Aplicando Métodos de Aprendizaje Sensible al Coste para Mejorar Problemas de *Big Data* Extremadamente Desbalanceados Usando *Random Forest*

Sara del Río, Victoria López, José Manuel Benítez, and Francisco Herrera

Departamento de Ciencias de la Computación e Inteligencia Artificial, CITIC-UGR (Centro de Investigación en Tecnologías de la Información y las Comunicaciones), Granada, España
{srio,vlopez,j.m.benitez,herrera}@decsai.ugr.es

Abstract. Debido a la explosión y proliferación de datos en los últimos años, surge la necesidad de aplicaciones que permitan procesar cantidades ingentes de información. Esto supone un desafío pues las técnicas de minería de datos no están bien preparadas para afrontar los requerimientos de tiempo y espacio. Por otra parte, en muchos dominios de aplicación los problemas de clasificación suelen presentar una distribución de datos compleja donde las clases aparecen desbalanceadas. En este trabajo proponemos el uso de técnicas de aprendizaje sensible al coste en un entorno *Hadoop* para el algoritmo *Random Forest* de la biblioteca *Mahout*. Se ha llevado a cabo un estudio experimental con datos extremadamente desbalanceados en el que los resultados obtenidos muestran una mejora de rendimiento tanto en precisión como en tiempos de respuesta.

Keywords: *Big Data*, *Mahout*, *Hadoop*, Datos no Balanceados, Aprendizaje Sensible al Coste, *Random Forest*

1 Introducción

En la actualidad, en multitud de sectores como el de las telecomunicaciones, el farmacéutico o el de la salud, se genera información constantemente y en cantidades astronómicas. La realización de análisis sobre grandes volúmenes de datos es de crucial importancia para la obtención y mejora del conocimiento. Debido a que las bases de datos han crecido a mayor velocidad que la capacidad de procesamiento y de análisis inteligente, los almacenes de datos y las tecnologías de minería de datos han quedado desbordados por no poder hacer frente a dichas cantidades masivas de datos. Estos datos se empiezan a conocer como *Big Data*.

Por otra parte, en la mayoría de aplicaciones los datos presentan una distribución de clases en los que una o más clases están representadas por un gran número de ejemplos, mientras que el resto se representan por unos pocos. Esta situación es conocida como clasificación sobre conjuntos de datos desbalanceados y su importancia reside en que las clases minoritarias son el foco de interés del problema [13, 20].

Las soluciones de *Big Data* representan un nuevo paradigma en la administración de grandes volúmenes de datos. Estas soluciones se han centrado en el procesamiento de los mismos de forma distribuida, escalable y confiable. Para lograr este objetivo, surgió un modelo de programación denominado *MapReduce* [8]. Dicho modelo de programación divide el conjunto de datos original en subconjuntos más fáciles de abordar para después combinar las soluciones parciales obtenidas. Sin embargo, esta distribución de los datos suele ocasionar un efecto negativo, especialmente en conjuntos de datos desbalanceados donde se agrava especialmente dicho problema [19]. Además, al realizar una partición adicional de los datos podemos inducir artificialmente un problema de cambio en la distribución de los datos [16].

Para tratar de solventar estos problemas, en este trabajo presentamos un nuevo método basado en el algoritmo *Random Forest* [5] para el manejo de datos extremadamente desbalanceados. Gracias a su naturaleza de *ensemble*, se facilita la creación de una versión paralela del algoritmo, donde el conjunto de datos original se distribuya entre los distintos nodos de un clúster [11]. Para su realización, proponemos el uso conjunto de *Hadoop* [22] y de técnicas sensibles al coste [7].

Con este propósito, se lleva a cabo un estudio experimental en el que nos centramos en conjuntos de datos extremadamente desbalanceados. Para evaluar el rendimiento de la propuesta haremos uso de la Media Geométrica [3].

Este trabajo se organiza de la siguiente forma. En primer lugar, en la sección 2, se presenta una breve introducción a *Big Data* y a los problemas de clasificación con conjuntos de datos desbalanceados. En la sección 3, se da una descripción de la propuesta. Los experimentos y resultados se presentan en la Sección 4. Por último, en la sección 5 se ofrecen algunas conclusiones.

2 Clasificación con *Big Data* y Conjuntos de Datos Desbalanceados

En esta sección proporcionamos una introducción a *Big Data* en problemas de clasificación (Sección 2.1) y, a continuación, una breve descripción de los problemas de clasificación con conjuntos de datos desbalanceados (Sección 2.2).

2.1 El Desafío de *Big Data* en Clasificación

El constante incremento en la velocidad a la que se generan los datos nos conduce a manejar cantidades masivas a través de los algoritmos tradicionales de aprendizaje automático. Esto es un problema pues dichos algoritmos no se encuentran adaptados para tratarlos. Por este motivo, y para un procesamiento eficiente de estas cantidades masivas de datos (*Big Data*), será necesario adaptar los algoritmos de aprendizaje automático adecuadamente.

Facebook y *Twitter* son ejemplos de aplicaciones reales en las que se generan tales cantidades de datos. Por ejemplo, en 2010, *Facebook* contaba con 21 Peta Bytes en su almacén de datos y su almacenamiento crecía a un ritmo aproximado de 12 Tera Bytes diariamente [21].

Entre las soluciones propuestas al problema, algunas de las tecnologías relacionadas se están convirtiendo en el estándar de facto de *Big Data*. En 2004, Google presentó *MapReduce* [8], un modelo de programación para el procesamiento de grandes conjuntos de datos. Este nuevo paradigma computacional consta de dos fases, conocidas como *Map* y *Reduce*. La fase *Map* consiste en distribuir y ejecutar en cada nodo del clúster pequeños programas conocidos como *mappers*. Una de las implementaciones más populares en estos momentos de *MapReduce* es *Hadoop* [1, 22], un *framework* de libre distribución escrito en Java que facilita la escritura de aplicaciones distribuidas. *Hadoop* usa para el almacenamiento el sistema de ficheros HDFS (*Hadoop Distributed File System*), que crea múltiples réplicas de los datos y los distribuye en los nodos de un clúster.

Mahout [2, 18] es una biblioteca de algoritmos de aprendizaje automático y de minería de datos que aprovecha *Hadoop* para proporcionar el almacenamiento de los datos y la implementación del paradigma *MapReduce*. La biblioteca *Mahout* incluye algoritmos de recomendación, agrupamiento y clasificación.

2.2 Clasificación con Conjuntos de Datos Desbalanceados

El problema de la clasificación con conjuntos de datos desbalanceados se produce cuando existe una notable diferencia entre el número de ejemplos que pertenecen a las distintas clases [20]. Este problema ha adquirido mucha importancia en los últimos años debido a su presencia en numerosas aplicaciones reales tales como diagnosis médica, finanzas o bioinformática. En estos problemas, el interés de los expertos se centra en la identificación de los casos minoritarios pues suelen ser los más importantes desde el punto de vista del aprendizaje, y conllevan altos costes cuando su identificación no se realiza adecuadamente [9].

Tradicionalmente, la relación de desequilibrio [17], que se define como la relación entre el número de instancias de la clase mayoritaria y la clase minoritaria, determina el nivel de dificultad asociado a un conjunto de datos determinado. Sin embargo, existen factores adicionales que influyen negativamente en la clasificación con conjuntos de datos desbalanceados como por ejemplo: el solapamiento de clases, la existencia de una muestra de pequeño tamaño, y las diferencias en la distribución de los datos entre los conjuntos de entrenamiento y test [16], principalmente.

Numerosas técnicas han sido usadas para abordar el problema de clasificación de datos desbalanceados [13, 20]. Podemos hacer una división en enfoques: a nivel de datos [4, 6] en los que las instancias del conjunto de entrenamiento se equilibran para obtener una distribución de clases equitativa, y a nivel de algorítmico [15], que suponen una modificación de los algoritmos de manera que intenten beneficiar la clasificación de la clase minoritaria. Los enfoques sensibles al coste combinan los enfoques anteriores para intentar minimizar el coste de cometer un error clasificando una instancia de la clase minoritaria como mayoritaria [9]. Las técnicas basadas en *ensembles* han demostrado un buen comportamiento al adaptarse a problemas de clasificación desbalanceada [10].

El acierto en clasificación no suele ser una métrica de rendimiento apropiada cuando la distribución de las clases es muy diferente, pues no considera el coste

de las clasificaciones incorrectas y no identifica la incorrecta clasificación de las instancias de las clases menos representadas. En este trabajo vamos a utilizar la Media Geométrica (MG) [3], que se define como $MG = \sqrt{VP_{tasa} \cdot VN_{tasa}}$, donde VP_{tasa} es el porcentaje de ejemplos de la clase positiva bien clasificados y VN_{tasa} es el porcentaje de ejemplos de la clase negativa bien clasificados. Esta medida trata de maximizar el acierto de cada una de las dos clases equiparando estos dos objetivos en una única medida.

3 *Random Forest* para Clasificación con Problemas de *Big Data* Extremadamente Desbalanceados

En esta sección describiremos nuestra propuesta para hacer frente a los problemas con conjuntos de datos desbalanceados mediante el uso del algoritmo *Random Forest* (RF). Para ello, primero presentamos diversas aproximaciones de RF que serán la base de nuestra aproximación final.

De esta manera, en la Sección 3.1, se presenta la versión de RF para *Big Data*. A lo largo de la Sección 3.2, se describe la versión del algortimo RF sensible al coste. Por último, en la Sección 3.3, se describe la propuesta final en base a las versiones anteriormente presentadas.

Random Forest [5], es uno de los *ensembles* de árboles de decisión más conocidos y utilizados en clasificación. Este algoritmo combina clasificadores basados en árboles de decisión aleatorios, de forma que la clase a la que pertenece una nueva instancia vendrá determinada por el voto mayoritario de cada uno de estos árboles.

3.1 *Random Forest* para Clasificación con *Big Data*

La implementación *Partial* de *Mahout* [11] es un algoritmo que construye múltiples árboles para diferentes porciones de los datos. Se trata de una implementación *MapReduce* donde cada *mapper* es responsable de construir un subconjunto del *ensemble* de árboles haciendo uso de los datos disponibles en su partición. Este algoritmo tiene dos fases:

- **Fase de construcción:** En esta fase se construye el *ensemble* de árboles. Para ello, se ejecuta en el clúster un *Job MapReduce* (Figura 1).
- **Fase de clasificación:** Se ejecuta un nuevo *Job MapReduce* para la estimación de las clases asociadas al conjunto de datos (Figura 2).

Cada una de las fases anteriores consta de tres etapas: inicial, *map* y final. La etapa inicial lleva a cabo una segmentación del conjunto de datos original en bloques HDFS independientes, que son distribuidos por los nodos de un clúster. En la etapa *map*, cada *mapper* procesa una porción de información y genera un fichero que contiene, en la primera fase, la información relativa a cada uno de los árboles y, en la segunda fase, un fichero con las predicciones. En la etapa final se combinan los ficheros de salida de cada uno de los *mappers* para construir, en la primera fase, el *ensemble* de árboles y, en la segunda fase, el fichero con todas las predicciones.

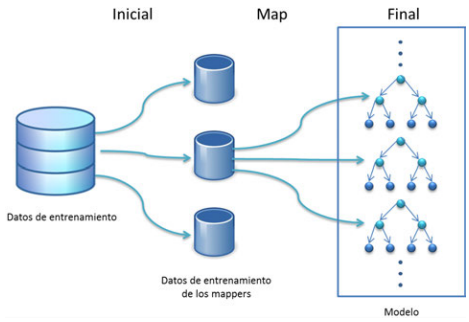

Fig. 1. Fase de construcción de un RF con la versión Partial de Mahout

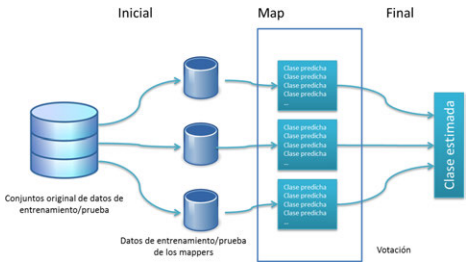

Fig. 2. Fase de clasificación de un RF con la versión Partial de Mahout

3.2 *Random Forest* para Clasificación con Conjuntos de Datos Desbalanceados

Weighted Random Forest [7] es una versión sensible al coste del algoritmo RF para tratar con conjuntos de datos desbalanceados. Esta aproximación modifica el algoritmo RF original en la etapa de construcción del árbol y en la etapa de clasificación. Durante la etapa de construcción del árbol, los costes son incorporados en todos los cálculos llevados a cabo internamente. Por ejemplo, el criterio utilizado para escoger una nueva división del árbol tiene en cuenta los costes asociados a cada clase en una instancia, en lugar de contabilizar las instancias en la misma proporción. Además, los pesos también se tienen en cuenta durante el cálculo de la clase asociada a la hoja del árbol. En la etapa de clasificación, modificamos el esquema de votación por mayoría a uno de votación ponderado. En este caso, calculamos el peso para cada hoja en cada árbol como una proporción de los pesos de las clases asociadas a cada una de las instancias clasificadas en la hoja en relación con el número de instancias clasificadas en dicha hoja.

Un problema de las técnicas sensibles al coste consiste en la propia estimación del coste, que no suelen proporcionar habitualmente los expertos. Para estimar el coste, se utiliza la estimación *out-of-bag* proporcionada por el algoritmo RF original [7].

3.3 RF-BigDataCS: *Random Forest* para Clasificación con Problemas de *Big Data* Extremadamente Desbalanceados

Inspirados en la implementación *Partial* del algoritmo RF de *Mahout*, hemos desarrollado un nuevo algoritmo que puede ser utilizado para problemas de clasificación de *Big Data* extremadamente desbalanceados: RF-BigDataCS. El hecho de que la implementación *Partial* del algoritmo RF de *Mahout* realice una segmentación del conjunto de datos original, dificulta aún más la clasificación con datos desbalanceados al reducir el tamaño de la muestra [19]. Por este motivo, debemos hacer uso de técnicas sensibles al coste.

Para adaptar el aprendizaje sensible al coste al entorno de *Mahout* es necesario incluir las modificaciones de la versión sensible al coste en serie en la implementación de la versión Partial de Mahout. En primer lugar, necesitamos estimar los pesos de cada una de las clases para que las clases minoritarias tengan un mayor peso. El peso de cada clase se calcula de la forma:

$$pesoClase_i = \frac{n_{claseMayoritaria}}{n_i}, \tag{1}$$

donde la clase mayoritaria es la clase con el mayor número de instancias y n_i es el número de instancias de la clase i. Esta estimación no puede llevarse a cabo teniendo en cuenta la estimación *out-of-bag* pues no es capaz de proporcionar pesos elevados para datos extremadamente desbalanceados.

Adicionalmente, se ha modificado el criterio utilizado para escoger una división al construir el árbol: en lugar de contabilizar las instancias de todas las clases en la misma proporción, consideramos el peso de cada clase asociado a la instancia. Además, los pesos también se tienen en cuenta para calcular cuál es la clase asociada a la hoja del árbol.

Una vez que cada *mapper* ha construido un subconjunto del *ensemble* de árboles, modificamos el método de razonamiento para clasificar nuevos ejemplos:

- Finalizada la etapa de construcción del modelo, el algoritmo calcula los pesos de las hojas de cada árbol. Para cada instancia del conjunto de datos, el algoritmo acumula en cada hoja el número de instancias clasificadas y el peso de la clase asociado a cada una de ellas. El peso de cada hoja es el peso acumulado dividido por el número de instancias clasificadas.
- Cuando se lleva a cabo la etapa de clasificación, para cada árbol que participa en la predicción de un ejemplo, el algoritmo acumula en la clase predicha por el mismo el peso asociado a la hoja. Para cada instancia y para todas las clases, se divide el peso acumulado anteriormente por el número de árboles que participan en la clasificación. Finalmente, se selecciona como clase asignada la que obtenga un mayor valor tras este proceso.

4 Estudio Experimental

En esta sección se describe el estudio experimental que se ha llevado a cabo para comprobar el rendimiento de nuestra propuesta RF-BigDataCS. En primer

Tabla 1. Resumen de conjuntos de datos no balanceados

Conjuntos de datos	#Ejemplos	#Atributos	Clase(may; min)	%Clase(may; min)	IR
kddcup_10_DOS_versus_normal	488736	41	(DOS; normal)	(80,10; 19,90)	4,02
kddcup_10_DOS_versus_PRB	395565	41	(DOS; PRB)	(98,96; 1,04)	95,31
kddcup_10_DOS_versus_R2L	392577	41	(DOS; R2L)	(99,71; 0,29)	349,83
kddcup_10_DOS_versus_U2R	391517	41	(DOS; U2R)	(99,98; 0,02)	6634,88
kddcup_10_normal_versus_PRB	101385	41	(normal; PRB)	(95,95; 4,05)	23,69
kddcup_10_normal_versus_R2L	98397	41	(normal; R2L)	(98,86; 1,14)	86,93
kddcup_10_normal_versus_U2R	97337	41	(normal; U2R)	(99,94; 0,06)	1648,78
kddcup_50_DOS_versus_normal	2428075	41	(DOS; normal)	(79,97; 20,03)	3,99
kddcup_50_DOS_versus_PRB	1962236	41	(DOS; PRB)	(98,95; 1,05)	94,48
kddcup_50_DOS_versus_R2L	1942248	41	(DOS; R2L)	(99,97; 0,03)	3448,82
kddcup_50_DOS_versus_U2R	1941711	41	(DOS; U2R)	(99,99; 0,01)	74680,19
kddcup_50_normal_versus_PRB	506941	41	(normal; PRB)	(95,95; 4,05)	23,67
kddcup_50_normal_versus_R2L	486953	41	(normal; R2L)	(99,88; 0,12)	863,93
kddcup_50_normal_versus_U2R	486416	41	(normal; U2R)	(99,99; 0,01)	18707,31
kddcup_full_DOS_versus_normal	4856151	41	(DOS; normal)	(79,97; 20,03)	3,99
kddcup_full_DOS_versus_PRB	3924472	41	(DOS; PRB)	(98,95; 1,05)	94,48
kddcup_full_DOS_versus_R2L	3884496	41	(DOS; R2L)	(99,97; 0,03)	3448,82
kddcup_full_DOS_versus_U2R	3883422	41	(DOS; U2R)	(99,99; 0,01)	74680,19
kddcup_full_normal_versus_PRB	1013883	41	(normal; PRB)	(95,95; 4,05)	23,67
kddcup_full_normal_versus_R2L	973907	41	(normal; R2L)	(99,88; 0,12)	863,93
kddcup_full_normal_versus_U2R	972833	41	(normal; U2R)	(99,99; 0,01)	18707,33

lugar, en la Sección 4.1 se presentan los problemas de clasificación utilizados en los experimentos. A continuación, la Sección 4.2 muestra los algoritmos y parámetros utilizados. Después, en la Sección 4.3 se presentan los resultados según la MG para cada una de las aproximaciones utilizadas. Finalmente, la Sección 4.4 muestra un análisis para evaluar la ganacia computacional (*speed-up*) por la utilización de técnicas para *Big Data*.

4.1 Conjuntos de Datos

Con el fin de analizar la calidad de nuestra propuesta se han seleccionado varios problemas de *Big Data* extremadamente desbalanceados derivados del conjunto de datos KDD Cup 1999 [14]. Puesto que este conjunto contiene múltiples clases, éstas se han agrupado para generar problemas binarios desbalanceados de gran tamaño. Concretamente se han creado nuevos conjuntos a partir de las clases *normal* y *DOS* como mayoritarias. Además, para llevar a cabo una prueba de escalabilidad se han generado versiones del 10% y del 50% del conjunto de datos original. Los datos utilizados se resumen en la tabla 1. Para el desarrollo de los experimentos se ha considerado un esquema de validación cruzada en 10 particiones.

4.2 Algoritmos y Parámetros

En este estudio, se ha comparado el rendimiento de nuestra propuesta, RF-BigDataCS, con respecto a las versiones básicas del algoritmo:

- ***Random Forest*** (RF) [5]: *Random Forest* original, *ensemble* de árboles de decisión disponible en la herramienta de minería de datos Weka [12].
- ***Random Forest* Sensible al Coste** (RF-CS) [7]: Versión adaptada de *Random Forest* que introduce aprendizaje sensible al coste.
- ***Random Forest* para *Big Data*** (RF-BigData) [11]: Versión Partial de *Random Forest* disponible en *Mahout* [18].

En la tabla 2 mostramos la configuración de parámetros para estos algoritmos, donde el parámetro *maxProfundidad* indica la profundidad de cada uno

de los árboles generados, *numCaracteristicas* es el número de atributos para construir los árboles aleatorios y *numArboles* representa el número de árboles que componen el *ensemble* de árboles. Además, podemos encontrar los atributos *estimacionCostes* y *numParticiones*. El primero ilustra cómo son calculados los costes de cada clase mientras que el segundo indica en cuántas partes se reparte el conjunto original para el procesamiento de los *mapper*.

Tabla 2. Parámetros para los algoritmos estudiados en la experimentación

Algoritmos	Parámetros
RF	maxProfundidad = ilimitada, numCaracteristicas = $log_2(N_{vars}) + 1$, numArboles = 100
RF-CS	maxProfundidad = ilimitada, numCaracteristicas = $log_2(N_{vars}) + 1$, numArboles = 100 estimacionCostes = basadaEnPesos
RF-BigData	maxProfundidad = ilimitada, numCaracteristicas = $log_2(N_{vars}) + 1$, numArboles = 20/10/5 numParticiones = 5/10/20 (respectivamente)
RF-BigDataCS	maxProfundidad = ilimitada, numCaracteristicas = $log_2(N_{vars}) + 1$, numArboles = 20/10/5 numParticiones = 5/10/20 (respectivamente), estimacionCostes = basadaEnPesos

4.3 Análisis de Rendimiento

Puesto que estamos interesados en problemas extremadamente no balaneados se han seleccionado las versiones *full* con mayor IR para analizar el rendimiento. En la tabla 3 se muestra la media de los resultados obtenidos para entrenamiento y test de todos los métodos estudiados en este trabajo. El símbolo NC (no computable) significa que la versión no fue capaz de completar el experimento. Esto es debido a que la implementación no está preparada para el uso de conjuntos de datos de este tamaño.

Podemos observar que los resultados de las versiones secuenciales, RF original y su modificación RF-CS, son ligeramente mejores con respecto a la adaptación a *Big Data*. Esta situación es común cuando se compara un enfoque secuencial con su paralelización pues parte de la precisión del modelo es sacrificada para la ganancia de velocidad. Además, se puede observar que RF-CS obtiene generalmente mejores resultados que RF original y que este comportamiento se extrapola a las versiones de RF-BigData.

También podemos comprobar el impacto que producen las distintas particiones sobre las versiones de RF-BigData pues cuantas más particiones se usan mayor es la pérdida de rendimiento del algoritmo. Nuestra propuesta, RF-BigDataCS, también se ve afectada por este problema, sin embargo, su variación no es tan dramática como en el caso de RF-BigData ya que mantiene un buen rendimiento tanto en precisión como en tiempos de respuesta, y por tanto, permite resolver el problema de clasificar grandes conjuntos de datos desbalanceados.

4.4 Estudio del *Speed-up*

Una de las principales cuestiones del uso de implementaciones distribuidas y paralelas es la obtención de mejores resultados en tiempo de ejecución. Por este motivo, es interesante evaluar el rendimiento de las versiones secuenciales frente a su adaptación distribuida. Para ello, haremos uso de la medida *Speed-up* o ganancia de velocidad, definida como $S_{up} = \frac{T_{sec}}{T_{par}}$, donde T_{sec} y T_{par} son los tiempos de ejecución de las versiones secuencial y paralela, respectivamente.

Tabla 3. Media de los resultados para las versiones de RF para conjuntos de datos extremadamente desbalanceados haciendo uso de la medida MG

Conjuntos de datos	kddcup_full_DOS_versus_U2R		kddcup_full_normal_versus_R2L		kddcup_full_normal_versus_U2R	
	MG_{entr}	MG_{tst}	MG_{entr}	MG_{tst}	MG_{entr}	MG_{tst}
Versiones secuenciales						
RF	NC	NC	1,0000	0,9832	0,9999	0,6836
RF-CS	NC	NC	0,9998	0,9976	0,9999	0,9813
Versiones para *Big Data*						
RF-BigData – 5 partes	0,7610	0,6731	0,9482	0,9376	0,3565	0,3333
RF-BigData-CS – 5 partes	0,8278	0,9608	0,9812	0,9835	0,9433	0,9960
RF-BigData - 10 partes	0,7045	0,6756	0,9274	0,9261	0,0000	0,0000
RF-BigData-CS – 10 partes	0,8032	0,9822	0,9793	0,9894	0,9381	0,9691
RF-BigData - 20 partes	0,0267	0,0000	0,8841	0,8767	0,0000	0,0000
RF-BigData-CS – 20 partes	0,9186	0,8853	0,9719	0,9657	0,9373	0,9593

La tabla 4 muestra los valores de *Speed-up* para cada uno de los conjuntos de datos considerados en este estudio. En ella podemos observar que las versiones de RF-BigData obtienen tiempos de respuesta mucho mejores. Además, podemos comprobar que el número de particiones influye en la ganancia en velocidad, sin embargo, no se mantiene una relación lineal, así como que la ganancia no es homogénea entre los conjuntos de datos considerados en este estudio.

Tabla 4. *Speed-up* para RF-BigData y RF-BigData-CS con 5, 10 y 20 particiones

Conjuntos de datos	RF-BigData - 5 partes			RF-BigData-CS - 5 partes			RF-BigData - 10 partes			RF-BigData-CS - 10 partes			RF-BigData - 20 partes			RF-BigData-CS - 20 partes		
	10%	50%	full	10%	50%	full	10%	50%	full	10%	50%	full	10%	50%	full	10%	50%	full
DOS_versus_normal	66,78	76,06	NC	39,88	50,81	NC	75,53	159,53	NC	40,83	76,82	NC	64,74	222,33	NC	29,07	66,04	NC
DOS_versus_PRB	48,74	57,18	NC	36,30	53,04	NC	58,14	115,74	NC	34,50	71,30	NC	48,25	154,94	NC	24,85	61,25	NC
DOS_versus_R2L	51,20	97,48	NC	35,96	73,35	NC	57,11	153,41	NC	34,03	91,79	NC	45,92	178,81	NC	23,24	70,85	NC
DOS_versus_U2R	47,52	102,80	NC	39,26	109,11	NC	49,99	187,68	NC	38,90	113,67	NC	41,39	184,88	NC	26,67	79,44	NC
normal_versus_PRB	5,39	77,07	NC	5,17	56,87	NC	6,09	98,55	NC	5,42	77,13	NC	4,99	103,65	NC	3,90	80,43	NC
normal_versus_R2L	4,01	66,29	7,32	3,37	62,44	7,37	4,68	80,90	35,60	3,54	78,49	16,39	3,96	122,39	60,01	2,55	78,49	15,31
normal_versus_U2R	3,70	75,96	10,54	3,91	68,81	8,15	3,84	92,03	40,10	3,56	32,66	16,47	3,18	92,03	62,28	2,59	23,67	14,19

5 Comentarios Finales

En este trabajo se ha presentado un algoritmo que basado en el algoritmo *Random Forest*, permite abordar problemas extremadamente desbalanceados con *Big Data*. Inspirado en las técnicas utilizadas para hacer frente a estas características de manera independiente, se han combinado ambas aproximaciones para construir un nuevo modelo que las integre y sea capaz de resolver ambos problemas de forma simultánea.

En la actualidad, los problemas de *Big Data* están ganando reconocimiento debido a las grandes cantidades de datos que se generan hoy en día. Los métodos de minería de datos tradicionales no son capaces de hacer frente a los nuevos requerimientos impuestos por *Big Data*. Por este motivo, hacemos uso del entorno *Hadoop*, que facilita el desarrollo de soluciones escalables y distribuidas. La biblioteca *Mahout*, construida sobre *Hadoop*, proporciona algoritmos de aprendizaje automático capaces de procesar grandes conjuntos de datos. El aprendizaje sensible al coste permite la transformación de los métodos de aprendizaje estándar en métodos que son capaces de hacer frente a conjuntos de datos desbalanceados en problemas de clasificación.

El rendimiento de nuestro modelo RF-BigData-CS ha sido contrastado en un estudio experimental que incluye varios problemas de *Big Data* extremadamente

desbalanceados. Los resultados obtenidos muestran que nuestra propuesta obtiene mejores resultados en cuanto a precisión y tiempo de respuesta. Además, señalan la necesidad de abordar conjuntamente problemas de *Big Data* y no balanceados, pues las distintas técnicas que por sí solas no son capaces de resolver el problema, al combinarse producen una buena sinergia.

Acknowledgements. Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia e Innovación bajo el proyecto TIN2011-28488 y los Planes Andaluces de Investigación P11-TIC-7765 y P10-TIC-6858. V.López posee una beca FPU del Ministerio de Educación, Cultura y Deporte.

Referencias

[1] Apache Hadoop Project: Apache Hadoop (2013), `http://hadoop.apache.org/`, [Online; consultado en junio 2013]
[2] Apache Mahout Project: Apache Mahout (2013), `http://mahout.apache.org/`, [Online; consultado en junio 2013]
[3] Barandela, R., Sánchez, J.S., García, V., Rangel, E.: Strategies for learning in class imbalance problems. Pattern Recognition 36(3), 849–851 (2003)
[4] Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behaviour of several methods for balancing machine learning training data. SIGKDD Explorations 6(1), 20–29 (2004)
[5] Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
[6] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over–sampling technique. Journal of Artificial Intelligent Research 16, 321–357 (2002)
[7] Chen, C., Liaw, A., Breiman, L.: Using random forest to learn imbalanced data. Tech. Rep. 666, Statistics Department, University of California Berkeley (2004)
[8] Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
[9] Elkan, C.: The foundations of cost–sensitive learning. In: Proceedings of the 17th IEEE International Joint Conference on Artificial Intelligence (IJCAI'01). pp. 973–978 (2001)
[10] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for class imbalance problem: Bagging, boosting and hybrid based approaches. IEEE Transactions on Systems, Man, and Cybernetics–part C: Applications and Reviews 42(4), 463–484 (2012)
[11] Hakim, D.A.: PartialData MapReduce Random Forests (2013), `http://cwiki.apache.org/confluence/display/MAHOUT/Partial+Implementation`, [Online; consultado en junio 2013]
[12] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explorations 11(1), 10–18 (2009)
[13] He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21(9), 1263–1284 (2009)
[14] International Knowledge Discovery Data Mining Tools Competition: KDD Cup 1999 Data (2013), `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, [Online; consultado en junio 2013]
[15] López, V., Fernández, A., del Jesus, M.J., Herrera, F.: A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets. Knowledge-Based Systems 38, 85–104 (2013)
[16] Moreno-Torres, J.G., Raeder, T., Aláiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition 45(1), 521–530 (2012)
[17] Orriols-Puig, A., Bernadó-Mansilla, E.: Evolutionary rule–based systems for imbalanced datasets. Soft Computing 13(3), 213–225 (2009)
[18] Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in Action. Manning Publications Co. (2012)
[19] Raudys, S.J., Jain, A.K.: Small sample size effects in statistical pattern recognition: Recommendations for practitioners. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(3), 252–264 (1991)
[20] Sun, Y., Wong, A.K.C., Kamel, M.S.: Classification of imbalanced data: A review. International Journal of Pattern Recognition and Artificial Intelligence 23(4), 687–719 (2009)
[21] Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sen Sarma, J., Murthy, R., Liu, H.: Data warehousing and analytics infrastructure at facebook. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2010). pp. 1013–1020 (2010)
[22] White, T.: Hadoop, The Definitive Guide. O'Reilly Media, Inc. (2012)