Ensemble Learning - Boosting: AdaBoost y XGBoost

Tomás Fontecilla

23 de septiembre de 2022

Ensemble Learning

La unión hace la fuerza

Los métodos de aprendizaje por ensamblaje o ensembling learning son utilizados para mejorar resultados al combinar *distintos modelos*.
Este enfoque permite la producción de un desempeño mejor en la predicción que un modelo único.

Ensemble Learning

Idea Básica

Aprender de un conjunto de modelos de clasificación para que éstos puedan votar.

Dos algoritmos (aunque no los únicos) de ensamblaje son los que vimos la clase pasada: **bagging** y **boosting**.
Estos algoritmos disminuyen la **varianza** de un estimador único al combinar varias estimaciones de diferentes modelos. Así el resultado puede ser un modelo de mayor estabilidad (disminuyendo la posibilidad de *sobreajuste*).

Ensembling Learning - Bagging y Boosting

De la clase pasada:

Bagging

También llamado *boostrap aggregation* es una técnica para reducir la varianza de una función de predicción estimada.

Boosting

Secuencialmente aplica un algoritmo de clasificación débil a versiones modificadas repetidamente de los datos para producir una secuencia de clasificadores débiles. Estas predicciones son combinadas para crear una mayoría de votos ponderado para producir una predicción final.

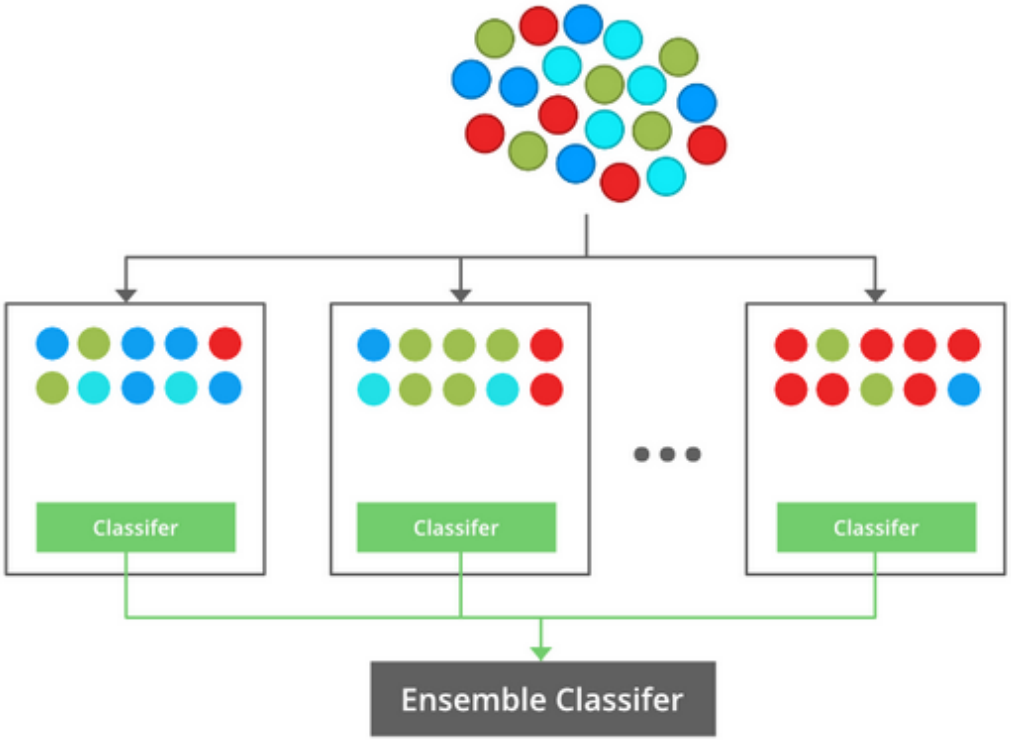
Ensemble - Bagging

Bootstrapping es la técnica de aumentar los datos al hacer muestreo con reemplazo sobre los datos originales y ajustar sobre ellos modelos de machine learning sea de clasificación o regresión.

Algoritmo:

- Generar múltiples subconjuntos de los datos originales con reemplazo
- Crear un modelo base para cada uno de los subconjuntos
- Cada modelo "aprende" en paralelo con cada conjunto de entrenamiento y es independiente de los otros
- Determinar predicciones para cada modelo y combinar esas predicciones con los otros.

Ensemble Learning - Bagging en gráfico



Original Data

Bootstrapping

Aggregating

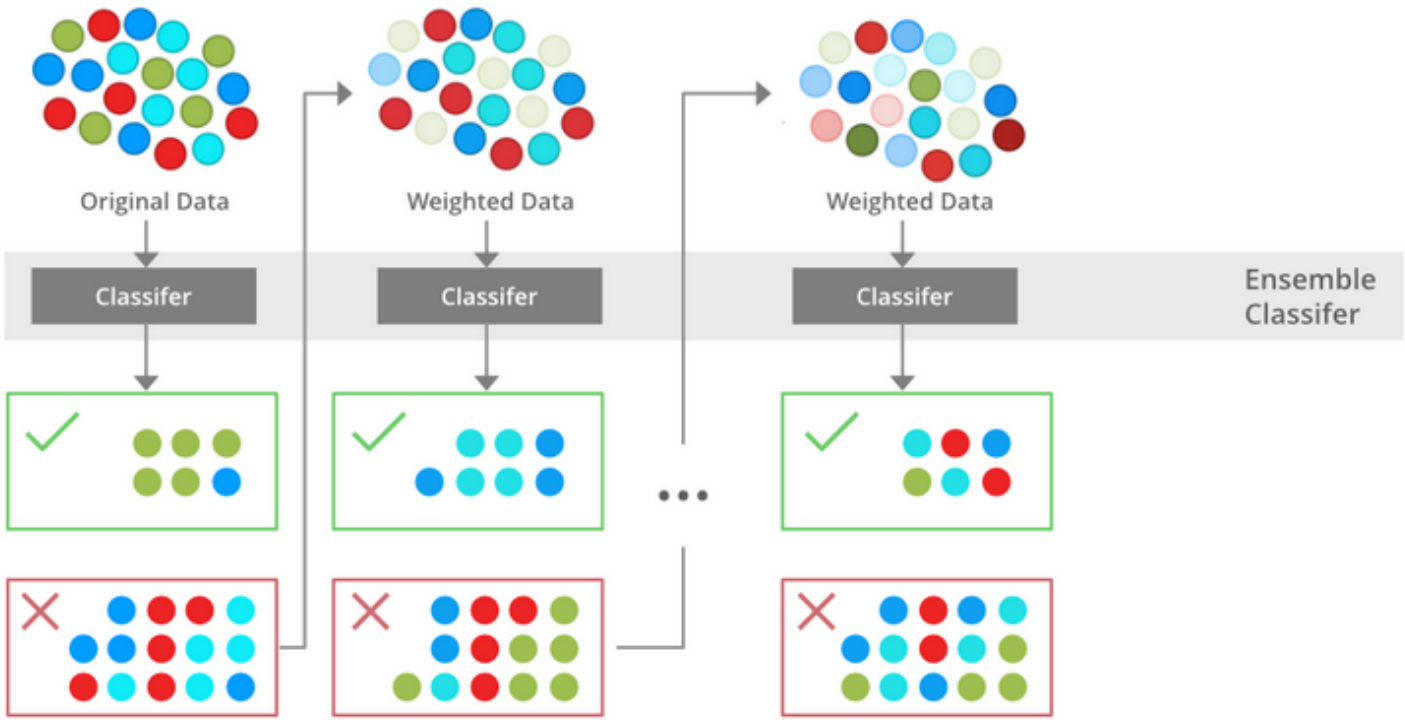
Bagging

Boosting

Boosting

Secuencialmente aplica un algoritmo de clasificación débil a versiones modificadas repetidamente de los datos para producir una secuencia de clasificadores débiles. Estas predicciones son combinadas para crear una mayoría de votos ponderado para producir una predicción final.

Boosting


Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer
Ensemble
Classifer

Ensemble Learning - Similitudes Bagging y Boosting

- Ambos son métodos de ensamblajes para obtener N learners desde 1 learner.
- Ambos generan varios set de entrenamiento al realizar muestreos aleatorios.
- Ambos generan la decisión final al promediar los N learners (o toman la mayoría de ellos, i.e. Mayoría de votos).
- Ambos reducen la varianza proveyendo una mayo estabilidad.

Ensemble Learning - Diferencias Bagging y Boosting

Bagging

1. la forma más simple de combinar predicciones que pertenecen al mismo tipo.
2. Apunta a disminuir la varianza, no el sesgo.
3. Cada modelo recibe el mismo peso.
4. Cada modelo es construido independientemente.

Boosting

1. Una forma de combinar predicciones que pertenezcan a distintos tipos.
2. Apunta a disminuir el sesgo, no la varianza.
3. Los modelos son ponderados según su desempeño.
4. Los nuevos modelos son influenciados por los modelos contruidos anteriormente.

Ensemble Learning - Diferencias Bagging y Boosting

Bagging

5. Diferentes conjuntos de datos de entrenamiento son seleccionados usando muestreo por fila con reemplazo y métodos de muestreo aleatorio de todo el conjunto de datos.
6. Intenta solucionar el problema de sobreajuste.
7. Si el clasificador es inestable (varianza alta), entonces se recomienda usar bagging.
8. Los clasificadores base son entrenados en paralelo.

Boosting

5. Cada nuevo subconjunto contiene elementos que fueron mal clasificados en el modelo anterior.
6. Apunta a disminuir el sesgo.
7. Si el clasificador es estable y simple (alto sesgo) se recomienda usar boosting.
8. Los clasificadores base son entrenados secuencialmente.

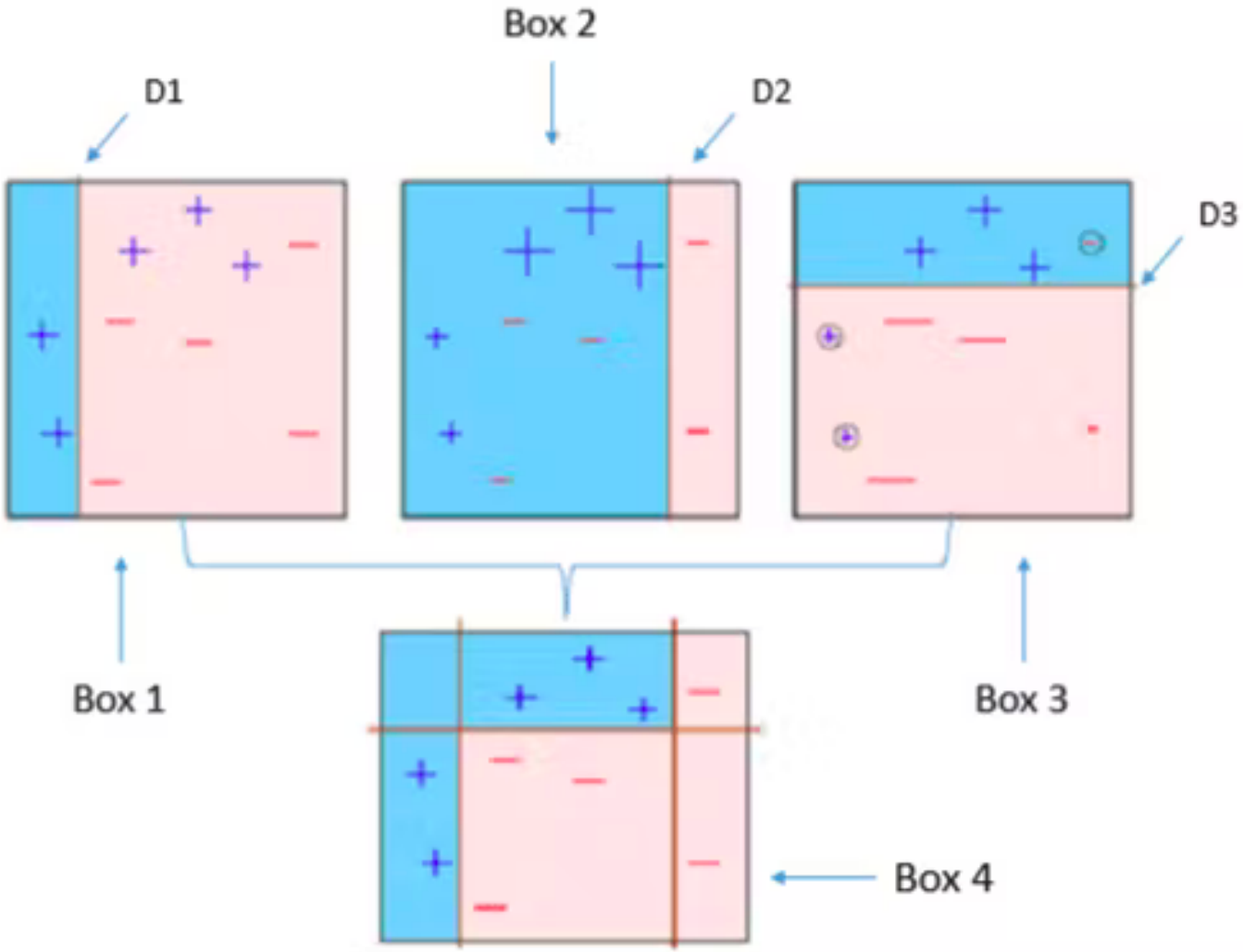
Adaboost

el algoritmo adaboost, uno de los más famosos algoritmos de boosting, es el siguiente:

AdaBoost

1. Inicializa los pesos de las observaciones $\omega_i = \frac{1}{N}, i = 1, ..., N$
2. Para $m = 1$ a M:
   a) Ajusta un clasificador $G_m(x)$ a los datos de entrenamiento usando los pesos ω_i
   b) Calcula
   $$err_m = \frac{\sum_{i=1}^{N} \omega_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i}$$
   c) Calcula $\alpha_m = log(\frac{1-err_m}{err_m})$
   d) Fija $\omega_i \leftarrow \omega_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))], i = 1, 2, ..., N$
3. Output: $G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$

AdaBoost - representación gráfica



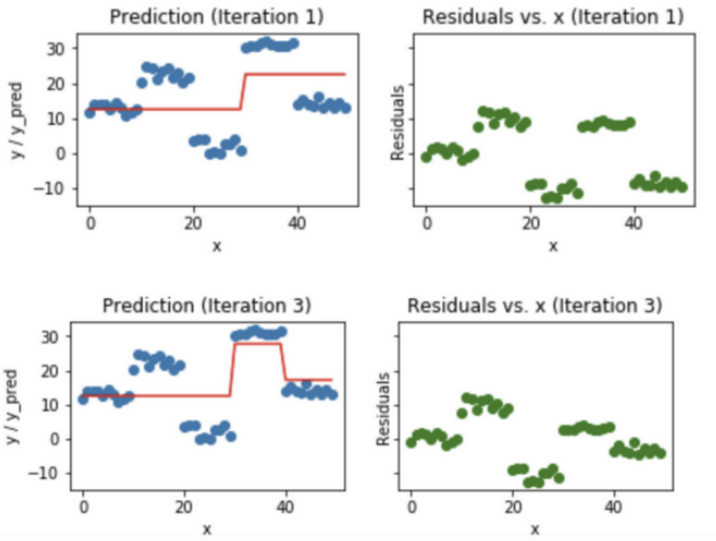
XGBoost - Gradient Boosting

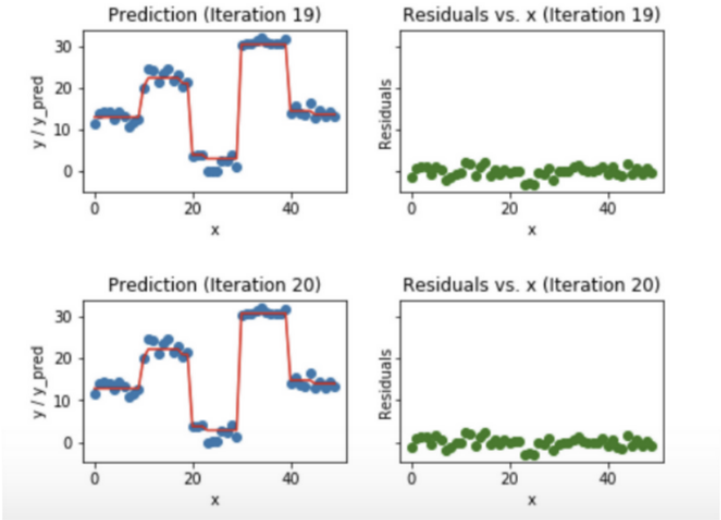
Para poder estudiar qué es XGBoost primero debemos saber qué hace Gradient Boosting.

Gradient Boosting

Como Adaboost, toma secuencialmente learners débiles, pero visto desde un punto de vista de optimización, toma una función de pérdida e intenta minimizar esa función. Esa pérdida son los residuos del modelo, y el Gradient Boosting intenta minimizar esos residuos.

XGBoost - gráficos Gradient Boosting





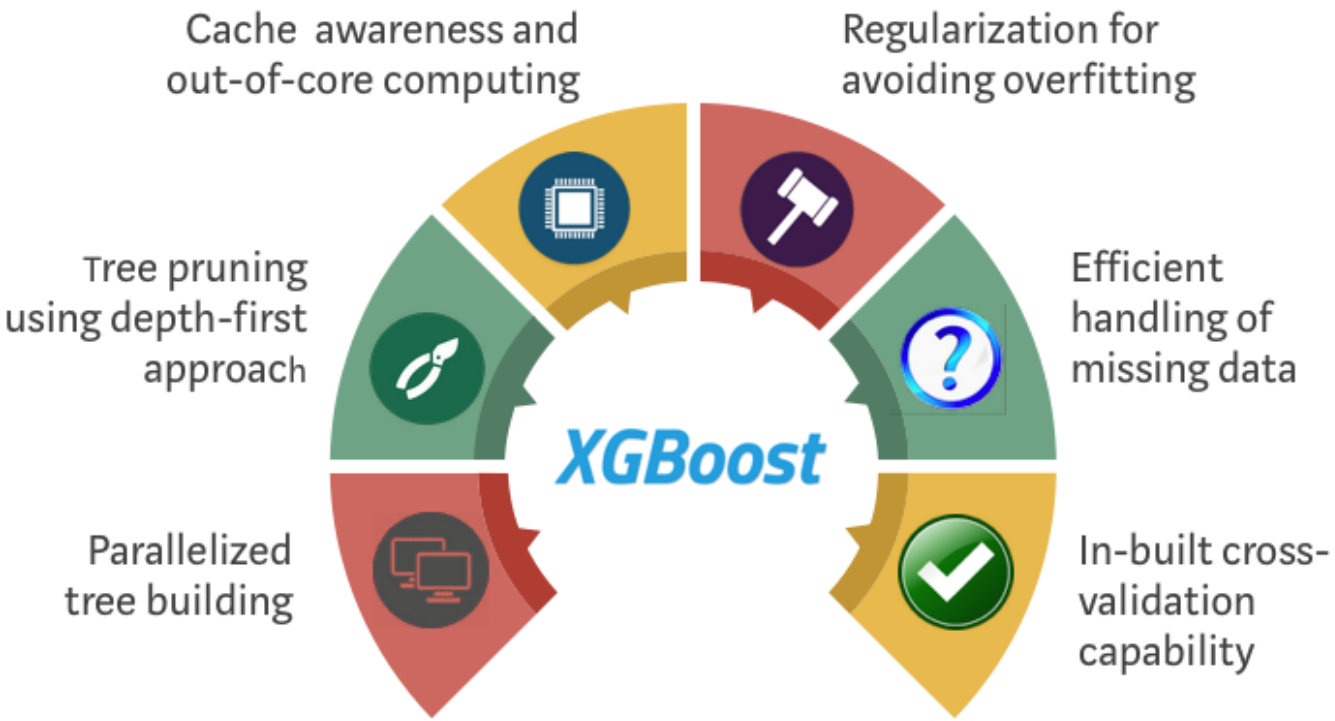
XGBoost

Similar a Gradient Boosting, XGBoost o Extreme Gradient Boosting tiene algunas diferencias

Caracteristicas XGBoost

- Penalización de árboles
- Encoje proporcionalmente los nodos hoja: como los árboles pueden tener distinto número de nodos terminales, asigna menor peso a los que presentan menos evidencia/
- Newton Boosting: En lugar de usar decenso por gradiente, utiliza newton-raphson para las aproximaciones, haciéndolo considerablemente más rápido.
- Parámetro extra de aleatorización: Reduce la correlación entre los clasificadores, mejorando el ensamblaje.

XGBoost


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

XGBoost - definiciones

- Construcción de árboles paralelizada: XGBoost enfoca el proceso de construcción secuencial usando implementación paralelizada
- Poda de árboles: A diferencia de otros métodos, donde la poda para una vez la pérdida es negativa, XGBoost hace crecer el árbol hasta una profundidad establecida con el parámetro 'max_depth' y luego poda hacia atrás hasta que la mejora en la función de pérdida está por debajo de la tolerancia (*threshold*)
- Conciencia de Cache y computación fuera de núcleo: XGBoost ha sido diseñado para reducir eficientemente el tiempo de computación y asignar un óptimo uso del recurso de memoria. Esto se logra concientizando el cache al asignar buffers internos en cada hilo para almacenar estadísticas de gradientes. Mayores mejoras como computación fuera de núcleo optimizan el espacio de disco disponible mientras maneja grandes volumenes de datos que no caben en memoria.

XGBoost - definiciones

- Regularización: La mayor ventaja de XGboost es la regularización. Ésta es una técnica usada para evitar el sobreajuste en modelos lineales y de árboles que limita, regula y encoje los coeficientes estimados hacia el cero.
- Se encarga de valores faltantes: Este algoritmo tiene características importantes para manejar los datos faltantes al aprender la mejor dirección de los valores faltantes. Estos valores son tratados para combinar una partición espaciada encontrando un algoritmo que maneje diferentes tipos de patrones espaciados en los datos
- Validación cruzada Built-in: El algoritmo viene con un método de validación cruzada pre construido en cada iteración, eliminando la necesidad de programar explicitamente esta búsqueda y especificar el número de iteraciones de boosting requeridas en una única corrida

Regularización y pérdida de entrenamiento

Una forma en que XGBoost controla la complejidad del modelo es a través de regularización y la función de pérdida.

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

Donde L es la función de pérdida en entrenamiento y Ω es la regularización. Así, la función de pérdida mide que tan bien ajusta el modelo los datos de entrenamiento mientras que Ω reduce la complejidad de las funciones árbol.

Regularización es la parte del modelo XGBoost donde se evita sobreajustar.

XGBoost - Función de pérdida usuales

Regresión - MSE

$$L(\theta) = \sum_i^n (y_i - \hat{y}_i)^2$$

Clasificación - Devianza

$$L(\theta) = \sum_i [(y_i ln(1 + e^{-\hat{y}_i}) + (1 - y_i) ln(1 + e^{\hat{y}_i})]$$