UNIVERSIDAD DE GRANADA

UNIVERSIDAD DE GRANADA E.T.S.I. INFORMÁTICA Y TELECOMUNICACIÓN - Course 2019/ 2019

BIG DATA II

PRÁCTICA FINAL

Spark

Jaime cloquell Capp

Email:
jaumecloquell@correo.ugr.es

Contents

1 Introducción

En la actualidad, uno de los mayores desafıos y retos en tecnologıas de la información es el procesamiento eficiente de grandes cantidades de datos con el objeto de obtener información valiosa que ayude a la toma de decisiones en distintas áreas. Estas cantidades de datos son popularmente conocidas como big-data. Debido a que las técnicas y herramientars tradicionales no son capaces de hacer frente a problemas de big data, están apareciendo numerosas soluciones para el análisis y la gestión de los datos. Estas enormes cantidades de información también afectan a los métodos de mineria de datos tradicionales, que necesitan ser adaptados para tratar tales cantidades de datos.

Uno de los retos que dificulta la extración de conocimiento es el problema de clasificación sobre conjuntos de datos desbalanceados. Esta situación se produce cuando existe una desproporción notable en el número de ejemplos ertenecientes a cada clase. Este problema ha ganado mucha importancia en los últimos años ya que se encuentra presente en muchas aplicaciones reales tales como finanzas o diagnósis médica. En estos casos, el interés de los expertos se centra en la deteción de las clases menos representadas. Los problemas de big data también están afectados por este desbalanceo de clases.

Para tratar de abordar de forma eficiente problemas de big data han aparecido numerosas soluciones, siendo una de las más relevantes el modelo de programación denominado MapReduce. Este modelo de programación divide el conjunto de datos original en pequeños subconjuntos que son procesados en paralelo de forma independinete y a continuación combinados para obtener una solución final. No obstante, esta división de los datos puede provocar un efecto negativo en dominios desbalanceados. Entre los factores que pueden degradar el rendimiento en clasificación podemos encontrar el problema de la falta de densidad, relacionado con el tamaño del conjunto de entrenamiento. Este problema se amplifica cuando la clase minoritaria tiene una representación baja, ya que provoca la aparición de los small disjuncts cuando el conjunto de datos original se divide y distribuye por el procedimiento MapReduce.

2 Modelado

El conjunto de datos utilizado para los experimentos es un dataset y tiene dos clases, donde la mayoritaria representa el 80% de los datos y la minoritaria representa el 20%. Es decir, nos enfrentamos a un problema de clasificación binario desbalanceada.

A continuación explicaremos detalladamente los pasos que hemos seguido en el proceso.

2.1 Técnicas empleadas

En este capítulo vamos a profundizar en algunas de las técnicas que se han mencionado hasta ahora. Como hemos introducido previamente, muchas de ellas se basarán en la construcción de árboles de clasificación y en función del número de clasificadores que confeccione, podemos agrupar estos métodos en dos nuevas categorías: un clasificador o varios clasificadores. Respecto a los primeros, veremos algoritmos basados en la construcción de un ínico árbol de decisión, mientras que los segundos combinan

varios clasificadores básicos con el fin de obtener uno más preciso (multiclasificadores).

En el trabajo se han usado tres algoritmos base: árboles de decisión (clasificador), random forest (multiclasificador) y PCARD (multiclasificador). Para las comparativas, se ejecutaron estos algoritmos con y sin preprocesamiento previo.

2.2 Preprocesamiento

Cuando nos encontramos ante el problema de desbalanceo de clases, una de las soluciones más intuitivas es la modificación de la distribución de los datos, en particular, incrementar el número de casos de la clase minoritaria, para que el modelo sea capaz de detectarlas y que pueda predecir un número razonable de valores de la clase minoritaria.

En la siguiente tabla recogemos algunas conclusiones obtenidas en este artículo **[2]** sobre cada uno de los métodos de desbalanceo que nos permiten razonar los motivos por los cuales han sido seleccionados para este trabajo.

Técnica	Ventaja	Desventaja
Undersampling	Menor tiempo de procesado	Pérdida de información
Oversampling	No hay pérdida de información	Repetición de muestras ruidosas y mayor tiempo de procesado

Table 1: Ventajas e Inconvenientes de las técnicas de desbalanceo

Dado que el objetivo del problema es predecir con mayor certeza los valores tanto de la clase mayoritaria como minoritaria, independeiente del coste computacional, en un principio nos decantaremos por utilizar oversampling a no ser que veamos que undersampling se comporta mejor con los datos.

Además de aplicar técnicas de desbalanceo, se han aplicado otros tipos de técnicas tales como:

- Filtro de ruido
- Selección de instancias
- Desbalanceo
- Normalización

2.3 Método de evaluación de los modelos

Para medir el rendimiento de un modelo clasificador debemos fijarnos en la tasa de error cometido por el mismo, es decir, tendremos que calcular el porcentaje de casos clasificados de forma incorrecta sobre el conjunto de datos con los que estamos trabajando.
Para medir la habilidad haciendo predicciones de los clasificadores se hace uso de una tabla conocida como la **matriz de confusión**, que constituye la herramienta de visualización más empleada en aprendizaje supervisado. Su papel es comparar las predicciones del clasificador con los valores reales.

En nuestro caso, compararemos las clases que el modelo predice para cada clase (positivo o negativo) con el valor real.

La matriz de confusión una vez hechas las predicciones puede parecer algo insuficiente a la hora de medir la eficiencia de un modelo clasificando datos. Una buena manera de explicar esto sería obtener una serie de medidas cuantitativas que nos permitan juzgar con argumentos sólidos el buen comportamiento de los clasificadores. A partir de los valores de la matriz de confusió podemos calcular los valores finales de Precision, Recall, Specifity y Accuracy.
El problema de estas medidas es que ninguna de estas métricas por sí misma, individualmente, es representativa del poder predictivo del clasificador. En problemas complejos como es nuestro caso un clasificador aumentará el número de True Positives (TP) a costa de incrementar también el de False Positives (FP) –no al mismo ritmo. Lo que se busca es un clasificador que sea capaz de incrementar TP a un ritmo (mucho) mayor que FP. Por tanto una vez calculados los valores anteriores, se puede calcular una medida que nos dará una idea de la eficiencia global del sistema. En nuestro caso nos basaremos en **Matthews Correlation Coefficient (MCC) [1]**

MCC es usado generalmente en aprendizaje automático como una medida de la calidad de un clasificador binario. Toma en cuenta los valores positivos verdaderos y positivos falsos y los negativos, y es considerado generalmente como una medida balanceada que puede ser usada incluso si las clases tienen tamaños muy diferentes. El coeficiente MCC es en esencia un coeficiente de correlación entre las clasificaciones binarias observadas y predichas.

3 Resultados

Una vez conocida la metodología y el funcionamiento de cada uno de los modelos, mostraremos los resultados obtenidos por los mismos con el fin de poder decantarnos por alguno de ellos. Con esto no nos referimos sólo al aspecto numérico, sino también a otras cuestiones que han ido surgiendo durante el análisis de las técnicas de desbalanceo abordadas en este trabajo y que a continuación explicamos.

3.1 Algoritmos de desbalanceo

En primer lugar empezamos ejecutando los métodos de aprendizaje sin ningún preprocesamiento, solo que algoritmos de desbalanceo para así tener un punto de partida y conocer que porcentaje de acierto tendremos como referencia a la hora de aplicar métodos preprocesamiento y conocer si mejora o no el modelo respecto al modelo base.
En la tabla 2 se recogen los resultados obtenidos para las distintas pruebas realizadas en los algoritmos de desbalanceo. En la primera fila contamos con las soluciones que proporciona el modelo PCARD, Random Forest y Decision Tree.

Los algoritmos de desbalanceo empleado son dos: ROS, RUS. Además, para aplicar ROS, probaremos con diferentes porcentajes de ratio, que irán desde 20 hasta 140, lo que supone casi duplicar el número de instancias de la clase minoritaria.

Hemos de mencionar que la notación empleada en la tabla 2 corresponden en primer lugar el preprocesamiento elaborado, seguido por el porcentaje de acierto los valores de la matriz de confusión y finalmente hemos indicado las medidas de sensibilidad, la especificidad y la precisión. Esta estructura se mantendra a lo largo del trabajo.

Modelo	**Preprocesamiento**	**MCC**	**Sensitivity**	**Specificity**	**Precision**
PCARD	-	0.1216	**0.9985**	0.0272	0.9023
PCARD	ROS[40]	0.3618	0.8859	0.5541	0.9470
PCARD	ROS[60]	**0.3657**	0.8074	0.7169	0.9625
PCARD	ROS[80]	0.3646	0.7591	0.7958	0.9710
PCARD	ROS[100]	0.3492	0.7110	0.8422	0.9759
PCARD	ROS[120]	0.3384	0.6769	0.8699	0.9791
PCARD	ROS[140]	0.3285	0.6459	**0.8922**	**0.9818**
PCARD	RUS	0.3468	0.7071	0.8436	0.9760
RF	ROS[40]	0.3476	0.9489	0.3225	0.8645
RF	ROS[60]	0.3475	0.9492	0.3208	0.8625
RF	ROS[80]	0.3430	0.9469	0.3259	0.8714
RF	ROS[100]	0.3385	0.9720	0.2432	0.7187
RF	ROS[120]	0.3399	0.9715	0.2453	0.7234
RF	ROS[140]	0.3395	0.9712	0.2456	0.7246
RF	RUS	0.3395	0.9722	0.2437	0.7191
DT	ROS[40]	0.3390	0.9439	0.3354	0.8839
DT	ROS[60]	0.3463	0.9531	0.3031	0.8394
DT	ROS[80]	0.3268	0.9741	0.2297	0.6887
DT	ROS[100]	0.3268	0.9740	0.2299	0.6896
DT	ROS[120]	0.3126	0.9793	0.2108	0.6342
DT	ROS[140]	0.3101	0.9807	0.2073	0.6220
DT	RUS	0.3247	0.9746	0.2273	0.6828

Table 2: Resultados de los algoritmos de desbalanceo

Tras los resultados mostrados en la tabla 2, podemos comentar varios aspectos sobre los algoritmos de desbalanceo.

- La máxima precisión obtenida ocurre cuando aplicamos ROS con un ratio con un de 140, aunque existen casos en los que se han conseguido clasificadores muy precisos balanceando las clases con ROS con un ratio inferior incluso aplicando RUS.
- En general, la sensibilidad de los clasificadores es baja ya que al haber pocos casos de la clase minoritaria en la muestra, la capacidad para detectarlos por el clasificador será también escasa.
- Con undersampling empeoran los resultados base en cuanto a precisión tanto en PCARd o RF, por lo que posiblemente estemos eliminando muestras con información discriminatoriarelevante para la clasificación.

- En cuanto a oversampling, podemos observar en la tabla 2 que su aplicación previa con un ratio oversampling de 60 provoca mejoras significativas en cuanto a rendimiento para PCARD con respecto a no aplicar este preprocesamiento al algoritmo, mientras que para random Forest va mejor un oversampling rate de 40.
- RF ofrece valores muy buenos en sensitivity a cambio de perder specificty.

Tras los resultados obtenidos en este ensayo, se han recomendado ciertos límites para proceder con el desbalanceo de los datos. En general, se propone no modificar la distribución de los datos de forma que se supere el 60 de ratio. Esto puede derivar en métricas desfavorables, como por ejemplo, el número de predicciones obtenidas es excesivamente elevado o se obtiene un clasificador con un valor bajo en sensitivity.

Basandonos en los resultados obtenido podemos concluir que aplicar oversampling con un ratio 60, es el que aporta mayor equilibrio entre TP y FP. Los resultados obtenidos por la función PCARD superaban, en general, a los obtenidos por random Forest y el árbol de decisión. Por tanto en las posteriores pruebas, PCARD será el modelo por defecto para aplicar los preprocesado correspondiente.

3.2 Algoritmos de preprocesamiento

En esta sección discutiremos los resultados proporcionados en función de los distintos técnicas de preprocesado que hemos aplicado.

Modelo	**Preprocesamiento**	**MCC**	**Sensitivity**	**Specificity**	**Precision**
PCARD	ROS[60]	0.3657	0.8074	0.7169	0.9625
PCARD	ROS[60] + Filtro de ruido	0.3672	0.8045	0.7245	0.9633
PCARD	ROS[60] + Selección de instancias	**0.3700**	0.8259	0.6896	0.9599
PCARD	ROS[60] + Filtro de ruido + Selección de instancias	0.3638	0.8084	0.7122	0.9619

Table 3: Resultados del algoritmo PCARD + ROS[60] con distintos preprocesamientos.

4 Conclusiones

Después de haber efectuado todas las pruebas pertinentes sobre los modelos de clasificación y las técnicas de desbalanceo y preprocesamiento desarrolladas en este trabajo, podemos elaborar una serie de conclusiones:

En general aplicando previamente undersampling empeoran los resultados base en cuanto a precisión con RF, por lo que posiblemente estemos eliminando muestras con información discriminatoria relevante para la clasificación.

En cuanto a ROS, podemos observar en la tabla de resultados que su aplicación previa con un ratio oversampling de 60 provoca mejoras significativas en cuanto a rendimiento para todos los modelos con

respecto a aplicar este preprocesamiento con distintos valores de ratio. Para Random Forest va mejor un oversampling rate de 120 y para Decision Tree con ninguno de los oversampling rates probados se consigue un incremento en precisión de clasificación.

En general, el algoritmo de preprocesamiento que consigue elevar el valor precision de PCARD es HME (Filtro de ruido), mientras que prácticamente iguala a al valor sensitivity base obtenida para Pcard, sin embargo, este algoritmo obtiene una tasa MCC inferior a la obtenida con SSMASFLSDE (selección de instancias).

References

[1] Wikipedia *Matthews correlation coefficient*
`https://en.wikipedia.org/wiki/Matthews_correlation_coefficient`

[2] Adnan Amin, Sajid Anwar *Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study*
`https://ieeexplore.ieee.org/abstract/document/7707454`