BLOQUE 2

MÓDULO:
TÉCNICAS DE MACHINE LEARNING

ALGORITMOS ENSAMBLADOS

LORENZO MARTÍNEZ MANERO

Ingeniero Industrial Superior por la Universidad Politécnica de Valencia.

Institut de Formació Contínua-IL3
UNIVERSITAT DE BARCELONA

ÍNDICE

OBJETIVOS ESPECÍFICOS

TEMA 1. RANDOM FOREST

- Comprender su funcionamiento interno.
- Entender los diferentes parámetros y cómo usarlos.
- Distinguir las formas en que se produce la predicción, cuando es Regresión o Clasificación.
- Aprender a calcular la importancia de las variables y su significado.
- Aplicar este tipo de algoritmos a diferentes tipos de datos usando la librería Scikit-Learn de Python.
- Comprender cuándo vale la pena usarlos, sus ventajas y sus desventajas.

TEMA 2. GRADIENT BOOSTING

- Comprender su funcionamiento interno.
- Entender los diferentes parámetros y cómo usarlos.
- Distinguir las formas en que se produce la predicción, cuando es Regresión o Clasificación.
- Conocer las diferentes formas de Parada Temprana o Early Stopping.
- Saber las diferencias y los parecidos respecto de Random Forest.
- Aplicar este tipo de algoritmos a diferentes tipos de datos usando la librería Scikit-Learn de Python.
- Comprender cuándo vale la pena usarlos, sus ventajas y sus desventajas.

TEMA 3. ADABOOST

- Comprender su funcionamiento interno.
- Entender los diferentes parámetros y cómo usarlos.
- Distinguir las formas en que se produce la predicción, cuando es Regresión o Clasificación.
- Aprender cómo calcula el algoritmo los pesos asociados a las instancias.
- Aprender cómo calcula el algoritmo los pesos asociados a las predicciones de cada árbol de decisión.
- Aplicar este tipo de algoritmos a diferentes tipos de datos usando la librería Scikit-Learn de Python.
- Comprender cuándo vale la pena usarlos, sus ventajas y sus desventajas.

ALGORITMOS ENSAMBLADOS

En el tema 1 de este módulo, hemos visto tres algoritmos supervisados:

- Árboles de Decisión.
- KNN.
- Naive Bayes.

En este segundo tema, nos vamos a centrar en el primer algoritmo de los que hemos nombrado arriba.

Debido a la facilidad de uso y a su funcionamiento interno, los Árboles de Decisión tienen una utilidad que va más allá de lo visto en el primer tema.

Se ha podido constatar que si usamos muchos Árboles de Decisión y los "ensamblamos" de ciertas formas podemos conseguir otros algoritmos más potentes en su predicción.

En los tres temas siguientes veremos tres formas diferentes de combinar los Árboles de Decisión:

- **Random Forest:** donde se utiliza, con algunas mejoras, una técnica denominada *Bootstrap Aggregation* o *Bagging.*
- **GradientBoosting:** que se basa en una técnica denominada Boosting.
- **AdaBoost:** que también se basa en Boosting pero con ciertas peculiaridades frente a los anteriores.

En Bagging se ensamblan los Árboles de Decisión lanzándolos en paralelo (lo que redundará en algoritmos robustos frente al overfitting).

Boosting, por el contrario, los ensambla en serie, de manera que cada uno trata de mejorar al anterior (lo que nos lleva a algoritmos con una gran potencia predictiva).

A partir de este punto, vamos a trabajar con Colab donde alternaremos teoría con ejemplos programados en Python.

IDEAS CLAVE

TEMA 1: RANDOM FOREST

- La importancia de los algoritmos ensamblados.
- La robustez de los modelos de Random Forest frente al overfitting
- Cómo funcionan los procesos aleatorios tanto en filas (instancias) como en columnas (variables).
- El cálculo de las predicciones atendiendo a si estamos ante un problema de Regresión o de Clasificación.
- El parámetro "max_features" para controlar el modelo final.
- El parámetro "random_state" para conseguir un modelo reproducible.
- Puesta en funcionamiento con la librería Scikit-Learn de Python.

TEMA 2: GRADIENT BOOSTING

- La potencia en la precisión de los modelos basados en Boosting.
- Gradient Boosting como algoritmo en serie de corrección de errores.
- Cómo utilizar los parámetros para obtener un buen rendimiento sin llegar al overfitting.
- El cálculo de las predicciones atendiendo a si estamos ante un problema de Regresión o de Clasificación.
- El parámetro "Learning Rate" para controlar su capacidad de aprendizaje.
- El parámetro "n_estimators" para controlar la profundidad del modelo (rendimiento vs overfitting).
- Puesta en funcionamiento con la librería Scikit-Learn de Python.

TEMA 3: ADABOOST

- AdaBoost como algoritmo en serie de corrección de errores, entendiendo la diferencia de funcionamiento respecto a Gradient Boosting.
- Cómo utilizar los parámetros para obtener un buen rendimiento sin llegar al overfitting.
- El cálculo de las predicciones atendiendo a si estamos ante un problema de Regresión o de Clasificación.
- El parámetro "Learning Rate" para controlar su capacidad de aprendizaje.

- El parámetro “max_depth” para controlar la profundidad de los diferentes árboles de decisión (stumps).
- Puesta en funcionamiento con la librería Scikit-Learn de Python.