\mu_s}$$

y para el caso en que $0 < \beta < 1$*:*

$$Pr[X_s > (1-\beta)\mu_s] \leq \left(\frac{e^{-\beta}}{(1-\beta)^{1-\beta}}\right)^{\mu_s}$$

La familia de algoritmos IADEM utiliza una formulación explícita más sencilla usada por Zhang [Zha02]. En este sentido, el siguiente corolario se usa para determinar el error de estimación absoluto en esta familia.

Corolario 11. *Sean* $X_1, X_2, ..., X_n$ *variables aleatorias independientes que se corresponden con experimentos de Bernoulli. Cada variable aleatoria* X_i *tiene una probabilidad de éxito igual a* p_i *y una probabilidad de fracaso igual a* $q_i = 1 - p_i$*. Sea* $X_s = \sum_{i=1}^{n} X_i$ *una variable aleatoria cuyo valor esperado es* $\mu_s = \sum_{i=1}^{n} p_i$*.*

$$Pr[X_s > (1+\beta)\mu_s] \leq \begin{cases} e^{-\frac{\beta^2\mu_s}{2+\beta}} & si\ \beta \geq 1 \\ e^{-\frac{\beta^2\mu_s}{3}} & si\ 0 < \beta < 1 \end{cases}$$

$$Pr[X_s < (1-\beta)\mu_s] \leq e^{-\frac{\beta^2\mu_s}{2}} si\ 0 < \beta < 1$$

Las cotas de Chernoff y Hoeffding no dependen de la distribución de probabilidad que genera las observaciones, exigiéndose únicamente que las observaciones sean independientes. Lógicamente, este aspecto las hace muy útiles en problemas en los cuales es necesario no presuponer ningún tipo de distribución. No obstante, el número de observaciones para alcanzar cotas ajustadas es mayor que usando cotas dependientes de la distribución.

Existen diferencias entre las cotas de Chernoff y Hoeffding que hacen usar enfoques diferentes para determinar el error de estimación calculado por ellas. La cota de Chernoff está expresada en forma multiplicativa y el error cometido se presenta mediante una aproximación relativa. Por su parte, la cota de Hoeffding se expresa en forma aditiva y el error cometido se presenta mediante una aproximación absoluta. Otra diferencia a destacar es que mientras que la cota de Chernoff hace los cálculos usando la suma de los eventos, la cota de Hoeffding usa directamente el valor medio. Por último y no menos importante, hay que considerar que para calcular la cota de Chernoff es preciso conocer el valor esperado para la suma de las variables, mientras que para la cota de Hoeffding sólo es preciso conocer el número de observaciones que se han registrado.

Otro enfoque interesante es usar las cotas de concentración para el muestreo secuencial (o adaptativo). En este caso, no se usa el tamaño de la muestra completa para determinar los intervalos de confianza, ni se fija el tamaño de la muestra necesario para hacer afirmaciones con determinada confianza. Con el enfoque de muestreo secuencial lo que se pretende es muestrear experiencias de forma constante hasta que el nivel de confianza exigido sea alcanzado. Esta es la razón para que el muestreo secuencial también se conozca como adaptativo: el tamaño necesario de la muestra dependerá de las experiencias que se hayan muestreado y variará según el problema. Ramos [Ram01] plantea este enfoque aplicándolo a la tarea de inducir árboles de decisión en el algoritmo IADEM-0, en este caso los intervalos definidos para las variables son calculados combinando las cotas de Chernoff y Hoeffding. La razón para usar ambas cotas es que los intervalos son más ajustados para una cota u otra dependiendo de las condiciones del problema, siendo

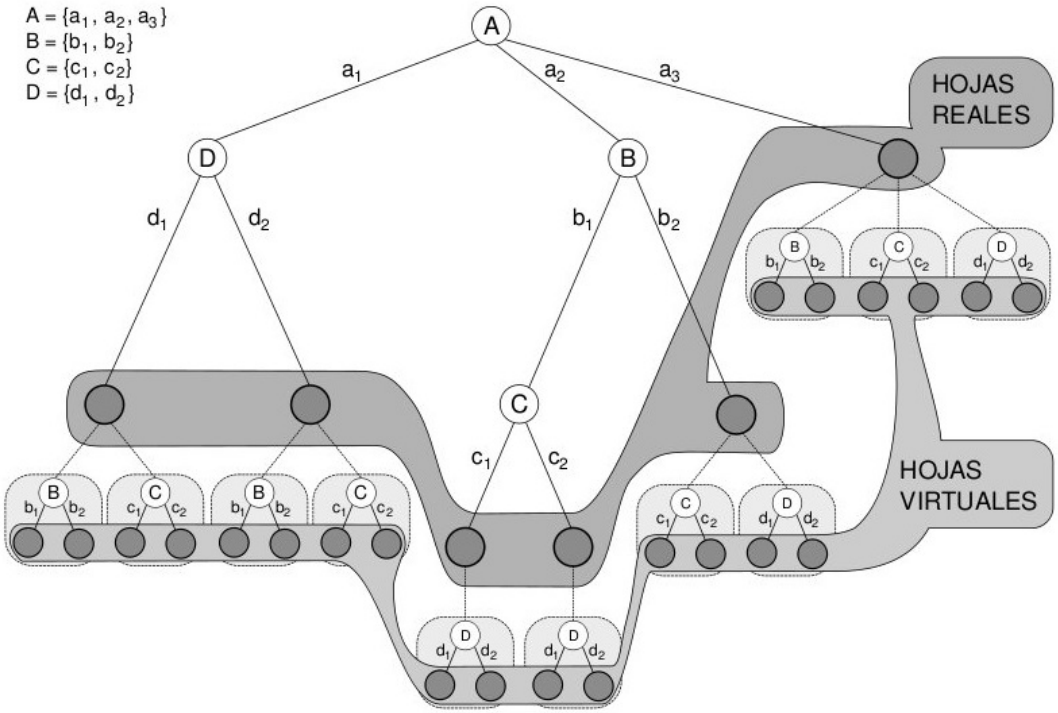


Figura 5.1: Representación de un árbol de decisión inducido por la familia IADEM [delo7].

en algunas circunstancias más ajustada la cota de Chernoff que la de Hoeffding y viceversa.

5.2 LA FAMILIA DE ALGORITMOS IADEM

En los árboles de decisión que induce la familia de algoritmos IADEM, cada nodo de decisión se etiqueta con un atributo y se define una frontera de hojas virtuales. En la Figura 5.1 se muestra un posible árbol de decisión generado por el algoritmo, en este caso considerando solo atributos nominales. El problema que aborda presenta cuatro atributos para explicar la estructura del mismo. Las hojas que forman la frontera real del árbol se denominan *hojas reales* y a todas las posibles hojas futuras *hojas virtuales*. En las hojas (tanto reales como virtuales) se incluyen contadores que permiten al algoritmo estimar un conjunto de variables con las que inducir el árbol de decisión. Una vez que se ha determinado la hoja real que va a ser expandida y se ha seleccionado el atributo que se va a usar para realizar la expansión, el proceso de expansión consiste en eliminar la hoja real y sustituirla por un nodo interno (o nodo de decisión) etiquetado con el atributo seleccionado. Las hojas virtuales que contenían información para ese atributo se convierten en nuevas hojas reales y descendientes del nuevo nodo de decisión. Al realizar el proceso de expansión de esta forma, el nuevo modelo resultante dispone de información previa (la almacenada en lo que antes eran hojas virtuales y ahora son reales) y no existe ninguna rama (camino desde la raíz hasta una hoja) sin información.

En el Algoritmo 2 se presentan los pasos generales de la inducción. La primera acción a realizar es la inicialización de la estructura (línea 1) cuyo objetivo es generar un árbol de decisión con un único nodo, que será una hoja real de la que

entrada:
ε: Error máximo tolerado;
δ: Confianza para el error;
E: Conjunto de datos;
salida : Árboles de decisión

```
1 Inicializar()
2 mientras ¬ Condición_Parada() ∧ ¬ Completamente_Expandido() hacer
3 |  Muestrear()
4 |  Recalcular()
5 |  si Condición_Expansión() ∧ ¬ Condición_Parada() ∧
  |  Es_Expansible(PEOR_HOJA) entonces
6 |  |  Expandir(PEOR_HOJA)
7 |  fin
8 fin
```

Algoritmo 2: Algoritmo base para la inducción en la familia IADEM.

dependerán tantas hojas virtuales como atributos y valores hayan sido definidos en la descripción del problema. Todos los contadores asociados a las hojas (reales o virtuales) serán inicializados a cero.

En la parte principal del algoritmo se encuentra una acción iterativa que será la que inducirá el árbol de decisión y que se repetirá mientras no se alcance la calidad exigida. Dicha acción iterativa se define con dos acciones fundamentales: la observación de nuevas experiencias ($Muestrear()$ y $Recalcular()$) y la expansión del árbol ($Expandir(PEOR\_HOJA)$) si se considera necesaria.

La información relevante de las experiencias observadas se conserva usando una representación probabilística soportada por la estructura del árbol de decisión. Usando esa información, el algoritmo puede calcular una estimación (en el mejor y en el peor caso) del error de clasificación cometido por el árbol. Con esa estimación se le puede ofrecer al usuario la posibilidad de que él establezca el error máximo tolerado (ϵ) para el problema que está afrontando, indicando también la confianza deseada para la estimación ($1 - \delta$).

La función $Condición\_Parada()$ controla que la precisión estimada del árbol sea la deseada, para ello considera el error máximo tolerado por el usuario (ϵ) y la estimación del error cometido por el árbol en el peor caso. El segundo motivo para detener la inducción del árbol es que el árbol no pueda seguir expandiéndose ($Completamente\_Expandido()$), es decir, que no queden más atributos libres por los que expandir.

Por otro lado, $Condición\_Expansión()$ y $Es\_Expansible()$ son usados para determinar la idoneidad de expandir el árbol en un determinado momento. La primera condición está diseñada para no expandir bajo cualquier circunstancia, retrasar la expansión a momentos en los que sea más útil y conseguir así árboles de decisión de menor tamaño. La segunda condición afecta únicamente a la hoja seleccionada para expandir el árbol ($PEOR\_HOJA$) y comprueba que exista suficiente evidencia de que el mejor atributo para expandir la hoja ha sido claramente identificado.

A pesar de su buen rendimiento, IADEM-0 carece de algunas características que limitan su uso: su funcionamiento puede no ser adecuado cuando trabaja con

conjuntos de datos en los que haya ruido, no es capaz de procesar directamente experiencias con atributos numéricos (precisándose una discretización estática previa), y no usa toda la información disponible en el árbol para realizar predicciones más fiables. Estas limitaciones han sido superadas junto con otras mejoras que han sido incorporadas en el diseño del algoritmo IADEM-2. Un paso intermedio entre IADEM-0 e IADEM-2 fue el desarrollo del algoritmo IADEM [RdM06] y del algoritmo IADEMc [dCRM06] en el que se solucionan parcialmente los defectos. El algoritmo que resuelve los problemas antes descritos y mejora sustancialmente el proceso de expansión es IADEM-2 [del07] y es el algoritmo en el cual nos enfocaremos en este capítulo.

El algoritmo IADEM-2 mejora la convergencia del modelo, incrementa la calidad de los modelos inducidos y aumenta la precisión. La calidad de los modelos inducidos es mejorada por un nuevo mecanismo que evalúa la idoneidad del mejor atributo para ser expandido, por la incorporación de técnicas que permiten procesar ejemplos con atributos continuos de forma directa (adaptando la propuesta de Gama y otros [GRM03]) y por el establecimiento de un desbalanceo máximo antes de poder expandir una hoja.

Las características fundamentales de esta familia de algoritmos se resumen a continuación:

ESTIMACIÓN DE INTERVALOS DE CONFIANZA Usa las cotas de Hoeffding (ε_H) y de Chernoff (ε_C) para calcular el intervalo de confianza final (ε_{IADEM}) para la media de un conjunto de variables aleatorias independientes ($\overline{X}_n = \sum_{i=1}^{n} X_i$):

$$\begin{aligned} \varepsilon_H &= \sqrt{\frac{1}{2n} \ln (2/\delta)} \\ \varepsilon_C &= \frac{1}{2n} \left(3 \ln \frac{2}{\delta} + \sqrt{9 \ln^2 \frac{2}{\delta} + 12 n \overline{X}_n \ln \frac{2}{\delta}} \right) \\ \varepsilon_{IADEM} &= \begin{cases} 1 & n = 0 \\ \min \{\varepsilon_H, \varepsilon_C, 1\} & n > 0 \end{cases} \end{aligned}$$

ELECCIÓN DE VARIABLES ALEATORIAS INDEPENDIENTES Asume como variables aleatorias independientes un flujo de valores booleanos (e.g. 1 si un ejemplo tiene una etiqueta de clase dada y 0 en otro caso). En los nodos hoja se calculan y almacenan frecuencias relativas correspondientes a estas variables aleatorias. Así, cada frecuencia relativa tiene asociado un intervalo de confianza. De esta forma, incluso con una sola estructura, podemos decir que la familia de algoritmos IADEM mantiene una variedad de árboles diferentes porque cada frecuencia relativa no es única (están en un intervalo de confianza). Por cada posible combinación de valores tenemos un posible árbol, y por cada árbol un vector cuyos componentes son frecuencias relativas, error al que contribuye cada una de sus hojas, etc. Cada vector debe satisfacer determinadas restricciones; por ejemplo, la suma de las probabilidades de alcanzar un nodo hoja debe ser 1.

ELECCIÓN DEL MEJOR ATRIBUTO A EXPANDIR Como cada frecuencia relativa en los nodos hoja tiene asociada un intervalo de confianza, un algoritmo propuesto por Ramos y Morales [RM00] calcula los intervalos de confianza para

cualquier medida de desorden. El mejor atributo (el de menor valor para la medida de desorden dada) se compara con el peor atributo asumiendo que la determinación estricta del mejor atributo no es determinante para alcanzar buenos árboles, siempre y cuando el mejor atributo sea realmente bueno. IADEM-2 expande por el mejor atributo si el intervalo de confianza de la medida de desorden en ambos nodos son muy diferentes (e.g. se destaca un atributo bueno) o muy similares (implicando que todos los atributos tienen importancia similar y la hoja puede expandirse por cualquiera de ellos).

5.3 MEJORA DE LA CALIDAD DE LAS EXPANSIONES

Para construir un árbol de decisión es necesario encontrar una prueba para cada nodo de decisión que divida los datos de entrada en subconjuntos. La tarea básica en la construcción de un árbol de decisión es ordenar los atributos (uno o una combinación de estos) acorde a su utilidad en discriminar las clases a partir de los datos de entrada. En el aprendizaje automático y la minería de datos, los atributos son usualmente ordenados acorde a su utilidad usando *reglas de evaluación de atributos*, y el mejor atributo o un buen subconjunto de estos se elige para formar parte de la prueba a instalar en un nodo de decisión. Las reglas de división son esencialmente heurísticas y evalúan la dependencia entre los atributos y la clase. Algunos autores han dividido estas reglas en tres categorías: las que se derivan desde la teoría de la información, reglas derivadas de medidas de distancia y derivadas de medidas de dependencia [Mur98].

Las reglas de evaluación de atributos son usadas más frecuentemente para elegir un solo atributo en cada nodo de decisión, como es el caso de la familia IADEM. Estas reglas son más simples y pueden ser derivadas con menos coste computacional, también han demostrado ser efectivas en las tareas de predicción [Kiro7, DHoo, RPDJ13, RJPD14, RMoo, delo7]. Una característica interesante de la familia IADEM es que a esta pueden ser incorporadas de forma natural varias reglas de evaluación que hacen uso de la información contenida en los nodos hoja y nodos virtuales.

Otro aspecto clave está relacionado con el número de salidas de la prueba a instalar en un nodo de decisión dado. Consideraremos, como es común, que los nodos de decisión tienen tantos hijos como salidas tiene su prueba correspondiente. En este caso, podemos distinguir las pruebas que solo tienen dos posibles salidas (que llamaremos binarias), y las que tienen más de dos salidas (no binarias o múltiples). Un árbol de decisión donde todas las pruebas tienen dos salidas da lugar a un árbol binario.

Dentro de las pruebas binarias y que consideran un solo atributo, una posible variante es dividir los valores de los atributos es dos subconjuntos disjuntos, cuya unión contemple todos los valores del atributo en cuestión. Estos dos subconjuntos pueden definir las dos posibles salidas de la prueba, en dependencia de si un ejemplo dado tiene el valor de ese atributo en un subconjunto o en el otro [RPDJ13, Qui86]. La tarea para el algoritmo de aprendizaje en este tipo de pruebas es encontrar dichos subconjuntos de forma tal que la división sea la mejor para la regla de evaluación de atributos dada. La principal desventaja de esta prueba es su coste computacional cuando el número de valores de un atributo dado es grande,

ya que el número de variantes a considerar es $2^n - 1$, quitando el caso simétrico cuando uno de los subconjuntos es vacío [Qui86].

En el aprendizaje incremental, para atributos nominales se ha usado la prueba binaria del tipo "$¿a_{ij} = x?$", donde a_{ij} es el valor del atributo a_i de un ejemplo dado, y x es uno de los posibles valores del atributo. En este caso, solo es necesario analizar la cantidad de valores de ese atributo nominal para estimar la mejor prueba.

Como hemos señalado, las pruebas binarias tienen algunas ventajas sobre las no binarias [Qui86]. A favor de las pruebas no binarias podemos decir que estas por lo general dan lugar a árboles de decisión más pequeños e interpretables, que es una de las ventajas fundamentales de los árboles de decisión. De esta forma, recientemente se ha propuesto combinar estas dos opciones en árboles de decisión para aprovechar sus ventajas [BHKP10], a este tipo de variante lo llamaremos pruebas mixtas. En este caso se evalúa cuál prueba (la binaria o la no binaria) es la mejor opción a la hora de expandir un nodo hoja determinado. Por ejemplo, al usar la ganancia de la precisión y considerando la observación de Quinlan [Qui86], esta variante se puede interpretar como que favorece la interpretabilidad y el tamaño pequeño de los árboles, instalando solo pruebas binarias cuando muestren ser significativamente una mejor opción.

Considerando lo anteriormente planteado, hemos extendido a IADEM-2 para contemplar pruebas binarias del tipo "$¿a_{ij} = x?$". De esta forma, dicha extensión hace que en los árboles inducidos por IADEM-2 puedan ser binarios o considerar de forma mixta ambas pruebas.

5.4 TRATAMIENTO DE ATRIBUTOS NUMÉRICOS

La habilidad de aprender a partir de atributos numéricos es importante ya que muchos problemas del mundo real se expresan a través de atributos numéricos. En el aprendizaje por lote la manipulación de atributos numéricos a través de árboles de decisión puede ser directa, ya que cada valor numérico es almacenado en memoria y está disponible para su posterior uso. En el aprendizaje por lotes, la manipulación de atributos numéricos ha sido ampliamente estudiada. Algunos algoritmos manipulan atributos numéricos de forma natural, como las máquinas de soporte vectorial, mientras que otros son más adecuados para valores discretos, como los árboles de decisión. En este caso, si el algoritmo de aprendizaje no puede manipular atributos numéricos de forma natural, también los datos numéricos pueden convertirse en discretos en la etapa de preprocesamiento, lo que es referido como discretización. Una categorización importante de los métodos de discretización tiene en cuenta cuatro dimensiones: métodos locales contra globales, supervisados contra no supervisados, estáticos contra dinámicos y paramétricos contra no paramétricos [Kir07, BHKP10].

Los métodos globales trabajan sobre el conjunto de ejemplos completo, mientras que los locales trabajan con un subconjunto de estos. Por ejemplo, los algoritmos de aprendizaje basados en árboles de decisión generalmente trabajan con métodos locales, dado en el conjunto de ejemplos es dividido en regiones más pequeñas acorde a las pruebas instaladas en los nodos de decisión.

Los métodos supervisados tienen en cuenta la etiqueta de clase de los ejemplos para realizar la discretización, mientras que los no supervisados no la tienen en cuenta. Los métodos estáticos consideran a cada atributo de forma independiente, mientras que los dinámicos consideran dependencia entre los atributos para realizar tal discretización. Por último, los paramétricos requieren de ajustes de alguna variable por el usuario mientras que su contrapuesto no.

Dentro del aprendizaje por lotes, entre los métodos de discretización más conocidos podemos citar al de igual anchura, que divide el rango numérico en un número dado de intervalos de igual ancho; de igual frecuencia, similar al anterior pero considerando la frecuencia de los valores; y el propuesto por Fayyad e Irani [FI93]. Sin embargo, la mayoría estos métodos requieren varias pasadas por los ejemplos de entrenamiento, lo cual no cumple uno de los requerimientos del aprendizaje incremental. Una excepción es el método de igual anchura, pero supone que el rango de los valores es conocido de antemano, lo cual puede ser violado en ambientes de flujos de datos.

5.4.1 *Métodos de discretización aplicables a la inducción incremental de árboles de decisión*

En la construcción incremental de árboles de decisión, la prueba "¿$a_{ij} \leq x$?" en los nodos de decisión es la más frecuente para manipular atributos numéricos, donde x es un número real y a_{ij} es un valor del atributo numérico a_i para un ejemplo dado. De esta forma, la tarea es buscar el valor x para obtener una prueba que contribuya efectivamente a la construcción del modelo, teniendo en cuenta las restricciones de complejidad computacional.

En la tarea de buscar este valor x, los métodos más usados en la inducción incremental de árboles de decisión son el árbol exhaustivo de búsqueda de Gama y otros [GRM03], la implementación en el paquete VFML [HD03] para manipular atributos numéricos con VFDT, el método propuesto por Greenwald y Khanna [GK01] con garantías matemáticas para acotar el error de las estimaciones, y la aproximación Gaussiana [Kir07].

IADEMc [dCRM06] manipula atributos numéricos mediante el árbol exhaustivo mencionado anteriormente. Este es un árbol binario de búsqueda donde cada valor real es almacenado. Esta estructura ahorra espacio recordando cada valor visto en las hojas del árbol binario. Si un valor se repite se incrementa el contador correspondiente en el nodo hoja correspondiente. Aún así, este ahorro solamente es significativo si se espera que los valores reales se repitan mucho. Sin embargo, tal como pasa en muchos conjuntos de datos, si la probabilidad de que se repita un valor real es prácticamente nula, este acercamiento tiene un alto coste computacional en la inducción de árboles de decisión. Específicamente, su tiempo de acceso a datos e inserción de nuevos datos es $O(\log_2 n)$ para todos los casos, donde n es la cantidad de valores vistos. Otro aspecto importante es definir la cantidad de puntos de corte a considerar como posibles candidatos del valor x en la prueba "¿$a_{ij} \leq x$?".

El método propuesto en el paquete VFML [HD03] resume los valores de los atributos numéricos ordenados en intervalos, creando un histograma. El rango de los valores que cubre cada intervalo es fijado al inicio de su creación y no cambia

mientras son vistos más valores. Al inicio, para cada valor numérico único visto se crea un nuevo intervalo. Después de alcanzar un número fijo de intervalos que debe definir el usuario, los contadores de cada intervalo son actualizados con los valores subsecuentes. El primer inconveniente de este método es que es sensible al orden de los datos. Por ejemplo, si los primeros valores se encuentran solo en una pequeña parte del rango de valores, los intervalos no serán capaces de representar con precisión el rango completo de valores. Otro asunto esta relacionado con la determinación del número óptimo de intervalos, mientras menos intervalos menos preciso es el método y mientras más intervalos aumentará su coste computacional.

El método propuesto por Greenwald y Khanna [GK01] pertenece al área de investigaciones de bases de datos. En esta área, este es el método con garantías de precisión más fuertes, representando así la mejor solución conocida. El método mantiene un conjunto ordenado de tuplas. Cada tupla almacena un valor del flujo de entrada junto con cotas implícitas que estiman el rango real de cada tupla. Greenwald y Khanna [GK01] proponen dos variantes del algoritmo, la primera variante asigna más espacio solo cuando el error de estimación está por exceder un límite fijado ϵ. La forma más usada en la inducción incremental de árboles de decisión fija un límite para la cantidad de memoria usada. Ambas variantes requieren del ajuste de un parámetro. La primera del error máximo deseado y la segunda del número máximo de tuplas. En lo adelante consideraremos la segunda variante ya que permite restringir la cantidad de memoria a usar, lo cual es una característica importante en el procesamiento de flujos de datos.

Por último, la aproximación Gaussiana [Kir07, BHKP10] usa la distribución normal para aproximar una distribución numérica por cada etiqueta de clase del problema. Esta distribución se mantiene incrementalmente almacenando un número fijo de variables en memoria, y no es sensible al orden de llegada de los datos. Una distribución normal es definida esencialmente por su valor medio, que es el centro de la distribución, y la desviación estándar o varianza, que define la dispersión de la distribución. La curva de la distribución tiene forma de campana. La misma es una buena representación de ciertos tipos de fenómenos naturales. En este caso también es necesario fijar el número de cortes a considerar para estimar la mejor opción de x en la prueba "¿$a_{ij} \leq x$?".

5.4.2 *Disminuyendo el coste computacional*

Como hemos visto, los métodos más sobresalientes de discretización aplicables a la inducción incremental de árboles de decisión tienen un parámetro en común, que está relacionado con el número máximo de intervalos a considerar en dicho proceso de discretización. Cuando la cantidad de valores numéricos distintos es muy grande, y la probabilidad de que dos valores numéricos sean iguales es cercana a cero, estos métodos rápidamente llegan a procesar esta cantidad máxima de intervalos. Luego estos métodos nunca reducen esta cantidad.

Supongamos que un atributo numérico a_i tiene valores a_{ij} en el intervalo $a_{ij} \in [m, n]$ y consideremos un número máximo k de intervalos (tuplas, etc.). Si el árbol se expande por dicho atributo a_i en un momento determinado, a partir del punto de corte x y por la prueba "¿$a_{ij} \leq x$?", entonces el intervalo inicial quedará dividido en dos ($[m, x]$ y $(x, n]$) que se corresponderán con cada hijo del nuevo nodo de

decisión. En cada uno de ellos nuevamente se considerarán k intervalos (ya que el árbol se puede expandir varias veces por el mismo atributo numérico contemplando distintos puntos de corte). Cuando el árbol se expande por un atributo nominal, análogamente podemos observar que aumenta también la especificidad a la hora de considerar dichos intervalos por parte del algoritmo de aprendizaje.

Así, en la aplicación directa de uno de estos métodos de discretización a la inducción incremental de árboles de decisión (y particularmente a IADEM-2), podemos observar que cuando el árbol crece la cantidad de intervalos a considerar en el modelo de aprendizaje también crece de forma significativa. A medida que el árbol se expande, el modelo aumenta la precisión en la estimación de posibles puntos de corte. Esto también puede interpretarse como que la especificidad de los puntos de corte a considerar en una determinada expansión gana importancia a medida que crece el árbol. Sin embargo, en los modelos de árboles de decisión lo más natural es que cuando la profundidad de un nodo de decisión es menor, los atributos y las pruebas contempladas en este sean más relevantes para el problema. Así, sería más lógico considerar mayor cantidad de intervalos al inicio de la inducción para tener más precisión en la estimación de los puntos de corte, y luego disminuir paulatinamente la cantidad de intervalos a considerar cuando el árbol va ganando en profundidad. Esta idea permitiría reducir el coste computacional del algoritmo de aprendizaje, lo que es de vital importancia en el procesamiento de flujos de datos.

Algunas investigaciones previas han aplicado los métodos de discretización anteriores a la inducción de árboles de decisión manteniendo fija la cantidad de intervalos a lo largo del entrenamiento del algoritmo [Kir07, BHKP10]. Algunas de estas investigaciones han observado empíricamente que mayor cantidad de intervalos no garantiza aumentar la precisión en la clasificación, o incluso que puede empeorarla; aunque sí aumenta significativamente el coste computacional. Esto también evidencia que a medida que crece el árbol la precisión en la estimación de puntos de corte no es tan significativa como a inicios de la inducción.

De esta forma, en el Algoritmo 3 proponemos disminuir la cantidad máxima de intervalos mientras la profundidad del árbol de decisión crece. La idea que persigue, además de lo discutido anteriormente, es mantener aproximadamente constante la cantidad de memoria a utilizar por el método de discretización durante la inducción del árbol. El algoritmo recibe como parámetro la cantidad máxima de intervalos al inicio de la inducción. También recibe como entrada el mínimo número de intervalos que puede devolver el algoritmo, para garantizar que el árbol pueda expandirse sin límites por cualquier atributo numérico (si dicha cantidad mínima es mayor que dos). Esta cantidad mínima la hemos fijado a 10. Estudios empíricos de investigaciones anteriores han mostrado que VFDT se desempeña bien con este número máximo fijo de intervalos [Kir07]. En la Sección 5.5.2 mostramos que IADEM-2c también tiene un buen comportamiento configurando esta cantidad fija máxima de intervalos.

5.5 EVALUACIÓN EMPÍRICA

En esta sección realizamos evaluaciones empíricas que muestran el rendimiento de IADEM-2c en conocidos conjuntos de datos. En estos experimentos es necesa-

entrada:
    h_i: nodo hoja
    max: cantidad máxima de intervalos
    min: cantidad mínima de intervalos
salida :
    cantidad máxima de intervalos final para el método de discretización en el nodo hoja

```
1  actual = padre (h_i)
2  contador = 0
3  mientras actual ≠ null hacer
4  |  contador = contador + cantidadDeHijos (actual)
5  |  actual = padre (actual)
6  fin
7  si contador = 0 entonces
8  |  contador = 1
9  fin
10 tmp = max/contador
11 si tmp < min entonces
12 |  tmp = min
13 fin
14 retornar tmp
```

Algoritmo 3: Algoritmo para determinar la cantidad máxima de intervalos para el método de discretización.

rio tener en cuenta las metodologías, métricas y herramientas existentes para la evaluación y comparación de algoritmos incrementales.

La Sección 3.3 muestra una revisión de los principales métodos, conjuntos de datos y herramientas que han sido utilizados para evaluar algoritmos de aprendizaje en flujos de datos. El aspecto clave es obtener la mayor precisión posible en la clasificación considerando las restricciones computaciones señaladas. De esta forma, es necesario evaluar tres características de interés: la precisión del algoritmo en la predicción, la cantidad de memoria necesaria para el aprendizaje y el tiempo de procesamiento requerido. Así, la efectividad del algoritmo de aprendizaje dependerá de su habilidad para obtener mayor precisión en una cantidad limitada de tiempo y espacio.

La cantidad de memoria y el límite del tiempo de procesamiento naturalmente influyen en la precisión del algoritmo de aprendizaje. La memoria y el tiempo también están relacionados. Mientras se pueda almacenar más información preprocesada, el algoritmo se ejecutará más rápido al no tener que calcular nuevamente soluciones anteriores, y así disminuir el tiempo de procesamiento a expensas del aumento de la memoria. Mientras más tiempo tenga disponible el algoritmo para el procesamiento, más información podrá procesar y más probabilidad tendrá de reducir el error de la predicción.

La idea fundamental que seguimos es considerar métodos cuya complejidad espacial y temporal no depende del número de ejemplos vistos, para así acotar teóricamente estas medidas de rendimiento. En los experimentos nos centraremos

en las tres características mencionadas anteriormente. Para cuantificar la precisión en la generalización emplearemos el método *holdout* en flujos de datos artificiales (ver Sección 3.3), aprovechando que los generadores de datos artificiales pueden generar una cantidad infinita de ejemplos. Como hemos observado anteriormente, en conjuntos de datos reales este acercamiento no es tan beneficioso, y en este caso a la llegada de cada ejemplo el algoritmo de aprendizaje es primero probado con el ejemplo y luego el aprendizaje continúa normalmente (*test-then-train*).

En el caso del tamaño del modelo, consideraremos el número de nodos para cuantificarlo, que ha sido común en muchos estudios anteriores [Mur98, Kir07, del07, BG09b]. Para medir el coste temporal utilizaremos el tiempo de procesamiento (en segundos) que un determinado algoritmo emplea tanto en el aprendizaje como en la etapa de prueba.

Todos los algoritmos propuestos han sido implementados sobre MOA [BHKP10], un marco de trabajo para el aprendizaje incremental. El mismo ofrece facilidades para el cálculo de estas medidas de rendimiento, también tiene implementados los generadores de flujos de datos artificiales presentes en los experimentos realizados.

5.5.1 *Pruebas binarias, múltiples y mixtas*

En este experimento hemos descartado generadores de flujos de datos donde no existen atributos nominales, ni donde todos los atributos son binarios. En estos casos, el tipo de pruebas no influye en las medidas de rendimiento analizadas, ya que todas las pruebas tendrían salida binaria. La Tabla 5.1 resume los generadores considerados.

Hemos configurado a IADEM-2c para predecir con el clasificador Naïve Bayes, ya que este usa más información almacenada en los nodos hojas para la predicción, dando lugar generalmente a valores más altos de precisión que la clase mayoritaria [del07]. También lo hemos configurado para manipular atributos numéricos con la aproximación Gaussiana y un número máximo de intervalos fijado a 10 [Kir07, BHKP10], debido al bajo coste temporal de este método de discretización. A pesar que mostramos el rendimiento de IADEM-2c para este método de discretización, hemos comprobado que para los restantes métodos de discretización mencionados las pruebas de división se comportan de forma similar.

En este experimento hemos entrenado a los algoritmos con 1 000 000 de instancias. De esta forma comprobamos la influencia del número de ejemplos de entrenamiento en el tamaño del modelo inducido por IADEM-2c a través de las pruebas descritas. En la Tabla 5.1 mostramos el comportamiento de los algoritmos en términos de precisión y número de nodos. Esta vez no hemos considerado el tiempo de procesamiento porque en este caso todas las variantes tuvieron un comportamiento similar. En la última fila se muestra el promedio de los valores de cada columna, en correspondencia con las distintas variantes del algoritmo y con las medidas de rendimiento.

Como se puede apreciar, al considerarse pruebas mixtas la precisión aumenta en la mayoría de los conjuntos de datos contemplados. Se puede observar además que en algunos flujos de datos las pruebas mixtas consiguen mayor precisión con árboles de aproximadamente igual tamaño.

Nombre	Abreviatura	nominales	numéricos	clases
Conceptos STAGGER (3 funciones)	STA1-3	3		2
Conceptos AGRAWAL (10 funciones)	AGR1-10	6	3	2
Generador de árboles aleatorios	RTG1	10	10	2
	RTG2	20	10	2

Tabla 5.1: Generadores de flujos de datos para la experimentación con distintos tipos de prueba instalada en los nodos de decisión en IADEM-2.

	IADEM-2c					
	Solo salidas múltiples		Solo salidas binarias		Salidas mixtas	
	precisión	nodos	precisión	nodos	precisión	nodos
STA1	99,99 (0,14)	16	99,99 (0,23)	*13*	99,99 (0,23)	15
STA2	**99,98** (**0,62**)	16	99,97 (0,96)	*15*	99,97 (0,67)	18
STA3	**99,97** (**0,75**)	16	99,96 (1,20)	*11*	99,95 (1,21)	14
AGR1	88,55 (1,01)	1	88,55 (1,01)	1	88,55 (1,01)	1
AGR2	64,53 (1,55)	1	64,53 (1,55)	1	64,53 (1,55)	1
AGR3	97,14 (0,99)	*16*	**97,26** (**1,55**)	*37*	97,15 (1,03)	*16*
AGR4	80,46 (4,09)	*7*	89,16 (1,36)	22	**89,49** (**1,27**)	32
AGR5	65,82 (1,55)	*1*	76,37 (1,52)	12	**76,38** (**1,51**)	12
AGR6	87,02 (3,09)	27	88,33 (1,84)	*26*	**88,46** (**1,86**)	43
AGR7	95,67 (1,55)	*25*	**96,04** (**1,17**)	33	96,02 (1,18)	33
AGR8	98,95 (0,33)	1	98,95 (0,33)	1	98,95 (0,33)	1
AGR9	95,24 (1,62)	*41*	95,98 (1,23)	68	**96,09** (**1,24**)	80
AGR10	99,85 (0,13)	1	99,85 (0,13)	1	99,85 (0,13)	1
RTG1	92,72 (4,74)	210	67,09 (3,22)	*24*	**94,18** (**3,71**)	241
RTG2	**93,59** (**5,22**)	312	60,56 (1,89)	*15*	92,24 (5,25)	284
$\overline{x}$	90,63	46,02	88,17	*18,58*	**92,12**	52,64

Tabla 5.2: Precisión y número de nodos de IADEM-2c comparando distintos tipos de pruebas para la expansión: salidas múltiples, salidas binarias y salidas mixtas (múltiples o binarias). Fueron 1 000 000 de ejemplos de entrenamiento, *holdout* cada 200 ejemplos con otros 200 ejemplos de prueba.

Nombre	Abreviatura	nominales	numéricos	clases
Conceptos AGRAWAL (10 funciones)	AGR1-10	6	3	2
Hiperplano	HYP		10	2
Generador de árboles aleatorios	RTG1	10	10	2
	RTG2	20	10	2
Funciones de base radial	RBF		10	3
Generador de forma de onda	WAV		21	3

Tabla 5.3: Generadores de flujos de datos para el experimento de métodos de discretización aplicados a IADEM-2c.

La Tabla 5.1 también refleja que en algunos conceptos es difícil para las pruebas binarias o no binarias decidirse por el mejor atributo. En estos casos, las pruebas mixtas pueden mejorar significativamente la precisión del modelo. Por ejemplo, al considerar solo pruebas binarias en RTG1, debido a las peculiaridades de este tipo de generador (los árboles que genera no tienen salidas binarias), el algoritmo IADEM-2c no diferencia con facilidad el mejor atributo para la expansión; sin embargo, esto no ocurrió en los restantes tipos de prueba considerados para ese conjunto de datos. Algo semejante puede interpretarse en RTG2.

5.5.2 *Métodos de discretización*

Varias investigaciones anteriores en el aprendizaje incremental han comparado la efectividad de métodos de discretización bajo diversas circunstancias. Por ejemplo, Kirkby [Kir07] estudia el parámetro relacionado con el número máximo de intervalos, aplicando estos métodos a los árboles de Hoeffding. Un aspecto interesante de este estudio es considerar la restricción de la cantidad de memoria que puede utilizar el algoritmo de aprendizaje en su entrenamiento. En este sentido, también estudia la influencia de esta cantidad de memoria para varias configuraciones de los algoritmos de discretización en conocidos conjuntos de datos.

Los métodos de discretización mencionados en la Sección 5.4 han sido incorporados a IADEM-2c para la manipulación de atributos numéricos, los cuáles serán evaluados empíricamente en este apartado. Para esta manipulación, al igual que IADEMc, se considera la prueba clásica "¿$a_{ij} \leq x$?" a instalar en posibles nodos de decisión. Este nodo de decisión correspondería a la expansión de una hoja por un atributo numérico a través de un punto de corte x, estimado a partir de alguna medida heurística (ganancia de información, índice Gini, etc.). Un nodo de decisión con esta prueba tiene dos hijos, que es la cantidad de respuestas a esta pregunta para un valor numérico a_{ij} dado: verdadero o falso. Como es usual, un mismo atributo numérico puede ser usado más de una vez para la expansión con distintos puntos de corte.

La Tabla 5.3 muestra los generadores de flujos de datos artificiales considerados en el experimento, para un total de 15 generadores. La Tabla 5.4 muestra el promedio de los resultados finales al ejecutar los algoritmos en todos los los conjuntos

				Precisión	Tiempo (*seg.*)	Nodos
IADEM-2	VFML	Estático	10	87,84	363,35	37,71
			50	89,45	1046,17	50,01
			100	91,65	2012,40	53,90
		Dinámico	100	88,63	437,02	43,26
			500	91,23	1100,27	55,74
			1000	91,43	1773,26	52,05
	GK	Estático	10	59,87	109,94	10,88
			50	74,22	690,96	49,01
			100	76,93	1012,72	39,64
		Dinámico	100	67,64	215,17	22,26
			500	76,36	1045,27	52,89
			1000	80,02	1816,25	56,72
	GAUS	Estático	10	89,10	634,75	58,09
			50	88,65	614,09	55,53
			100	88,85	826,47	56,12
		Dinámico	100	89,05	548,05	58,86
			500	88,78	1019,45	59,21
			1000	88,49	1728,87	52,84

Tabla 5.4: Promedio de los resultados finales sobre todos los flujos de datos para manipular atributos numéricos. Para cada flujo de datos hubieron 1 000 000 ejemplos de entrenamiento, cada 200 ejemplos de entrenamiento se evalúan los clasificadores con 200 ejemplos de prueba (*holdout*).

de datos mencionados. Esta última muestra tres medidas importantes en el aprendizaje en línea, relacionadas con la precisión del algoritmo y el coste computacional (temporal y espacial). Los mejores valores de rendimiento están sobresaltados en negritas. Los algoritmos de discretización que se evaluaron en combinación con IADEM-2c son el que se encuentra en el paquete VFML [HD03], el método propuesto por Greenwald y Khanna (GK) [GK01] y la aproximación Gaussiana (GAUS) [Kir07, BHKP10]. Se evaluaron dos variantes del algoritmo, en la primera (Estático) el parámetro relacionado con el número máximo de intervalos no varía durante todo el aprendizaje; en la segunda (Dinámico) dicho parámetro varía acorde al Algoritmo 3 propuesto anteriormente. La Tabla 5.4 muestra las medidas de rendimiento para cada variante con su respectivo parámetro. Como se mencionó anteriormente, en el Algoritmo 3 se configuró el parámetro del número mínimo de intervalos a 10.

VFML es más simple que GK y GAUSS, además puede retornar con precisión la información necesaria, que se encuentra en los intervalos definidos, para predecir con el clasificador Naïve Bayes. Sin embargo, los valores numéricos iniciales definen estos intervalos y estos no cambian durante el proceso de discretización, por lo que estos intervalos pueden estar sesgados por los valores iniciales. GK y GAUS son mucho menos afectados por el orden de llegada de los valores numéricos, pero solo pueden dar una aproximación de la información necesaria para el clasificador Naïve Bayes. Adicionalmente, GAUS asume valores reales distribuidos normalmente por etiqueta de clase. De ahí que VFML haya alcanzado los valores de precisión más altos en combinación con el clasificador Naïve Bayes.

Aumentar el número máximo de intervalos en VFML mejora significativamente la precisión del algoritmo de aprendizaje, manteniendo los árboles aproximadamente con el mismo tamaño. Una mayor cantidad de intervalos en VFML disminuye su sesgo hacia los primeros valores numéricos vistos por el método. En la Tabla 5.4 también podemos observar la eficacia de las variantes VFML-Dinámico y GK-Dinámico. Por ejemplo, VFML-Dinámico-500 y VFML-Estático-100 alcanzaron valores de precisión similares, mientras que VFML-Dinámico-500 tuvo un coste temporal mucho menor.

También queremos resaltar el comportamiento de las variantes de GAUS. Su coste temporal es bajo para todas sus variantes, en particular para GAUS-Estático-10, cuya precisión también fue la más alta. En GAUS, mientras mayor sea la cantidad máxima de intervalos a considerar, más fuerte es la restricción de normalidad del método, puesto que más estimaciones de probabilidad estarán calculadas acordes a la curva normal. De forma general, los valores de los atributos numéricos en los flujos de datos considerados no están distribuidos normalmente con respecto a su clase, de ahí que el valor más alto de su precisión lo haya alcanzado con la menor cantidad máxima de intervalos.

5.5.3 *Conjuntos de datos reales*

En esta sección mostramos el rendimiento de IADEM-2c en siete conjuntos de datos ampliamente usados en el aprendizaje incremental: ELEC2 (ELEC) [HSH98], COVERTYPE (COVE), NURSERY (NURS), ADULT (ADUL) [FA10], USENET1 (USE1), USENET2 (USE2), y SPAM [KTV08]. Los algoritmos son evaluados procesando los

ejemplos de entrenamiento en su orden temporal. La precisión y el número de nodos son evaluados mediante un acercamiento predictivo secuencial (*test-then-train*). La Tabla 5.5 muestra el promedio de los valores calculados cada 100 ejemplos procesados. Estos valores son calculados a partir de la fracción entre el número de ejemplos clasificados correctamente y el total de ejemplos procesados, con respecto a una ventana de tamaño 100 [BHKP10, GSaR13].

En este experimento incluimos, para cada método de discretización (VFML, GK y GAUS), la variante que alcanzó mejor rendimiento en la Sección 5.5.2. También hemos incluido la versión de VFDT implementada en MOA con su configuración por defecto [BHKP10]. Así, VFDT manipula atributos numéricos con la aproximación Gaussiana y un número máximo de intervalos fijado a 10 durante todo el proceso de aprendizaje.

Acorde al experimento con datos sintéticos, las variantes VFML-Dinámico-500 y GAUS-Estático-10 alcanzaron los mayores valores de precisión en el algoritmo IADEM-2c, manteniendo aproximadamente igual el número de nodos del modelo inducido y el tiempo de procesamiento. Como se puede observar en la Tabla 5.5, la variante GK-Estático-50 tuvo un desempeño pobre comparada con las restantes variantes. Como es natural, en conjuntos de datos donde no existen atributos numéricos, las tres variantes estudiadas del algoritmo IADEM-2c tienen medidas de rendimiento idénticas (NURSERY, USENET1, USENET2 y SPAM).

También podemos notar que VFDT por lo general es más preciso que IADEM-2c, aunque también induce árboles de decisión más grandes. Esto es debido a que el intervalo de confianza usado en VFDT converge a cero mucho más rápido que el usado en IADEM-2c, causando así que VFDT expanda nodos hojas con mayor frecuencia. Desafortunadamente, este intervalo de confianza no está matemáticamente bien justificado.

Finalmente, destacamos que IADEM-2c no manipula cambio de concepto, mientras que varias investigaciones previas han observado que en varios de los conjuntos de datos reales estudiados está presente el cambio de concepto. Como hemos señalado, el cambio de concepto deteriora el rendimiento del modelo de aprendizaje, ya que los datos presentes no están acorde a los pasados y así, un modelo inducido previamente puede convertirse obsoleto o incluso adverso son respecto a los datos actuales.

5.6 CONCLUSIONES DEL CAPÍTULO

En este capítulo le incorporamos a IADEM-2 mecanismos para inducir árboles binarios, a través de la instalación de pruebas de división binarias en los nodos de decisión. También hemos incluido la posibilidad de inducir árboles de decisión que contemplen tanto pruebas de división binarias como con salidas múltiples. Este último acercamiento resultó en una mejora significativa de la precisión del algoritmo en la clasificación, manteniendo acotado el coste computacional del mismo e induciendo igualmente árboles pequeños. La principal ventaja de considerar pruebas de división binarias es que estas permiten modelar problemas más complejos, mientras que las pruebas con salidas múltiples por lo general dan lugar a árboles de decisión más pequeños e interpretables.

Algoritmo				ELEC	COVE	NURS	ADUL	USE1	USE2	SPAM	Promedio
IADEM-2c	VFML-Dinámico-500	precisión	$\overline{x}$	76,13	74,51	86,41	79,14	**64,33**	**71,93**	**90,59**	77,58
			s	11,16	15,70	12,88	4,50	**21,83**	**10,64**	**8,68**	
		nodos	$\overline{x}$	24,35	56,23	*4,08*	4,48	1,00	1,00	5,62	13,82
			s	8,04	26,27	*1,14*	1,09	0,00	0,00	1,91	
		tiempo	*seg.*	33	545	3	12	1	1	19	87,61
	GK-Estático-50	precisión	$\overline{x}$	62,30	65,94	86,41	79,95	**64,33**	**71,93**	**90,59**	74,49
			s	17,68	17,02	12,88	4,62	**21,83**	**10,64**	**8,68**	
		nodos	$\overline{x}$	34,94	65,09	*4,08*	4,62	1,00	1,00	5,62	16,62
			s	11,06	33,97	*1,14*	1,02	0,00	0,00	1,91	
		tiempo	*seg.*	43	2569	3	53	1	1	18	384,05
	GAUS-Estático-10	precisión	$\overline{x}$	76,05	73,30	86,41	82,06	**64,33**	**71,93**	**90,59**	77,81
			s	13,57	16,13	12,88	3,92	**21,83**	**10,64**	**8,68**	
		nodos	$\overline{x}$	*17,11*	*48,70*	*4,08*	*2,90*	1,00	1,00	5,62	*11,49*
			s	*9,91*	*18,39*	*1,14*	*0,44*	0,00	0,00	1,91	
		tiempo	*seg.*	11	550	3	7	1	1	18	84,39
VFDT		precisión	$\overline{x}$	**78,87**	**80,58**	**89,85**	**83,95**	63,13	72,00	90,17	**79,79**
			s	**12,39**	**13,28**	**7,79**	**3,85**	21,05	10,63	12,77	
		nodos	$\overline{x}$	30,97	182,92	12,58	30,11	1,00	1,00	*4,38*	37,57
			s	12,56	104,95	5,42	19,65	0,00	0,00	*1,82*	
		tiempo	*seg.*	8	1093	2	9	1	1	12	160,86

Tabla 5.5: Precisión, número de nodos y tiempo de procesamiento de IADEM-2c sobre varios conjuntos de datos reales.

Hemos estudiado además el comportamiento de IADEM-2 en combinación con algunos de los métodos de discretización más populares en el área del aprendizaje en flujos de datos. Estos son la implementación en el paquete VFML [HD03] para manipular atributos numéricos con VFDT, el método propuesto por Greenwald y Khanna [GK01] con garantías matemáticas para acotar el error de las estimaciones, y la aproximación Gaussiana [Kir07, BHKP10]. En la adaptación de estos métodos a la inducción incremental de árboles de decisión, hemos propuesto un algoritmo para disminuir el coste computacional de los métodos de discretización a medida que el árbol crece, suponiendo que la precisión de estos métodos de discretización no es tan importante en los nodos con mayor profundidad en el árbol. Este acercamiento permite también tener más precisión en las estimaciones de puntos de corte relevantes al inicio de la inducción, sin aumentar significativamente el coste computacional del algoritmo resultante.

Sin embargo, IADEM-2c no presenta mecanismos para la manipulación de cambio de concepto, un problema inherente al aprendizaje incremental. En este problema nos enfocaremos en el siguiente capítulo.

6

IADEM-3: MANIPULANDO CAMBIO DE CONCEPTO

Los algoritmos de aprendizaje en línea son adecuados para minar grandes cantidades de datos generados constantemente a una alta velocidad. Internet, telefonía móvil y redes de sensores son fuentes comunes de estos flujos de datos [GG07]. Los algoritmos para la minería de flujos de datos deben ejecutarse con recursos computacionales limitados: todos los datos no pueden ser procesados debido a su gran tamaño (restricción de memoria), y los algoritmos de aprendizaje deben procesar datos que llegan con una tasa de llegada alta (un limitado tiempo de procesamiento). Adicionalmente, como hemos discutido en capítulos anteriores, los datos pueden ser no estacionarios.

Los árboles de decisión son uno de los modelos de clasificación más estudiados, comparados con otros métodos de aprendizaje automático y con expertos humanos [Mur98]. Sin embargo, los métodos clásicos de aprendizaje por lotes, como ID3 o C4.5, no son adecuados para la minería de flujos de datos; ellos asumen que todos los datos de entrenamiento están disponibles durante el proceso de aprendizaje, además suponen que estos datos son estacionarios. A pesar de esto, muchos algoritmos de aprendizaje han seguido su idea básica para construir árboles de decisión a partir de flujos de datos [DH00, RM00, dRGM08, RPDJ13]. Para manipular cambio de concepto se usa comúnmente un detector de cambio, el cuál puede poner en marcha un mecanismo de olvido para eliminar o decrecer la importancia de partes del modelo que se estima que ya no son importantes. Por ejemplo, VFDT [DH00] es un método conocido para la minería de flujos de datos y varios métodos lo extienden para manipular cambio de concepto usando esta estrategia [HSD01, GFR06, BG09a].

La mayoría de los estudios para manipular cambio de concepto a través de árboles de decisión han estado basados en VFDT. Sin embargo, al igual que VFDT, estos algoritmos deben revisarse ya que el uso de la cota Hoeffding no es adecuado para el problema de estimar intervalos de confianza para cualquier regla de división de atributos [RPDJ13]. Una excepción es OnlineTree2, el cual almacena algunos datos previos para la inducción del árbol (usa una memoria parcial de instancias) y considera varias heurísticas en los mecanismos de detección y olvido. Aunque estas heurísticas son intuitivas y tienen un buen rendimiento en varios dominios, ellas carecen de una justificación estadística adecuada.

Como hemos visto, la familia de algoritmos IADEM [RM00, RdM06, dCRM06, dRGM08] también puede inducir árboles de decisión a partir de flujos de datos. Estos algoritmos solo almacenan frecuencias relativas en los nodos hoja ($\overline{X}$) y estiman intervalos de confianza ($[E(\overline{X}) - \varepsilon, E(\overline{X}) + \varepsilon]$) para estas frecuencias relativas usando las cotas de Hoeffding y de Chernoff. De esta forma, la inducción del árbol considera estas estimaciones para ejecutar diferentes acciones (expansión de un nodo, selección del atributo más apropiado para expandir, etc.). Varios algoritmos basados en IADEM tratan de mejorar la calidad del modelo inducido y de

extender sus áreas de aplicación [dCRM06, dRGM08], pero ningún algoritmo de esta familia puede manipular cambio de concepto.

Este capítulo persigue solucionar estas deficiencias y presenta IADEM-3, un algoritmo basado en IADEM-2c (introducido en el Capítulo 5), para la minería de flujos de datos con cambio de concepto. Este nuevo algoritmo satisface los requerimientos comunes para el aprendizaje en línea: tiene complejidad temporal y espacial constante por ejemplo procesado, aprende con una sola pasada sobre los ejemplos de entrenamiento, y no depende de parámetros a ajustar por el usuario para la manipulación de cambio de concepto. IADEM-3 sigue una estrategia común para manipular cambio de concepto monitorizando constantemente la consistencia de partes del modelo (subárboles). Cuando este estima una inconsistencia con respecto a los datos más recientes, se supone un cambio de concepto y se ejecutan algunas acciones para reconstruir los subárboles afectados por el cambio (induciendo subárboles alternativos). En la literatura, esta estrategia es vista como una ventana temporal [GMR05] moviéndose sobre los datos más recientes y acorde al concepto actual. De esta forma, el modelo de aprendizaje es entrenado con los ejemplos más recientes almacenados en esta ventana. La idea principal de este mecanismo es reducir el tamaño de la ventana (e.g. eliminando subárboles viejos) cuando un concepto nuevo aparece, y permitir su crecimiento tanto como sea posible cuando los conceptos son estables. Algunos algoritmos de aprendizaje previos han fijado el tamaño de la ventana [HSD01] y también lo han ajustado dinámicamente para lidiar efectivamente con conceptos estables y cambiantes [HSD01, NFM07, BHKP10].

IADEM-3 también usa subárboles alternativos para proveer predicciones más exactas, ya que el modelo de aprendizaje completo puede ser visto como un árbol de opciones [PHK07, IGZD11]. En los árboles de opciones, un ejemplo sin etiqueta de clase puede tomar múltiples caminos desde la raíz hasta las hojas y así puede tomar también múltiples etiquetas de clase. De esta forma, es necesario un método adicional para dar una única predicción.

Previamente, en la Sección 3.2.1 hemos realizado una revisión de los principales algoritmos incrementales para manipular cambio de concepto basados en árboles de decisión, hemos discutido además los principales problemas a considerar en su diseño, cómo los algoritmos existentes intentan resolver estos problemas y sus limitaciones fundamentales. En la siguiente sección discutimos los mecanismos de adaptación del cambio que incorporamos a IADEM-3. Luego, en la Sección 6.3 comparamos nuestro nuevo algoritmo IADEM-3 con su versión previa IADEM-2c y con otros algoritmos relevantes basados en árboles de decisión en el área del aprendizaje incremental: VFDT [DH00], Hoeffding Adaptive Tree (HAT) [BHKP10], y OnlineTree2 [NFM07]. Mostramos que nuestra propuesta es competitiva teniendo en cuenta medidas de rendimiento importantes (precisión, tamaño del modelo y tiempo de procesamiento), así como considerando varios conjuntos de datos sintéticos y del mundo real. Finalmente, presentamos las conclusiones del capítulo en la Sección 6.4.

6.1 DISCUSIÓN DE LA ESTRATEGIA DE ADAPTACIÓN AL CAMBIO

Para manipular cambio de concepto, uno de los métodos más simples que se pudiera usar es una ventana de tamaño fijo moviéndose sobre las últimas instan-

cias, manteniendo actualizado el modelo con respecto a esta ventana (por ejemplo, como en CVFDT [HSD01]). Sin embargo, como hemos discutido, no existe un tamaño fijo de esa ventana para manipular distintas velocidades de cambio (abrupto y gradual). Además, cuando el concepto es estable, es deseable que dicha ventana crezca tanto como sea posible para así tener entrenado al modelo con mayor cantidad de ejemplos y obtener mejor generalización.

De esta forma, acorde a como se señala en la literatura, manipular cambio de concepto de forma explícita y con ventanas de tamaño variable conlleva generalmente a un mejor rendimiento del algoritmo de aprendizaje resultante. Para ello es necesario estimar si el concepto es estable (para que la ventana crezca) o si existe un cambio de concepto (para reducir su tamaño y olvidar ejemplos antiguos). Para la estimación de tales cambios de concepto, como hemos visto en el Capítulo 4, la monitorización de alguna medida de rendimiento (e.g. precisión) en el tiempo es un método efectivo. Un deterioro de esta medida puede significar un cambio de concepto. De esta forma, es posible el uso de herramientas estadísticas para tal monitorización, dándole fundamento teórico a la estimación de posibles cambios distribucionales en los datos de entrada.

Nuevamente, uno de los métodos más simples para dicha estrategia fue presentado en el Capítulo 4, donde un detector de cambio recibe como entrada un flujo de valores reales (correspondiente a una medida de rendimiento dada) y su salida es el estado estimado por el detector. Cuando el detector dispara una señal de cambio se descarta el clasificador antiguo (que se supone esté entrenado con el concepto anterior) y un clasificador nuevo se entrena con las instancias más recientes. También hemos visto que una señal adicional de alerta puede utilizarse para el entrenamiento previo del nuevo clasificador antes de la señal definitiva de cambio.

El primer inconveniente de este método ocurre cuando el cambio estimado por el detector no se corresponde con un cambio de concepto. Por ejemplo, debido a que el deterioro de la medida de rendimiento no estuvo dado por un cambio de concepto sino por ruido en los datos de entrada, o porque el detector tuvo un falso positivo. Incluso si la señal de cambio del detector se corresponde con un verdadero cambio de concepto, esta estrategia solo es eficaz si el cambio subyacente es abrupto y los conceptos consecutivos comparten poco o nada en común, ya que el clasificador antiguo se descarta por completo inmediatamente. No obstante, en aplicaciones reales muchas veces los cambios son graduales y los conceptos consecutivos comparten muchas características en común [GŽB+]. De esta forma, en estos casos un clasificador antiguo entrenado con muchas experiencias puede ser más preciso que un clasificador nuevo entrenado con pocas, incluso aunque el rendimiento del clasificador antiguo haya decaído significativamente. Este hecho es más sustancial cuando el algoritmo de aprendizaje tiene una curva de aprendizaje lenta en los conceptos subyacentes. Por ejemplo, en el caso de los árboles de decisión que usan estimación por intervalos, en situaciones determinadas estos intervalos de confianza pueden demorarse en ajustarse para decidir por el mejor atributo a expandir un nodo hoja dado, siendo posible así que su aprendizaje sea lento. Por otro lado, al ser semejantes los conceptos consecutivos, los cambios pueden que no afecten al modelo completo sino solo a algunas partes del mismo (e.g. en presencia de cambios locales), por lo que descartar el modelo completo

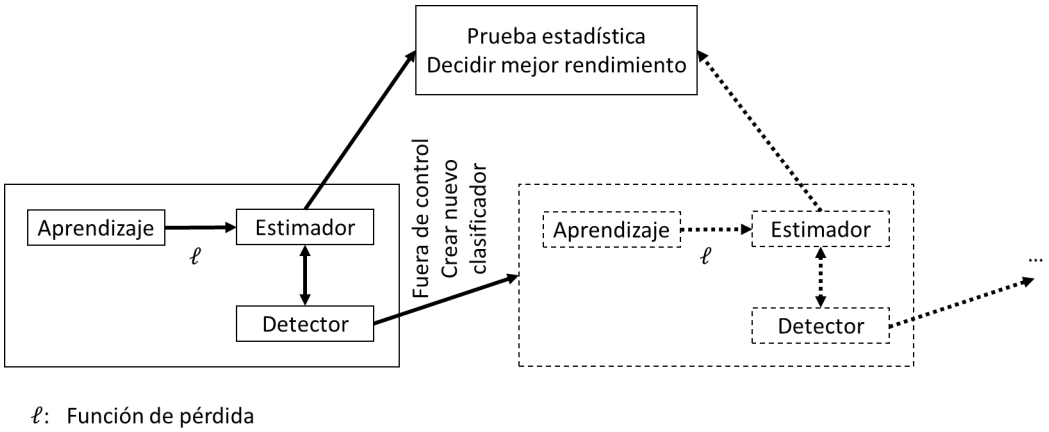


Figura 6.1: Esquema para la manipulación de cambio de concepto.

podría repercutir negativamente en el poder de generalización del algoritmo de aprendizaje.

IADEM-3 cambia un árbol de decisión antiguo (o partes del mismo) por uno más reciente cerciorándose primero que dicho cambio es efectivo. Para ello, es necesario estimar cuál es el valor real de una medida de rendimiento dada, calculada continuamente para ambos modelos de aprendizaje. A partir de estas dos estimaciones de rendimiento (una para cada modelo, el antiguo y el más reciente), IADEM-3 aplica una prueba estadística para decidir con determinada confianza cuál es el modelo con mejor rendimiento. En este punto, el clasificador alternativo sustituye al antiguo solo si demuestra ser más efectivo. Por otro lado, cuando los cambios son graduales y lentos, incluso el modelo alternativo puede bajar su rendimiento debido al cambio gradual, por lo que este mecanismo de adaptación puede aplicarse recursivamente a los modelos alternativos (ver Figura 6.1). Note que el esquema de la Figura 6.1 puede complementarse con el de la Figura 4.1 para la actualización de los estimadores, algoritmos de aprendizaje, etc.

En IADEM-3, la monitorización del rendimiento de ambos modelos de aprendizaje (el antiguo y el más reciente) se lleva a cabo por un acercamiento predictivo secuencial a la llegada de cada ejemplo (como lo hacen por ejemplo DDM [GMCR04], EDDM [BdF+06], HDDM, etc.). Cabe recalcar que los métodos usados en este mecanismo (estimadores, detectores, pruebas estadísticas, algoritmos de aprendizaje) naturalmente deben cumplir con las restricciones computacionales comunes para el aprendizaje en línea.

En el caso particular de los árboles de decisión, la monitorización de esta consistencia en todos los nodos del árbol ha demostrado ser efectiva [HSD01, NFM07, BHKP10]. De esta forma, se pueden detectar cambios en partes del espacio de instancias (en los nodos de decisión) cuando ocurren cambios locales. Adicionalmente, reconstruir solo partes del modelo es más eficaz (e.g., teniendo en cuenta el coste computacional y la precisión del modelo durante dicha reconstrucción) que reconstruir el árbol completo. Así, en IADEM-3 cada nodo de decisión sigue el esquema de la Figura 6.1. Debido a que en un momento dado pueden existir varias instancias de los componentes de este esquema (una por nodo de decisión), la eficiencia computacional de estos componentes es un punto clave.

Al ocurrir un cambio, también los nodos hojas pueden sufrir un deterioro en relación a su poder de predicción o clasificación. Así, a pesar de que pocos métodos han definido como manipular cambios de concepto en los nodos hojas [NFM07], estas deben estar también contempladas en la estrategia para la manipulación de cambio de concepto.

6.2 EL NUEVO ALGORITMO

IADEM-3 monitoriza la consistencia en subárboles chequeando continuamente el error de clasificación en cada nodo. Un incremento significativo en la tasa de error es interpretado como un cambio de concepto. En los nodos hojas, este incremento puede indicar efectivamente que la información almacenada en el respectivo nodo hoja no está acorde con el concepto actual. De esta forma, IADEM-3 reinicia todas las variables en las hojas cuando una señal de cambio es disparada por el detector de cambio correspondiente. Viendo el nodo hoja como un clasificador, es posible aplicar un método más sofisticado como mecanismo de adaptación [GMCR04, BM08]. Sin embargo, elegimos el más simple teniendo en cuenta además su bajo coste computacional.

Este mecanismo de olvido es muy drástico para los nodos de decisión. Olvidar repentinamente en un nodo de decisión, por ejemplo, podando el subárbol correspondiente, puede deteriorar significativamente el rendimiento del modelo de aprendizaje en presencia de cambios graduales si este subárbol es relativamente grande. Como hemos indicado, en estos casos un modelo de aprendizaje antiguo entrenado con muchos ejemplos puede ser más preciso que un alternativo nuevo entrenado con pocos.

En el caso de los nodos de decisión, IADEM-3 crea subárboles alternativos que son inducidos en paralelo con el árbol principal. Un subárbol alternativo sustituye a su respectivo subárbol principal cuando esta sustitución mejora significativamente la precisión del modelo de aprendizaje. Sin embargo, para implementar esta estrategia necesitamos resolver dos problemas fundamentales: la detección en línea de incrementos significativos en la tasa de error en subárboles, y la manipulación (promoción y eliminación) de subárboles alternativos.

Los siguientes apartados están dirigidos fundamentalmente a resolver estas cuestiones.

6.2.1 *Detección de cambio*

Cuando un cambio de concepto ocurre, el modelo de aprendizaje actual no está acorde con los datos más recientes y como consecuencia, su error en la clasificación aumenta. El mecanismo de detección de estos cambios en la tasa de error que proponemos se lleva a cabo en cada nodo de decisión, para manipular cambios locales [IGD11].

En IADEM-3 se monitoriza el error de predicción a través de un acercamiento predictivo secuencial (*test-then-train*) [GMCR04, GSaR13], donde a la llegada de cada instancia, los atributos de la misma se les proporcionan al algoritmo para realizar una predicción, y luego el aprendizaje continúa normalmente. Gama y otros [GSaR13] dan garantías matemáticas al método predictivo secuencial como

un marco de trabajo general que provee estimación del error para algoritmos de aprendizaje incremental. En esencia, en cada nodo de decisión de IADEM-3 se sigue el esquema mostrado en la Figura 4.1 con una función de pérdida 0-1.

Instalar un detector de cambio en cada nodo puede tener un coste computacional alto. Para ello, un detector de cambio en línea es una opción factible. Entre estos detectores podemos encontrar varios enfoques paramétricos [BN93], pero los más conocidos asumen que los datos de entrada están distribuidos normalmente. En la comunidad del aprendizaje automático también podemos encontrar otros métodos relacionados como DDM [GMCR04], EDDM [BdF+06] y ECDD [RATH12], pero estos no están equipados con garantías matemáticas de rendimiento. ADWIN2 [BG07] tiene garantías matemáticas en forma de cotas para las tasas de falsos positivos y falsos negativos, pero no procesa los datos de entrada con una cantidad limitada de memoria y tiempo de procesamiento.

Particularmente, elegimos a HDDM$_{A\text{-test}}$ [FdR+14], el cual presentamos en el Capítulo 6. El mismo procesa los valores de entrada con complejidad temporal y espacial constante. HDDM$_{A\text{-test}}$ también tiene garantías matemáticas para las tasas de falsos positivos y falsos negativos. La prueba W-test con el estadístico EWMA, también estudiada en el Capítulo 6, no es igualmente efectiva en este caso, principalmente porque es más susceptible al ruido y así tiene una tasa de falsos positivos más alta. Como hemos analizado en el caso de los árboles de decisión, un falso positivo puede tener el coste adicional de un aprendizaje lento (necesidad del ajuste de intervalos de confianza para expandir un nodo hoja, etc.).

6.2.2 *Adaptación*

IADEM-3 mantiene actualizado un detector de cambio en línea en cada nodo del árbol. Como hemos señalado, una señal de cambio en un nodo hoja provoca la inicialización de todas sus variables. En los nodos de decisión, IADEM-3 crea un subárbol alternativo y lo induce en paralelo con el árbol principal a una señal de cambio. Este subárbol alternativo reemplazará al original si IADEM-3 estima que este reemplazo mejora significativamente la precisión del modelo de aprendizaje resultante.

Para estimar cuál subárbol tiene mejor desempeño, lo primero es estimar el desempeño de cada uno de ellos (el original y el correspondiente alternativo) para luego comparar dichas estimaciones. En general seguimos el esquema de la Figura 6.1. La estimación del desempeño de los subárboles se realiza a través de la instalación de una instancia del algoritmo HDDM$_{A\text{-test}}$. Para ello HDDM$_{A\text{-test}}$ monitoriza incrementos y decrementos en el valor medio del estimador correspondiente (la media aritmética), reiniciando sus contadores si se detecta un cambio en cualquiera de los dos casos. Naturalmente, las señales de cambio solo se dispararan cuando se estiman incrementos en la media poblacional. Así, en cada momento, HDDM$_{A\text{-test}}$ es capaz de estimar el valor medio del error cometido por el subárbol respectivo. HDDM$_{A\text{-test}}$ cumple entonces dos funciones en un nodo de decisión: estimar la precisión del subárbol correspondiente y detectar cambios de concepto.

Usando este detector de cambio, en cada nodo de decisión y en la raíz de cada subárbol alternativo existe entonces un estimador de su desempeño, que tiene como entrada mediciones de error relativas a la función de pérdida 0-1 y un acerca-

miento predictivo secuencial. Suponiendo que estas mediciones de error se corresponden con variables aleatorias independientes (ver Capítulo 4), podemos aplicar la prueba A-test para comparar la precisión entre subárboles como sigue. Supongamos que el detector de cambio correspondiente a un nodo del árbol principal ha estimado un concepto estable por n mediciones, en este nodo de decisión tendríamos una estimación de su rendimiento a través de la media aritmética $\overline{X}_n$ calculada a partir de la secuencia de variables aleatorias independientes $X_1, X_2, \ldots, , \ldots, X_n$. Del mismo modo, en la raíz del subárbol alternativo correspondiente, a m mediciones de su desempeño tendríamos el estadístico $\overline{Y}_m$ calculado a partir de las variables aleatorias independientes $Y_1, Y_2, \ldots, , \ldots, Y_m$. Entonces, a partir de la prueba A-test podemos estimar con nivel de confianza $1 - \alpha$ cuál de los dos subárboles es el de menor tasa de error a través de $\overline{X}_n$ y $\overline{Y}_m$.

Por otro lado, puede ser que un subárbol alternativo no logre mejorar la precisión de su respectivo subárbol original. Esto puede estar dado, por ejemplo, porque este subárbol alternativo no fue creado debido a un cambio de concepto (debido a ruido, a una detección falsa, etc.), porque el correspondiente subárbol original se recuperó del cambio de concepto (debido a una alternación de subárboles en un nodo de decisión con mayor profundidad), porque incluso con el deterioro significativo del rendimiento del subárbol original este sigue siendo adecuado para el concepto actual, etc. Entonces, un árbol alternativo puede dejar de ser relevante. Naturalmente, los subárboles alternativos que no son útiles deben eliminarse lo más rápido posible, debido a que ellos consumen recursos críticos computacionales (tiempo de procesamiento y memoria).

En general, es difícil estimar si un subárbol alternativo dado puede incrementar su precisión en el tiempo y así llegar a formar parte del árbol principal. Los acercamientos más prominentes siguen una idea simple [BG09a, IGD11]: si un subárbol alternativo parece ser incapaz de cumplir su propósito (mejorar la precisión del correspondiente subárbol alternativo), este es eliminado. Así, IADEM-3 elimina subárboles alternativos por dos razones diferentes: (1) el subárbol alternativo tiene una precisión significativamente peor que su correspondiente original, (2) o el subárbol original se ha recobrado del cambio de concepto estimado.

Similar a cuando un subárbol alternativo promueve, IADEM-3 usa $\text{HDDM}_{A\text{-test}}$ como estimador de la media y la prueba A-test para eliminar subárboles alternativos cuando este tiene significativamente una peor precisión que el subárbol original. Con respecto a la segunda alternativa, cuando la causa de creación de un subárbol alternativo (la tasa de error de su respectivo subárbol original) ya no persiste, el subárbol alternativo correspondiente es eliminado. Esta comparación es llevada a cabo nuevamente a través de $\text{HDDM}_{A\text{-test}}$ y la prueba A-test.

IADEM-3 crea un nuevo subárbol alternativo en los nodos de decisión incluso si uno previo ya se estaba induciendo debido a una señal previa de cambio. Entonces, un subárbol alternativo también puede ser eliminado indirectamente si un detector de cambio dispara una segunda señal de cambio. Este método es capaz de manipular tipos graduales de cambio con un coste computacional bajo. Otro método conocido que puede lidiar con cambios graduales, como el usado en CVFDT [HSD01], es monitorizar la consistencia de subárboles en todos los nodos, ya sea pertenecientes al árbol principal o a un subárbol alternativo creado previamente, y crear subárboles alternativos a partir de cualquier nodo inconsistente. Sin em-

bargo, este método puede tener un coste computacional alto cuando los conceptos cambian con alta frecuencia.

En el Algoritmo 4 se muestra el pseudocódigo de IADEM-3.

entrada:
$e_i = (\overrightarrow{a_i}, c_i)$: ejemplo del flujo de datos
A: Árbol de decisión
salida :
Árbol de decisión

```
1  algoritmoCIADEM(A, e_i)
2      predecir la clase ĉ_i de la observación a_i usando el árbol A
3      para cada nodo n_j en el camino desde la raiz hasta la hoja acorde al ejemplo
       e_i = (a_i, c_i) hacer
4          actualizar estimador a partir de la función de pérdida ℓ(c_i, ĉ_i)
5          si n_j es un nodo de decisión entonces
6              si se estima un incremento significativo en el valor del estimador
               entonces
7                  crear subárbol alternativo con raíz en n_j
8              fin
9              si existe un subárbol alternativo subÁrbol induciéndose con raiz en n_j
               entonces
10                 chequear para intercambiar subárboles
11                 chequear para eliminar subÁrbol
12                 actualizar subárbol con (a_i, c_i)
13             fin
14         fin
15         sino si n_j es un nodo hoja entonces
16             si el nivel del detector es de cambio inicializar estructuras de
               datos
17         fin
18         actualizar n_j con (a_i, c_i)
19     fin
20 fin
```

Algoritmo 4: Algoritmo para la manipulación de cambio de concepto aplicable en algoritmos de inducción de árboles de decisión.

6.2.3 *Método de predicción*

Como los nodos de decisión pueden tener subárboles alternativos, los ejemplos pueden recorrer el árbol por medio de diferentes caminos. De esta forma, el modelo de aprendizaje resultante puede ser visto como un árbol de opciones. Los árboles de opciones han sido explorados tanto en el aprendizaje por lotes [**?**] como en flujos de datos [PHK07, IGZD11], aunque no en la presencia de cambio de concepto.

El punto clave en la creación de árboles de opciones es el criterio para crear nodos de opciones y el método de predicción. En IADEM-3, podemos ver la creación de un subárbol alternativo como una opción insertada en el nodo de decisión correspondiente. Los algoritmos más conocidos de aprendizaje adaptativo basados en árboles de decisión, tal como CVFDT [HSD01] o HAT [BHKP10], no consideran los subárboles alternativos en las tareas de clasificación y realizan una predicción clásica usando solamente el árbol principal. El beneficio de los árboles de opciones sobre un ensamble de clasificadores tradicional es una representación más eficiente de varios modelos de aprendizaje. Viendo el modelo de aprendizaje generado por IADEM-3 como un ensamble, existen muchas variantes para combinar múltiples predicciones en flujos de datos [WFYH03]. Sin embargo, nuevamente elegimos la variante más simple y eficiente, donde las predicciones son combinadas por el promedio en los nodos de decisión [PHK07].

6.3 ESTUDIO EXPERIMENTAL

En esta sección evaluamos empíricamente el nuevo algoritmo en conceptos estables y cambiantes, comparando la calidad del modelo con respecto a otros algoritmos en línea basados en árboles de decisión: IADEM-2c, VFDT [DH00], HAT [BG09a], y OnlineTree2 [NFM07]. En los conceptos estables, evaluamos fundamentalmente la estabilidad del modelo inducido, la evolución del aprendizaje en el tiempo, así como cuán robusto es el nuevo algoritmo al ruido y a las detecciones de cambio falsas. Cuando los conceptos varían en el tiempo, evaluamos el rendimiento de los algoritmos bajo varios tipos comunes de cambio (abrupto y gradual), cuán rápido IADEM-3 se adapta a los cambios de concepto, y la tasa de detecciones de cambio falsas.

Todos los experimentos fueron llevados a cabo sobre MOA [BHKP10], un marco de trabajo para el aprendizaje incremental. El rendimiento de IADEM-3 es evaluado considerando tres medidas de rendimiento importantes: precisión, tamaño del modelo (en término de número de nodos) y tiempo de procesamiento (en segundos). Con el objetivo de medir la evolución del proceso de aprendizaje, calculamos estas medidas durante todo el proceso de aprendizaje [GC06, Bif10, BHKP10].

Sobre los conjuntos de datos sintéticos, los algoritmos de aprendizaje entran cada cierto tiempo en una fase de prueba con un número fijado de ejemplos sin etiqueta de clase, que varían en cada prueba (*holdout*). En este caso, tomamos ventaja de los generadores de flujos de datos ya que estos pueden generar un número infinito de ejemplos. Adicionalmente, el evaluador del algoritmo de aprendizaje conoce en cada momento la función objetivo subyacente en los ejemplos generados, pudiendo así usar los generadores según el concepto actual para cada fase de prueba.

Tales condiciones no se mantienen cuando los algoritmos aprenden en flujos de datos del mundo real, por lo que en este caso usamos un acercamiento predictivo secuencial [Daw84, GSaR13]. Este acercamiento consiste básicamente en calcular las medidas a la llegada de cada ejemplo de entrenamiento (paso de prueba); y luego, el ejemplo se le proporciona al algoritmo para continuar normalmente con el aprendizaje (paso de entrenamiento). Esta metodología está basada en la suma cumulativa de los valores de una función de pérdida dada. El valor del error pre-

dictivo secuencial es más pesimista cuando son vistos más ejemplos, porque no existe un mecanismo de olvido y el rendimiento actual tiene igual importancia que el rendimiento previo. Para solucionar este problema, calculamos las métricas por medio de una ventana deslizante que almacena solo los últimos ejemplos [GRC09]. Gama y otros [GSR09] han mostrado que la precisión calculada con un acercamiento predictivo secuencial y con mecanismos de olvido (e.g. una ventana deslizante) converge a la medida calculada con un acercamiento de tipo *holdout*.

Para verificar las diferencias significativas seguimos a Dietterich [**?**] usando la prueba de Mcnemar para comparar pares de algoritmos. Debido al número grande de observaciones usadas, todos los casos mostraron diferencias significativas con valores de confianza menores que 10^{-3}.

6.3.1 *Configuración de los algoritmos*

En todos los experimentos configuramos a VFDT y HAT con sus parámetros por defecto en MOA, ya que varios estudios previos han mostrado que estos algoritmos tienen un buen rendimiento con esta configuración en muchas circunstancias [Kir07]. De forma similar, para comparar de forma apropiada el tamaño del modelo y el tiempo de procesamiento, IADEM-2c y IADEM-3 manipulan atributos numéricos a través de la aproximación Gaussiana [Kir07, BHKP10] con un número máximo de intervalos fijado a 10. Por otro lado, OnlineTree2 requiere un método de discretización con un procedimiento para olvidar un número real dado (e.g. cuando una hoja está en un estado de degradación), entonces usar la aproximación Gaussiana en este caso no es posible. Así, para OnlineTree2 adaptamos el método implementado en el paquete VFML[1] con un número máximo de intervalos fijado a 50.

VFDT y HAT predicen en los nodos hojas con el método propuesto por Kirkby [Kir07, BHKP10] (su configuración por defecto en MOA). IADEM-2c y IADEM-3 predicen en los nodos hojas con el clasificador Naïve Bayes, mientras que IADEM-3 combina las predicciones del árbol principal y los posibles alternativos por el método propuesto en la Sección 6.2.3. Por otro lado, OnlineTree2 usa la clase mayoritaria [NFM07].

En IADEM-3, el nivel de confianza $(1 - \delta)$ para estimar incrementos significativos en la tasa de error es configurado a 0,001. Varios estudios empíricos han mostrado que la prueba A-test y $\text{HDDM}_{A\text{-test}}$ tienen un buen desempeño (tasas de falsos positivos y falsos negativos, así como retardo en la detección de cambios) con esta configuración [FdR+14].

6.3.2 *Datos sintéticos*

En este experimento usamos varios generadores de flujos de datos implementados en MOA para evaluar el rendimiento de los algoritmos. La Tabla 6.1 resume las principales características de estos generadores. De esta forma, los algoritmos se ejecutan sobre flujos de datos con ruido y atributos irrelevantes, atributos nomi-

1 http://www.cs.washington.edu/dm/vfml/

Generadores	Acrónimos	Nominales	Numéricos	Clases	Simulación de cambio de concepto
Conceptos STAGGER (3 funciones objetivo)	STA	3	0	2	Cambiando la función objetivo
Conceptos AGRAWAL (10 funciones objetivo)	AGR	6	3	2	Cambiando la función objetivo
Pantalla LED de 7 segmentos (5 % de ruido)	LED	24	0	10	Intercambiando atributos relevantes e irrelevantes
Hiperplano	HYP	0	10	2	Variando el peso en el hiperplano
Funciones de base radial	RBF	0	10	3	Variando la posición de los centroides

Tabla 6.1: Principales características de los generadores de flujos de datos usados en el estudio experimental con datos sintéticos.

nales y numéricos, así como considerando diferentes tipos de funciones objetivo (e.g. con frontera de clases lineales y no lineales).

En conceptos estables (Tabla 6.2), los algoritmos son entrenados con 1 000 000 de ejemplos usando un acercamiento *holdout* para realizar pruebas a los algoritmos cada 1 000 ejemplos de entrenamiento con otros 1 000 ejemplos de prueba. La Tabla 6.2 muestra los resultados promediados de los algoritmos en conceptos estables, considerando todos los generadores de flujos de datos descritos en la Tabla 6.1. En los resultados promediados de esta tabla también se incluyen las tres funciones objetivo de STAGGER y las diez de AGRAWAL.

Como se esperaba, IADEM-2c y IADEM-3 tienen un rendimiento similar en promedio, aunque IADEM-3 tiene una precisión ligeramente superior. En la mayoría de los casos, ambos algoritmos tuvieron medidas de precisión idénticas, lo que indica que IADEM-3 es robusto al ruido y a las detecciones falsas. Tal robustez está dada fundamentalmente por HDDM$_{A\text{-test}}$, el método usado para la detección de cambios significativos en la precisión. La Tabla 6.2 también muestra que ambos algoritmos, en promedio, inducen árboles mucho más pequeños y tienen un tiempo de procesamiento menor que el resto de los algoritmos; aunque fueron superados en precisión.

Con respecto a VFDT y su versión adaptativa (HAT), en los conjuntos de datos considerados, ellos raramente alcanzaron la misma precisión debido a que HAT llevó a cabo varios intercambios de subárboles durante el proceso de inducción. Esto sugiere que el mecanismo de adaptación de HAT es más susceptible al ruido. También podemos apreciar que la diferencia en el número de nodos en este caso también es más apreciable.

La Figura 6.2 muestra las curvas de aprendizaje de los algoritmos en el generador LED. Podemos ver un comportamiento típico de IADEM-2c y IADEM-3 en

		Media
IADEM-2	Precisión	91,00
	Nodos	*23,08*
	Tiempo (seg.)	68
IADEM-3	Precisión	91,59
	Nodos	24,82
	Tiempo (seg.)	101
VFDT	Precisión	94,00
	Nodos	251,68
	Tiempo (seg.)	157
HAT	Precisión	93,63
	Nodos	297,74
	Tiempo (seg.)	230
OnlineTree2	Precisión	**94,02**
	Nodos	1279,11
	Tiempo (seg.)	1245

Tabla 6.2: Resultados promediados del rendimiento de los algoritmos sobre todos los flujos de datos sintéticos en conceptos estables. Los algoritmos fueron entrenados con 1 000 000 de ejemplos.

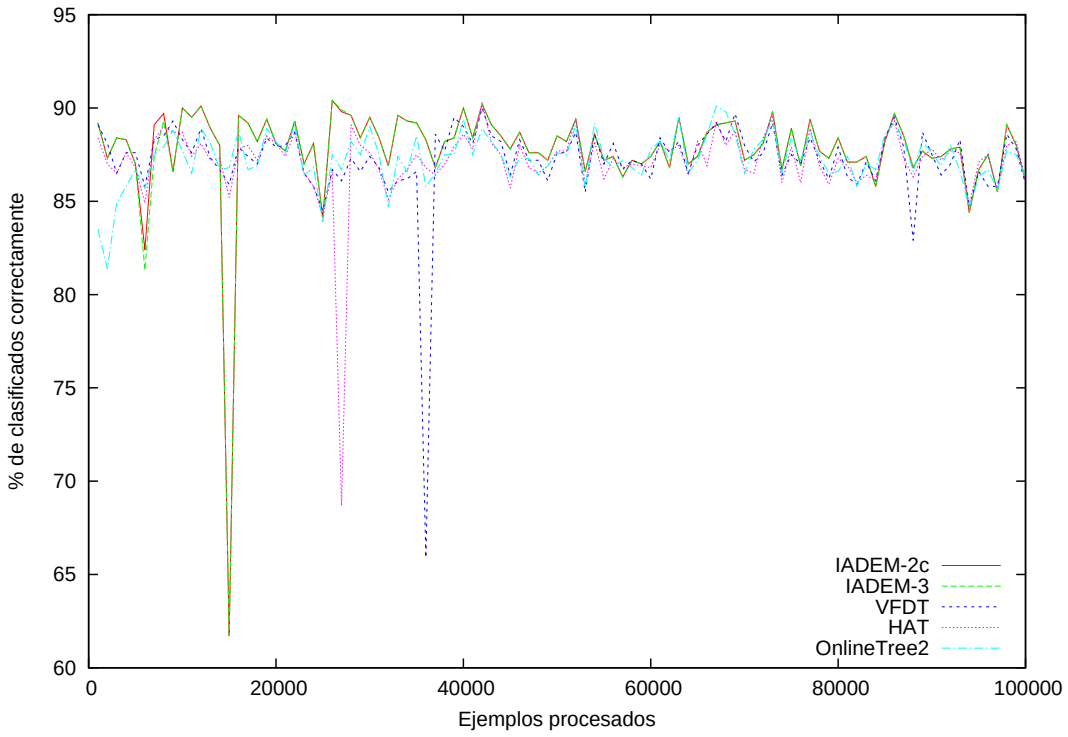


Figura 6.2: Curvas de aprendizaje de los algoritmos en conceptos estables sobre el generador LED.

conceptos estables, en los que tienen una curva de aprendizaje muy similar. También podemos notar que VFDT y HAT no tienen tal comportamiento debido a la detección de falsos positivos.

Los algoritmos también fueron ejecutados en flujos de datos no estacionarios (Tablas 6.3 y 6.4), cambiando los conceptos acorde a la Tabla 6.1. Para simular cambios graduales usamos la función sigmoide incrementando la probabilidad que a cada marca de tiempo los nuevos ejemplos pertenezcan al nuevo concepto [BHKP10]. En estas tablas se generaron 30 000 ejemplos por concepto estable, 30 cambios de concepto, y en cambios graduales (Tabla 6.4), configuramos el período de transición entre conceptos a 3 000. Para evaluar el rendimiento de los algoritmos, la precisión y el número de nodos fue calculado cada 100 ejemplos de entrenamiento. La precisión fue calculada a través de 100 ejemplos de prueba diferentes en cada paso de prueba (cada 100 ejemplos de entrenamiento). Las Tablas 6.3 y 6.4 resumen el rendimiento de los algoritmos en términos de promedio ($\overline{x}$) y desviación estándar (s) para las medidas de precisión y número de nodos. Estas tablas también muestran el tiempo de procesamiento (en segundos) que cada algoritmo tardó en procesar todos los ejemplos de entrenamiento y prueba. Las diferencias significativas de estas medidas con respecto los restantes algoritmos están mostradas en negritas para la precisión, y en itálicas para el número de nodos. El menor tiempo de procesamiento para cada flujo de datos está subrayado.

Las Tablas 6.3 y 6.4 muestran que las versiones adaptativas de IADEM-2c y VFDT (IADEM-3 y HAT respectivamente) claramente tiene un mejor rendimiento que sus pares. Podemos apreciar que IADEM-3 es competitivo en las tres medidas de rendimiento (precisión, número de nodos y tiempo de procesamiento) con respecto a HAT y OnlineTree2.

IADEM-3 detectó todos los cambios de conceptos y adaptó el modelo de aprendizaje con eficacia a estos cambios. Hemos observado que el método para manipular cambio de concepto en los nodos hojas, que inicializa todas sus variables cuando el detector de cambio estima un cambio de concepto, mejora significativamente la precisión en la clasificación de IADEM-3. El método de predicción usado en los nodos de decisión para combinar las predicciones de los subárboles alternativos y el árbol principal también se mostró efectivo. Sin embargo, estos acercamientos no han sido considerados en algoritmos previos como CVFDT, OnlineTree2 y HAT.

HAT tiene un tiempo de procesamiento pequeño e induce árboles pequeños. Sin embargo, su algoritmo base (VFDT) no induce árboles tan pequeños, lo que sugiere que están ocurriendo frecuentemente cambio de subárboles. Naturalmente, los cambios de subárboles innecesarios pueden deteriorar significativamente la precisión del algoritmo de aprendizaje (por ejemplo, ver la Tabla 6.2), por lo que dar más nivel de confianza a la prueba estadística usada en HAT para alternar subárboles podría mejorar su rendimiento.

Las Figuras 6.3 y 6.4 muestras curvas de aprendizaje de los algoritmos en LED (cambios abruptos) y AGRAWAL (cambios graduales). En correspondencia con las Tablas 6.3 y 6.4, los cambios de concepto ocurren cada 30 000 ejemplos de entrenamiento. Podemos notar como los algoritmos que manipulan cambio de concepto (IADEM-2, HAT y OnlineTree2) se recuperan cuando su precisión cae debido a un cambio de concepto. Por ejemplo, la Figura 6.3 muestra que IADEM-3 puede

			STA	LED	AGR	HYP	RBF
IADEM-2	Precisión	$\overline{x}$	72,85	53,29	62,51	60,67	86,21
		s	13,98	21,25	14,01	20,25	2,22
	Nodos	$\overline{x}$	64,20	35,13	*17,63*	30,85	40,07
		s	13,67	9,63	*6,75*	9,20	8,92
	Tiempo	*sec.*	157	372	143	201	249
IADEM-3	Precisión	$\overline{x}$	99,79	**89,86**	**90,61**	**92,05**	87,05
		s	0,62	**2,03**	**6,67**	**2,35**	2,34
	Nodos	$\overline{x}$	15,30	52,18	62,78	23,26	*38,90*
		s	1,22	16,76	56,16	12,42	*9,98*
	Tiempo	*sec.*	69	879	642	240	310
VFDT	Precisión	$\overline{x}$	78,11	58,90	66,05	64,69	91,70
		s	21,20	18,25	11,12	20,03	2,84
	Nodos	$\overline{x}$	454,29	138,60	589,90	310,64	389,82
		s	283,58	94,04	343,53	178,71	214,12
	Tiempo	*sec.*	94	1046	1723	965	1155
HAT	Precisión	$\overline{x}$	99,57	86,64	89,60	**92,06**	**91,72**
		s	2,28	3,20	10,80	**3,70**	**2,95**
	Nodos	$\overline{x}$	6,83	*1,88*	27,37	*21,32*	502,31
		s	3,05	*2,15*	17,34	*17,00*	274,46
	Tiempo	*sec.*	39	77	130	111	1564
OnlineTree2	Precisión	$\overline{x}$	**99,95**	64,71	89,92	85,61	90,24
		s	**0,39**	17,39	8,92	10,25	4,10
	Nodos	$\overline{x}$	*5,16*	3323,19	601,37	1574,35	2272,37
		s	*2,51*	1284,94	307,02	307,02	920,81
	Tiempo	*sec.*	48	14339	2208	11453	19624

Tabla 6.3: Medidas de rendimiento de los algoritmos sobre cambios abruptos. Los cambios ocurren cada 30 000 ejemplos de entrenamiento. Fueron 30 cambios de concepto.

			STA	LED	AGR	HYP	RBF
IADEM-2	Precisión	$\overline{x}$	72,70	52,33	62,63	62,87	86,09
		s	13,48	20,88	14,49	27,93	2,28
	Nodos	$\overline{x}$	64,32	32,17	38,09	31,47	*36,86*
		s	13,33	7,87	13,36	10,42	*8,34*
	Tiempo	*sec.*	157	359	289	205	241
IADEM-3	Precisión	$\overline{x}$	97,69	**88,60**	**89,94**	**91,91**	86,95
		s	5,25	**1,94**	**6,68**	**4,29**	2,35
	Nodos	$\overline{x}$	*16,86*	32,26	79,18	*19,79*	38,48
		s	*5,00*	14,16	45,67	*7,55*	8,91
	Tiempo	*sec.*	79	661	710	240	326
VFDT	Precisión	$\overline{x}$	77,16	55,78	66,02	68,06	91,97
		s	19,86	18,71	12,11	26,91	2,81
	Nodos	$\overline{x}$	455,37	124,12	556,43	307,84	396,79
		s	283,72	63,86	321,48	175,04	218,49
	Tiempo	*sec.*	99	873	1575	805	1034
HAT	Precisión	$\overline{x}$	**97,96**	85,18	88,15	91,65	**91,61**
		s	**6,72**	6,16	11,92	8,11	**2,85**
	Nodos	$\overline{x}$	16,89	*4,44*	*32,66*	33,35	491,74
		s	11,37	*4,79*	*25,87*	39,11	265,61
	Tiempo	*sec.*	51	105	152	155	1704
OnlineTree2	Precisión	$\overline{x}$	97,60	62,92	88,89	87,22	89,94
		s	7,00	16,80	9,64	9,62	3,81
	Nodos	$\overline{x}$	33,58	3686,24	801,06	1588,64	2082,51
		s	9,18	1481,32	411,11	773,68	740,11
	Tiempo	*sec.*	145	14509	2741	11014	14547

Tabla 6.4: Medidas de rendimiento de los algoritmos sobre cambios graduales. Los cambios graduales ocurren cada 30 000 ejemplos de entrenamiento con 3 000 ejemplos en el período de transición entre conceptos consecutivos. Fueron 30 cambios de concepto.

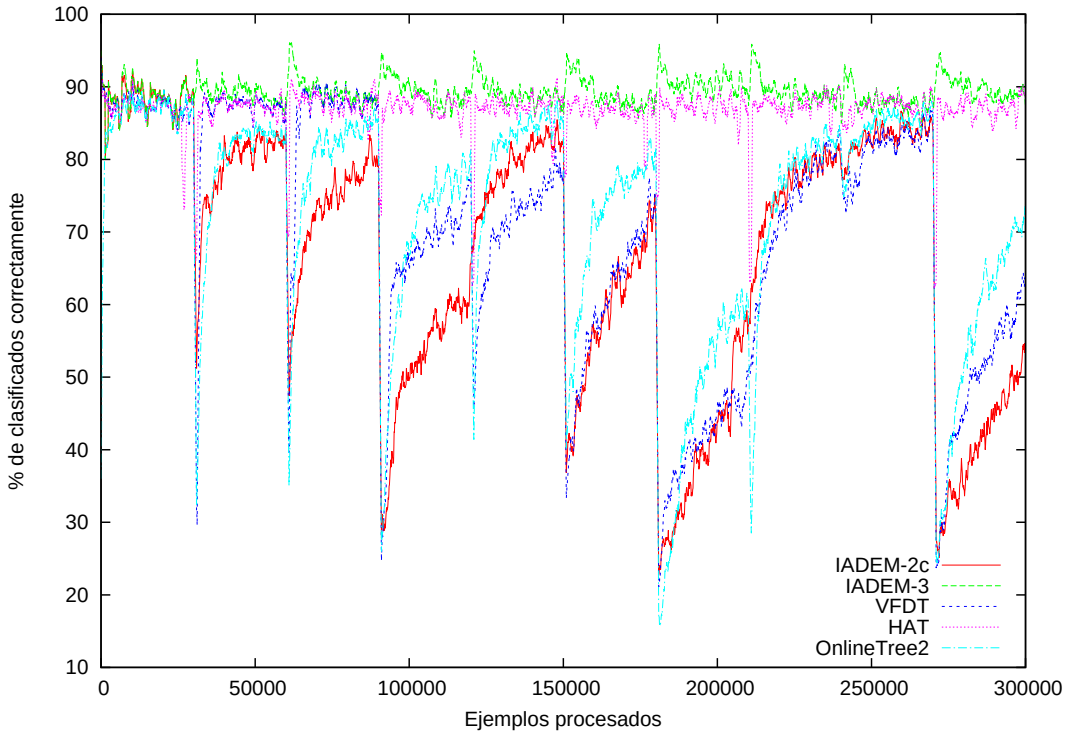


Figura 6.3: Curvas de aprendizaje de los algoritmos en los conceptos LED con cambios abruptos. Los cambios ocurren cada 30 000 ejemplos de entrenamiento.

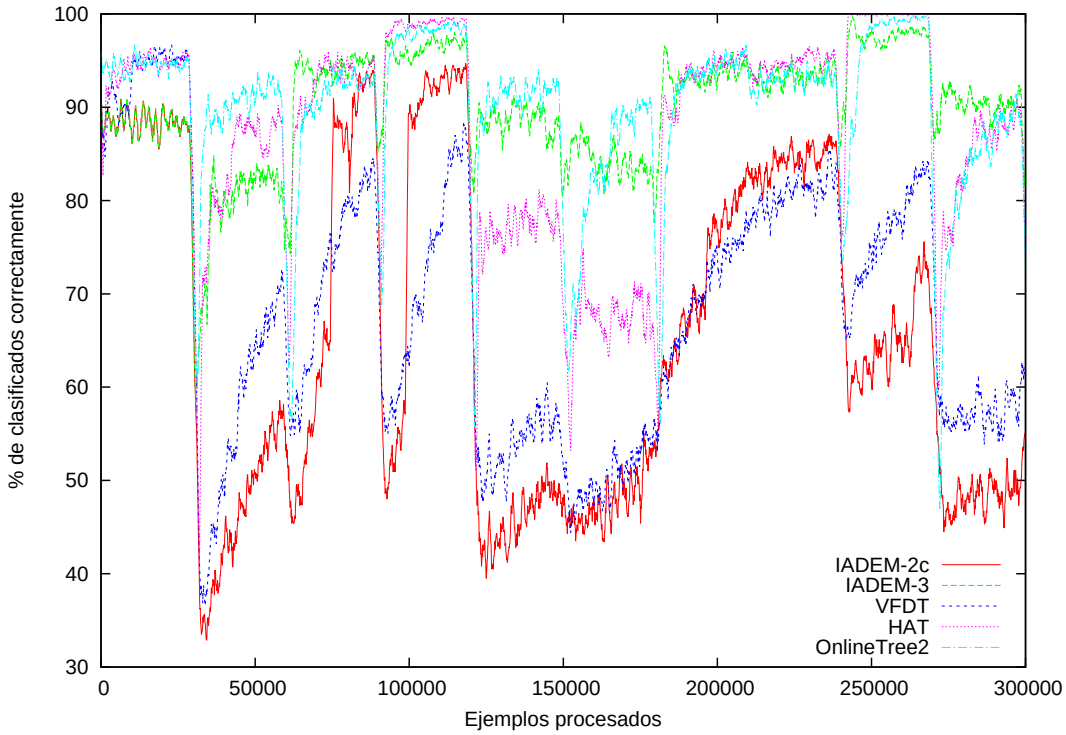


Figura 6.4: Curvas de aprendizaje de los algoritmos en conceptos AGRAWAL con cambios graduales. Los cambios ocurren cada 30 000 ejemplos y con 3 000 ejemplos de entrenamiento en el período de transición entre conceptos.

adaptarse a estos cambios rápidamente, ya que el deterioro de su precisión no es siquiera visible.

Estas figuras también muestran que IADEM-2 y VFDT algunas veces se recuperan de la caída de precisión instantáneamente en algunos cambios de concepto, aunque estos no los manipulan. Esto es debido a que en estos experimentos los conceptos anteriores pueden reaparecer. De esta forma, una estrategia para manipular conceptos que reaparecen es una opción interesante, más que en algunas aplicaciones reales pueden estar presentes este tipo de cambio [Žlio9b, GŽB+].

6.3.3 *Datos del mundo real*

Los conjuntos de datos reales que estudiamos en esta sección han sido usados en varios trabajos investigativos relacionados con cambio de concepto. Para estos conjuntos de datos, no hay afirmaciones fuertes de la presencia o tipo de cambio subyacentes en los datos. En este experimento evaluamos los métodos procesando los ejemplos en su orden temporal. A la llegada de cada nuevo ejemplo de entrenamiento, al clasificador primero se le realiza una prueba eliminado la etiqueta de clase del ejemplo, luego el aprendizaje continúa normalmente con el ejemplo original. La precisión es calculada con respecto a una ventana deslizante de tamaño fijo igual a 1 000 [BHKP10, GSaR13]. Para la precisión, la Tabla 6.5 muestra el promedio y la desviación estándar de la fracción entre el número de ejemplos bien clasificados y el total de ejemplos cada 100 ejemplos procesados. Las medidas relacionadas con el número de nodos son calculadas también cada 100 ejemplos. De forma similar, la Tabla 6.5 muestra el tiempo de procesamiento que demoró cada algoritmo en procesar todos los ejemplos.

Específicamente, los algoritmos se ejecutan en el conjunto de datos *Electricity Market* (ELEC)[2] y otros cinco obtenidos del repositorio de la UCI:[3] *Forest Cover Type* (COV), dos conjuntos de datos basados en una colección de 20 grupos de noticias conocidos como *usenet1* y *usenet2* (USE1 y USE2), una colección de correo spam (SPAM) [KTV08] y el conjunto de datos *Nursery* (NURS).

IADEM-3 también tiene un buen rendimiento en estos conjuntos de datos del mundo real. El método para manipular cambio de concepto en las hojas permite reaccionar más rápidamente a los cambios cuando estos ocurren con mucha frecuencia (por ejemplo, en ELEC [RATH12]). En estos casos el entrenamiento de subárboles alternativos puede no ser suficiente. La inducción del árbol puede venir aparejada con un aprendizaje lento para los algoritmos basados en cotas de concentración probabilísticas (IADEM-2c, IADEM-3, VFDT y HAT), ya que estas cotas pueden requerir de muchos ejemplos para decidir por el mejor atributo para la expansión. Así, la alternación de subárboles puede tardar mucho y el modelo aprende conceptos unificados, deteriorando significativamente su precisión.

La Figura 6.5 muestra las curvas de aprendizaje de los algoritmos ejecutándose en el conjunto de datos ELEC. En esta figura podemos ver que la precisión de los algoritmos no cae tan drásticamente cuando ocurre un cambio como en los conjuntos de datos sintéticos. Esto indica que los cambios son graduales, lentos

2 http://moa.cms.waikato.ac.nz/datasets/
3 http://www.ics.uci.edu/mlearn/MLRepository.html

			ELEC	COV	USE1	USE2	SPAM	NURS
IADEM-2	Precisión	$\overline{x}$	76,35	73,28	61,68	70,01	90,82	85,99
		s	8,72	9,42	4,54	2,62	6,11	9,95
	Nodos	$\overline{x}$	17,11	48,70	*1,00*	*1,00*	5,62	4,08
		s	9,91	18,39	*0,00*	*0,00*	1,91	1,14
	Tiempo	*sec.*	11	593	1	1	20	3
IADEM-3	Precisión	$\overline{x}$	**87,88**	**91,83**	**71,85**	**73,04**	**91,37**	88,10
		s	**2,74**	**3,89**	**3,76**	**3,17**	**6,30**	4,15
	Nodos	$\overline{x}$	11,23	164,66	*1,00*	*1,00*	*4,09*	*3,79*
		s	4,73	88,57	*0,00*	*0,00*	*1,48*	*1,35*
	Tiempo	*sec.*	20	1689	1	1	26	3
VFDT	Precisión	$\overline{x}$	78,94	80,56	60,14	69,82	90,39	89,23
		s	8,22	8,03	4,95	2,74	8,78	5,78
	Nodos	$\overline{x}$	30,97	182,92	*1,00*	*1,00*	4,38	12,58
		s	12,56	104,95	*0,00*	*0,00*	1,82	5,42
	Tiempo	*sec.*	2	195	0	0	4	0
HAT	Precisión	$\overline{x}$	83,68	82,58	61,68	69,88	90,87	**90,25**
		s	5,18	9,34	5,09	3,35	8,96	**3,68**
	Nodos	$\overline{x}$	*7,45*	*5,32*	*1,00*	*1,00*	5,33	6,81
		s	*4,53*	*4,20*	*0,00*	*0,00*	3,11	3,32
	Tiempo	*sec.*	2	58	0	0	6	1
OnlineTree2	Precisión	$\overline{x}$	82,77	84,66	70,37	71,66	85,28	89,38
		s	3,58	6,24	3,31	3,08	8,91	5,44
	Nodos	$\overline{x}$	17,07	284,34	40,47	55,27	4,57	36,75
		s	7,51	83,37	25,33	19,21	5,19	14,93
	Tiempo	*sec.*	6	630	1	1	5	1

Tabla 6.5: Medidas de rendimiento de los algoritmos en conjuntos de datos del mundo real.

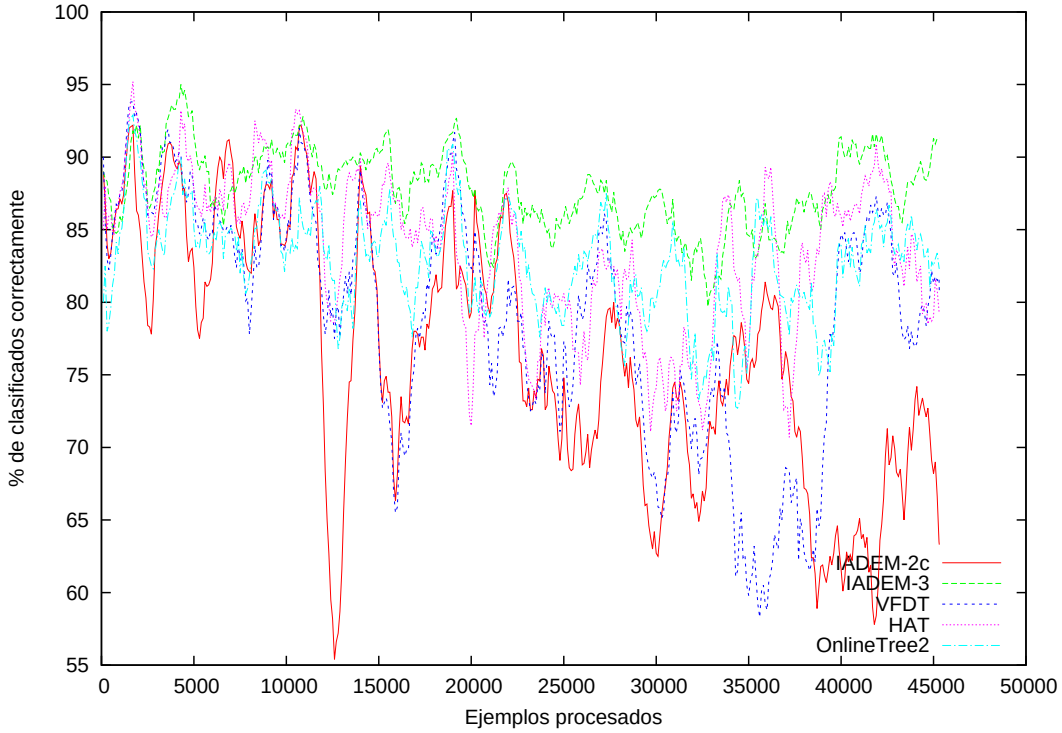


Figura 6.5: Curvas de aprendizaje de los algoritmos en el conjunto de datos *Electricity Market*.

y locales. Note además cómo la precisión de los algoritmos que no manipulan cambio de concepto (IADEM-2c y VFDT) se deteriora gradual y continuamente.

6.4 CONCLUSIONES DEL CAPÍTULO

En este capítulo hemos presentado un nuevo algoritmo para la inducción en línea de árboles de decisión, el cual es capaz de aprender en flujos de datos no estacionarios. IADEM-3, el nuevo algoritmo, reconstruye el modelo de aprendizaje cuando estima que ha ocurrido un cambio de concepto construyendo subárboles alternativos. Estos subárboles alternativos reemplazan partes del árbol principal si este reemplazo mejora significativamente la precisión del modelo de aprendizaje resultante. Un detector de cambio monitoriza el rendimiento de subárboles, el cual dispara el mecanismo de adaptación cuando estima un deterioro significativo del rendimiento.

En las tareas de clasificación, el modelo de aprendizaje inducido por IADEM-3 es visto como un árbol de opciones, dado que un ejemplo puede tomar distintos caminos teniendo en cuenta el árbol original y los subárboles alternativos. De esta forma, la predicción del árbol principal y los subárboles alternativos son combinadas para clasificar ejemplos sin etiqueta de clase.

IADEM-3 satisface importantes requerimientos para el aprendizaje en línea: tiene complejidad temporal y espacial constante por ejemplo procesado, aprende con una sola pasada por los ejemplos vistos, y no depende del ajuste de parámetros por el usuario para la manipulación de cambio de concepto.

En este capítulo también hemos comparado a IADEM-3 con los principales algoritmos incrementales basados en árboles de decisión. Hemos mostrado que IADEM-3 es competitivo en precisión, tiempo de procesamiento y tamaño del modelo inducido considerando varios conjuntos de datos sintéticos y del mundo real, así como los tipos más comunes de cambio. IADEM-3 mejora la precisión de su predecesor IADEM-2c no solo en flujos de datos no estacionarios, sino también cuando los conceptos son estables.

Por otro lado, IADEM-3 no es capaz de manipular explícitamente conceptos recurrentes, otro tipo de cambio que puede estar presente en muchos problemas del mundo real.

Parte III

CONCLUSIONES Y ANEXOS

7

CONCLUSIONES Y TRABAJOS FUTUROS

En esta tesis hemos abordado, como objetivo general, aumentar la precisión de los modelos de clasificación o predicción bajo cambio de concepto, considerando las restricciones computacionales comunes en estos escenarios y los tipos de cambios más frecuentes. En este último capítulo mostramos las principales aportaciones de la tesis y resumimos las futuras líneas de investigación.

7.1 CONCLUSIONES

En este trabajo hemos definido un marco de trabajo para el aprendizaje en flujos de datos no estacionarios, teniendo en cuenta los tipos de cambio más comunes que están presentes en muchos problemas reales, así como los requerimientos generales para el aprendizaje en línea. Dentro de este marco de trabajo, hemos realizado un estado del arte que incluye las estrategias, algoritmos, metodologías de evaluación y herramientas de software para el aprendizaje en línea adaptativo.

Posteriormente hemos presentado un método para desarrollar algoritmos que pueden aprender en flujos de datos y adaptarse a cambios de concepto. Este método tiene como componente fundamental un detector de cambio de concepto, que dispara señales de cambio cuando estima, a través de la media aritmética o del estimador EWMA, que un algoritmo de aprendizaje dado ha sufrido un deterioro significativo en su rendimiento. En particular, hemos propuesto una familia de detectores de cambio (HDDM) que no asume conocimiento relacionado con la función de distribución de probabilidad que rige a los valores de rendimiento, provee garantías probabilísticas de desempeño para las tasas de falsos positivos y falsos negativos, procesa los datos de entrada con complejidad computacional constante en tiempo y espacio, y no requiere del ajuste de parámetros por parte del usuario. Otra característica interesante del método propuesto es que al procesar las medidas de rendimiento no depende del algoritmo de aprendizaje, por lo que este puede ser aplicado a cualquier clasificador para la manipulación de cambio de concepto.

Hemos probado empíricamente que el método propuesto es efectivo en comparación con otros métodos similares en el área, así como en el desarrollo de versiones adaptativas a partir de algoritmos clásicos en escenarios de flujos de datos como el clasificador Naïve Bayes y el Perceptrón simple. Para ello hemos considerado diversas características del cambio y de las funciones objetivo (cambios abruptos y graduales, locales y globales, conjuntos de datos con diferentes niveles de ruido, con atributos nominales y numéricos, etc.).

También extendimos a la familia de algoritmos de inducción de árboles de decisión IADEM, específicamente a IADEM-2, en tres dimensiones fundamentales: instalación de pruebas de división binarias en los nodos de decisión, manipulación de atributos numéricos y manipulación de cambio de concepto. Al algoritmo resultante de todas estas modificaciones lo hemos nombrado IADEM-3. IADEM-3 satisface importantes requerimientos para el aprendizaje incremental: tiene com-

plejidad temporal y espacial constante por ejemplo procesado, aprende con una sola pasada por los ejemplos vistos, y no depende del ajuste de parámetros por el usuario para la manipulación de cambio de concepto.

De esta forma, IADEM-3 presenta mecanismos para inducir árboles binarios, así como la posibilidad de inducir árboles de decisión tanto con pruebas de división binarias como con salidas múltiples. Este último acercamiento alcanzó, en la mayoría de los experimentos realizados, una mejora significativa de la precisión del algoritmo en la clasificación, manteniendo acotado el coste computacional del algoritmo e induciendo árboles de aproximadamente igual tamaño al inducido por IADEM-2. Considerar pruebas de división binarias permite modelar problemas más complejos, mientras que las pruebas con salidas múltiples por lo general dan lugar a árboles de decisión más pequeños e interpretables.

Estudiamos además el comportamiento de IADEM-3 en combinación con algunos de los métodos de discretización más conocidos en el aprendizaje en flujos de datos. En la adaptación de estos métodos a IADEM-3, propusimos un algoritmo para disminuir el coste computacional de los métodos de discretización a medida que el árbol crece, suponiendo que la precisión de estos métodos de discretización no es tan importante en los nodos con mayor profundidad en el árbol. La idea fundamental es tener más precisión en las estimaciones de puntos de corte relevantes al inicio de la inducción, sin aumentar significativamente el coste computacional del algoritmo.

Por otro lado, IADEM-3 manipula cambios de concepto reconstruyendo el modelo de aprendizaje cuando estima que ha ocurrido un cambio. Esta reconstrucción se lleva a cabo induciendo subárboles alternativos. Estos subárboles alternativos pueden reemplazar partes del árbol principal en respuesta a un cambio. Una instancia de HDDM monitoriza constantemente el rendimiento de los subárboles, y dispara el mecanismo de adaptación cuando estima un deterioro significativo del rendimiento. De esta forma, el mecanismo de adaptación al cambio considera las características particulares de los árboles de decisión inducidos por la familia IADEM, siendo así más efectiva dicha adaptación.

El modelo de aprendizaje inducido por IADEM-3 también es visto como un árbol de opciones, dado que un ejemplo puede tomar varios caminos desde la raíz hasta las hojas teniendo en cuenta el árbol original y los subárboles alternativos. De esta forma, la predicción del árbol principal y los subárboles alternativos son combinadas para clasificar ejemplos sin etiqueta de clase.

Finalmente, hemos comparado a IADEM-3 con respecto a los principales algoritmos incrementales basados en árboles de decisión. Hemos mostrado que IADEM-3 es competitivo en precisión, tiempo de procesamiento y tamaño del modelo inducido considerando varios conjuntos de datos sintéticos y del mundo real, así como los tipos más comunes de cambio.

7.2 LIMITACIONES Y TRABAJOS FUTUROS

Como trabajo futuro nos enfocaremos fundamentalmente en dos direcciones: proveerle más generalidad al método de detección de cambio presentado en el Capítulo 4, y aumentar la convergencia del modelo y el área de aplicación de IADEM-3 (descrito en los Capítulos 5 y 6).

7.2.1 *Detectores de cambio en línea*

Las garantías teóricas de desempeño de los detectores de cambio en el área del aprendizaje en flujos de datos (y de nuestro nuevo método HDDM) han estado enfocadas en las tasas de falsos positivos y falsos negativos. Adicionalmente, en el Capítulo 4 hemos calculado empíricamente el retardo en la detección del cambio bajo diversas condiciones. Sin embargo, algunos autores han propuesto otras medidas teóricas de rendimiento a considerar en el diseño y evaluación de los algoritmos de detección de cambio [BN93]; por ejemplo la media del retardo en la detección de cambio, el tiempo medio entre falsas alarmas, la precisión en la estimación de la magnitud del cambio y en el tiempo de cambio, etc.

Algunas investigaciones recientes han extendido las garantías teóricas de los detectores de cambio al algoritmo de aprendizaje, como por ejemplo a árboles de decisión [BG09a]. Los detectores de cambio que usan como estimadores a la media aritmética y al estadístico EWMA también han sido usados en la evaluación del rendimiento de los algoritmos de aprendizaje en línea en ambientes no estacionarios [GSaR13].

7.2.2 *Inducción de árboles de decisión*

Una limitación de IADEM-3, heredada de la familia de algoritmos IADEM, es que en ciertos conceptos él puede tardar en ajustar los intervalos de confianza y así decidir por el mejor atributo para la expansión, implicando que en estos casos el algoritmo necesite grandes cantidades de datos para alcanzar niveles adecuados de precisión.

En la literatura, a este problema se le ha dado solución mediante la generación de árboles aleatorios al principio de la inducción; pero el método no es tan efectivo si el problema presenta muchos atributos irrelevantes. Otra posibilidad es la combinación de IADEM-3 con otros algoritmos basados en árboles de decisión. Por ejemplo, OnlineTree2 [NFM07] por lo general aprende los conceptos subyacentes más rápidamente, pero luego induce árboles de decisión mucho más grandes comparado con IADEM-3 sin aumentar significativamente la precisión. La hibridación de ambas técnicas, induciendo al inicio del proceso de aprendizaje con OnlineTree2 y luego con IADEM-2 es un objetivo futuro de investigación. Emparejado a esto, el cálculo de intervalos de confianza para reglas de división de atributos adicionales a través de las cotas de McDiarmid es una propuesta reciente de investigación [RPDJ13].

Otra limitación que no hemos observado en la experimentación realizada, pero que puede aparecer en IADEM-3, sería consecuencia de una alta dimensionalidad del conjunto de datos. Entre nuestras líneas futuras de investigación se encuentra modificar a IADEM-3 para afrontar esta dificultad, controlando el tamaño del modelo inducido de forma que solo se almacene la información más útil, inhabilitando temporalmente las ramas menos prometedoras [BHKP10].

Otra extensión de utilidad es la posibilidad de ejecutarse sobre conjuntos de datos con valores de atributos desconocidos. El tratamiento de atributos con valores desconocidos ampliaría aún más el ámbito de aplicación de nuestros algoritmos. Por otro lado, IADEM-3 no es capaz de manipular explícitamente conceptos recu-

rrentes, otro tipo de cambio que puede estar presente en muchos problemas del mundo real.

A

DEMOSTRACIONES MATEMÁTICAS

A.1 DEMOSTRACIÓN DEL COROLARIO 3

Primero consideremos la cota de probabilidad para el error de tipo I (probabilidad de detección falsa). Si $E\left[\overline{X}\right] \leq E\left[\overline{Y}\right]$ (o equivalentemente $E\left[\overline{X}\right] - E\left[\overline{Y}\right] \leq 0$), por inclusión (i.e. si $A \Rightarrow B$ entonces $\Pr(A) \leq \Pr(B)$) tenemos que ($D = \overline{X} - \overline{Y}$ y $E_D = E\left[\overline{X}\right] - E\left[\overline{Y}\right]$)

$$\Pr\left\{D \geq \varepsilon_\alpha\right\} \leq \Pr\left\{D - E_D \geq \varepsilon_\alpha\right\} \tag{A.1}$$

Aplicando inversión en la ecuación (4.1):

$$\Pr\left\{D - E_D \geq \varepsilon_\alpha\right\} \leq \alpha \tag{A.2}$$

Aplicando transitividad en las ecuaciones (A.1) y (A.2), obtenemos $\Pr\left\{\overline{X} - \overline{Y} \geq \varepsilon_\alpha\right\} \leq \alpha$ como queríamos demostrar.

Consideremos ahora la cota para el error de tipo (probabilidad de no detección). Si $E\left[\overline{X}\right] \geq E\left[\overline{Y}\right] + \varsigma$ entonces para $\varsigma > \varepsilon_\alpha$

$$\begin{aligned} \Pr\left\{D < \varepsilon_\alpha\right\} &= \Pr\left\{D - \varsigma < \varepsilon_\alpha - \varsigma\right\} \\ &\leq \Pr\left\{D - E_D < \varepsilon_\alpha - \varsigma\right\} \end{aligned} \tag{A.3}$$

Ya que para $\varepsilon > 0$, $\Pr\left\{D - E_D > \varepsilon\right\} = \Pr\left\{D - E_D < -\varepsilon\right\}$; como $\varepsilon_\alpha - \varsigma < 0$, a partir de la ecuación (4.1):

$$\Pr\left\{D - E_D < \varepsilon_\alpha - \varsigma\right\} \leq e^{\frac{-2(\varsigma - \varepsilon_\alpha)^2}{\left(n^{-1} + m^{-1}\right)(b-a)^2}} \tag{A.4}$$

Finalmente, nuevamente aplicando transitividad en las ecuaciones (A.3) y (A.4), tenemos

$$\Pr\left\{\overline{X} - \overline{Y} < \varepsilon_\alpha\right\} \leq e^{\frac{-2(\varsigma - \varepsilon_\alpha)^2}{\left(n^{-1} + m^{-1}\right)(b-a)^2}}$$

A.2 DEMOSTRACIÓN DEL COROLARIO 5

Sea $\overrightarrow{X} = (X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m)$ y $f(\overrightarrow{X}) = \hat{X}_n - \hat{Y}_m$. Entonces, si $\hat{X}_n$ y $\hat{Y}_n$ satisfacen la condición de diferencias acotadas independientes en el Teorema 4, es fácil de demostrar que para $f(\overrightarrow{X})$ la condición de diferencias acotadas independientes es satisfecha tomando $d_i = (b-a)v_i$ para todo $i \in \{1, 2, \ldots, n\}$, y $d_j = (b-a)v_j'$ para todo $j \in \{1, 2, \ldots, m\}$. Entonces

$$\sum_{i=1}^{n+m} d_i^2 = (b-a)^2 \left[\sum_{i=1}^{n} (v_i)^2 + \sum_{i=1}^{m} \left(v_i'\right)^2\right] = \mathcal{D}_{n,m}.$$

Finalmente, tomándolo todo junto, obtenemos la cota deseada a partir de la ecuación (4.4).

A.3 DEMOSTRACIÓN DEL EJEMPLO 7

A n variables aleatorias vistas, los pesos en el estadístico EWMA toman la forma $v_1 = (1-\lambda)^{n-1}$, y $v_i = \lambda(1-\lambda)^{n-i}$ para $1 < i \leq n$ (note que $\sum_{i=1}^{n} v_i = 1$). Entonces para $n > 1$ ($\mathcal{D}_1 = 1$) tenemos:

$$\begin{aligned}
\mathcal{D}_n &= \sum_{i=1}^{n} d_i^2 = (b-a)^2 \sum_{i=1}^{n} v_i^2 \\
&= (b-a)^2 \left[\sum_{i=2}^{n} \lambda^2 (1-\lambda)^{2(n-i)} + (1-\lambda)^{2(n-1)} \right] \\
&= (b-a)^2 \left[\lambda^2 \frac{1-(1-\lambda)^{2(n-1)}}{1-(1-\lambda)^2} + (1-\lambda)^{2(n-1)} \right]
\end{aligned}$$

Cuando $n \to \infty$, $\mathcal{D}_n$ converge a un valor constate ya que

$$\lim_{n\to\infty} \mathcal{D}_n = \frac{(b-a)^2 \lambda}{2-\lambda}$$

.

Como $\sum_{i=1}^{n} d_i^2 = \mathcal{D}_n$, $f(\overrightarrow{X}) = \hat{X}_n$ y $E[f(\overrightarrow{X})] = E\left[\hat{X}_n\right]$; a partir de la ecuación (4.4) obtenemos la cota deseada.

A.4 DEMOSTRACIÓN DEL EJEMPLO 8

Esta vez la condición de diferencias acotadas independientes es

$$\begin{aligned}
\mathcal{D}_n &= \sum_{i=1}^{n} d_i^2 = \left(\frac{b-a}{\mathcal{W}_n}\right)^2 \sum_{i=1}^{n} \beta^{2(n-i)} \\
&= (b-a)^2 \left(\frac{1-\beta}{1-\beta^n}\right)^2 \left(\frac{1-\beta^{2n}}{1-\beta^2}\right)
\end{aligned}$$

Como $0 < \beta \leq 1$, $\mathcal{D}_n$ converge a $\lim_{n\to\infty} \mathcal{D}_n = (b-a)^2 (1-\beta)/(1+\beta)$. Similar al Ejemplo 7, a partir de la ecuación (4.4) obtenemos la cota deseada.

B

CURSOS DE DOCTORADO

Esta investigación ha sido desarrollada en el marco del Doctorado Iberoamericano en Soft Computing, patrocinado por la Junta de Andalucía y basado en un convenio de colaboración entre las Universidades Andaluzas, la Universidad Central "Martha Abreu" de las Villas y la Asociación Universitaria Iberoamericana de Postgrado. Durante la parte lectiva de este programa se recibieron los cursos de doctorado que se muestran a continuación, de los cuales se especifican sus profesores y respectivos temarios.

- **Conceptos de Recuperación de Información Probabilística.** Profesores: Dr. Juan Huete; Dr. Juan Manuel Fernández Luna (Universidad de Granada); Dr. Ramiro Pérez (UCLV).

  Introducción a la recuperación de información. Indexación. Introducción a las Redes Bayesianas. Modelos de recuperación de información. Técnicas de modificación de consultas. Sistemas de recuperación de información. Recuperación de información Web. Recuperación de información estructurada. Sistemas de recomendación. Agrupamiento y clasificación documental.

- **Procesamiento Digital de Imágenes.** Profesores: Dr. Miguel García Silvente (Universidad de Granada); Dr. Juan Lorenzo (UCLV).

  Visión por ordenador: Introducción, filtrado, extracción de rasgos simples (fronteras y esquinas), extracción de rasgos más complejos (líneas, círculos, etc.), segmentación, representación y descripción de formas. Visión estéreo: Correspondencia entre imágenes, rectificación, cálculo de disparidad y profundidad. Detección de movimiento: Flujo óptico. Seguimiento de objetos: Filtro de partículas, reconocimiento de gestos.

- **Procesado de Datos y Aprendizaje basado en Reglas.** Profesores: Dr. Jesús Aguilar (Universidad de Pablo de Olavide); Dra. Yanet Rodríguez (UCLV).

  Aprendizaje, Minería de Datos y KDD: Clasificación del aprendizaje. Fases del Proceso KDD. Tareas de la Minería de Datos. Enfoques de aprendizaje y modelos de conocimiento. Heurísticas de búsqueda. Preprocesado de datos: La calidad de los datos; tratamiento de valores perdidos, outliers y ruido; normalización y estandarización; discretización y transformación; selección de ejemplos; selección de atributos. Aprendizaje basado en reglas: Diferencias entre modelos comprensibles y no comprensibles, conceptos de complejidad y precisión, medidas de complejidad y precisión, clasificación mediante reglas de decisión, técnicas de visualización de reglas.

- **Aprendizaje basado en Árboles Decisión.** Profesores: Dr. Rafael Morales; Dr. Gonzalo Ramos (Universidad de Málaga); Dra. Yailé Caballero (Universidad de Camagüey).

Algoritmos TDIDT. Árbol de decisión. Estructura general de los TDIDT. Objetivos. Clasificación y Predicción. Conceptos de base: medidas, condición hoja, prepoda y postpoda, discretización, binarización. Cotas de concentración: Chernoff y Hoeffding. Mejoras de los algoritmos TDIDT: Poda con control predictivo, inducción con atributos desconocidos, inducción con costes, inducción con olvido, técnicas de votación. IADEM: Inducción de Árboles de Decisión por Muestreo. CIDIM: Control de Inducción por División Muestral. MultiCIDIM. Experimentación.

- **Redes Neuronales Evolutivas: Aplicaciones.** Profesores: Dr. César Hervás (Universidad de Córdoba); Dra. María García (UCLV).

  Redes Neuronales Artificiales en Modelado de Sistemas Dinámicos. Redes Neuronales de base spline, sigmoide y potencial. Algoritmos Evolutivos para el modelado y entrenamiento de Redes Neuronales. Algoritmos Híbridos para la optimización de modelos de Redes Neuronales de base sigmoide y potencial. Aplicaciones en agronomía de precisión, microbiología, aerobiología y cinética química.

- **Soft Computing: fundamentos, hibridaciones y aplicaciones.** Profesores: Dr. David Pelta (Universidad de Granada); Dr. Rafael Bello (UCLV).

  Fundamentos y componentes de Soft Computing. Elementos básicos de optimización y búsqueda. Esquemas y posibilidades de hibridación. Estrategias cooperativas de resolución de problemas. Aplicaciones de Soft Computing en bioinformática.

- **Algoritmos Genéticos y Sistemas Difusos Evolutivos.** Profesores: Dr. Jorge Casillas (Universidad de Granada); Dr. Carlos Morell (UCLV).

  Algoritmos genéticos. Diversidad versus convergencia. Algoritmos evolutivos multiobjetivo. Introducción a la extracción de conocimiento con algoritmos evolutivos. Sistemas difusos evolutivos: Sistemas basados en reglas difusas (SBRD). SBRD en diferentes problemas: control, regresión, clasificación y modelos descriptivos (reglas de asociación, descubrimiento de subgrupos). Aprendizaje y ajuste de SBRD con algoritmos evolutivos. Algunas aplicaciones: control, robótica móvil, marketing, etc.

- **Modelos Gráficos Probabilísticos.** Profesores: Dr. Serafín Moral (Universidad de Granada); Dr. R. Grau (UCLV).

  Conceptos básicos sobre probabilidad. Algunos conceptos sobre grafos. Independencia y d-separación. Construcción de redes Bayesianas. Inferencia. Aprendizaje. Toma de decisiones con diagramas de influencia. Temas abiertos.

- **Informática Gráfica.** Profesores: Dr. Juan Carlos Torres (Universidad de Granada); Dr. Carlos Pérez (UCLV).

  Gráficos por ordenador. Hardware gráfico. Pipeline. Visualización con OpenGL. Visualización realista. Interacción. Modelado geométrico: Funciones del modelo geométrico, mallas, curvas y superficies, sólidos, volúmenes. Digitalización 3D: Métodos de captura, escáner láser, triangulación, registrado, fu-

sión, simplificación. Realidad virtual: Visualización estéreo, tecnología, interacción, detección de colisiones, software, visualización adaptativa. Modelos gráficos en sistemas GIS: Representación raster, representación vectorial, análisis raster, GIS 3D.

C

PUBLICACIONES

- I. Frías Blanco, J. del Campo Ávila, G. Ramos Jiménez, R. Morales Bueno, A. Ortiz Díaz, y Y. Caballero Mota, "Online and non-parametric drift detection methods based on Hoeffding's bound," IEEE Transactions on Knowledge and Data Engineering, 2014. DOI 10.1109/TKDE.2014.2345382.
- ——, "Aprendiendo con detección de cambios online," Computación y Sistemas, vol. 18, págs. 169–183, 2014.
- ——, "Aprendiendo incrementalmente con cambio de concepto," en II Conferencia Internacional de Ciencias Computacionales e Informáticas, Informática 2013, La Habana, 2013.
- ——, "Detectando cambio de concepto en flujos de datos continuos," en FIMAT 2011, Holguín, Cuba, 2011.
- A. Ortiz Díaz, R. Ramírez Tasé, I. Frías Blanco, G. Ramos Jiménez, R. Morales Bueno, y Y. Caballero Mota, "Comparación entre algoritmos de clasificación basados en árboles de decisión," en V Conferencia Científica de la Universidad de las Ciencias Informáticas, UCIENCIA 2010, Cuba, 2010.
- I. Frías Blanco, A. Ortiz Díaz, G. Ramos Jiménez, R. Morales Bueno, y Y. Caballero Mota, "Clasificadores y multiclasificadores con cambio de concepto basados en árboles de decisión," Revista Iberoamericana de Inteligencia Artificial, vol. 13, págs. 32–43, 2010.
- ——, Tendencias de Soft Computing. Editorial Feijóo, UCLV, 2009, cap. Clasificadores incrementales y cambio de concepto.
- A. Ortiz Díaz, I. Frías Blanco, G. Ramos Jiménez, R. Morales Bueno, y Y. Caballero Mota, Tendencias de Soft Computing. Editorial Feijóo, UCLV, 2009, cap. Algoritmos multiclasificadores incrementales que detectan cambios de concepto.

[AF10] F. Angiulli y F. Fassetti. Distance-based Outlier Queries in Data Streams: The Novel Task and Algorithms. *Data Mining and Knowledge Discovery*, 20(2):290–324, 2010. (Citado en la página 41.)

[Aha91] D. Aha. Incremental Constructive Induction: An Instance-Based Approaches. En *Proc. 8th Int. Workshop on Machine Learning*, págs. 117–121, 1991. (Citado en las páginas 36 y 40.)

[Aha97] D. Aha. *Lazy Learning*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. (Citado en la página 36.)

[AHT94] N. Anderson, P. Hall, y D. Titterington. Two-Sample Test Statistics for Measuring Discrepancies between Two Multivariate Probability Density Functions Using Kernel-Based Density Estimates. *Journal of Multivariate Analysis*, 50(1):41–54, 1994. (Citado en la página 23.)

[AHWY03] C. Aggarwal, J. Han, J. Wang, y P. Yu. A Framework for Clustering Evolving Data Streams. En *Proc. 29th Int. Conf. on Very Large Data Bases*, volumen 29, págs. 81–92, 2003. (Citado en la página 50.)

[AIS93] R. Agrawal, T. Imielinski, y A. Swami. Database Mining: A Performance Perspective. *IEEE Trans. on Knowledge and Data Engineer*, 5(6):914–925, 1993. (Citado en las páginas 46 y 80.)

[AKA91] D. Aha, D. Kibler, y M. Albert. Instance-Based Learning Algorithm. *Machine Learning*, 6(1):37–66, 1991. (Citado en las páginas 36 y 40.)

[AM07] R. Adams y D. MacKay. Bayesian online changepoint detection. Reporte técnico arXiv:0710.3742v1 [stat.ML], University of Cambridge, 2007. (Citado en la página 24.)

[APK+00] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, y P. Stamatopoulos. Learning to Filter Spam Email: A Comparison of a Naïve Bayesian and a Memory-Based Approach. En *Proc. Workshop on Machine Learning and Textual Information Access*, págs. 1–13, 2000. (Citado en la página 28.)

[Bar92] P. Bartlett. Learning with a Slowly Changing Distribution. En *Proc. 5th Annual Workshop on Computational Learning Theory*, págs. 243–252, 1992. (Citado en la página 17.)

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, y J. Widom. Models and Issues in Data Stream Systems. En *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, págs. 1–16, 2002. (Citado en la página 25.)

[BBDK96] P. Bartlett, S. Ben-David, y S. Kulkarni. Learning Changing Concepts by Exploiting the Structure of Change. En *Workshop on Computational Learning Theory*, págs. 131–139, 1996. (Citado en las páginas 17 y 18.)

[BC92] A. Blum y P. Chalasani. Learning Switching Concepts. En *Proc. 5th Annual Workshop on Computational Learning Theory*, págs. 231–242, 1992. (Citado en la página 17.)

[BCG+10] P. Bonissone, J. Cadenas, M. Garrido, R. Díaz, y A. Mu noz. Toma de Decisiones en Ensambles Basados en Árbol Fuzzy. En *XV Congreso Español sobre Tecnologías y Lógica Fuzzy*, págs. 637–642, 2010. (Citado en la página 39.)

[BdF+06] M. Baena, J. del Campo, R. Fidalgo, A. Bifet, R. Gavaldà, y R. Morales. Early Drift Detection Method. En *4th Int. Workshop on Knowledge Discovery from Data Streams*, 2006. (Citado en las páginas 6, 15, 21, 22, 24, 25, 27, 42, 46, 47, 51, 58, 59, 69, 70, 112 y 114.)

[BDM02] B. Babcock, M. Datar, y R. Motwani. Sampling from a Moving Window over Streaming Data. En *Proc. 13th Annual ACM-SIAM Symposium on Discrete algorithms*, págs. 633–634, 2002. (Citado en la página 25.)

[BFOS84] L. Breiman, J. Friedman, R. Olshen, y C. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984. (Citado en las páginas 4, 7, 46 y 47.)

[BG07] A. Bifet y R. Gavaldà. Learning from time-changing data with adaptive windowing. En *Proc. 7th SIAM Int. Conf. on Data Mining*, 2007. (Citado en las páginas 6, 21, 22, 24, 25, 27, 30, 39, 58, 61, 69, 70 y 114.)

[BG09a] A. Bifet y R. Gavaldà. Adaptive Learning from Evolving Data Streams. En *Proc. 8th Int. Symposium on Intelligent Data Analysis*, págs. 249–260, 2009. (Citado en las páginas 50, 109, 115, 117 y 133.)

[BG09b] A. Bifet y R. Gavaldà. Adaptive parameter-free learning from evolving data streams. En N. Adams, C. Robardet, A. Siebes, y J. Boulicaut, editores, *8th International Symposium on Intelligent Data Analysis*, Berlin, Heidelberg, 2009. Springer-Verlag. (Citado en las páginas 22, 25 y 101.)

[BH96] P. Bartlett y D. Helmbold. Learning Changing Problems. Reporte técnico, Australian National University, 1996. (Citado en las páginas 17 y 39.)

[BH99] M. Black y R. J Hickey. Maintaining the Performance of a Learned Classifier Under Concept drift. *Intelligent Data Analysis*, 3(6):453–474, 1999. (Citado en la página 42.)

[BH07] J. Beringer y E. Hüllermeier. Efficient Instance-Based Learning on Data Streams. *Intelligent Data Analysis*, 11(6):627–650, 2007. (Citado en las páginas 36, 37 y 40.)

[BHGH96] H. Bum, S. Hoon, T. Gon, y K. Ho. Fast Learning Method for Back-Propagation Neural Network by Evolutionary Adaptation of Learning Rates. *Neurocomputing*, 11(1):101–106, 1996. (Citado en la página 35.)

[BHKP10] A. Bifet, G. Holmes, R. Kirkby, y B. Pfahringer. MOA: Massive Online Analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010. (Citado en las páginas 7, 23, 28, 29, 30, 49, 52, 70, 75, 81, 85, 90, 96, 98, 99, 101, 105, 106, 108, 110, 112, 117, 118, 121, 125 y 133.)

[BHPF10] A. Bifet, G. Holmes, B. Pfahringer, y E. Frank. Fast perceptron decision tree learning from evolving data streams. En *Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part II*, PAKDD'10, págs. 299–310, Berlin, Heidelberg, 2010. Springer-Verlag. (Citado en la página 31.)

[BHPG09] A. Bifet, G. Holmes, B. Pfahringer, y R. Gavaldà. New Ensemble Methods for Evolving Data Streams. En *Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 139–148, 2009. (Citado en las páginas 46, 47, 48, 59 y 80.)

[Bif09] A. Bifet. *Adaptive Learning and Mining for Data Streams and Frequent Patterns*. Tesis de doctorado, Universitat Politècnica de Catalunya, 2009. (Citado en las páginas 3 y 5.)

[Bif10] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volumen 207. IOS Press, 2010. (Citado en las páginas 39, 41, 42, 51, 59 y 117.)

[BJ90] G. Box y G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990. (Citado en la página 26.)

[BL96] R. Barve y P. Long. On the Complexity of Learning from Drifting Distributions. En *Proc. 9th Annual Conf. on Computational Learning Theory*, págs. 122–130, 1996. (Citado en la página 17.)

[Blu97] A. Blum. Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain. *Machine Learning*, 26(1):5–23, 1997. (Citado en las páginas 38 y 40.)

[BM02] H. Brighton y C. Mellish. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery*, 6:153–172, 2002. (Citado en la página 37.)

[BM08] S. Bach y M. Maloof. Paired Learners for Concept Drift. En *Proc. 8th IEEE Int. Conf. on Data Mining*, págs. 23–32, 2008. (Citado en las páginas 6, 75 y 113.)

[BM10] S. Bach y M. Maloof. A Bayesian Approach to Concept Drift. En *Advances in Neural Information Processing Systems 23*, págs. 127–135, 2010. (Citado en las páginas 6 y 24.)

[BN93] M. Basseville y I. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Englewood Cliffs, NJ, 1993. (Citado en las páginas 6, 9, 14, 21, 22, 23, 24, 26, 67, 114 y 133.)

[Bou11] A. Bouchachia. Fuzzy classification in dynamic environments. *Soft Computing*, 15(5):1009–1022, 2011. (Citado en la página 31.)

[BPRH13] A. Bifet, B. Pfahringer, J. Read, y G. Holmes. Efficient data stream classification via probabilistic adaptive windows. En *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, págs. 801–806, New York, NY, USA, 2013. ACM. (Citado en la página 24.)

[Brz10] Dariusz Brzezinski. Mining Data Streams with Concept Drift. Tesis de maestría, Poznan University of Technology, 2010. (Citado en las páginas 41, 43, 46, 50 y 51.)

[BRŽ+13] A. Bifet, J. Read, I. Žliobaitė, B. Pfahringer, y G. Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. En *Machine Learning and Knowledge Discovery in Databases*, volumen 8188 de *Lecture Notes in Computer Science*, págs. 465–479. Springer Berlin Heidelberg, 2013. (Citado en la página 53.)

[BS93] M. Biehl y H. Schwarze. Learning Drifting Concepts with Neural Networks. *Journal of Physics A: Mathematical and General*, 26(11):2651–2665, 1993. (Citado en las páginas 36 y 40.)

[BV13] A. Bouchachia y C. Vanaret. Gt2fc: An online growing interval type-2 self-learning fuzzy classifier. *Fuzzy Systems, IEEE Transactions on*, PP(99), 2013. (Citado en la página 31.)

[CAW98] S. Chawathe, S. Abiteboul, y J. Widom. Representing and Querying Changes in Semistructured Data. En *Proc. 14th Int. Conf. on Data Engineering*, págs. 4–13, 1998. (Citado en la página 23.)

[CCW08] T. Cheng, L. Chien, y Y. Wei. An Efficient and Sensitive Decision Tree Approach to Mining Concept-Drifting Data Streams. *Informatica*, 19(1):135–156, 2008. (Citado en las páginas 29, 30 y 40.)

[CCW09] T. Cheng, L. Chien, y Y. Wei. Mining decision rules on data streams in the presence of concept drifts. *Expert Systems with Applications*, 36:1164–1178, 2009. (Citado en la página 32.)

[CE02] A. Chen y E. Elsayed. Design and performance analysis of the exponentially weighted moving average mean estimate for processes subject to random step changes. *Technometrics*, 44(4), November 2002. (Citado en las páginas 26 y 67.)

[CGM+92] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, y D. Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *Neural Networks, IEEE Transactions on*, 3(5):698–713, 1992. (Citado en la página 34.)

[Che52] H. Chernoff. A Measure of Asymptotic Efficiency for Tests or a Hypothesis Based on the Sum of Obsevation. *Annals of Mathematical Statistics*, 23(4):493–507, 1952. (Citado en las páginas 7 y 89.)

[CKN08] J. Cheng, Y. Ke, y W. Ng. A survey on algorithms for mining frequent itemsets over data streams. *Knowledge and Information Systems*, 16(1):1–27, July 2008. (Citado en la página 24.)

[CNDH03] P. Cunningham, N. Nowlan, S. Delany, y M. Haahr. A Case-Based Approach to Spam Filtering that Can Track Concept Drift. En *Proc. Workshop on Long-Lived CBR Systems (in ICCBR-2003)*, 2003. (Citado en las páginas 6, 28 y 50.)

[Col99] R. Colomb. Representation of Propositional Expert Systems as Partial Functions. *Artificial Intelligence*, 109(1-2):187–209, 1999. (Citado en la página 31.)

[CP01] G. Cauwenberghs y T. Poggio. Incremental and decremental support vector machine learning. En *Adv. Neural Information Processing Systems*, volumen 13 de *NIPS'00*. Cambridge MA: MIT Press, 2001. (Citado en la página 33.)

[CS06] E. Cohen y M. Strauss. Maintaining time-decaying stream aggregates. *Journal of Algorithms*, 59(1):19–36, 2006. (Citado en las páginas 27 y 65.)

[Das90] B. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society, 1990. (Citado en la página 36.)

[Daw84] 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. (Citado en las páginas 43 y 117.)

[DB07] S. Delany y D. Bridge. Catching the Drift: Using Feature-Free Case-Based Reasoning for Spam Filtering. En *Proc. 7th Int. Conf. on Case Based Reasoning*, 2007. (Citado en las páginas 40 y 51.)

[DC04] S. Delany y P. Cunningham. An Analysis of Case-Based Editing in a Spam Filtering System. En *Proc. 7th European Conf. in Case-Based Reasoning*, págs. 128–141, 2004. (Citado en las páginas 13, 28, 37, 40, 50 y 51.)

[dCRM06] J. del Campo, G. Ramos, y R. Morales. Improving prediction accuracy of an incremental algorithm driven by error margins. En *Proc. 4th Int. Workshop on Knowledge Discovery from Data Streams*, págs. 57–66, 2006. (Citado en las páginas 94, 97, 109 y 110.)

[DCTC05] S. Delany, P. Cunningham, A. Tsymbal, y L. Coyle. A Case-based Technique for Tracking Concept Drift in Spam Filtering. *Knowledge-Based Systems*, 18(4-5):187–195, 2005. (Citado en las páginas 13, 37, 40 y 51.)

[Dec13] M. Deckert. Incremental rule-based learners for handling concept drift: an overview. *Foundations of Computing and Decision Sciences*, 38(1):35–65, 2013. (Citado en la página 31.)

[del07] J. del Campo. *Nuevos Enfoques en Aprendizaje Incremental*. Tesis de doctorado, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, 2007. (Citado en las páginas xi, 3, 6, 7, 8, 28, 29, 38, 40, 61, 62, 89, 92, 94, 95 y 101.)

[DGIM02] M. Datar, A. Gionis, P. Indyk, y R. Motwani. Maintaining Stream Statistics over Sliding Windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002. (Citado en las páginas 24, 25, 57 y 69.)

[DH00] P. Domingos y G. Hulten. Mining High-Speed Data Streams. En *Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 71–80, 2000. (Citado en las páginas 7, 28, 29, 32, 41, 50, 80, 89, 95, 109, 110 y 117.)

[DKP11] T. Dasu, S. Krishnan, y G. Pomann. Robustness of change detection algorithms. En *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis*, págs. 125–137, 2011. (Citado en la página 70.)

[DKVY06] T. Dasu, S. Krishnan, S. Venkatasubramanian, y K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. En *38th Symposium on the Interface of Statistics, Computing Science, and Applications*, 2006. (Citado en la página 23.)

[DR09] A. Dries y U. Rückert. Adaptive Concept Drift Detection. *Statistical Analysis and Data Mining*, 2(5–6):311–327, 2009. (Citado en las páginas 15, 22, 34, 40 y 41.)

[dRGM08] J. del Campo, G. Ramos, J. Gama, y R. Morales. Improving the performance of an incremental algorithm driven by error margins. *Intelligent Data Analysis*, 12(3):305–318, 2008. (Citado en las páginas 7, 28, 29, 61, 62, 89, 109 y 110.)

[DWV99] H. Drucker, D. Wu, y V. Vapnik. Support Vector Machines for Spam Categorization. *IEEE Trans. on Neural Networks*, 10(5):1048 –1054, 1999. (Citado en la página 28.)

[EP11] R. Elwell y R. Polikar. Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Trans. on Neural Networks*, 22(10):1517–1531, 2011. (Citado en las páginas 39 y 40.)

[FA10] A. Frank y A. Asuncion. UCI Machine Learning Repository, 2010. (Citado en las páginas 45, 50, 85 y 105.)

[FAR05] F. Ferrer, J. Aguilar, y J. Riquelme. Incremental Rule Learning and Border Examples Selection from Numerical Data Streams. *Journal of Universal Computer Science*, 11(8):1426–1439, 2005. (Citado en las páginas 14, 32, 40, 41, 42, 43, 45 y 48.)

[FdR+14] I. Frías, J. del Campo, G. Ramos, R. Morales, A. Ortiz, y Y. Caballero. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering. In press, DOI: 10.1109/TKDE.2014.2345382*, 2014. (Citado en las páginas 114 y 118.)

[FH89] K. Fukunaga y R. Hayes. Estimation of Classifier Performance. *IEEE Trans. on Pattern Analysis Machine Intelligence*, 11(10):1087–1101, 1989. (Citado en las páginas 6 y 25.)

[FHT00] J. Friedman, T. Hastie, y R. Tibshirani. Additive Logistic Regression: A Statistical View of Boosting. *Annals of Statistics*, 28(2):337–374, 2000. (Citado en la página 28.)

[FHWY04] W. Fan, Y. Huang, H. Wang, y P. Yu. Active Mining of Data Streams. En *Proc. 4th SIAM Int. Conf. on Data Mining*, págs. 457–461, 2004. (Citado en la página 14.)

[FHY04] W. Fan, Y. Huang, y P. Yu. Decision tree evolution using limited number of labeled data items from drifting data streams. En *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, págs. 379–382, Washington, DC, USA, 2004. IEEE Computer Society. (Citado en la página 50.)

[FI93] M. Fayyad y K. Irani. Multi-interval discretization of continuos-valued attributes for classification learning. En *EInternational Joint Conference on Artificial Intelligence*, págs. 1022–1027, 1993. (Citado en la página 97.)

[FID+07] F. Fernández, E. Iglesias, F. Díaz, J. Méndez, y J. Corchado. Applying Lazy Learning Algorithms to Tackle Concept Drift in Spam Filtering. *Expert Systems with Applications*, 33(1):36–48, 2007. (Citado en la página 28.)

[FM97] Y. Freund y Y. Mansour. Learning under Persistent Drift. En *Proc. 3rd European Conf. on Computational Learning Theory*, págs. 109–118, 1997. (Citado en las páginas 17 y 18.)

[For06] G. Forman. Tackling Concept Drift by Temporal Inductive Transfer. En *Proc. 29th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, págs. 252–259, 2006. (Citado en la página 24.)

[FS97] Y. Freund y R.E. Schapire. A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. (Citado en la página 39.)

[Gam10] J. Gama. *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC, 2010. (Citado en las páginas 42 y 44.)

[GBR+06] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, y A. Smola. A Kernel Method for the Two-Sample-Problem. En *Advances in Neural*

Information Processing Systems 19, págs. 513–520, 2006. (Citado en la página 23.)

[GC06] J. Gama y G. Castillo. Learning with Local Drift Detection. En *Proc. 2nd Int. Conf. on Advanced Data Mining and Applications*, págs. 42–55, 2006. (Citado en las páginas 42, 47, 48, 59 y 117.)

[GFH07] J. Gao, W. Fan, y J. Han. On Appropriate Assumptions to Mine Data Streams: Analysis and Practice. En *Proc. 7th IEEE Int. Conf. on Data Mining*, págs. 143–152, 2007. (Citado en la página 14.)

[GFHY07] J. Gao, W. Fan, J. Han, y P. Yu. A general framework for mining concept-drifting data streams with skewed distributions. En *SDM*. SIAM, 2007. (Citado en la página 52.)

[GFR06] J. Gama, R. Fernandes, y R. Rocha. Decision trees for mining data streams. *Intelligent Data Analysis*, 10(1):23–45, 2006. (Citado en las páginas 29 y 109.)

[GG07] J. Gama y M. Gaber. *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer-Verlag, 1 edición, 2007. (Citado en la página 109.)

[GGC08] G. Grinblat, P. Granitto, y A. Ceccatto. Guillermo l. grinblat, pablo m. granitto, alejandro ceccatto. *Revista Iberoamericana de Inteligencia Artificial*, 12:39–50, 2008. (Citado en las páginas 19 y 33.)

[GK01] M. Greenwald y S. Khanna. Space-efficient online computation of quantile summaries. *SIGMOD Rec.*, 30(2):58–66, 2001. (Citado en las páginas 97, 98, 105 y 108.)

[GK09] J. Gama y P. Kosina. Tracking Recurring Concepts with Meta-learners. En *Proc. 14th Portuguese Conf. on Artificial Intelligence*, págs. 423–434, 2009. (Citado en la página 46.)

[GK11] J. Gama y P. Kosina. Learning decision rules from data streams. En *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, AAAI '11, págs. 1255–1260, 2011. (Citado en las páginas 31 y 32.)

[GKS01] J. Gehrke, F. Korn, y D. Srivastava. On computing correlated aggregates over continual data streams. *SIGMOD Rec.*, 30(2):13–24, 2001. (Citado en la página 24.)

[GMCR04] J. Gama, P. Medas, G. Castillo, y P. Rodrigues. Learning with Drift Detection. En *Proc. 20th Brazilian Symposium on Artificial Intelligence*, págs. 286–295, 2004. (Citado en las páginas 6, 14, 15, 18, 21, 22, 24, 25, 27, 29, 32, 37, 41, 45, 46, 51, 58, 59, 67, 69, 70, 112, 113 y 114.)

[GMR04] J. Gama, P. Medas, y R. Rocha. Forest Trees for On-Line Data. En *Proc. 2004 ACM symposium on Applied computing*, págs. 632–636, 2004. (Citado en las páginas 6 y 40.)

[GMR05] J. Gama, P. Medas, y P. Rodrigues. Learning decision trees from dynamic data streams. En *Proceedings of the 2005 ACM symposium on Applied computing*, SAC '05, págs. 573–577, New York, NY, USA, 2005. ACM. (Citado en las páginas 29, 30 y 110.)

[GMS11] J. Gomes, E. Menasalvas, y P. Sousa. Learning recurring concepts from data streams with a context-aware ensemble. En *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, págs. 994–999, New York, NY, USA, 2011. ACM. (Citado en la página 15.)

[GR07] J. Gama y P. Rodríguez. *Learning from Data Streams: Processing Techniques in Sensor Networks*. Springer-Verlag, 1 edición, 2007. (Citado en las páginas 3, 13 y 24.)

[GRC09] J. Gama, P. Rodrigues, y G. Castillo. Evaluating Algorithms that Learn from Data Streams. En *Proc. 2009 ACM Symposium on Applied Computing*, págs. 1496–1500, 2009. (Citado en las páginas 43, 47 y 118.)

[GRM03] J. Gama, R. Rocha, y P. Medas. Accurate Decision Trees for Mining High-Speed Data Streams. En *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 523–528, 2003. (Citado en las páginas 90, 94 y 97.)

[GSaR13] J. Gama, R. Sebastião, y P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013. (Citado en las páginas 70, 85, 106, 113, 117, 125 y 133.)

[GSR09] J. Gama, R. Sebastião, y P. Rodrigues. Issues in Evaluation of Stream Learning Algorithms. En *Proc. 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 329–338, 2009. (Citado en las páginas 30, 43 y 118.)

[GT12] V. Grossi y F. Turini. Stream mining: a novel architecture for ensemble-based classification. *Knowledge and Information Systems*, 30(2):1–35, 2012. (Citado en las páginas 39 y 40.)

[GUCG11] G. Grinblat, L. Uzal, H. Ceccatto, y P. Granitto. Solving nonstationary classification problems with coupled support vector machines. *Neural Networks, IEEE Transactions on*, 22(1):37–51, 2011. (Citado en las páginas 19 y 33.)

[GŽB+] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, y A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, in press. (Citado en las páginas 3, 13, 19, 57, 111 y 125.)

[Har99] M. Harries. SPLICE-2 Comparative Evaluation: Electricity Pricing. Reporte técnico, The University of New South Wales, Sydney, Australia, 1999. (Citado en las páginas 41, 51 y 59.)

[HD03] Geoff Hulten y Pedro Domingos. VFML – A toolkit for mining high-speed time-changing data streams. Software toolkit, 2003. (Citado en las páginas 51, 97, 105 y 108.)

[HFH+09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, y I. Witten. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009. (Citado en la página 51.)

[HK06] J. Han y M. Kamber. *Data Mining Concepts and Techniques*. Morgan Kaufmann, San Mateo, 2006. (Citado en la página 24.)

[HL91] D. Helmbold y P. Long. Tracking Drifting Concepts Using Random Examples. En *4th Annual Workshop on Computational Learning Theory*, págs. 13–23, 1991. (Citado en las páginas 15, 17 y 18.)

[HL94] D. Helmbold y P. Long. Tracking Drifting Concept by Minimizing Disagreements. *Machine Learning*, 14(1):27–45, 1994. (Citado en las páginas 15, 17, 18 y 39.)

[Ho05] S. Ho. A Martingale Framework for Concept Change Detection in Time-Varying Data Streams. En *Proc. 22nd Int. Conf. on Machine Learning*, págs. 321–327, 2005. (Citado en las páginas 46 y 48.)

[Hoe63] W. Hoeffding. Probabilities inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58(301):13–30, 1963. (Citado en las páginas 7, 61, 62 y 89.)

[HRF04] J. Hernández, M. Ramírez, y C. Ferri. *Introducción a la minería de datos*. Prentice Hall, 2004. (Citado en las páginas 3 y 4.)

[HSD01] G. Hulten, L. Spencer, y P. Domingos. Mining Time-Changing Data Streams. En *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 97–106, 2001. (Citado en las páginas 6, 7, 13, 28, 29, 30, 40, 43, 48, 80, 109, 110, 111, 112, 115 y 117.)

[HSH98] M. Harries, C. Sammut, y K. Horn. Extracting Hidden Context. *Machine Learning*, 32(2):101–126, 1998. (Citado en las páginas 6 y 105.)

[IGD11] E. Ikonomovska, J. Gama, y S. Džeroski. Learning Model Trees from Evolving Data Streams. *Data Mining and Knowledge Discovery*, 23(1):128–168, 2011. (Citado en las páginas 6, 7, 29, 30, 51, 113 y 115.)

[IGZD11] E. Ikonomovska, J. Gama, B. Zenko, y S. Dzeroski. Speeding-up Hoeffding-based regression trees with options. En *28th International Conference on Machine Learning*, págs. 537–544, 2011. (Citado en las páginas 10, 110 y 116.)

[JA03] R. Jin y G. Agrawal. Efficient Decision Tree Construction on Streaming Data. En *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 571–576, 2003. (Citado en las páginas 41 y 42.)

[Joa97] T. Joachims. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. En *Proc. 14th Int. Conf. on Machine Learning*, págs. 143–151, 1997. (Citado en la página 28.)

[Joa98] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. En *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, págs. 137–142, London, UK, UK, 1998. Springer-Verlag. (Citado en las páginas 33 y 50.)

[Joa00] T. Joachims. Estimating the Generalization Performance of a SVM Efficiently. En *Proc. 17th Int. Conf. on Machine Learning*, págs. 431–438, 2000. (Citado en la página 34.)

[Joa06] T. Joachims. Training linear svms in linear time. En *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, págs. 217–226, New York, NY, USA, 2006. ACM. (Citado en la página 33.)

[KBG04] D. Kifer, S. Ben, y J. Gehrke. Detecting Change in Data Streams. En *Proc. 30th Int. Conf. on Very Large Data Bases*, volumen 30, págs. 180–191, 2004. (Citado en las páginas 21, 22, 23, 53 y 58.)

[KG12] P. Kosina y J. Gama. Handling time changing data with adaptive very fast decision rules. En *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ECML PKDD'12, págs. 827–842, Berlin, Heidelberg, 2012. Springer-Verlag. (Citado en las páginas 29, 31, 32 y 40.)

[KHA99] M. Kelly, D. Hand, y N. Adams. The impact of changing populations on classifier performance. En *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, págs. 367–371, New York, NY, USA, 1999. ACM. (Citado en la página 16.)

[Kir07] R. Kirkby. *Improving Hoeffding Trees*. Tesis de doctorado, University of Waikato, 2007. (Citado en las páginas 7, 95, 96, 97, 98, 99, 101, 103, 105, 108 y 118.)

[KJ00] R. Klinkenberg y T. Joachims. Detecting Concept Drift with Support Vector Machines. En *Proc. 17th Int. Conf. on Machine Learning*, págs. 487–494, 2000. (Citado en las páginas 6, 18, 34 y 40.)

[Kli04] R. Klinkenberg. Learning Drifting Concepts: Example Selection vs. Example Weighting. *Intelligent Data Analysis*, 8(3):281–300, 2004. (Citado en las páginas 6, 19, 26, 41, 50, 65 y 66.)

[KM05] J.Z. Kolter y M.A. Maloof. Using Additive Expert Ensembles to Cope with Concept Drift. En *Proc. 22nd Int. Conf. on Machine Learning*, págs. 449–456, 2005. (Citado en la página 39.)

[KM07] J. Kolter y M. Maloof. Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. *Journal of Machine Learning Research*, 8:2755–2790, 2007. (Citado en las páginas 28, 38, 40, 45, 47 y 51.)

[KPR90] A. Kuh, T. Petsche, y R. Rivest. Learning Time-Varying Concepts. En *Advances in Neural Information Processing Systems 3*, págs. 183–189, 1990. (Citado en las páginas 15 y 17.)

[KR98] R. Klinkenberg y I. Renz. Adaptive Information Filtering: Learning in the Presence of Concept Drifts. En *Proc. Workshop Learning for Text Categorization (in ICML/AAAI-98)*, págs. 33–40, 1998. (Citado en las páginas 24 y 27.)

[KS09] Y. Kawahara y M. Sugiyama. Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation. En *Proc. SIAM International Conference on Data Mining*, págs. 389–400, 2009. (Citado en las páginas 23 y 24.)

[KSW04] J. Kivinen, A. Smola, y R. Williamson. Online learning with kernels. *Signal Processing, IEEE Transactions on*, 52(8):2165–2176, 2004. (Citado en la página 33.)

[KTV08] I. Katakis, G. Tsoumakas, y I. Vlahavas. An Ensemble of Classifiers for coping with Recurring Contexts in Data Streams. En *Proc. 18th European Conf. on Artificial Intelligence*, págs. 763–764, 2008. (Citado en las páginas 26, 85, 105 y 125.)

[KTV10] I. Katakis, G. Tsoumakas, y I. Vlahavas. Tracking Recurring Contexts using Ensemble Classifiers: an Application to Email Filtering. *Knowledge and Information Systems*, 22(3):371–391, 2010. (Citado en las páginas 41 y 51.)

[Kun08] L. Kuncheva. Classifier Ensembles for Detecting Concept Change in Streaming Data: Overview and Perspectives. En *Proc. 2nd Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, págs. 5–9, 2008. (Citado en la página 19.)

[Kun13] L. Kuncheva. Change detection in streaming multivariate data using likelihood detectors. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1175–1180, 2013. (Citado en la página 23.)

[KW95] M. Kubat y G. Widmer. Adapting to Drift in Continuous Domains . Reporte técnico ÖFAI-TR-94-27, Austrian Research Institute for Artificial Intelligence, Vienna, 1995. (Citado en las páginas 6, 18, 34, 35, 40, 42, 44, 45, 46 y 48.)

[KYM07] Y. Kawahara, T. Yairi, y K. Machida. Change-Point Detection in Time-Series Data Based on Subspace Identification. En *Proc. 7th IEEE Int. Conf. on Data Mining*, págs. 559–564, 2007. (Citado en la página 23.)

[LCG10] D. Leite, P. Costa, y F. Gomide. Evolving Granular Neural Network for Semi-supervised Data Stream Classification. En *Proc. 2010 Int. Joint Conf. on Neural Networks*, págs. 1–8, 2010. (Citado en las páginas 34, 35 y 40.)

[LDM08] P. Lindstrom, S. Delany, y B. MacNamee. AUTOPILOT: Simulating Changing Concepts in Real Data. En *Proc. 19th Irish Conf. on Artificial Intelligence and Cognitive Science*, págs. 272–281, 2008. (Citado en la página 50.)

[Lit91] N. Littlestone. Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. En *Proc. 4th annual workshop on Computational learning theory*, págs. 147–156, 1991. (Citado en la página 26.)

[Lon99] P. Long. The Complexity of Learning According to Two Models of a Drifting Environment. *Machine Learning*, 37(3):337–354, 1999. (Citado en la página 17.)

[LSB+90] J. Lucas, M. Saccucci, R. Baxley, W. Woodall, H. Maragh, F. Faltin, G. Hahn, W. Tucker, J. Hunter, J. MacGregor, y T. Harris. Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics*, 32(1):1–29, 1990. (Citado en la página 26.)

[LV04] M. Lazarescu y S. Venkatesh. Using Multiple Windows To Track Concept Drift. *Intelligent Data Analysis*, 8(1):29–59, 2004. (Citado en la página 26.)

[LW94] N. Littlestone y M. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108:212–261, 1994. (Citado en la página 38.)

[Malo3] M. Maloof. Incremental Rule Learning with Partial Instance Memory for Changing Concepts. En *Proc. Int. Joint Conf. on Neural Networks*, volumen 4, págs. 2764 – 2769 vol.4, 2003. (Citado en la página 31.)

[McD89] C. McDiarmid. On the method of bounded differences, 1989. (Citado en las páginas 7 y 63.)

[MGK+11] M. Masud, J. Gao, L. Khan, J. Han, y B. Thuraisingham. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):859–874, 2011. (Citado en la página 52.)

[Mit97] T. Mitchell. *Machine Learning*. McGraw Hill, 1997. (Citado en la página 59.)

[MM04] M. Maloof y R. Michalski. Incremental Learning with Partial Instance Memory. *Artificial Intelligence*, 154:95–126, 2004. (Citado en las páginas 31, 32 y 40.)

[Mono1] D. Montgomery. *Introduction to Statistical Quality Control*. Wiley: New York, 2001. (Citado en las páginas 6, 9, 21, 23, 24, 26, 58, 65, 67, 69 y 70.)

[MRA+12] J. Moreno, T. Raeder, R. Alaiz, N. Chawla, y F. Herrera. A unifying view on dataset shift in classification. *Pattern Recognition*, 45(1):521–530, 2012. (Citado en las páginas 14 y 52.)

[Mur98] S. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, December 1998. (Citado en las páginas 6, 28, 89, 95, 101 y 109.)

[Mus98] D.R. Musicant. NDC: Normally Distributed Clustered Datasets, 1998. www.cs.wisc.edu/dmi/svm/ndc. (Citado en la página 48.)

[Muto5] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Found. Trends Theoretical Computer Science*, 1(2):117–236, 2005. (Citado en las páginas 25 y 49.)

[MvdBW07] S. Muthukrishnan, E. van den Berg, y Y. Wu. Sequential change detection on data streams. En *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, págs. 551–550, 2007. (Citado en la página 53.)

[MWK+06] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, y T. Euler. YALE: Rapid Prototyping for Complex Data Mining Tasks. En *Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 935–940, 2006. (Citado en la página 52.)

[MWY10] L.L. Minku, A.P. White, y X. Yao. The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift. *IEEE Trans. on Knowledge and Data Engineering*, 22:730–742, 2010. (Citado en las páginas 14, 42, 46 y 49.)

[NFMo5] M. Núñez, R. Fidalgo, y R. Morales. On-Line learning of decision trees in problems with unknown dynamics. En *Proc. 4th Mexican Int. Conf. on Advances in Artificial Intelligence*, págs. 443–453. Springer-Verlag, 2005. (Citado en las páginas 26 y 29.)

[NFM07] M. Núñez, R. Fidalgo, y R. Morales. Learning in Environments with Unknown Dynamics: Towards more Robust Concept Learners. *Journal of Machine Learning Research*, 8:2595–2628, 2007. (Citado en las páginas 7, 26, 29, 30, 40, 66, 110, 112, 113, 117, 118 y 133.)

[NK07] A. Narasimhamurthy y Ludmila I. Kuncheva. A Framework for Generating Data to Simulate Changing Environments. En *Proc. 25th IASTED Int. Multi-Conf.: artificial intelligence and applications (AIAP'07)*, págs. 384–389, 2007. (Citado en las páginas 47 y 49.)

[NLo4] J. Natwichai y X. Li. Knowledge Maintenance on Data Streams with Concept Drifting. En *Proc. 1st Int. Symposium on Computational and Information Science*, págs. 705–710, 2004. (Citado en las páginas 31 y 40.)

[ORo1] N.C. Oza y S. Russell. Online Bagging and Boosting. En *Proc. 8th Int. Workshop on Artificial Intelligence and Statistics*, págs. 105–112, 2001. (Citado en las páginas 39 y 40.)

[PHK07] B. Pfahringer, G. Holmes, y R. Kirkby. New options for Hoeffding trees. En *Proceedings of the 20th Australian Joint Conference on Advances in Artificial Intelligence*, págs. 90–99. Springer-Verlag, 2007. (Citado en las páginas 10, 110, 116 y 117.)

[PM07] R. Prescott y D. MacKay. Bayesian Online Changepoint Detection. Reporte técnico arXiv:0710.3742v1 [stat.ML], University of Cambridge, 2007. (Citado en la página 24.)

[PUUH01] R. Polikar, L. Udpa, S. Udpa, y V. Honavar. Learn++: An Incremental Learning Algorithm for Supervised Neural Networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 31(4):497 –508, 2001. (Citado en la página 34.)

[Qui86] J. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986. (Citado en las páginas 4, 7, 90, 95 y 96.)

[Qui93] J. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993. (Citado en las páginas 4, 5 y 7.)

[Ram01] G. Ramos. *Nuevos desarrollos en apredizaje inductivo* . Tesis de doctorado, Universidad de Málaga, 2001. (Citado en las páginas 7, 28, 29, 61, 62, 89 y 91.)

[RATH12] G. Ross, N. Adams, D. Tasoulis, y D. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191–198, 2012. (Citado en las páginas 27, 58, 71, 73, 86, 114 y 125.)

[Rau01] S. Raudys. *Statistical and Neural Classifiers: An Integrated Approach to Design*. Springer-Verlag, London, UK, 2001. (Citado en la página 26.)

[RB07] S. Ramamurthy y R. Bhatnagar. Tracking Recurrent Concept Drift in Streaming Data Using Ensemble Classifiers. En *Proc. 6th Int. Conf. on Machine Learning and Applications*, págs. 404–409, 2007. (Citado en las páginas 15 y 42.)

[RdCM05] G. Ramos, J. del Campo, y R. Morales. Induction of Decision Trees Using an Internal Control of Induction. En *Proc. 8th Int. Conf. on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems*, págs. 795–803, 2005. (Citado en las páginas 7, 28, 29, 38, 61, 62 y 89.)

[RdM06] G. Ramos, J. del Campo, y R. Morales. Incremental algorithm driven by error margins. *Lecture Notes in Artificial Intelligence*, 4265:358–362, 2006. (Citado en las páginas 94 y 109.)

[RE12] A. Rakitianskaia y A. Engelbrecht. Training feedforward neural networks with dynamic particle swarm optimisation. *Swarm Intelligence*, 6(3):233–270, 2012. (Citado en las páginas 34 y 35.)

[RGCL04] J. Rushing, S. Graves, E. Criswell, y A. Lin. A Coverage Based Ensemble Algorithm (CBEA) for Streaming Data. En *Proc. 16th IEEE Int. Conf. on Tools with Artificial Intelligence*, 2004. (Citado en la página 40.)

[RJPD14] L. Rutkowski, M. Jaworski, L. Pietruczuk, y P. Duda. Decision trees for mining data streams based on the gaussian approximation. *IEEE Transactions on Knowledge and Data Engineering*, 26(1):108–119, 2014. (Citado en las páginas 7, 29, 53, 89 y 95.)

[RMoo] G. Ramos y R. Morales. A new method for induction decision trees by sampling. En *Neurocolt Workshop on Applications of Learning Theory*, Barcelona, Spain, 2000. (Citado en las páginas 7, 28, 29, 61, 62, 89, 94, 95 y 109.)

[RMd04] G. Ramos, R. Morales, y J. del Campo. *Tendencias de la Minería de Datos en España*, capítulo IADEM-0: Un nuevo algoritmo incremental, págs. 91–98. Red Española de Minería de Datos, 2004. (Citado en las páginas 7, 28, 29, 61, 62 y 89.)

[RPDJ13] L. Rutkowski, L. Pietruczuk, P. Duda, y M. Jaworski. Decision trees for mining data streams based on the mcdiarmid's bound. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1272–1279, 2013. (Citado en las páginas 7, 29, 32, 53, 89, 95, 109 y 133.)

[RTA11] G. Ross, D. Tasoulis, y N. Adams. Nonparametric monitoring of data streams for changes in location and scale. *Technometrics*, 53(4):379–389, 2011. (Citado en las páginas 21, 24 y 57.)

[Rüp01] S. Rüping. Incremental Learning with Support Vector Machines. En *Proc. 1st IEEE Int. Conf. on Data Mining*, págs. 641–642, 2001. (Citado en las páginas 19, 33, 34 y 40.)

[Rut04] L. Rutkowski. Adaptive Probabilistic Neural Networks for Pattern Classification in Time-Varying Environment. *IEEE Trans. on Neural Networks*, 15(4):811–827, 2004. (Citado en las páginas 35 y 40.)

[Sal97] M. Salganicoff. Tolerating Concept and Sampling Shift in Lazy Learning Using Prediction Error Context Switching. *Artificial Intelligence Review*, 11(1-5):133–155, 1997. (Citado en las páginas 6, 14, 19, 37 y 40.)

[SF86] J. Schlimmer y D. Fisher. A Case Study of Incremental Concept Induction. En *Proc. 5th National Conf. on Artificial Intelligence*, págs. 495–501, 1986. (Citado en las páginas 6, 41, 42 y 45.)

[SG86] J. Schlimmer y R. Granger. Incremental Learning from Noisy Data. *Machine Learning*, 1(3):317–354, 1986. (Citado en las páginas 31, 32, 36, 40 y 80.)

[SK01] W. Street y Y. Kim. A Streaming Ensemble Elgorithm (SEA) for Large-Scale Classification. En *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 377–382, 2001. (Citado en las páginas 28, 38, 40, 42, 43 y 46.)

[SK07] Martin Scholz y Ralf Klinkenberg. Boosting Classifiers for Drifting Concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007. (Citado en las páginas 15, 39 y 40.)

[SLC+12] Y. Seldin, F. Laviolette, N. Cesa, J. Shawe, y P. Auer. PAC-Bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58(12):7086–7093, 2012. (Citado en la página 26.)

[SLH+99] N. Syed, H. Liu, S. Huan, L. Kah, y K. Sung. Handling Concept Drifts in Incremental Learning with Support Vector Machines. En *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 317–321, 1999. (Citado en las páginas 33, 42 y 44.)

[SM81] E. Smith y D. Medin. *Categories and concepts*. Harvard University Press, 1981. (Citado en la página 89.)

[SMLL07] Y. Sun, G. Mao, X. Liu, y C. Liu. Mining Concept Drifts from Data Streams Based on Multi-Classifiers. En *Proc. 21st Int. Conf. on Advanced Information Networking and Applications Workshops*, págs. 257–263, 2007. (Citado en las páginas 38 y 40.)

[Sta03] K. Stanley. Learning Concept Drift with a Committee of Decision Trees. Reporte técnico UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA, 2003. (Citado en las páginas 6 y 16.)

[SW07] D. Sculley y Gabriel M. Wachman. Relaxed online svms for spam filtering. En *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, págs. 415–422, New York, NY, USA, 2007. ACM. (Citado en las páginas 19, 33 y 50.)

[Taj02] N. Tajvidi. Permutation Tests for Equality of Distributions in High-Dimensional Settings. *Biometrika*, 89(16):354–374, 2002. (Citado en la página 23.)

[TPCP08] A. Tsymbal, M. Pechenizkiy, P. Cunningham, y S. Puuronen. Dynamic Integration of Classifiers for Handling Concept Drift. *Information Fusion*, 9(1):56–68, 2008. (Citado en la página 43.)

[Val84] L. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. (Citado en la página 4.)

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. (Citado en la página 33.)

[WFYH03] H. Wang, W. Fan, P. Yu, y J. Han. Mining Concept-Drifting Data Streams Using Ensemble Classifiers. En *Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 226–235, 2003. (Citado en las páginas 5, 26, 31, 38, 40, 41, 42, 48, 50 y 117.)

[WK93] G. Widmer y M. Kubat. Effective Learning in Dynamic Environments by Explicit Context Tracking. En *Proc. European Conf. on Machine Learning*, págs. 227–243, 1993. (Citado en las páginas 6, 13, 14, 15 y 42.)

[WK96] G. Widmer y M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23(1):69–101, 1996. (Citado en las páginas 6, 14, 18, 19, 31, 32, 40, 42, 43, 44, 46, 47 y 48.)

[WL07] Yi-Min Wen y Bao-Liang Lu. Incremental learning of support vector machines by classifier combining. En *Proceedings of the 11th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'07, págs. 904–911, Berlin, Heidelberg, 2007. Springer-Verlag. (Citado en la página 33.)

[WM97] D. Wilson y T. Martinez. Instance pruning techniques. En Morgan Kaufmann, editor, *Proceedings of the 14th International Conference on Machine Learning*, págs. 404–411, 1997. (Citado en la página 37.)

[Woz11] M. Wozniak. A hybrid decision tree training method using data streams. *Knowledge and Information Systems*, 29(2):335–347, November 2011. (Citado en la página 31.)

[Yao99] X. Yao. Evolving Artificial Neural Networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999. (Citado en las páginas 34, 35 y 40.)

[Ye03] N. Ye. *The handbook of data mining*. Lawrence Erlbaum Associates, 2003. (Citado en las páginas 3 y 4.)

[YT02] K. Yamanishi y J. Takeuchi. A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time-Series Data. En *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, págs. 676–681, 2002. (Citado en la página 23.)

[YWZ06] Y. Yang, X. Wu, y X. Zhu. Mining in Anticipation for Concept Change: Proactive-Reactive Prediction in Data Streams. *Mining and Knowledge Discovery*, 13(3):261–289, 2006. (Citado en la página 47.)

[ŽBG+12a] I. Žliobaitė, A. Bifet, M. Gaber, B. Gabrys, J. Gama, L. Minku, y K. Musial. Next challenges for adaptive learning systems. *ACM SIGKDD Newsletter*, 14(1), 2012. (Citado en la página 24.)

[ŽBG+12b] I. Žliobaitė, A. Bifet, M. Gaber, B. Gabrys, J. Gama, L. Minku, y K. Musial. Next challenges for adaptive learning systems. *SIGKDD Explor. Newsl.*, 14(1):48–55, 2012. (Citado en la página 52.)

[ŽBPH13] I. Žliobaitė, A. Bifet, B. Pfahringer, y G. Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems, in press*, 2013. (Citado en las páginas 52 y 58.)

[ŽG14] I. Žliobaitė y B. Gabrys. Adaptive preprocessing for streaming data. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):309–321, 2014. (Citado en la página 52.)

[Zha00] G. Zhang. Neural networks for classification: A survey. *IEEE Transactions on Systems, Man and Cybernetics—Part C*, 30(4):451–462, 2000. (Citado en la página 34.)

[Zha02] H. Zhang. The additive vs. multiplicative Chernoff bounds. A few other forms of Chernoff-Hoeffding bounds. Reporte técnico, Lecture Notes for Course CS 174: Combinatorics and Discrete Probability University of California, Berkeley, 2002. (Citado en la página 91.)

[ZJYH11] P. Zhao, R. Jin, T. Yang, y S. Hoi. Online auc maximization. En Lise Getoor y Tobias Scheffer, editores, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, págs. 233–240, New York, NY, USA, 2011. ACM. (Citado en la página 41.)

[ŽK09] I. Žliobaitė y L. Kuncheva. Determining the Training Window for Small Sample Size Classification with Concept Drift. En *Proc. ICDM Workshops (in IEEE Int. Conf. on Data Mining)*, págs. 447–452, 2009. (Citado en las páginas 25 y 27.)

[Žli08] I. Žliobaitė. Expected classification error of the euclidean linear classifier under sudden concept drift. En *Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 02*, FSKD '08, págs. 29–33, Washington, DC, USA, 2008. IEEE Computer Society. (Citado en la página 22.)

[Žli09a] I. Žliobaitė. Combining Time and Space Similarity for Small Size Learning under Concept Drift. En *Proc. 18th Int. Symposium on Foundations of Intelligent Systems*, págs. 412–421, 2009. (Citado en las páginas 39 y 50.)

[Žli09b] I. Žliobaitė. Learning under Concept Drift: an Overview. Reporte técnico arXiv:1010.4784v1 [cs.AI], Vilnius University, 2009. (Citado en las páginas 3, 13, 16, 49 y 125.)

[Žli10] I. Žliobaitė. Change with Delayed Labeling: When is it Detectable? En *Proc. 2010 IEEE Int. Conf. on Data Mining Workshops*, págs. 843–850, 2010. (Citado en la página 51.)

[ZZTG10] P. Zhang, X. Zhu, J. Tan, y L. Guo. SKIF: a Data Imputation Framework for Concept Drifting Data Streams. En *Proc. 19th ACM Int. Conf. on Information and Knowledge Management*, págs. 1869–1872, 2010. (Citado en la página 51.)