UNIVERSIDAD
DE GRANADA

Clasificación con Conjuntos de Datos No Balanceados
Laboratorio de Programación en R

Minería de Datos, Aspectos Avanzados

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Índice

1. Introducción

En esta práctica se pretende que el estudiante comprenda las implicaciones que tiene un conjunto con clases desequilibradas (imbalanced) en el rendimiento de los clasificadores estándar.

La práctica se divide en tres partes bien diferenciadas:

1. Una primera parte donde se analizará el comportamiento a nivel básico con varios conjuntos de datos sencillos mediante el software R (o python si así el estudiante lo desea). Este estudio se realizará durante la misma clase de prácticas.
2. Una segunda parte, también a realizar durante la sesión de prácticas, en la que se analizará el comportamiento de los algoritmos de SMOTE híbridos, a saber, SMOTE+ENN y SMOTE+TomekLinks en comparación con el SMOTE clásico.
3. A continuación se añaden dos ejercicios de carácter opcional:
    a. En primer lugar, se propone la implementación en R (o python) del algoritmo Borderline-SMOTE a partir del código inicial de SMOTE.
    b. En segundo lugar, se propone realizar un análisis relativamente exhaustivo de los parámetros del algoritmo SMOTE (número de vecinos, porcentaje de oversampling, enfoque de interpolación, etc.) para chequear su comportamiento en función de distintos valores.

Nota Importante: A lo largo de todo este guión de prácticas, las indicaciones se realizan para la solución implementada bajo R. Si decide guiarse por Python, tendrá que trabajar con el paquete "imbalanced-learn" de manera totalmente autónoma.

También es importante hacer notar que el guión de prácticas NO está diseñado como un tutorial "paso a paso", si no que se da por sentado la capacidad del estudiante para llevar a cabo las distintas tareas propuestas sin especificar todo el contenido.

2. Objetivos

Para esta sesión de laboratorio, se necesitará descargar desde PRADO2 el fichero `imbalanced.R`, además de los conjuntos de datos `subclus.txt` y `circle.txt`.

Para la evaluación, se deberán enviar a la actividad correspondiente de PRADO2 todos los ficheros R o Python que se hayan utilizado. Hay que asegurarse de que el código entregado esté suficientemente comentado.

También, será necesario proporcionar un análisis breve de los resultados obtenidos usando comentarios en los ficheros R o Python. Particularmente se valorará el uso de herramientas de generación de informes tipo KnitR.

La fecha de entrega para las diferentes actividades se indicarán una vez concluya el presente curso. Es importante hacer notar la "optatividad" de esta práctica, toda vez que la evaluación final se llevará a cabo mediante el trabajo sobre Deep Learning.

3. Análisis del efecto del desbalanceo en problemas de clasificación

En este ejercicio, se analizarán las estrategias a nivel de datos ROS, RUS y SMOTE para tratar con distribuciones de datos no balanceadas.

3.1 Preparación de datos y rendimiento básico de clasificación

Abra el fichero imbalanced.R. El código carga el conjunto de datos bidimensional subclus y lo visualiza. Calcula el ratio de imbalanceo (IR) y configura las particiones para un 5-folds cross validation (5-FCV), respetando el desbalanceo de clase en cada partición.

Finalmente, evalúe el rendimiento del clasificador base 3NN sobre este conjunto de datos por medio de la medida g-mean.

Hágalo también sobre el conjunto de datos circle.

3.2 Random Oversampling (ROS)

Estudie la implementación de la técnica de random oversampling (ROS) en el fichero imbalanced.R.

La aplicación de ROS debería obtener un conjunto de entrenamiento perfectamente equilibrado, duplicando aleatoriamente las instancias de la clase minoritaria.

Evalúe su rendimiento sobre el conjunto de datos subclus, por medio de un 5-FCV, utilizando el clasificador 3NN.

Hágalo también sobre el conjunto de datos circle.

3.3 Random Undersampling (RUS)

Estudie la implementación de la técnica de random undersampling (RUS) en el fichero imbalanced.R.

La aplicación de RUS debería obtener un conjunto de entrenamiento perfectamente equilibrado, borrando instancias de la clase mayoritaria de forma aleatoria.

Evalúe su rendimiento sobre el conjunto de datos subclus, por medio de un 5-FCV, utilizando el clasificador 3NN. Se visualizan las acciones de este técnica de preprocesamiento.

Hágalo también sobre el conjunto de datos circle.

3.4 Synthetic Minority Oversampling Technique (SMOTE)

3.4.1 Distancia

El fichero imbalanced.R contiene una función de distancia que calcula la distancia entre dos instancias en un conjunto de datos, teniendo en cuenta que algunas características son nominales (la distancia por característica para atributos nominales debería ser de 0 si las instancias tienen el mismo valor y de 1 en caso contrario).

3.4.2 getNeighbors

Escriba una función que encuentre los 5 vecinos de la clase minoritaria más cercano a un elemento, dados los índices de los elementos minoritarios y un conjunto de entrenamiento:

```
getNeighbors <- function(x, minority.instances, train){
      ...hacer algo  ...
      return( ...)
}
```

El algoritmo básicamente trataría de recorrer todos los ejemplos minoritarios de train (salvo el mismo ejemplo x), calculando la distancia entre x y cada uno de ellos. Almacenaría tanto las 5 distancias mínimas como el índice de los ejemplos a quiénes corresponden.

Puede extender la función para que busque un número K de ejemplos.

3.4.3 syntheticInstance

Implemente una función syntheticInstance que, dados los vecinos seleccionados de x, se escoja uno al azar y construya una instancia sintética en una posición aleatoria sobre el segmento de la línea entre x y su vecino. Para atributos nominales, la nueva instancia escoge aleatoriamente entre el valor de x o de su vecino. La función debería devolver la nueva instancia.

```
syntheticInstance <- function(x, neighbors, data){
      ...hacer algo  ...
      return( ...)
}
```

Consejo: considerar usar la función runif.

3.4.4 Evaluación

Poner en conjunto las funciones anteriores y usarla en la implementación del método SMOTE. Dicho método puede estar directamente implementado sobre el código (no en una función) como ROS y RUS.

Verificar su rendimiento sobre el conjunto de datos subclus y circle.

Proporcionar visualización.

4. Paquete 'unbalanced' y combinación de técnicas

Existe un paquete en CRAN llamado 'unbalanced' que implementa algunas de las técnicas más conocidas de preprocesamiento de datos para clasificación no balanceada. La documentación se puede encontrar en

https://cran.r-project.org/web/packages/unbalanced/unbalanced.pdf.

Utilizando el paquete 'unbalanced', se pide combinar dos técnicas de undersampling con SMOTE. La primera de ellas será la combinación SMOTE + TomekLinks y la segunda será SMOTE + ENN. Para ello, usa las funciones ubSMOTE, ubENN y ubTomek.

Verificar su rendimiento sobre el conjunto de datos subclus y circle. Proporcionar visualización.

También puedes seguir los pasos en el siguiente tutorial para una comparación de métodos de preprocesamiento más exhaustiva:

https://shiring.github.io/machine_learning/2017/04/02/unbalanced

5. Borderline-SMOTE *(parte optativa)*

Una de las extensiones más conocidas de SMOTE es el algoritmo Borderline-SMOTE (Han et al., ICIC 2005). Puedes encontrar la descripción del método aquí: http://sci2s.ugr.es/keel/keel-dataset/pdfs/2005-Han-LNCS.pdf.

Extender la implementación previa de SMOTE para que ahora se realice el proceso descrito en Borderline-SMOTE1. Para ello hay que tener en cuenta varios detalles:

- La función `getNeighbors` debe considerar ahora ejemplos minoritarios y mayoritarios.
- Considere también calcular los 5 vecinos más cercanos.
- Los puntos minoritarios ruidosos (aquellos cuyos vecinos cercanos son TODOS de la clase mayoritaria) no se consideran.
- Calcular el conjunto *DANGER* de ejemplos *borderline* descrito en el paper descriptivo. Serán aquellos en los que más de la mitad de sus vecinos más cercanos son de la clase mayoritaria.
- Continuar el proceso restante de *SMOTE* solo generando instancias artificiales de los puntos contenidos en *DANGER* hasta equilibrar el conjunto de datos.

Verificar su rendimiento sobre el conjunto de datos subclus y circle. Proporcionar visualización.

6. Análisis de los parámetros de SMOTE *(parte optativa)*

En esta última parte de la práctica se pretende analizar la influencia de los distintos parámetros de SMOTE. En efecto, durante las clases teóricas, se indicó que muchos de los parámetros de SMOTE podían tener cierta importancia de cara

a la calidad de los nuevos ejemplos sintéticos generados sobre el conjunto de entrenamiento.

Por ello, el objetivo de esta tarea es contrastar algunos de ellos para comprobar cuáles pueden tener más influencia, o cuáles pueden ser valores significativos para observar diferencias en los resultados.

Para llevar a cabo este objetivo, lo primero es determinar el marco experimental, que queda a disposición del alumno.

En cuanto a conjuntos de datos a utilizar, en principio se pueden seleccionar por defecto “subclus” y “circle”, si bien el estudio será más relevante cuando mayor sea el número de problemas, tanto sintéticos como reales. En cualquier caso, deberá utilizar una técnica de validación cruzada para chequear los resultados.
Como clasificador base para analizar el comportamiento, se puede utilizar por defecto kNN con K = 1 ó K = 3. También podría ser interesante analizar los resultados con el árbol de decisión C4.5.

Por último, quedaría por discutir qué parámetros son apropiados para el estudio, y qué rango de valores utilizar. Los parámetros más directos serían K para el número de vecinos escogidos (por ejemplo entre K = 1, K = 5, K = N/2 con N número de instancias positivas, etc.), y el porcentaje de oversampling (duplicar la clase minoritaria, 50-50 de ratio de clases, un 50% de clase minoritaria sobre mayoritaria, etc.).

Construye las tablas correspondientes de resultados y haz un breve análisis si observas algún patrón interesante en ellas.