Árbol de decisión C4.5 basado en entropía minoritaria para clasificación de conjuntos de datos no balanceados

Luis A. Caballero-Cruz, Asdrúbal López-Chau, Jorge Bautista-López

Ingeniería en Computación, Centro Universitario UAEM Zumpango, Zumpango, Estado de México, México

Resumen. En el área de aprendizaje automático, el problema de desbalance en los datos es uno de los más desafiantes. Desde hace más de una década, se han desarrollado nuevos métodos para mejorar el desempeño de los métodos de clasificación para este tipo de problema. En este artículo se presenta una modificación al algoritmo C4.5 usando el concepto de entropía minoritaria. La propuesta está basada en la corrección de un error que observamos en una publicación previa. La implementación del nuevo método presentado es probada con conjuntos públicamente disponibles. Los resultados obtenidos muestran la utilidad del método desarrollado.

Palabras clave: desbalance, clasificación, entropía minoritaria, árbol de decisión.

1. Introducción

En minería de datos, la tarea de clasificación se refiere al proceso de utilizar un modelo —llamado clasificador— que permita realizar predicciones sobre el tipo, clase o categoría al que pertenecen los objetos que se le presentan a dicho modelo. Las posibles clases de objetos son conocidas de antemano, y el modelo es construido utilizando un conjunto de datos de entrenamiento, en el que los objetos o instancias están previamente etiquetados.

Un problema que se presenta con los métodos de clasificación tradicionales, sucede cuando los datos usados para construir el clasificador se encuentran no balanceados, es decir; cuando las clases o categorías del conjunto de datos de entrenamiento no se encuentran en cantidades aproximadamente proporcionales [8]. Un ejemplo típico ocurre en el análisis financiero, ya que en los datos de las personas que poseen créditos bancarios, la cantidad de personas que presentan adeudo de cantidades muy grandes, es notablemente menor a la cantidad de personas con adeudo de cantidades pequeñas. Otro ejemplo, en un ámbito muy diferente, se puede encontrar con datos de alumnos de nivel escolar primaria en México. La experiencia indica que son más los alumnos que aprueban el ciclo escolar que aquellos que no aprueban. En la literatura, se han detectado muchos otros ejemplos del mundo real donde sucede este tipo de desbalance en los datos.

En este tipo de aplicaciones, es de suma importancia la correcta predicción de instancias que pertenecen a la clase con la menor cantidad de elementos en el conjunto de datos de entrenamiento [6], llamada clase minoritaria.

En este artículo, se propone una modificación al método de clasificación C4.5, que es un árbol de decisión. Esta modificación permite seleccionar, de forma diferente a la tradicional, los puntos de separación de datos que realiza el clasificador de manera recursiva. De esta manera, la mayor parte de los objetos de clase minoritaria son concentrados al centro o en los extremos de las tres particiones que se realiza en cada iteración del algoritmo. La propuesta está basada en una observación que realizamos al artículo en [3]. Notamos que en dicho artículo existen contradicciones y un error en el cálculo de la entropía minoritaria propuesta, por lo que hicimos la corrección correspondiente, y proponemos algunas otras mejoras como trabajo futuro.

Este artículo está organizado de la siguiente manera. Una revisión sobre trabajos relacionados es descrita en la sección 2. En la sección 3, se presentan los conceptos básicos relacionados con la medición del desempeño de métodos de clasificación con conjuntos de datos no balanceados. También se incluye una revisión breve sobre Weka, que es el framework utilizado para implementar la modificación propuesta. En la sección 4, se presenta el algoritmo original que se propuso en el artículo [3]. En esa sección mostramos la observación que realizamos, y un contra-ejemplo para demostrar un error en el cálculo de la entropía minoritaria. La corrección propuesta, así como la implementación, se detalla en la sección 5. Los resultados se presentan en la sección 6. Las conclusiones generales, así como trabajos futuros, se encuentran en la sección 7. En la última parte de este artículo se encuentran las referencias consultadas.

2. Trabajos relacionados

El efecto de desbalance en los datos sobre el desempeño de los clasificadores ha sido ampliamente estudiado durante los últimos años. Uno de los primeros trabajos sobre ello fue el descrito en [10], en el cual se usaron conjuntos de datos sintéticos para realizar una exploración sobre la relación que existe entre la complejidad de los conceptos subyacentes en los datos, el tamaño de los conjuntos de datos de entrenamiento y el nivel de desbalance entre las clases. Los métodos usados en [10] fueron arboles de decisión, redes neuronales y máquinas de soporte vectorial. En [5] y [11], se llevaron acabo algunos experimentos aplicando el árbol de decisión C4.5 con conjuntos de datos reales. En [2] fue realizado un estudio de los algoritmos especialmente diseñados para la limpieza de los datos, y balance de los conjuntos de datos. Recientemente, en [9], fueron revisados los efectos del ruido en los datos sobre varios métodos de clasificación.

Algunas conclusiones sobre el problema de desbalance en los datos, que se han obtenido son las siguientes. El desbalance entre clases perjudica el desempeño de los métodos de clasificación, pero este no es el factor más importante [14]; el desbalance interno tiene un impacto más negativo que el desbalance entre

clases [7,10]; el ruido en los datos y los casos raros son perjudiciales para los clasificadores [9,13].

Probablemente, el algoritmo más famoso, diseñado específicamente para atender el desbalance de los conjuntos de datos es Synthetic Minority Oversampling Technique (SMOTE) [4]. Este sobre muestrea la clase minoritaria tomando cada muestra de la clase minoritaria e introduce nuevos objetos a lo largo de los segmentos de línea que unen a la instancia con los k vecinos más cercanos. Un problema con SMOTE es que tiende a sobre generalizar, y también introduce traslape entre clases, esto es por que las nuevas muestras son generadas sin considerar la distribución de los datos.

Una revisión más amplia sobre clasificación con conjuntos de datos no balanceados puede encontrarse en [8].

Uno de los algoritmos propuestos recientemente, y que utiliza un árbol de decisión es el presentado en [3]. En ese artículo, se propone usar tres particiones en la construcción del árbol. La idea es concentrar a las instancias de clase minoritaria en la partición central, usando el criterio de entropía minoritaria que se presenta más adelante en el presente artículo. De esta manera, el método le brinda mayor oportunidad a las instancias de clase minoritaria, de ser incluídas en nodos más puros. El la sección 4, se realiza una observación al algoritmo en [3], y se presenta una corrección del mismo.

3. Preliminares

3.1. Árbol de decisión C4.5, o J48 en Weka

En 1993, la Universidad de Waikato, en Nueva Zelanda, desarrolló la primera versión del software llamado *Weka* (*Waikato Environment for Knowledge Analysis*, que en español se traduce como "Entorno para análisis del conocimiento de la Universidad de Waikato"). Este software es un entorno de trabajo que provee la implementación de diferentes algoritmos para aprendizaje automático (*machine learning*). La finalidad principal fue hacer que los algoritmos de aprendizaje automático estén disponibles para el público en general, y que sean aplicados tanto en investigación, como en la industria. La versión actual de Weka es software libre publicado bajo la Licencia Pública General GNU, está escrito en lenguaje Java y soporta aplicaciones para *Big Data*.

Uno de los métodos para clasificación más populares en la comunidad de investigadores y practicantes del área de aprendizaje automático es C4.5 [12]. Este método es un árbol de decisión, ampliamente utilizado para realizar comparativas de desempeño de clasificadores en conjuntos de datos no balanceados. La implementación de C4.5 que ofrece Weka es llamada J48.

Los árboles de decisión son modelos cuya estructura tiene cierta similitud a los diagramas de flujo. La idea subyacente de este tipo de método es realizar particiones al conjunto de datos repetidamente, con la finalidad de que las particiones sean cada vez más *puras*, es decir, que contengan elementos de una sola clase. Cada partición o separación de un conjunto de datos (S), se realiza

probando todos los posibles valores de las instancias en cada dimensión o atributo (A), y después se selecciona a la mejor partición de acuerdo a algún criterio. Este proceso se realiza de manera recursiva.

Algunos de los criterios para elegir la mejor partición de conjuntos de datos, son basados en Entropia(E), Information Gain(IG), Split Info(SI) y Gain Ratio(GR), cuyas ecuaciones son mostradas a continuación :

$$E(S) = -\sum_{i=1}^{N} p_i \log(p_i) \quad (1)$$

$$SI(S) = -\sum_{i=1}^{N} \frac{S_i}{S} \left(\log(\frac{S_i}{S}) \right) \quad (3)$$

$$IG(S, A) = E(S) - \sum_{i=1}^{N} \frac{S_i}{S} E(S_i) \quad (2)$$

$$GR(S, A) = \frac{IG(S, A)}{SI(S)} \quad (4)$$

Los logaritmos son base dos para todas las fórmulas en este artículo.

3.2. Métricas para evaluar el desempeño de clasificadores en conjuntos de datos no balanceados

Una forma conveniente de observar el desempeño de un método de clasificación en conjuntos de datos con dos clases, es usar una matriz de confusión. Ésta contiene información sobre las predicciones hechas por el clasificador usando el número de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. La tabla 1 muestra un ejemplo de matriz de confusión.

Tabla 1. Matriz de confusión

Predicción ⇓		
positivo	negativo	
d	c	positivo
b	a	negativo
		⇑
		Clase real

Utilizar únicamente la precisión de clasificación (ecuación (5)) para determinar el desempeño de un clasificador, no es conveniente cuando se trata de conjuntos de datos no balanceados. Esto es porque en este tipo de aplicaciones, se pueden obtener valores altos de precisión de clasificación, simplemente ignorando a la clase minoritaria, sin embargo, esta última es importante en casos como detección de transacciones fraudulentas, detección de intrusiones en la red, minería web, marketing directo, y diagnósticos médicos [4].

$$AC = \frac{a+d}{a+b+c+d} \tag{5}$$

Es común usar medidas tales como el recuerdo (*recall*) o true positive rate(TP), que indica la proporción de casos positivos que fueron correctamente identificados. El false positive rate (FP), que hace referencia a la proporción de elementos positivos que fueron clasificados incorrectamente como negativos. El true negative rate (TN), que indica la proporción de casos negativos que fueron clasificados correctamente. La proporción de elementos positivos que fueron clasificados incorrectamente como negativos, se obtiene con el false negative rate (FN). Es común que a la clase cuyo número de instancias es la menor (clase minoritaria), se le asigne la categoría de positivo, mientras que a la clase contraria (clase mayoritaria) se le asigne la de negativo. Las ecuaciones siguientes muestran la forma de calcular cada una de las medidas anteriores.

$$TN = \frac{a}{a+b} \tag{6}$$

$$FP = \frac{b}{a+b} \tag{7}$$

$$FN = \frac{c}{c+d} \tag{8}$$

$$TP = \frac{d}{c+d} \tag{9}$$

$$P = \frac{d}{b+d} \tag{10}$$

4. Árbol de decisión basado en entropía minoritaria

El árbol de decisión presentado en [3], divide el conjunto de datos en tres particiones, basándose en los conceptos de entropía minoritaria y tasa de ganancia. La idea es usar dos puntos de separación en el atributo seleccionado, de modo que permitan capturar la mayor cantidad de elementos de clase minoritaria en la partición central.

En [3], se propone la ecuación (11) para calcular la entropía minoritaria., en donde N representa la cantidad de particiones, n_i representa la cantidad de instancias de clase minoritaria en la $i-esima$ partición y n representa la cantidad de elementos de clase minoritaria.

$$MinorityEntropy(n) = -\sum_{i=1}^{N} \frac{n_i}{n} \left(\log \left(\frac{n_i}{n} \right) \right) \tag{11}$$

El propósito de usar esta medida, es dar mayor importancia a las instancias de clase minoritaria. Con la finalidad de reducir el costo computacional para la construcción del árbol de decisión, los puntos de separación se calculan usando un centroide c (ecuación (12)) con $y_i = +1$.

$$c = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{12}$$

Este último es obtenido con los valores de instancias de clase minoritaria sobre el atributo bajo prueba.

En general, en cada etapa de la construcción del árbol, se calculan dos puntos de separación para cada atributo, que permiten dividir los datos en un nodo en tres particiones. Desde ahora, se denominarán SP_1 y SP_2 a los puntos de separación. El primero de estos es calculado desde la primera instancia de clase minoritaria del conjunto, hasta c, mientras que el segundo se calcula desde c, hasta la última instancia de clase minoritaria.

Todos los candidatos cuya tasa de ganancia sea mayor a un umbral T (ecuación 13), son almacenados y posteriormente se selecciona al candidato con el menor valor de entropía minoritaria.

$$T = 1 - R \tag{13}$$

En la ecuación 13, el valor de R, se calcula con la proporción de la tasa de varianza de las instancias minoritarias y la tasa de varianza de todas las instancias.

$$R = \frac{\sigma^2(MinorityClassInstances)}{\sigma^2(AllInstances)} \tag{14}$$

El árbol de decisión es construido de manera similar a C4.5, pero usando la medida de entropía mencionada anteriormente.

4.1. Una observación sobre el cálculo de la entropía minoritaria

La idea subyacente en el algoritmo presentado en [3], es concentrar la mayor cantidad de instancias de clase minoritaria en una sola partición durante la construcción de un árbol de decisión. Lo anterior es logrado mediante el uso del concepto de entropía minoritaria, explicado en la sección 3. Esta idea resulta atractiva para atacar el problema de clasificación sobre conjuntos de datos no balanceados, ya que se evita que el clasificador ignore a las instancias de clase minoritaria, tal como ocurre comúnmente con los algoritmos tradicionales.

Una observación que realizamos en el artículo [3], es que el valor de la entropía minoritaria no se encuentra acotado entre los valores cero y uno. En dicho artículo se puede encontrar la siguiente afirmación: "*This value lies between 0 and 1. If the minority entropy is 0, then all of minority class instances are located in the middle partition*". Hemos encontrado que esto no es siempre cierto. Para mostrarlo, exponemos un contra ejemplo.

Suponer que tenemos un conjunto de datos con *100* elementos, de los cuales *10* son de clase minoritaria. Supongamos también que en alguna iteración del algoritmo, los puntos de separación SP_1 y SP_2 propician la formación de tres particiones con la distribución de los datos que se muestra en la figura 1.

Cada una de las particiones contiene la cantidad de datos mostrada en la tabla 2.

Al realizar el cálculo de la entropía minoritaria para cada partición, obtenemos lo siguiente:

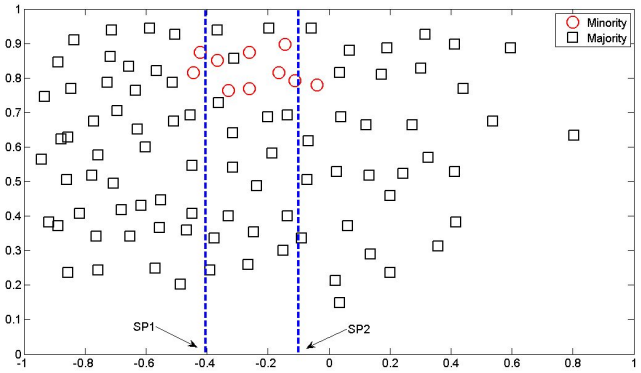

Fig. 1. Conjunto de datos sintético particionado

Tabla 2. Distribución de las instancias en las tres particiones de la figura 1

D	$Instancias_{maj}$	$Instancias_{min}$	Total
1	42	2	44
2	17	7	24
3	31	1	32

$$MinorityEntropy(n) = -\left(\frac{2}{10}\log\left(\frac{2}{10}\right)\right) - \left(\frac{7}{10}\log\left(\frac{7}{10}\right)\right) - \left(\frac{1}{10}\log\left(\frac{1}{10}\right)\right)$$

$$MinorityEntropy(n) \approx 1.16$$

Con lo obtenido en el contra ejemplo expuesto, es claro que la ecuación para calcular la entropía minoritaria produce valores que no se encuentran acotados entre cero y uno, como se afirma en [3]. Esto no sería un inconveniente, sino fuera porque al no estar acotada, cada posible valor en el atributo bajo prueba puede llegar a ser punto de separación, lo que hace que el algoritmo se comporte de manera similar a un árbol de decisión C4.5. En la siguiente sección, realizamos una propuesta para calcular de manera diferente la entropía minoritaria.

5. Método propuesto

Habiendo identificado el problema con el cálculo de la entropía minoritaria, se propone una modificación al algoritmo presentado en [3]. Descrito de manera

breve, se propone seguir manteniendo a las tres particiones para la construcción del árbol de decisión, sin embargo, para el cálculo de la entropía minoritaria se propone usar unicamente dos particiones, la idea es considerar a las particiones extremas como si se tratara de una sola.

El surgimiento de esta propuesta, está basado en observaciones que realizamos usando algunos conjuntos de datos sintéticos. Al analizar las condiciones para las cuales los valores de la entropía minoritaria pertenecían al intervalo $[0, 1]$, observamos que esto sólo ocurre cuando las instancias de clase minoritaria están presentes en dos de las tres particiones. En otros casos, el resultado obtenido puede estar fuera del rango indicado. La forma corregida para el cálculo de entropía minoritaria que proponemos en este artículo es la mostrada en la ecuación (15).

Es importante hacer notar, que pese a que nuestra corrección tiene cierta similitud a la fórmula de cálculo de entropía (ecuación (1)), entre ellas existe una gran diferencia. En la entropía se consideran todas las instancias de cada clase en el conjunto de datos de entrenamiento, en contraste, en la fórmula propuesta únicamente se consideran las instancias de clase minoritaria.

$$MinorityEntropy(n) = -\left(\frac{n_1}{n}\log\left(\frac{n_1}{n}\right)\right) - \left(\frac{n_2}{n}\log\left(\frac{n_2}{n}\right)\right) \tag{15}$$

Donde

n_1 representa la cantidad de instancias de clase minoritaria en la partición central,

n_2 representa la suma de la cantidad de instancias de clase minoritaria que se encuentran en las particiones izquierda y derecha, es decir, la suma de ambas,

n representa la cantidad de elementos de clase minoritaria antes de realizar particiones.

Para los casos en los que únicamente hay dos particiones, la partición central se considera la de la derecha. El pseudocódigo en Algoritmo 1 muestra en detalle la propuesta presentada.

Como ejemplo, suponer que los datos son los mismos del ejemplo anterior. Las instancias de la partición 1 han formado un sólo conjunto con los de la partición 2, por lo que el cálculo de la entropía minoritaria es ahora:

$$MinorityEntropy(n) = -\left(\frac{7}{10}\log\left(\frac{7}{10}\right)\right) - \left(\frac{3}{10}\log\left(\frac{3}{10}\right)\right) \approx 0.88129$$

La correción del cálculo de entropía minoritaria ha sido probada tanto en casos triviales, así como en los casos en los que la fórmula original entrega resultados mayores a uno, aparentemente ambas formulas son similares.

En el Algoritmo 1, el intercambio de los atributos del conjunto L al conjunto R no debe afectar a este conjunto, ya que se realiza únicamente en el algoritmo, pero no en los datos, es decir; el recorrido que realiza el punto de separación SP_2 con cada atributo del conjunto R, no debe considerar los elementos del conjunto L que han sido agregados a R. Nuestra propuesta no asegura que en todos los caso las instancias de clase minoritaria se posicionen en la partición central.

Algorithm 1: C4.5 con Entropía minoritaria

Input : D: Conjunto de datos
Output: Árbol de decisión

begin
  Crear un nodo del árbol;
  if $\forall x \in MajorityClass$ **then**
    return *nodo como clase mayoritaria*
  end
  if $\forall x \in MinorityClass$ **then**
    return *nodo como clase minoritaria*
  end
  foreach *atributo* **do**
    $c := \bar{D}_{min} : \{D_{min} : \exists x : x = instClaseMin\} \& D_{min} \subset D$;
    $L := \{L : \exists x : x \leq c\}$;
    $R := \{R : \exists x : x > c\}$;
    foreach l *en* L **do**
      foreach r *en* R **do**
        Calcular la tasa de ganancia para l y r;
        Calcular la entropía minoritaria;
      end
      Mover el atributo l al conjunto R;
    end
    $G_{multiple} := \{\forall candidato : candidato > T\}$;
    Seleccionar la menor entropía minoritaria de $G_{multiple}$;
  end
  Seleccionar el mejor candidato de los atributos como punto de separación;
  Separar las instancias en las particiones D_1, D_2 y D_3 correspondientes a los puntos de separación seleccionados;
  Invocar recursivamente el algoritmo con D_1, D_2 y D_3;
end

6. Resultados

Para probar el desempeño de nuestro algoritmo, se usaron los conjuntos de datos, cuyas características son mostradas en la tabla 3. Estos datos están públicamente disponibles en la Internet [1], y son usados para realizar experimentos con conjuntos de datos no balanceados. En la versión actual de la implementación, el tipo de atributo considerado por el algoritmo es únicamente numérico, es decir, no se soportan atributos nominales, ni tipo cadena, etc.

El código fuente del algoritmo propuesto, fue escrito en lenguaje Java, modificando varias clases que componen al algoritmo J48 de Weka.

Los datos reportados, son el promedio de 30 repeticiones de los experimentos para cada conjunto de datos. En cada experimento, se usaron el 70 % de los datos elegidos pseudo aleatoriamente para entrenamiento, y el 30 % restante para prueba.

Tabla 3. Conjuntos de datos usados en los experimentos

Nombre	Mayoritaria	Minoritaria	Atributos	Total	IR
ecoli4	316	20	8	336	15.80
glass-0-1-6_vs_5	175	9	10	184	19.44
page-blocks-1-3_vs_4	444	28	11	472	15.85
pima	500	268	9	768	1.86
shuttle-c0-vs-c4	1,706	123	10	1,829	13.86
vehicle1	629	217	19	846	2.89
vowel0	898	90	14	988	9.97
yeast-2_vs_4	463	51	9	514	9.07

La tabla 4 muestra los resultados obtenidos. El significado de las columnas está descrito en la sección 3.

Es interesante notar que aunque a lo largo del artículo [3], se explica un método que pareciera trabajar con atributos numéricos, en los resultados de ese trabajo, los autores presentan solamente conjuntos de datos con atributos categóricos. Debido a lo anterior, y a que la implementación actual de nuestro algoritmo sólo soporta atributos numéricos, fue imposible realizar una comparativa.

Como puede observarse, el algoritmo propuesto tiene un desempeño que supera al del C4.5 original. Es necesario que nuestra propuesta soporte otros tipos de atributos, y que se realicen más experimentos mostrando resultados como áreas bajo la curva (ROC), para poder llegar a conclusiones más generales.

7. Conclusiones y trabajo futuro

El problema de clasificación con conjuntos de datos no balanceados, ha sido un área de interés para la comunidad tanto teórica como práctica de minería de datos. Se han realizado una cantidad grande de estudios para tratar de entender este problema complejo, y también se han propuesto una amplia plétora de métodos para atacar la dificultad de obtener un buen desempeño de los algoritmos de clasificación en este tipo de conjuntos de datos.

En este artículo, se propone una corrección a un método publicado recientemente en [3], está basado en un árbol de decisión y en el concepto de entropía minoritaria. El error en dicha publicación fue encontrado al tratar de implementar el método, y se observaron las condiciones para las cuales sucede una contradicción en la definición de la entropía minoritaria. Al momento de la publicación de este artículo, el método propuesto solamente soporta conjuntos de datos con atributos numéricos, y se encuentra implementado en lenguaje de programación Java, además, continúa nuestra investigación con el objetivo de realizar una comparativa más justa entre el algoritmo de [3] y el propuesto, e incluir más resultados con métricas como F-measure, AUC ROC y PCR.

Los trabajos futuros de esta investigación son los siguientes:

Tabla 4. Resumen de resultados

Método	Recall	FP	TN	FN	Precisión
yeast-2_vs_4					
Propuesta	0.732	0.035	0.965	0.268	0.685
C4.5	0.771	0.040	0.960	0.229	0.636
vowel0					
Propuesta	0.921	0.012	0.988	0.079	0.882
C4.5	0.935	0.014	0.986	0.065	0.857
vehicle1					
Propuesta	0.530	0.176	0.824	0.470	0.456
C4.5	0.536	0.179	0.821	0.464	0.443
shuttle-c0-vs-c4					
Propuesta	0.988	0.000	1.000	0.012	0.998
C4.5	0.989	0.000	1.000	0.011	1.000
pima					
Propuesta	0.628	0.218	0.782	0.372	0.578
C4.5	0.643	0.222	0.778	0.357	0.560
page-blocks-1-3_vs_4					
Propuesta	0.948	0.008	0.992	0.052	0.884
C4.5	0.915	0.005	0.995	0.085	0.925
glass-0-1-6_vs_5					
Propuesta	0.426	0.040	0.960	0.574	0.238
C4.5	0.595	0.024	0.976	0.405	0.560
ecoli4					
Propuesta	0.826	0.020	0.980	0.174	0.672
C4.5	0.779	0.022	0.978	0.221	0.638

- Realizar una implementación paralela, distribuida o en GPU del método propuesto, para aplicaciones con datos grandes.
- Mejorar el método para poder procesar más tipos de atributos.
- Realizar una comparativa con otros métodos.
- Proponer otras reglas para particionar datos durante la construcción del árbol de decisión.
- Reducir el tiempo de entrenamiento.
- Realizar híbridos con métodos externos para mejorar el desempeño del clasificador.
- Realizar más pruebas y analizar profundamente los resultados obtenidos.

Agradecimientos. Los autores de este artículo agradecen a los revisores anónimos por sus valiosas observaciones para mejorar este trabajo. También, agradecen a la Universidad Autónoma del Estado de México, por el apoyo económico a través del proyecto de investigación 3790/2014/CID.

Referencias

1. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011), http://www.oldcitypublishing.com/MVLSC/MVLSCabstracts/MVLSC17.2-3abstracts/MVLSCv17n2-3p255-287Alcala.html
2. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor. Newsl. 6(1), 20–29 (Jun 2004), http://doi.acm.org/10.1145/1007730.1007735
3. Boonchuay, K., Sinapiromsaran, K., Lursinsap, C.: Minority split and gain ratio for a class imbalance. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on. vol. 3, pp. 2060–2064 (2011)
4. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: Smote: Synthetic minority oversampling technique. Journal of Artificial Intelligence Research 16, 321–357 (2002), http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a.pdf
5. Drummond, C., Holte, R.C.: C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. pp. 1–8 (2003)
6. Elkan, C.: The foundations of cost-sensitive learning. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2. pp. 973–978. IJCAI'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
7. Gong, W., Luo, H., Fan, J.: Extracting informative images from web news pages via imbalanced classification. In: Proceedings of the 17th ACM International Conference on Multimedia. pp. 1123–1124. MM '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1631272.1631529
8. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. on Knowl. and Data Eng. 21(9), 1263–1284 (Sep 2009), http://dx.doi.org/10.1109/TKDE.2008.239
9. Hulse, J.V., Khoshgoftaar, T.: Knowledge discovery from imbalanced and noisy data. Data & Knowledge Engineering 68(12), 1513 – 1542 (2009), http://www.sciencedirect.com/science/article/pii/S0169023X09001141
10. Japkowicz, N., Stephen, S.: The class imbalance problem: A systematic study. Intelligent Data Analysis 6(5), 429 (2002)
11. Luengo, J., Fernandez, A., Herrera, F., Herrera, F.: Addressing data-complexity for imbalanced data-sets: A preliminary study on the use of preprocessing for c4.5. In: Intelligent Systems Design and Applications, 2009. ISDA '09. Ninth International Conference on. pp. 523–528 (2009)
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
13. Seiffert, C., Khoshgoftaar, T., Van Hulse, J.: Hybrid sampling for imbalanced data. In: Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference on. pp. 202–207 (2008)
14. Weiss, G.M.: Mining with rarity: A unifying framework. SIGKDD Explor. Newsl. 6(1), 7–19 (Jun 2004), http://doi.acm.org/10.1145/1007730.1007734