Aprendizaje escalable de clasificadores A1DE y A2DE sobre MapReduce

Jacinto Arias, José A. Gámez, and José M. Puerta

Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Albacete, España {jacinto.arias,jose.gamez,jose.puerta}@uclm.es

Resumen El paradigma de programación MapReduce ha supuesto un importante avance en el análisis de grandes volúmenes de datos, no obstante al no ser un modelo de programación general es difícil adaptar algunas técnicas de procesado a este nuevo contexto. En este trabajo nos centraremos en el problema de la clasificación automática en el que la escalabilidad es un punto clave para abordar problemas de alta complejidad. Para ello estudiaremos dos clasificadores de gran popularidad como son A1DE y A2DE identificando las propiedades que permiten su implementación bajo MapReduce y evaluando su rendimiento contra una serie de problemas de gran volumen y alta dimensionalidad sobre distintas configuraciones de un clúster de computadores.

Keywords: MapReduce, Hadoop, Big Data, Alta Dimensionalidad, Clasificadores basados en Redes Bayesianas, *ensembles*

1. Introducción

Durante los últimos años se ha podido observar un aumento progresivo de la capacidad de generación y almacenamiento de datos en formato digital [7]. La manipulación de estos grandes volúmenes de datos con el objetivo de extraer información valiosa de los mismos es conocido como 'Big Data'. Es necesario un nuevo enfoque para poder tratar con estos conjuntos de datos masivos, ya que las técnicas tradicionales basadas en paradigmas de computación secuencial no proporcionan la escalabilidad necesaria. Como respuesta a este problema han aparecido herramientas como el paradigma de computación paralela MapReduce (MR) [3], cuyo éxito radica en su capacidad de abstracción, independizando la algorítmica de la tecnología subyacente y del tamaño de los datos a procesar.

En el campo del Aprendizaje Automático y la Minería de Datos la escalabilidad ha sido siempre un campo de investigación de gran relevancia, aun así, el tamaño y la complejidad de los problemas que se presentan en Big Data no siempre permiten el uso de las técnicas tradicionales, y es por ello que también se requiere una adaptación de las mismas. Es importante destacar que MR no es un paradigma con un enfoque general, y por ello no todos los problemas poseen una aplicación directa sobre este modelo, siendo un caso extremo las aplicaciones con una naturaleza iterativa. En la literatura se pueden encontrar formalismos que encajan de manera natural bajo el paradigma MR, por lo que resulta relevante

estudiar su rendimiento en dominios de alta complejidad; donde los problemas no solo presentan un tamaño masivo en cuanto al número de ejemplos, sino también en cuanto a su alta dimensionalidad.

Este trabajo se centra en el problema de la clasificación supervisada, y en especial sobre una familia de modelos probabilísticos que han alcanzado una alta popularidad por su alto equilibrio entre calidad y eficiencia como son los clasificadores A1DE [10] y A2DE [11]. La contribución de este trabajo consiste en identificar la idoneidad de este formalismo para su implementación bajo MR; diseñando y evaluando un algoritmo de entrenamiento capaz de abordar simultáneamente problemas de gran tamaño y alta dimensionalidad.

La estructura del documento es la siguiente: En la Sección 2 se introducen los modelos estudiados y el paradigma MapReduce. En la Sección 3 se describe la implementación MR propuesta. En la Sección 4 se discuten los resultados experimentales obtenidos sobre grandes conjuntos de datos y diversas configuraciones de recursos. Por último, en la Sección 5, se exponen las conclusiones.

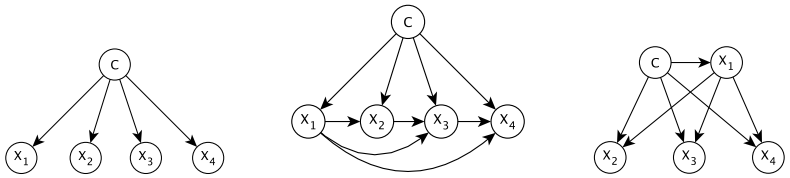
2. Preliminares

2.1. Clasificadores Bayesianos

El problema de la clasificación supervisada consiste en asignar una clase $c \in \Omega_C$, de un conjunto finito de etiquetas previamente definidas a un objeto $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ caracterizado por n valores de atributos $\mathbf{A} = \{X_1, X_2, \ldots, X_n\}$. Asumiremos que estos atributos son discretos. De entre los formalismos más utilizados destacan los basados en Redes Bayesianas, 'Bayesian Network Classifiers' (BNCs) [5] [2]. Una red Bayesiana se define mediante un par $(\mathbf{G}, \mathbf{\Theta})$, siendo $\mathbf{G}$ un grafo dirigido acíclico compuesto por un conjunto de nodos que representan las variables en $\mathbf{A} \cup \{C\}$ y un conjunto de enlaces que representan relaciones de dependencia entre las variables. Por otra parte $\mathbf{\Theta}$ es la representación paramétrica del modelo, un conjunto de tablas de probabilidad condicional (CPTs) $P(X_i \mid \mathbf{pa}(X_i))$, donde $\mathbf{pa}(X_i)$ es el conjunto de padres de X_i codificados en $\mathbf{G}$.

El caso más simple de BNC es el modelo de Bayes Ingenuo (NB), donde se asume que todos los atributos son condicionalmente independientes conocido el valor de la variable clase C, lo que implica una estructura fija del modelo (Figura 1a) y reduce el aprendizaje a la estimación de los parámetros. Los clasificadores denominados 'Bayes Ingenuo Aumentados' relajan la suposición de independencia incluyendo nuevas relaciones en la estructura (Figura 1b) para aumentar la precisión de sus respuestas pero sin complicar en exceso el aprendizaje.

Clasificadores A1DE y A2DE El clasificador A1DE (Averaged One Dependence Estimators) combina una gran capacidad predictiva con un proceso de aprendizaje eficiente. Se define como un conjunto de modelos independientes de los cuales se obtiene el promedio de la clasificación individual, esta técnica es conocida como clasificadores 'ensemble' [4] y aprovecha la diversidad existente



(a) Bayes Ingenuo (NB) (b) NB Aumentado (c) SPODE

Figura 1: Estructura de clasificadores basados en el modelo Bayes Ingenuo.

entre los clasificadores individuales para corregir posibles errores mediante la agregación, redundando en una predicción de mayor calidad.

El ensemble está formado por clasificadores del tipo de NB Aumentado. Concretamente, está formado por todos los individuos de una familia de modelos que presentan la siguiente restricción estructural: Cada atributo solo puede tener un padre además de la variable clase, estos modelos se conocen como clasificadores 1-dependientes (ODE). Además, para restringir el número de posibles modelos sólo se incluyen aquellos en los que el atributo padre es un atributo X_p común al resto de atributos del modelo, denominado super-padre (SP). Estos modelos se conocen como SPODEs (Figura 1c). Un clasificador A1DE aprende todos los $|\mathbf{A}|$ SPODEs posibles utilizando cada atributo $X_i \in \mathbf{A}$ como SP cada vez.

La ausencia de aprendizaje estructural permite a A1DE obtener unos tiempos de entrenamiento muy competitivos, siendo necesaria una única pasada sobre los datos para estimar los parámetros. Sin embargo, esto implica un incremento drástico de la complejidad espacial del algoritmo que crece de manera lineal con $|\mathbf{A}| = n$. La complejidad computacional de A1DE es $O(mn^2)$, donde m es el número de ejemplos en el conjunto de entrenamiento; mientras que la complejidad espacial sigue $O(|\Omega_C|n^2v^{n-1})$ con v la cardinalidad media de los atributos.

La popularidad del clasificador A1DE llevó a los autores a extender su trabajo [11] generalizando el modelo al uso de clasificadores k-dependientes, AkDE, en los que el nodo SP esta formado por combinaciones de k atributos. Como instancia particular, el clasificador A2DE ha demostrado tener una potencia predictiva muy superior a la de A1DE a expensas de requerir un mayor esfuerzo computacional. En el caso del clasificador A2DE, el número de modelos que componen el ensemble crece cuadráticamente respecto al número de atributos, siendo su complejidad espacial $O(|\Omega_C|n^3v^{n-1})$. Al contrario que A1DE, esta extensión no posee las mismas propiedades de escalabilidad y solo puede utilizarse en dominios de tamaño moderado.

2.2. Paradigma de Programación MapReduce

El paradigma de programación MapReduce fue presentado en 2004 [3] con el fin de proporcionar una plataforma de computación paralela orientada al análisis de grandes volúmenes de datos que hiciera uso de clústers de computadores de

hardware convencional. El principal potencial de MR viene dado por su fuerte capacidad de abstracción, que permite dividir el procesamiento en funciones individuales *Map* y *Reduce* que serán distribuidas y procesadas de manera individual por los distintos nodos del clúster. De este modo, el programador únicamente debe proporcionar el diseño de estas dos funciones para abordar su problema, evitando la necesidad de adaptar el procesamiento a la arquitectura subyacente o al tamaño de los datos de entrada. El entorno MR proporciona así una gran capacidad de escalabilidad y tolerancia a fallos.

Un procedimiento MR se estructura de la siguiente forma: Inicialmente se dividen los datos de entrada en una serie de particiones que son distribuidas entre los distintos nodos de cómputo; seguidamente se aplica la función Map a cada partición individual generando una salida intermedia que será utilizada como entrada de las funciones Reduce. Durante todo el procedimiento MR los datos se estructuran como pares $\langle$*clave, valor*$\rangle$, de modo que todos los pares que conforman los datos de salida intermedia obtenida de la fase Map se ordenan y agrupan en función de su clave, creando nuevos pares para cada clave cuyo valor es la colección de todos los valores individuales que compartían dicha clave. Cada uno de estos nuevos pares es procesado por una función Reduce, que agrega el resultado. La salida final viene dada por todas las salidas de las tareas Reduce.

3. Aprendizaje y Clasificación Escalable Sobre MapReduce

La mayor parte de la carga computacional en el aprendizaje del clasificador A1DE viene dada por el cómputo de las CPTs de cada atributo dada la clase y el super-padre de cada modelo; el mayor impacto en el rendimiento viene dado a la hora de obtener las frecuencias conjuntas de cada pareja de atributos junto a la clase, necesarias para estimar las CPTs, a partir del conjunto de datos de entrenamiento. La complejidad de este proceso está dirigida por el número de ejemplos en la base de datos m de manera lineal y por el número de atributos n de manera cuadrática, por lo que existe un equilibrio en el coste final del algoritmo conforme estos dos valores aumentan, pudiéndonos encontrar problemas con un gran número de ejemplos (Big Data), o con una gran cantidad de atributos (alta dimensionalidad), en casos extremos, es posible que ambos problemas confluyan.

El cómputo de frecuencias es un problema que encaja cómodamente dentro del paradigma MR a la hora de diseñar un procedimiento escalable. Concretamente, es posible abordar ambas dimensiones partiendo de principios similares [8]:

- Denominaremos **paralelismo horizonal** al procedimiento que reduce la complejidad respecto a m. Esto es posible debido a que la frecuencia conjunta de ocurrencia de un conjunto de variables $\mathbf{S_l}$ para un conjunto de datos $\mathbf{D}$ es agregable si se computa de manera parcial para distintos subconjuntos D_k del conjunto de datos original.

- Denominaremos **paralelismo vertical** al procedimiento que reduce la complejidad respecto a n. Se puede observar que el cómputo de las frecuencias conjuntas de cada combinación de atributos es independiente uno de otro y puede ser realizado de manera independiente, no obstante la complejidad de este caso viene dada a la hora de distribuir los datos y sincronizar los nodos de cómputo para obtener el resultado final.

3.1. Fase de Entrenamiento

Partiendo de los casos anteriores es posible definir el aprendizaje del clasificador A1DE como un procedimiento MR. Durante la fase Map se calcularán las distribuciones de frecuencias conjuntas para cada par de atributos y la clase, $\#(X_i, X_j, C)/X_i, X_j \in \mathbf{A}$, de manera que cada tarea actuará en paralelo sobre una partición D_k de los datos de entrenamiento D y un subconjunto $\mathbf{S_l}$ del total de combinaciones de atributos $\mathbf{S} = \{(X_i, X_j), /X_i, X_j \in \mathbf{A}, j > i\}$. Las tareas deberán organizarse de manera jerárquica con el fin de que para cada partición de datos se calculen todas las parejas, es por ello por cada partición de los datos será necesaria una tarea Map para cada subconjunto de combinaciones. El número de tareas Map total será entonces igual al producto del número de particiones de datos y del número de subconjuntos de combinaciones de atributos a los que denominaremos M_h y M_v respectivamente.

El procedimiento para el cómputo de la frecuencia conjunta se describe en el Algoritmo 1. La tarea Map, que recibe como entradas una partición de los datos D_k y un subconjunto de parejas de variables $\mathbf{S_l}$, se ha diseñado en base a un esquema de implementación que optimiza la cantidad de información de salida intermedia que se genera. El proceso está dividido en tres pasos: (1) *Setup* se realiza una única vez al comienzo para inicializar las estructuras de datos que almacenarán las distribuciones de frecuencias parciales; (2) *Map* concentra toda la carga computacional, procesando cada ejemplo contenido en la partición D_k mediante la función Map, actualizando las frecuencias para cada ocurrencia particular de las parejas de variables en $\mathbf{S_l}$ y la clase; (3) *CleanUp* consistente en emitir como salida de la tarea Map un par $\langle N_{ij}, \#_k(X_i, X_j, C)\rangle$ para cada elemento de $\mathbf{S_l}$, donde la clave identifica una pareja (X_i, X_j) concreta y el valor es la distribución de frecuencias parciales dada la clase computada para D_k.

Seguidamente, la salida de las tareas Map se ordena agrupando los pares por su clave, esto es por parejas de variables, y se utiliza como entrada de la fase Reduce. Cada tarea procesará un par $\langle N_{ij}, \mathbf{V}\rangle$, donde la clave identifica a un par de variables (X_i, X_j) y $\mathbf{V}$ es una colección que contiene la distribución de frecuencias parciales $\#_k(X_i, X_j, C)$ para todas las particiones del conjunto de entrenamiento. En esta fase del algoritmo se agregan las frecuencias de todas las particiones para seguidamente proceder a su normalización con el fin de obtener las CPTs deseadas. A partir de una distribución de frecuencias $\#k(X_i, X_j, C)$ es posible obtener dos CPTs que son necesarias para el clasificador A1DE: $P(X_i \mid X_j, C)$ y $P(X_j \mid X_i, C)$. Por último es necesario formatear la salida del procedimiento MR para obtener el modelo A1DE final, para ello emitiremos un par para cada CPT identificándola con el modelo SPODE al que corresponde usando el

Algoritmo 1: Fase *Map* del aprendizaje del algoritmo A1DE sobre MR.

Input: D_k: Una partición del conjunto de entrenamiento.
Input: $\mathbf{S_l}$: Un subconjunto de parejas (X_i, X_j) del conjunto S.

Setup *()*
  //Inicia la estructura de datos para el cómputo de frecuencias:
1 **for** ***each*** $(X_i, X_j) \in \mathbf{S_l}$ **do**
2 $\#_k(X_i, X_j, C) \leftarrow integer[|\Omega_{X_i}|][|\Omega_{X_j}|][|\Omega_C|]$

Map *($\langle key, d\rangle$) //Donde key es arbitrario y $d = (x_1, x_2, \ldots x_n, c) \in D_k$*
  //Para cada combinación se calcula de distribución de frecuencias conjunta:
1 **for** ***each*** $(X_i, X_j) \in \mathbf{S_l}$ **do**
2 $(x_i, x_j, c) \leftarrow d \downarrow \{x_i, x_j, c\}$
3 $\#_k(X_i, X_j, C)[x_i][x_j][c] + +$

CleanUp *()*
  //Se emite un par ⟨key, value⟩ para cada combinación:
1 **for** ***each*** $(X_i, X_j) \in \mathbf{S_l}$ **do**
2 **Output:** $\langle N_{ij}, \quad \#_k(X_i, X_j, C)\rangle$

índice del super-padre como clave; por ejemplo, para $\Theta_{i|j,C} = P(X_j \mid X_i, C)$ emitiremos el par $\langle X_j, \Theta_{i|j,C}\rangle$. Adicionalmente será necesario calcular y emitir las probabilidades marginales de la variable clase así como las probabilidades condicionales de cada super-padre con respecto a la clase; nótese que el número de estos parámetros es mucho menor al anterior y que éstos pueden ser obtenidos fácilmente marginalizando las distribuciones de probabilidad anteriores.

3.2. Extensión para A2DE

El procedimiento anterior puede extenderse para el caso del clasificador A2DE atendiendo a varios puntos que requieren modificación. En primer lugar en el clasificador A2DE es necesario computar las frecuencias conjuntas para todas las posibles combinaciones de tres variables junto a la clase, en este punto el subconjunto S que se utiliza durante la fase Map debe definirse como $\mathbf{S} = (X_i, X_j, X_k)\ X_i, X_j, X_k \in \mathbf{A}, k > j > i$. La estructura de la fase Map se mantiene modificando únicamente los bucles y cálculos para incluir el tercer atributo X_k en las distribuciones de frecuencias parciales.

Para la fase Reduce la diferencia reside en el número de distribuciones de probabilidad que se pueden obtener a partir de una misma distribución de frecuencias, por ejemplo, en el caso de $\#(X_i, X_j, X_k, C)$ es posible obtener $\Theta_{i|j,k,C} = P(X_i \mid X_j, X_k, C)$, $\Theta_{j|i,k,C} = P(X_j \mid X_i, X_k, C)$ y $\Theta_{k|i,j,C} = P(X_k \mid X_i, X_j, C)$. Una vez agregadas las distribuciones de frecuencias se obtienen las CPTs y se emiten como en A1DE, indicándose como clave del par el identificador del SPODE que está formado por dos variables; siguiendo con el ejemplo emitiríamos: $\langle SP_{kj}, \Theta_{i|j,k,C}\rangle$, $\langle SP_{ik}, \Theta_{j|i,k,C}\rangle$ y $\langle SP_{ij}, \Theta_{k|i,j,C}\rangle$. Un esquema gráfico del proceso MR para el aprendizaje del clasificador A2DE se muestra en la figura 2.

Algoritmo 2: Fase *Reduce* del aprendizaje del algoritmo A1DE sobre MR.

Input: $\langle N_{ij}, \mathbf{V} \rangle$, $\mathbf{V}$: lista de distribuciones de frecuencias parciales para el par (X_i, X_j)

1 $\mathbf{V} = [\#_1(X_i, X_j, C), \ldots, \#_{\frac{m}{t}}(X_i, X_j, C)]$.
//Inicializa las estructuras de datos para la suma:
2 $T \leftarrow integer[|\Omega_{X_i}|][|\Omega_{X_j}|][|\Omega_C|]$
//Sumar sobre las distribuciones de frecuencias parciales:
3 **for** ***each*** $\mathbf{v} \in \mathbf{V}$ **do**
  //Suma sobre todos los estados de la distribución:
4 $T \leftarrow T + v$
//Normaliza las frecuencias sobre el atributo hijo para obtener las distribuciones:
5 $P(X_i \mid X_j, c) \leftarrow Normalize(T, X_i)$
6 $P(X_j \mid X_i, c) \leftarrow Normalize(T, X_j)$
7 **Emite:** $\langle SP_j, P(X_i \mid X_j, c) \rangle$
8 **Emite:** $\langle SP_i, P(X_j \mid X_i, c) \rangle$

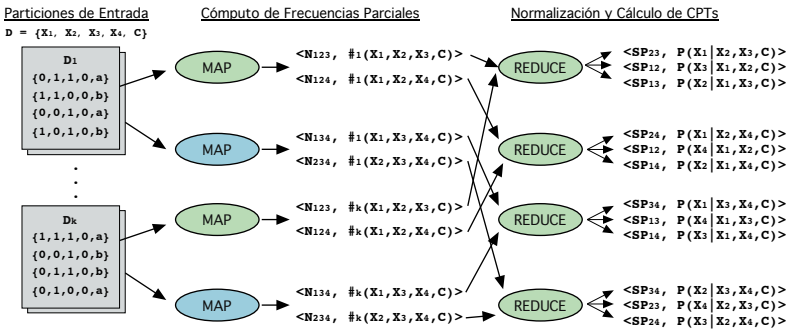


Figura 2: Representación gráfica del proceso de aprendizaje de A2DE sobre MR.

4. Evaluación Experimental

Se han realizado una serie de experimentos sobre diversas configuraciones en un clúster de computadores. El entorno de ejecución MapReduce elegido es la distribución Hadoop, gestionado mediante la plataforma Cloudera Manager 5.4. El clúster se compone de un nodo maestro y seis nodos de trabajo, todos ellos dotados de dos procesadores Intel Xeon E5-2609v3 1.90GHz hexacore, 64GB de memoria principal y 4TB de disco duro. El banco de pruebas seleccionado está formado por conjuntos de datos que combinan en distinto grado problemas de escalabilidad con respecto a número de ejemplos y dimensionalidad. Los datos han sido obtenidos de repositorios públicos y preprocesados previamente a la experimentación para su uso sobre los algoritmos implementados; concretamente, se han discretizado todas las variables continuas por igual anchura como variables categóricas de tres estados. Dado que los modelos A1DE y A2DE evaluados son idénticos a los originales no se ha contemplado el estudio de los resultados

Tabla 1: Propiedades de los conjuntos de datos utilizados en la evaluación de los clasificadores A1DE y A2DE. ECBLD14* ha sido muestreada aleatoria y estratificadamente del original manteniendo una proporción 3:1 de la distribucion de la clase para evitar el desbalanceo.

Nombre	**#Atrib.**	**#Ej.**	**Tamaño**	**Ref.**	**Nombre**	**#Atrib.**	**#Ej.**	**Tamaño**	**Ref.**
KDDCup99	40	4.8M	387M	[6]	ECBLD14*	630	4.3M	5 GB	[1]
Splice	141	50M	14 GB	[9]	Epsilon	2000	500k	1.9 GB	[12]

Tabla 2: Resultados para A1DE sobre los bancos de datos de prueba y las configuraciones del cluster. **T** representa el tiempo en segundos, **S** el SpeedUp contrastado con el algoritmo secuencial y **E** la eficiencia, como la proporción de SpeedUp con respecto al número de tareas en paralelo.

			M_h4				M_h8				M_h16			M_h32		M_h64
			M_v1	M_v2	M_v4	M_v8	M_v1	M_v2	M_v4	M_v8	M_v1	M_v2	M_v4	M_v1	M_v2	M_v1
Dataset	**Secuencial**		R_4	R_8	R_{16}	R_{32}	R_8	R_{16}	R_{32}	R_{64}	R_{16}	R_{32}	R_{64}	R_{32}	R_{64}	R_{64}
KDDCup99	**T**	70	31	28	20	22	19	18	18	34	16	18	31	18	30	29
	S	-	2.26	2.50	3.50	3.18	3.68	3.89	3.89	2.06	4.38	3.89	2.26	3.89	2.33	2.41
	E	-	0.56	0.31	0.22	0.10	0.46	0.24	0.12	0.03	0.27	0.12	0.04	0.12	0.04	0.04
Splice	**T**	4930	1357	1023	690	555	688	522	494	491	355	383	359	356	325	290
	S	-	3.63	4.82	7.14	8.88	7.17	9.44	9.98	10.04	13.89	12.87	13.73	13.85	15.17	17.00
	E	-	0.91	0.60	0.45	0.28	0.90	0.59	0.31	0.16	0.87	0.40	0.21	0.43	0.24	0.27
ECBLD14*	**T**	13873	5919	1308	771	859	2657	1819	896	638	2057	1656	905	1971	1548	1703
	S	-	2.34	10.61	17.99	16.15	5.22	7.63	15.48	21.74	6.74	8.38	15.33	7.04	8.96	8.15
	E	-	0.59	1.33	1.12	0.50	0.65	0.48	0.48	0.34	0.42	0.26	0.24	0.22	0.14	0.13
Epsilon	**T**	16307	5942	3422	2297	1950	3197	2263	2133	2061	2246	1971	1924	1961	1944	1900
	S	-	2.74	4.77	7.10	8.36	5.10	7.21	7.65	7.91	7.26	8.27	8.48	8.32	8.39	8.58
	E	-	0.69	0.60	0.44	0.26	0.64	0.45	0.24	0.12	0.45	0.26	0.13	0.26	0.13	0.13

de clasificación, únicamente se estudia la escalabilidad. Los conjuntos de datos utilizados se describen en la Tabla 1.

Con el objeto de evaluar el impacto del aprendizaje sobre MR se ha estudiado el rendimiento del algoritmo variando los recursos disponibles en el clúster para obtener diversos grados de paralelismo. Se han configurado ejecuciones repartiendo el número total de tareas Map en distinta proporción en lo que respecta a paralelismo horizontal M_h y vertical M_v, el número de tareas Reduce R_i será siempre el producto de estos dos valores. A modo de referencia se ha implementado una versión secuencial optimizada de los algoritmos que requiere una única pasada sobre los datos. Los tiempos medidos incluyen el aprendizaje del modelo y su serialización en disco. Estos experimentos se han ejecutado en un único nodo del clúster, utilizando un procesador y toda la memoria principal disponible.

Como podemos observar, la implementación MR del algoritmo A1DE escala casi de manera lineal con respecto a la ejecución secuencial hasta un número moderado de tareas, salvo en el caso de la base de datos KDDCup99 cuyo menor tamaño la hace irrelevante para este estudio. Nótese que el rendimiento se reduce conforme se aumentan los recursos disponibles, esto es debido a que según el tamaño del problema y una vez que se alcanza un grado de paralelis-

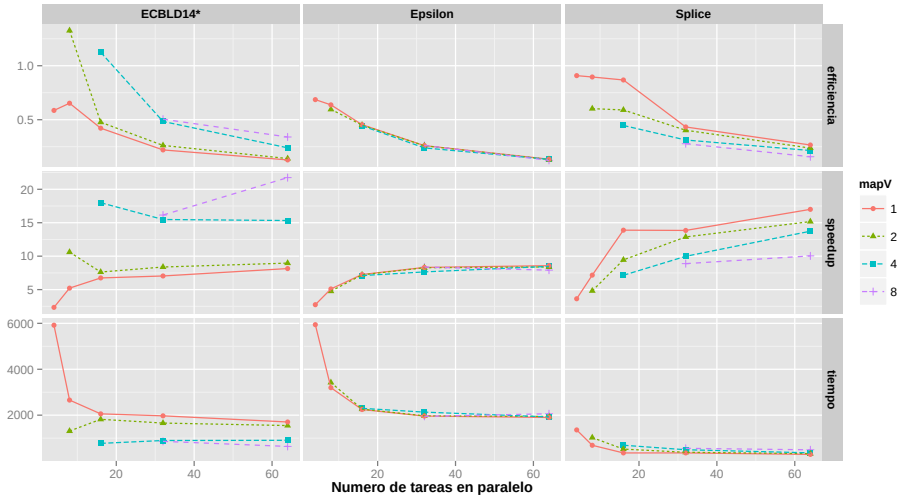


Figura 3: Resultados de tiempo de ejecución, speed-up y eficiencia para el clasificador A1DE y las distintas bases de datos. Las distintas líneas muestran los niveles de paralelismo vertical.

mo óptimo, aumentar el número de recursos supone una penalización por un exceso de tráfico en la sincronización. Otro resultado relevante que se puede extraer de los experimentos es el efecto resultante de aplicar el mismo número de tareas en distinta proporción vertical/horizontal. Únicamente se observa un efecto favorable en la verticalización para el dataset *ECBLD14**. Si se analiza la complejidad del algoritmo se puede observar que la reducción de aplicar ambas estrategias es lineal: Sea $O(n^2m)$ la complejidad del algoritmo y M_h y M_v el número de tareas paralelas horizontales y verticales respectivamente, la fórmula quedaría $O(\frac{n^2}{M_v}\frac{m}{M_h})$, donde en el caso vertical la reducción es lineal y equivalente al caso horizontal, ya que la división de tareas se realiza sobre combinaciones de atributos. Este comportamiento puede comprobarse en la Figura 3 donde se representan los datos de la tabla 2.

En el aprendizaje del clasificador A2DE se imponen ciertas restricciones a nivel de recursos que denotan el alto incremento de complejidad con respecto a A1DE. Dado que el tamaño del ensemble aumenta de manera cuadrática con respecto al número de atributos, es posible sobrepasar la memoria disponible. Por otra parte, el elevado tamaño de los conjuntos de datos seleccionados supone un aumento del tiempo de ejecución que en algunas ocasiones se puede considerar excesivo. Para los experimentos sobre A2DE se ha fijado un tiempo máximo de ejecución de 24 horas que es insuficiente en la mayoría de los casos. Por ello se ha decidido mostrar una aproximación del rendimiento del algoritmo para una muestra aleatoria reducida de un 2 % del número total de ejemplos de los conjuntos de datos.

Tabla 3: Resultados detallados para el aprendizaje del clasificador A2DE sobre los bancos de datos de prueba reducidos (al 2 %) y distintas configuraciones del clúster.

			M_h4				M_h8				M_h16			M_h32		M_h64
			M_v1	M_v2	M_v4	M_v8	M_v1	M_v2	M_v4	M_v8	M_v1	M_v2	M_v4	M_v1	M_v2	M_v1
Dataset	**Secuencial**		R_4	R_8	R_{16}	R_{32}	R_8	R_{16}	R_{32}	R_{64}	R_{16}	R_{32}	R_{64}	R_{32}	R_{64}	R_{64}
Splice	**T**	41410	14786	5724	3765	3567	6288	4387	3034	2974	4572	3954	3203	4004	3098	2598
	S	-	2.80	7.23	11.00	11.61	6.59	9.44	13.65	13.92	9.06	10.47	12.93	10.34	13.37	15.94
	E	-	0.70	0.90	0.69	0.36	0.82	0.59	0.43	0.22	0.57	0.33	0.20	0.32	0.21	0.25
ECBLD14*	**T**	-	-	-	-	37517	-	-	-	35869	-	-	-	-	-	-
KDDCup99	**T**	1237	185	104	77	68	112	65	51	66	87	50	60	63	60	64
	S	-	6.69	11.89	16.06	18.19	11.04	19.03	24.25	18.74	14.22	24.74	20.62	19.63	20.62	19.33
	E	-	1.67	1.49	1.00	0.57	1.38	1.19	0.76	0.29	0.89	0.77	0.32	0.61	0.32	0.30

Tabla 4: Espacio en disco de los clasificadores entrenados. Para el dataset *epsilon* el tamaño se ha estimado en base a la fórmula de la complejidad.

	Splice	ECBLD14*	Epsilon	KddCup99
A1DE	7,7M	75.2M	634.8M	2.7M
A2DE	919.4M	70.7G	*>700G*	466.4M

Los resultados obtenidos se muestran en la Tabla 3, donde se puede observar que aún con las restricciones impuestas el rendimiento del clasificador es muy inferior al de A1DE siendo en algunas ocasiones inviable. Cabe destacar que en el caso de los dos conjuntos de mayor complejidad existen limitaciones importantes en cuanto a los requerimientos de memoria. En el caso de *ECBLD14** además de excederse el tiempo máximo de ejecución en muchos casos existe una limitación de memoria principal, puesto que la configuración de MR asigna un total de 4GB a cada tarea en paralelo solo es posible aprender el modelo cuando la verticalización se lleva a cabo con 8 tareas o más. El caso de *epsilon* es más extremo, donde el fallo de memoria se produce incluso al intentar ejecutar el algoritmo secuencial utilizando en su totalidad la memoria principal.

Los resultados anteriores indican dificultades a la hora de escalar el clasificador A2DE mediante MR, sobre todo si pretendemos hacer uso de hardware convencional como su definición indica; si bien una posible opción podría basarse en incrementar el paralelismo vertical, lo que implicaría un alto sobrecoste al ser necesaria la lectura de mucha información redundante. En este punto resulta relevante estudiar el tamaño de los modelos obtenidos midiendo el espacio ocupado en disco (Tabla 4), en base a estos resultados queda patente que el clasificador A2DE en dominios de alta dimensionalidad puede llegar incluso a sobrepasar el tamaño original de los datos de entrenamiento, lo cual implica la inviabilidad de su uso para clasificación escalable.

5. Conclusiones y Trabajo Futuro

Se ha estudiado una implementación de los clasificadores A1DE y A2DE sobre el paradigma MapReduce a partir de la cual se ha podido observar un

comportamiento escalable en su aplicación a problemas de gran tamaño y en dominios de alta dimensionalidad, donde se han obtenido resultados muy favorables en comparación con su implementación secuencial. Por otra parte, ha quedado patente que en el caso del clasificador A2DE la propia definición del modelo no es escalable, ya que el tamaño de los modelos obtenidos es muy elevado.

Sería por tanto relevante extender este estudio para incluir también la fase de clasificación, dado el elevado tamaño de los modelos obtenidos. Además será necesario proceder al diseño de nuevas técnicas de aprendizaje que, en base a los principios del clasificador A2DE, puedan proporcionar un equilibrio entre calidad en la clasificación y escalabilidad. Por ejemplo, se podría aprovechar la gran cantidad de datos disponibles para aprender a partir de muestras aleatorias de los datos y proyecciones del conjunto de atributos haciendo uso de técnicas que ya han resultado exitosas en el caso de otros modelos basados en ensembles.

Agradecimientos. Este trabajo ha sido parcialmente financiado por fondos FEDER, el Gobierno de España (MINECO) y la JCCM a través de los proyectos TIN2013-46638-C3-3-P y PEII-2014-049-P. Jacinto Arias también obtiene financiación mediante una beca FPU del MECD con referencia FPU13/00202.

Referencias

1. Bacardit, J., Widera, P., Márquez, A., Divina, F., Aguilar, J.S., Krasnogor, N.: Contact map prediction using a large-scale ensemble of rule sets and the fusion of multiple predicted structural features. Bioinformatics 28(19), 2441–2448 (2012)
2. Bielza, C., Larrañaga, P.: Discrete Bayesian Network Classifiers. ACM Computing Surveys. 47(1), 1–43 (2014).
3. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM pp. 1–13 (2008)
4. Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple classifier systems, pp. 1–15. Springer (2000)
5. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine learning 29(2-3), 131–163 (1997)
6. Lichman, M.: UCI ML repository (2013), `http://archive.ics.uci.edu/ml`
7. Manyika, J., Chui, M., Brown, B., Bughin, J.: Big data: The next frontier for innovation, competition, and productivity. Tech. Rep. June (2011)
8. Madsen, A. L., Jensen, F., Salmerón, A., Karlsen, M., Langseth, H., Nielsen, T. D.: A new method for vertical parallelisation of TAN learning based on balanced incomplete block designs. Lecture Notes in Artificial Intelligence, 8754, 302–317 (2014)
9. Sonnenburg, S., Franc, V.: Coffin: A computational framework for linear SVMs. In: Proceedings of the 27th ICML-10. pp. 999–1006 (2010)
10. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive Bayes: Aggregating One-Dependence Estimators. Machine Learning 58(1), 5–24 (2005)
11. Webb, G.I., Boughton et al.: Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. Machine Learning 86(2), 233–272 (2011)
12. Yuan, G.X., Ho, C.H., Lin, C.J.: An improved GLMNET for L1-regularized logistic regression. JMLR 13(1), 1999–2030 (2012)