Boosting

Analítica de Datos, Universidad de San Andrés

Si encuentran algún error en el documento o hay alguna duda, mandenmé un mail a rodriguezf@udesa.edu.ar y lo revisamos.

1. Introducción al Boosting

Boosting es una técnica de ensamble que combina múltiples modelos "débiles"para crear un modelo "fuerte". A diferencia de Random Forest, que construye árboles independientes en paralelo, Boosting construye modelos secuencialmente, donde cada nuevo modelo intenta corregir los errores de los modelos anteriores.

Los principios fundamentales del Boosting son:

- Construcción secuencial de modelos
- Ponderación de observaciones
- Combinación ponderada de predicciones
- Enfoque en errores previos

2. Algoritmo AdaBoost

AdaBoost (Adaptive Boosting) fue desarrollado en 1995 por Yoav Freund y Robert Schapire. Fue el primer algoritmo de Boosting exitoso. Sus principales características son:

- Inicialmente, todas las observaciones tienen el mismo peso
- En cada iteración:
    - Se entrena un modelo débil
    - Se identifican las observaciones mal clasificadas
    - Se aumenta el peso de las observaciones mal clasificadas
    - Se disminuye el peso de las observaciones bien clasificadas

- La predicción final es un promedio ponderado de las predicciones de todos los modelos

3. Gradient Boosting

Gradient Boosting es una generalización del Boosting que:

- Utiliza el gradiente del error para guiar el aprendizaje porque permite identificar la dirección de máximo descenso en la función de pérdida
- Puede aplicarse a cualquier función de pérdida diferenciable ya que utiliza el cálculo diferencial para minimizar el error, lo que lo hace versátil para diferentes tipos de problemas
- Es más flexible y potente que AdaBoost debido a que no se limita a clasificación binaria y puede manejar diferentes funciones objetivo
- Permite mayor control sobre el proceso de aprendizaje al poder ajustar parámetros como la tasa de aprendizaje y la profundidad de los árboles

3.1. Parámetros Importantes

- **n_estimators:** Número de árboles (iteraciones)
- **learning_rate:** Tasa de aprendizaje (shrinkage)
- **max_depth:** Profundidad máxima de los árboles
- **subsample:** Fracción de datos para cada árbol
- **min_samples_split:** Mínimo de muestras para dividir

4. XGBoost

XGBoost (eXtreme Gradient Boosting) es una implementación optimizada de Gradient Boosting que ofrece:

- Mayor velocidad de entrenamiento: Utiliza técnicas de paralelización y optimización de memoria que permiten procesar grandes volúmenes de datos más rápidamente que otras implementaciones de Gradient Boosting.

- Mejor manejo de memoria: Implementa estructuras de datos especializadas y técnicas de compresión que reducen significativamente el uso de memoria durante el entrenamiento.
- Regularización incorporada: Incluye regularización L1 (Lasso) y L2 (Ridge) directamente en el algoritmo, lo que ayuda a prevenir el overfitting y mejora la generalización del modelo.
- Manejo de valores faltantes: Puede manejar automáticamente valores faltantes en los datos sin necesidad de preprocesamiento adicional, lo que simplifica el pipeline de machine learning.
- Paralelización eficiente: Aprovecha múltiples núcleos de CPU y puede distribuir el trabajo entre diferentes máquinas, lo que permite escalar el entrenamiento a conjuntos de datos muy grandes.

4.1. Características Principales

- **Regularización:** Implementa regularización L1 (Lasso) para selección de características y L2 (Ridge) para control de complejidad
- **Early Stopping:** Monitorea el rendimiento en conjunto de validación y detiene el entrenamiento cuando no hay mejora en un número especificado de iteraciones
- **Feature Importance:** Proporciona métricas detalladas sobre la contribución de cada variable al modelo final
- **Manejo de Datos Dispersos:** Utiliza estructuras de datos especializadas para procesar eficientemente matrices sparse
- **Paralelización:** Permite entrenamiento distribuido y procesamiento en múltiples núcleos
- **Monitoreo de Progreso:** Ofrece herramientas para visualizar el progreso del entrenamiento y métricas de rendimiento

5. Comparación con Random Forest

Característica	Random Forest	Boosting
Construcción	Paralela	Secuencial
Velocidad	Más rápido	Más lento
Overfitting	Menos propenso	Más propenso
Ajuste	Menos parámetros	Más parámetros
Interpretabilidad	Mayor	Menor

6. Implementación en Python

```python
# AdaBoost
from sklearn.ensemble import AdaBoostClassifier
ada_model = AdaBoostClassifier(
    n_estimators=100,
    learning_rate=1.0
)

# Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier
gb_model = GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3
)

# XGBoost
import xgboost as xgb
xgb_model = xgb.XGBClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    reg_lambda=1,  # L2
    reg_alpha=0    # L1
)

# Entrenamiento
model.fit(X_train, y_train)

# Prediccion
y_pred = model.predict(X_test)
```

7. Buenas Prácticas

- **Preprocesamiento:** Escalar variables numéricas y codificar categóricas
- **Validación Cruzada:** Usar para evaluar y ajustar hiperparámetros
- **Early Stopping:** Implementar para evitar overfitting
- **Learning Rate:** Comenzar con valores pequeños (0.01-0.1)
- **Monitoreo:** Observar error de entrenamiento y validación