

Decision Tree에서 XGBoost까지

2021.11.07 김은희

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple- decision trees through a majority voting mechanism

Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models








Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias



A graphical representation of possible solutions to a decision based on certain conditions

Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees

Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

-  : Decision Tree
-  : Bagging
-  : Random Forest
-  : Boosting
 -  : Adaptive Boosting
 -  : Gradient Boosting
 -  : XGBoost

Decision Tree

- 끊임없는 yes/no 질문 → 결정에 다다르기
- 🤔 Overfitting의 문제는 어떻게 해결하는데? : Pruning (사전 가지치기)
 - 트리의 최대 깊이를 제한하거나 (max_depth)
 - 리프의 최대 개수를 제한하거나 (max_leaf_nodes)
 - 노드의 분할을 위한 포인트의 최소 개수를 지정(min_samples_leaf)

Decision Tree

- 장점
 - 직관적
 - 트리 생성에 각 특성이 얼마나 중요하게 기여* 했는지 평가 가능 (* 단, 그것이 어떤 클래스에 속하는지를 지지하는지는 알 수 없음)
- 단점
 - 여전히 overfitting 되는 경향

Bagging

- Bootstrap(데이터의 '복원 추출') + aggregating(합쳐서 예측하기)
- 데이터를 복원 추출하고, 베이스 모델의 결과값을 합쳐서 예측

Random Forest

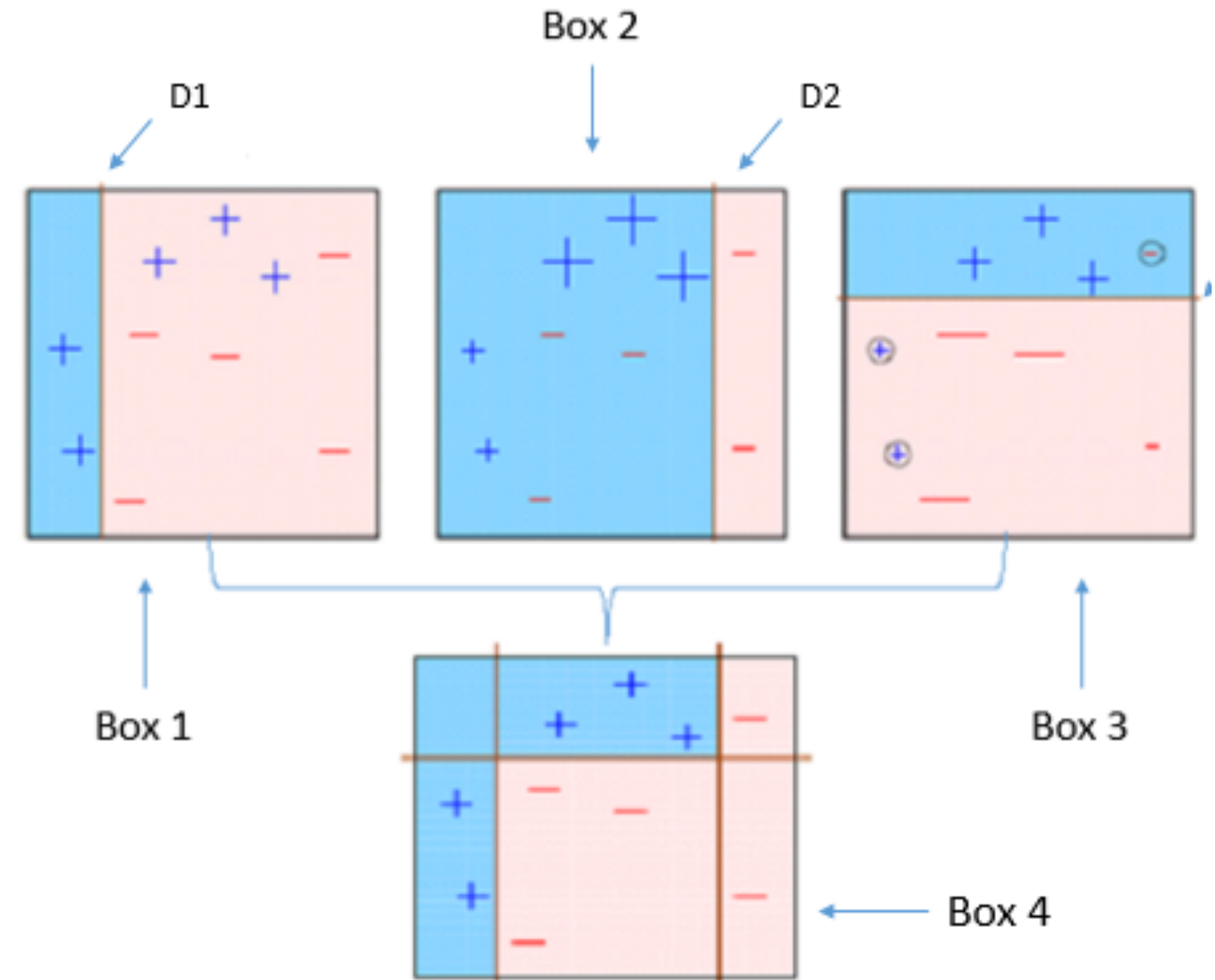
- Decision Tree + Bagging : 결정 트리 + 배깅의 아이디어
- 랜덤하게 선택된 여러 training data + 랜덤하게 선택된 변수들을 기준으로 분기점

Boosting

- Bagging
 - 병렬 (A, B, C, D 각자 할 일 → 합치기)
 - 상대적으로 느린 속도
- Boosting
 - 직렬 (A가 못한 일 → B가 노력 → C가 노력 → ...) “weak learner들의 성장서사”
 - 상대적으로 빠른 속도

Adaptive Boosting

- weak learner 의 오류 데이터에 가중치
- 잘 맞추지 못하는 샘플을 더 많이 추출



Adaptive Boosting

- 전체 데이터에서 무작위 추출
- 모든 sample data 들의 가중치 초기화 (가중치 $\alpha_t = \frac{1}{2} \ln(\frac{1 - \epsilon_t}{\epsilon_t})$)
- 다음 dataset 선택의 가중치 업데이트 $D_{t+1} = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
- weak learner 들이 학습 과정에서 보인 성능의 가중치 부여 → 최종 모델 만들기

Gradient Boosting

- 잘 못 맞추고 있는 만큼을 맞출 수 있도록, 추정 목표에 변화 주기
- $\hat{y} = f_1(x)$ 이라는 예측의 오차 : $y - f_1(x) = f_2(x)$ 를 예측의 목표로 \rightarrow
 $y - f_1(x) - f_2(x) = f_3(x) \dots$ 의 반복
- Gradient 개념과의 연계
- Loss function 을 $L = \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2$ 라 하고 gradient를 계산 후 변형
- $y_i - f(x_i) = \text{잔차} = -\frac{\partial L}{\partial f(x_i)}$ ‘잔차에 대해 업데이트’ = ‘gradient의 반대 방향으로 학습’

Gradient Boosting

- 장점
 - 우수한 성능
 - 각 변수의 기여 정도를 계산 가능
- 단점
 - noise 조차 예측의 대상 \rightarrow overfitting 의 문제 발생

XGBoost

- GBM 의 ‘직렬’ 방식의 한계 : 데이터가 메모리에 한번에 로딩되어야 함
- GBM을 최대한 효율적으로 사용할 수는 없을까? → 분산처리 가능한 XGBoost