

SMOTE-I: mejora del algoritmo SMOTE para balanceo de clases minoritarias

J Moreno¹, D Rodríguez¹, MA Sicilia, JC Riquelme² y R Ruiz³

¹Departamento de ciencias de la Computación
Campus Externo. Universidad de Alcalá, 287

Ctra. Barcelona km. 33.6, - 28871 Alcalá de Henares (Madrid)

²Escuela Técnica Superior de Ingeniería Informática
Avda. Reina Mercedes, s/n, 41012 – Sevilla, España

³Escuela Politécnica Superior
Universidad Pablo de Olavide
Ctra. Utrera Km 1, 41013 Sevilla

{daniel.rodriguezg,msicilia}@uah.es,robertoruiz@upo.es,riquelme@us.es

Resumen. Las técnicas de minería de datos están encaminadas a desarrollar algoritmos que sean capaces de tratar y analizar datos de forma automática con objeto de extraer de cualquier tipo de información subyacente en dichos datos. El problema del desbalanceo de los datos consiste en la predominancia de ciertos valores en los datos y la escasez o ausencia de otros que dificulta o impide la extracción de información. En este trabajo se presenta un nuevo algoritmo basado en SMOTE y llamado SMOTE-I que mejora al original con la la base de datos analizada.

Palabras clave: SMOTE, ENN, desbalanceo de datos.

1 Introducción

En un caso ideal todos los datos pertenecientes a cada clase se encuentran agrupados entre ellos y claramente diferenciables del resto de clases. La realidad es bien distinta y con frecuencia los datos presentan diferentes problemas que dificultan la labor de los clasificadores y disminuyen la calidad de la clasificación realizada. Los problemas más destacables son (i) el ruido es un problema derivado de la naturaleza de los datos consistente en el gran parecido que presentan entre sí datos pertenecientes a clases distintas o datos erróneos; (ii) el solapamiento entre clases ocurre cuando datos de clases distintas ocupan un espacio común debido a que algunos de los atributos de dichas clases comparten un mismo rango de valores; y (iii) el desbalanceo de clases ocurre cuando el número de instancias de cada clase es muy diferente

En estas circunstancias los clasificadores presentan una tendencia de clasificación hacia la clase mayoritaria, minimizando de ésta manera el error de clasificación y clasificando correctamente instancias de clase mayoritaria en detrimento de instancias de clase minoritaria. Entre las medidas que podemos tomar para el tratamiento del ruido se encuentran (i) usar algoritmos tolerantes al ruido; (ii) usar algoritmos que

reduzcan el ruido filtrando las instancias ruidosas y (iii) usar algoritmos que corrijan las instancias que generan el ruido. Para el tratamiento del solapamiento podemos utilizar algoritmos de “limpieza” que reduzcan las áreas de solapamiento. Finalmente, hay más opciones para el tratamiento del desbalanceo, entre las que se encuentran:

- *Sampling*: Consiste en balancear la distribución de las clases añadiendo ejemplos de la clase minoritaria. A ésta técnica se le denomina *oversampling*. Algunos de los algoritmos más representativos son SMOTE [1], y *Resampling* [5]. También es posible realizar lo contrario: eliminar ejemplos de la clase mayoritaria. Ésta técnica se conoce como *undersampling*. Un algoritmo bastante representativo es RUS [3]. Ambas técnicas tienen ventajas e inconvenientes. Entre los inconvenientes del *undersampling* está la pérdida de información que se produce al eliminar instancias de la muestra. Sin embargo tiene la ventaja de que reduce el tiempo de procesamiento del conjunto de datos. *Oversampling* tiene la ventaja de no perder información pero puede repetir muestras con ruido además de aumentar el tiempo necesario para procesar el conjunto de datos.
- *Oversampling*:
 - *SMOTE*: Genera nuevas instancias de la clase minoritaria interpolando los valores de las instancias minoritarias más cercanas a una dada.
 - *Resampling*: Duplica al azar instancias de la clase minoritaria
- *Undersampling*:
 - *Random undersampling*: Elimina al azar instancias de la clase mayoritaria
 - *Tomek Links* [4]: Elimina sólo instancias de la clase mayoritaria que sean redundantes o que se encuentren muy cerca de instancias de la clase minoritaria.
 - *Wilson Editing* [6]. También conocido como ENN (*Editing Nearest Neighbor*) elimina aquellas instancias donde la mayoría de sus vecinos pertenecen a otra clase.
- *Boosting* consiste en asociar pesos a cada instancia que se van modificando en cada iteración del clasificador. Inicialmente todas las instancias tienen el mismo peso y después de cada iteración, en función del error cometido en la clasificación se reajustan los pesos con objeto de reducir dicho error:
 - *AdaBoost*: Implementa el algoritmo de *Boosting* descrito. En cada iteración *AdaBoost* genera nuevas instancias utilizando *Resampling*
 - *SMOTEBoost*: Es similar a *AdaBoost* pero usa *SMOTE* en lugar del *Resampling* para generar nuevas instancias.
 - *RUSBoost*: Aplica *AdaBoost* pero en cada iteración utiliza *RUS* (*Random Undersampling*) que reducen el tamaño de la muestra de datos y simplifican y aumentan el rendimiento del clasificador.

2 Trabajos relacionados: SMOTE y ENN

Fix y Hodges [2] publicaron el algoritmo de la regla del vecino más cercano. La idea básica del algoritmo es suponer que instancias próximas entre sí tienen mayor probabilidad de pertenecer a la misma clase. Para clasificar una nueva instancia, se realiza un cálculo de la distancia entre cada atributo de la nueva instancia y el resto de instancias del conjunto de datos y se asocia a la clase de la instancia más cercana. El principal inconveniente del algoritmo es el alto coste computacional que tiene.

SMOTE (*Syntetic Minority Over-sampling Technique*) [1] es un algoritmo de *oversampling* que genera instancias “sintéticas” o artificiales para equilibrar la muestra de datos basado en la regla del vecino más cercano. La generación se realiza extrapolando nuevas instancias en lugar de duplicarlas como hace el algoritmo de *Resampling*. Para cada una de las instancias minoritarias se buscan las instancias minoritarias vecinas (más cercanas) y se crean N instancias entre la línea que une la instancia original y cada una de las vecinas. El valor de N depende del tamaño de *oversampling* deseado. Para un caso del 200% por cada instancia de la clase minoritaria deben crearse dos nuevas instancias genéricas.

SMOTE es un algoritmo de sobre-muestreo de ejemplos utilizado para la clase minoritaria:

- Crea ejemplos sintéticos en lugar de hacer un sobre-muestreo con reemplazo.
- Opera en el espacio de atributos *feature space*, en lugar del espacio de datos *data space*.
- Crea un ejemplo sintético a lo largo de los segmentos de línea que unen alguno o todos los k vecinos más cercanos de la clase minoritaria.
- Se eligen algunos de los k vecinos más cercanos de manera aleatoria (no se utilizan todos).
- SMOTE utiliza típicamente $k = 5$.

El algoritmo de SMOTE realiza los siguientes pasos:

- Recibe como parámetro el porcentaje de ejemplos a sobre-muestrear.
- Calcula el número de ejemplos que tiene que generar.
- Calcula los k vecinos más cercanos de los ejemplos de la clase minoritaria.
- Genera los ejemplos siguiendo este proceso:
 - Para cada ejemplo de la clase minoritaria, elige aleatoriamente el vecino a utilizar para crear el nuevo ejemplo.
 - Para cada atributo del ejemplo a sobre-muestrear, calcula la diferencia entre el vector de atributos muestra y el vecino elegido.
 - Multiplica esta diferencia por un número aleatorio entre 0 y 1.
 - Suma este último valor al valor original de la muestra.
 - Devuelve el conjunto de ejemplos sintéticos.

SMOTE es el algoritmo herramienta más utilizada para realizar el sobre muestreo pero presenta los siguientes inconvenientes: (i) puede generar muchos ejemplos artificiales cuyas semillas son ejemplos con ruido; (ii) al generar un nuevo ejemplo, interpola entre dos ejemplos de la clase minoritaria, sin embargo, pueden existir muchos ejemplos cercanos o inclusive entre ellos de la clase mayoritaria, generando modelos incorrectos; (iii) sólo funciona con variables continuas y (iv) no tiene una forma clara de decidir cuántos ejemplos generar. Para minimizar los inconvenientes

citados después de usar SMOTE pueden aplicarse otros algoritmos de limpieza para reducir el ruido y el solapamiento de las clases:

- *Tomek Links*: Elimina las instancias de las clases mayoritaria y minoritaria que estén muy próximas entre sí.
- *Wilson Editing* (ENN): Elimina cualquier instancia cuya etiqueta sea distinta de las clases que la rodean. ENN suele eliminar más instancias que Tomek Link.
- Además en éste trabajo se presenta una versión modificada SMOTE-I que ofrece una solución “automática” para balancear conjuntos de datos con más de una clase minoritaria aportando a la vez una sencilla solución al problema de determinar el número de ejemplos a generar sintéticamente.

El algoritmo de Edición de Wilson, también conocido como la regla del vecino más cercano editado (*Edited Nearest Neighbor*) elimina las instancias mal clasificadas de un conjunto de datos mediante la regla *k-NN*. Para identificar estas instancias el algoritmo elimina aquellas instancias cuya clase no coincide con la clase de la mayoría de sus vecinos. Este algoritmo elimina la superposición entre clases y por tanto parte de posible ruido que pueda haber en la muestra. El algoritmo de ENN realiza los siguientes pasos:

1. Recibe como parámetro el número de vecinos a buscar para cada clase
2. Calcula los *k* vecinos más cercanos de cada instancia.
3. Si la mayoría de los vecinos son de clase distinta y marca la instancia.
4. Elimina las instancias marcadas.

3 SMOTE-I

La definición original de SMOTE únicamente contempla datos en los que hay una única clase minoritaria. La modificación propuesta en éste trabajo permite *C* clases minoritarias ajustando automáticamente los pesos de cada clase para generar las instancias apropiadas de cada una de ellas. Adicionalmente también se ha añadido la opción de invertir los pesos de las clases minoritarias. Dado un conjunto de datos con un total de *T* instancias pertenecientes a *C* clases distintas, y una probabilidad de máxima para considerar cualquier clase como minoritaria *P*: Una clase *C_n* con *I_n* instancias, siendo $0 \leq n < C$ será minoritaria si su ocurrencia, $P_n = I_n/T$ es menor que *P*. El número de instancias sintéticas a generar para cada clase *C_n* minoritaria será igual al factor de sobremuestreo *N*. El número total de instancias sintéticas generadas *TS* será:

$$TS = \sum I_n * N$$

Como puede observarse, el número de instancias generadas en caso de haber más de una clase minoritaria será proporcional al número de instancias de cada clase. De ésta forma, se generarán más instancias sintéticas para las clases minoritarias con mayor número de instancias cuando lo deseable es justamente lo contrario: Generar mayor número de instancias sintéticas para las clases con menor número de instancias. Para realizar ésta inversión y teniendo en cuenta las definiciones anteriores: Definiremos la probabilidad invertida de cada instancia *PI_n* como:

$$PI_n = \frac{1 - P_n}{C - 1}$$

y determinaremos el número de instancias sintéticas a generar para cada clase C_n minoritaria, SN de la siguiente manera:

$$SN = \frac{PI_n \cdot T}{I_n}$$

La siguiente tabla muestra un ejemplo de un conjunto de 238 instancias distribuidas en 5 clases siendo $P = 50\%$:

I_n	P_n	$1 - P_n$	PI_n	$PI_n * T$	$(PI_n * T) - I_n$	SN	Nro instancias sintéticas por clase
12	0.05	0.95	0.24	56.5	44.5	4	48
21	0.09	0.91	0.23	54.25	33.25	2	42
23	0.10	0.90	0.23	53.75	30.75	1	23
59	0.25	0.75	0.19	44.75	-14.25	0	0
Clase mayoritaria:							
123	0.52						
Totales:							
238	1	4	1	238			113

Tabla 1. Ejemplo cálculos SMOTE-I

Si observamos la anterior tabla, la columna $PI_n * T$ y su total, $PI_n * T$ indica el número total de instancias que deberían generarse de cada clase y puesto que cada clase ya cuenta con I_n instancias, lo correcto es restar ese valor tal como muestra la columna $(PI_n * T) - I_n$. De ésta manera, redefinimos SN como:

$$SN = \frac{(PI_n * T) - I_n}{I_n}$$

La última columna de la tabla nos muestra el resultado de la inversión de probabilidades que favorece la creación de instancias sintéticas de las clases con menor número de instancias reales. En el apartado 5. Resultados Experimentales puede observarse la aplicación de éste algoritmo.

Resultados Experimentales

En éste proyecto se han utilizado un conjunto de datos obtenidos del repositorio de PROMISE1, llamado *usp5* y contienen clases nominales y ha sido utilizados para experimentar con el algoritmo SMOTE-I y comparar los resultados con el SMOTE original y ENN en problemas de clasificación. Más concretamente se ha utilizado

¹ <http://promisedata.org/>

C4.5 (implementado bajo el nombre de J48) en la herramienta Weka. J48 es un árbol de decisión cada nivel corresponde a una clase, cada nodo a un atributo y las ramas a los valores asociados a dichos atributos.

El conjunto de datos *usp5* de PROMISE contiene 15 atributos que representan información acerca métricas del esfuerzo en proyectos de ingeniería del software. Los atributos son los siguientes:

- *ID*: Identificador de tres dígitos
- *ObjType*: Tipo de objeto (*PJ-project*, *FT-feature*, *RQ-requirement*)
- *Effort*: Esfuerzo actual en horas gastadas en tareas relacionadas con la implementación del objeto por todas las personas participantes.
- *Funct%*: Porcentaje de funcionalidad desarrollada.
- *IntComplx*: Complejidad interna de cálculo.
- *DataFile*: Número de ficheros de datos/tablas accedidos.
- *DataEn*: Número de puntos de entrada de datos.
- *DataOut*: Número de puntos de salida de datos.
- *UFP*: Puntos de Función sin ajustar.
- *Lang*: Lenguaje de desarrollo utilizado.
- *Tools*: Herramientas de desarrollo y plataformas.
- *ToolExpr*: Experiencia de los desarrolladores con el lenguaje y las herramientas de desarrollo.
- *AppExpr*: Nivel de experiencia desarrollando aplicaciones.
- *TeamSize*: Tamaño del equipo de desarrollo.
- *DBMS*: Sistema gestor de bases de datos utilizado
- *Method*: Metodología de desarrollo utilizada.
- *AppType*: Tipo de sistema/arquitectura. Es la clase del *dataset*. Puede contener los siguientes valores: *B/S*, *C/S*, *BC/S*, *Centered*, *Other*

A continuación en las tablas 2 y 3 se presentan los resultados utilizando validación cruzada de 10 (10 CV).

Clase	J48	Resample	ENN	SM-I	ENN	
					Resamp	SM-I
BC/S	0,878	0,973	0,831	0,897	0,973	0,899
B/S	0,853	0,952	0,866	0,831	0,952	0,852
NULL	0,737	0,833	0,778	0,833	0,833	0,857
C/S	0,678	0,818	0,667	0,721	0,818	0,806
C	0,000	0,000	0,000	0,950	0,000	0,927
S	0,000	0,800	0,000	0,977	0,800	0,977
B	0,000	0,000	0,000	0,909	0,000	0,976
3-Tier_B/S	0,000	0,000	0,000	0,000	0,000	0,000
BS	0,667	0,800	0,667	1,000	0,800	0,952
BC	0,000	0,000	0,000	0,000	0,000	0,000
B/C	0,000	0,000	0,000	0,000	0,000	0,000

Tabla 2. Usp05, F-Measure clasificando con J48

Clase	J48	Resamp	ENN	SM-I	ENN	
					Resamp	SM-I
BC/S	0,936	0,981	0,915	0,955	0,981	0,973
B/S	0,984	0,999	0,979	0,996	0,999	0,997
NULL	0,985	0,926	0,984	0,961	0,926	0,964
C/S	0,872	0,930	0,870	0,885	0,930	0,960
C	0,690	?	0,680	0,972	?	0,974
S	0,808	0,997	0,804	0,998	0,997	0,998
B	0,433	0,995	0,408	0,998	0,995	1,000
3-Tier_B/S	0,463	?	0,456	0,475	?	0,474
BS	1,000	0,999	0,999	1,000	0,999	0,999
BC	0,463	?	0,459	0,480	?	0,478
B/C	0,443	?	0,424	0,472	?	0,470

Tabla 3. Usp05, A-ROC clasificando con J48

En el conjunto de datos utilizado con más de dos clases, en los resultados obtenidos se observa cómo a pesar de haber balanceado el conjunto de instancias mediante SMOTE el porcentaje de instancias clasificadas correctamente no aumenta significativamente, e incluso es inferior a los resultados obtenidos aplicando directamente el clasificador a los datos. Además, a pesar de obtenerse menor porcentaje de instancias clasificadas correctamente, los valores de *AUC* y *F-Measure* obtenidos después de aplicar SMOTE mejoran algo en relación a los resultados obtenidos sin aplicarlo. También se observa una mejora significativa en éstos valores al aplicar la versión SMOTE-I en conjunción con el clasificador J48.

5 Conclusiones y trabajo futuro

En este trabajo se ha presentado una modificación del conocido algoritmo de SMOTE. Este es uno de los problemas que presenta SMOTE ya que al interpolar entre dos instancias minoritarias es posible que entre ellas existan instancias de la clase mayoritaria dificultando así la labor del clasificador. La variante de SMOTE presentada en este trabajo tiene en cuenta la distribución de las clases y genera las instancias sintéticas acorde a esta distribución.

Los trabajos futuros deben ir encaminados al desarrollo de nuevos algoritmos de balanceo de datos y a mejorar los existentes como por ejemplo el estudio del número óptimo de instancias sintéticas a generar por SMOTE o SMOTE-I.

Agradecimientos

Esta investigación está financiada por el Ministerio de Educación y Ciencia (TIN2007-68084-C02-00) y las universidades de Alcalá, Sevilla y Pablo de Olavide.

Referencias

1. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer W.P. (2002). SMOTE: Syntetic Minority Over-sampling Technique. *Journal of Artificial Inteligence Research* 16, 321-357. Disponible en: <http://www.jair.org/media/953/live-953-2037-jair.pdf>
2. Fix, E. Hodges, J. (1952). Discriminatory analysis, nonparametric discrimination: small simple performance. Technical Report 11, US Air Force, School of Aviation Medicine, Randolph Field, TX.
3. Kamei, Y., Otros (2007). The Effects of Over and Under Sampling on Fault-prone Module Detection. *Empirical Software Engineering and Measurement*, 2007. First International Symposium on Volume, Issue, 20-21 Sept. 2007. Page(s)196-204. IEEE Computer Society Press, Los Alamos
4. Tomek , I. (1976), Two modifications of CNN, *IEEE Transactions on Systems, Man and Cybernetics* 6. IEEE Computer Society Press, Los Alamos.
5. Wilson , D. L. (1972), Asymptotic properties of nearest neighbor rules using edited data, *IEEE Transactions on Systems, Man and Cybernetics* . IEEE Computer Society Press, Los Alamos.
6. Witten, I. Frank, E. (2005), *Data Mining: Practical machine learning tools and techniques*, 2nd ed., Morgan Kaufmann, San Francisco