Teoria-Resuelta-Examen-Enero-202...

user_2335920

Especialidad: Sistemas de Información

4º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Accede al documento original

(3 ptos.) En un problema de clasificación con tres clases, donde la clase minoritaria es 10 veces menos frecuente que la clase mayoritaria, responde justificadamente a las siguientes cuestiones:

a) ¿Qué medida emplearías para valorar el acierto de un clasificador?

En un problema de clasificación con clases desbalanceadas, las métricas tradicionales como **Precisión** o **Exactitud** no son adecuadas, ya que pueden ser engañosas al estar dominadas por la clase mayoritaria. En su lugar, se recomienda usar métricas que consideren explícitamente el rendimiento en la clase minoritaria, como:

- **F1-Score**: Combina precisión y recall, siendo útil para evaluar el balance entre falsos positivos y falsos negativos.
- **AUC-ROC (Área bajo la curva ROC)**: Evalúa la capacidad del modelo para discriminar entre clases, ignorando el desbalance.
- **G-Mean (Media geométrica entre TPR y TNR)**: Es útil porque mide el equilibrio entre la tasa de verdaderos positivos y la tasa de verdaderos negativos.

De estas, elegiría **F1-Score** si el enfoque está en maximizar el rendimiento en la clase minoritaria o **G-Mean** si se busca un equilibrio entre ambas clases.

b) ¿Qué método usarías para comparar la eficacia de diferentes algoritmos de clasificación?

Para comparar algoritmos de clasificación en un escenario de desbalance de clases, utilizaría:

- **Validación cruzada estratificada (Stratified K-Fold Cross-Validation)**: Garantiza que la proporción de clases en cada partición sea consistente con la proporción global, lo cual es fundamental para evitar sesgos en problemas de desbalance.
- Para la evaluación, utilizaría **curvas ROC y AUC**, o una combinación de métricas como el **F1-Score** y **G-Mean**, dependiendo de los objetivos específicos del problema. Al comparar los algoritmos, se pueden usar pruebas estadísticas como **Wilcoxon Signed-Rank Test** para determinar si las diferencias observadas son estadísticamente significativas.

c) ¿Qué solución propondrías para mejorar la capacidad predictiva de los clasificadores generados por un algoritmo de aprendizaje de árboles de decisión?

Para mejorar la capacidad predictiva de los árboles de decisión en problemas con clases desbalanceadas:

1. **Muestreo**:
    - **Oversampling** de la clase minoritaria (como SMOTE).
    - **Undersampling** de la clase mayoritaria.
    - Uso de combinaciones (por ejemplo, Tomek Links con SMOTE).

2. **Pesos en las clases**: Configurar pesos inversamente proporcionales a la frecuencia de las clases, haciendo que el algoritmo penalice más los errores en la clase minoritaria.
3. **Pruning (Poda)**: Aplicar técnicas de poda para reducir el sobreajuste, que es común en árboles profundos.
4. **Ajuste de hiperparámetros**: Optimizar parámetros como la profundidad máxima del árbol, el número mínimo de muestras por nodo o el criterio de división para mejorar el equilibrio entre precisión y recall.

d) ¿Y de un algoritmo ensemble learning?

Para un algoritmo de ensemble learning (como Random Forest o Gradient Boosting), las siguientes estrategias pueden mejorar la capacidad predictiva:

1. **Pesos en las clases**: Similar a los árboles individuales, asignar pesos a las clases para que los errores en la clase minoritaria sean penalizados más. Esto es especialmente útil en algoritmos como Random Forest y XGBoost.
2. **Muestreo**:
    - Realizar oversampling o undersampling en cada bootstrap del ensemble para equilibrar las clases durante la construcción de cada modelo base.
    - Algunos algoritmos como XGBoost y LightGBM tienen opciones integradas para manejar clases desbalanceadas (e.g., `scale_pos_weight`).
3. **Ensembles adaptativos**:
    - Usar algoritmos como **AdaBoost**, que ajustan los pesos de las instancias mal clasificadas, permitiendo enfocarse en la clase minoritaria.
4. **Threshold tuning**: Ajustar el umbral de decisión para mejorar el recall de la clase minoritaria sin perjudicar demasiado la precisión.
5. **Stacking**: Combinar diferentes algoritmos de ensemble para aprovechar sus puntos fuertes, como Random Forest y Gradient Boosting, ajustados específicamente para manejar desbalance de clases.

Estas soluciones no solo mejoran el rendimiento en problemas desbalanceados, sino que también aseguran que el modelo sea más generalizable.

(2 ptos.) Explica en qué consiste la clasificación multi-etiqueta, el aprendizaje multi-instancia y el aprendizaje semisupervisado. En cada caso, propón un ejemplo de resolución de un problema apropiado para él.

1. Clasificación Multi-Etiqueta

La clasificación **multi-etiqueta** es un problema de aprendizaje supervisado donde cada instancia (ejemplo) puede estar asociada a **más de una clase** simultáneamente. A diferencia de la clasificación tradicional (donde cada instancia pertenece a una sola clase), aquí una instancia puede tener múltiples etiquetas activas.

Problema: Clasificación de películas en géneros.
Una película puede pertenecer a múltiples géneros, como "Acción", "Ciencia Ficción" y "Aventura" al mismo tiempo.
Solución: Usar un clasificador multi-etiqueta como **Binary Relevance** (creando un modelo para cada etiqueta) o técnicas avanzadas como **ML-KNN** (Multi-Label k-Nearest Neighbors).

2. Aprendizaje Multi-Instancia

El aprendizaje **multi-instancia** es un enfoque en el que los datos están organizados en bolsas (*bags*), y cada bolsa contiene múltiples instancias. Una etiqueta se asigna a toda la bolsa, no a las instancias individuales. El objetivo es predecir la etiqueta de una bolsa basándose en sus instancias.

Problema: Identificación de tumores en imágenes médicas.
Una imagen (la bolsa) puede contener varias regiones (instancias). Si al menos una región contiene tejido canceroso, la imagen se etiqueta como positiva para cáncer.
Solución: Usar algoritmos diseñados para aprendizaje multi-instancia, como **MI-SVM** (Support Vector Machines adaptadas) o **APR** (Axis-Parallel Rectangle).

3. Aprendizaje Semisupervisado

El aprendizaje **semisupervisado** es una técnica que combina datos **etiquetados** y **no etiquetados** para entrenar un modelo. Esto es útil cuando el etiquetado de datos es costoso o laborioso, pero hay gran disponibilidad de datos no etiquetados.

Problema: Clasificación de correos como "spam" o "no spam".
Supongamos que tenemos un pequeño conjunto de correos etiquetados y un gran conjunto de correos no etiquetados.
Solución: Usar un algoritmo semisupervisado, como el **Self-Training**, que utiliza el modelo inicial entrenado en datos etiquetados para predecir etiquetas en datos no etiquetados y mejorar su rendimiento con iteraciones.