Ch07.

다양한 모델을 결합한 앙상블 학습

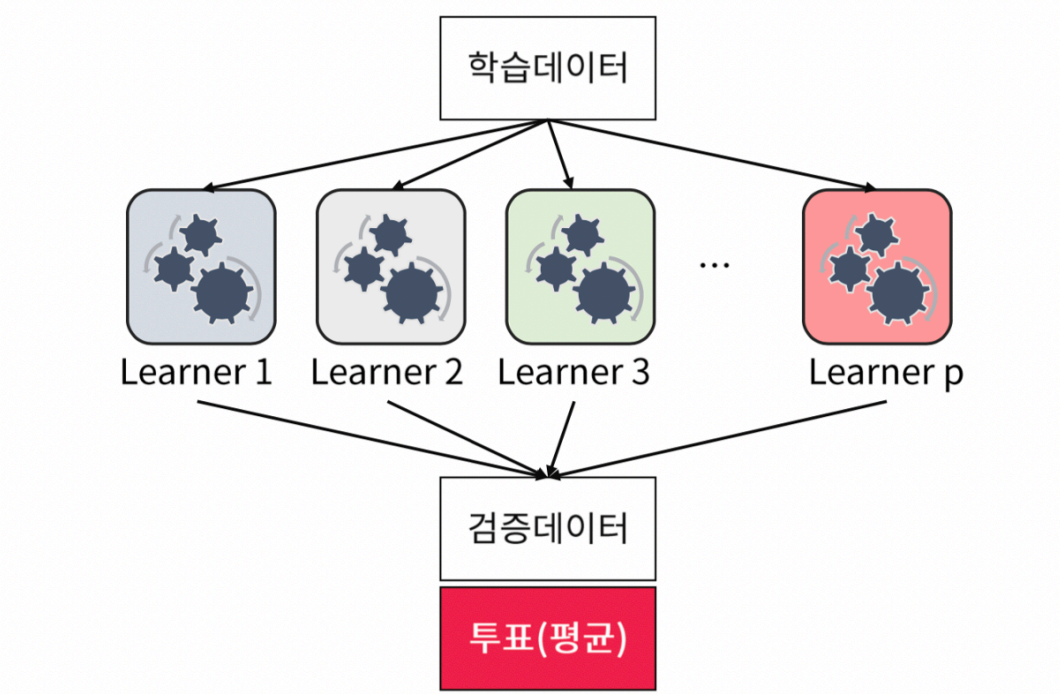
202STG18 이재빈

CONTENTS

01 ,
Ensemble

1. Ensemble

- definition

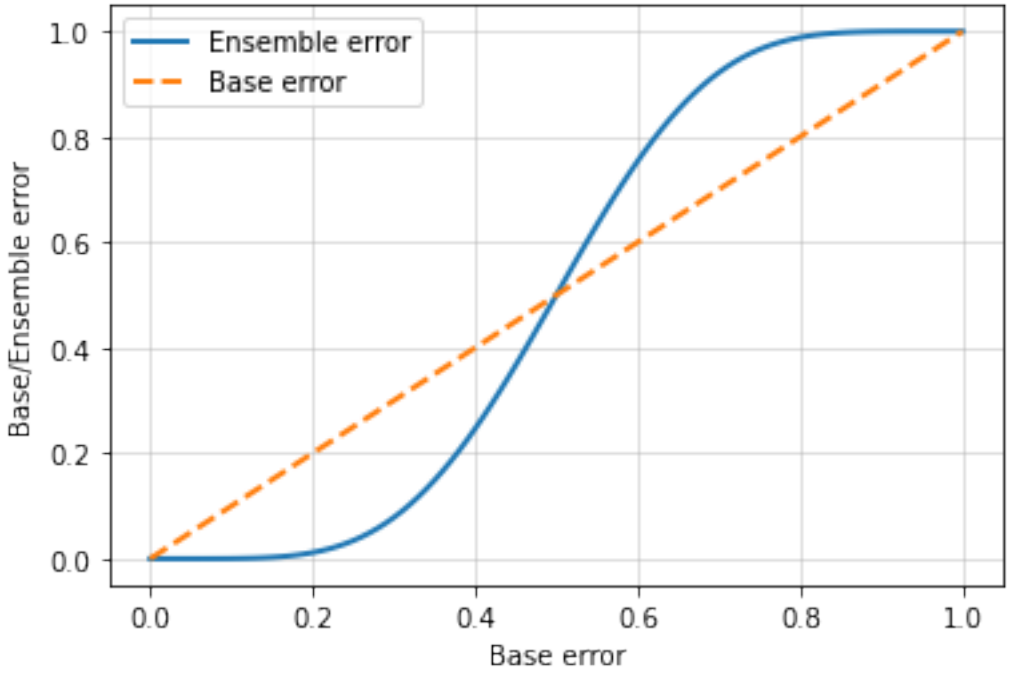


여러 개의 약분류기를 모아서 강한 하나의 분류기 만들기

전문가 10명의 예측을 묶어, 전문가 한 명 보다 더 정확하고 안정된 예측을 만든다!

1. Ensemble

- 이진분류 에러율



이진분류 에러율 >>> 앙상블 에러율

$$P(y \geq k)$$

$$= \sum_{k}^{n} \binom{n}{k} \varepsilon^{k}(1-\varepsilon)^{n-k}$$

$$= \varepsilon_{ensemble}$$

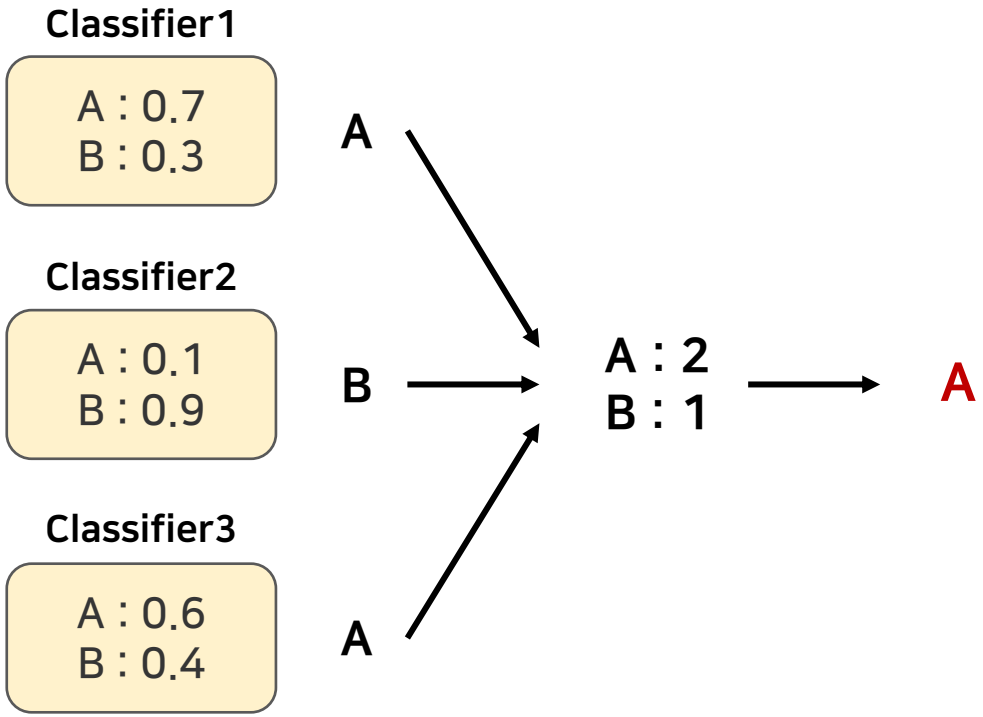
$$P(y \geq k)$$

$$= \sum_{k=6}^{11} \binom{11}{k} 0.25^{k}(1-0.25)^{11-k}$$

$$= 0.034$$

02 ,
Voting

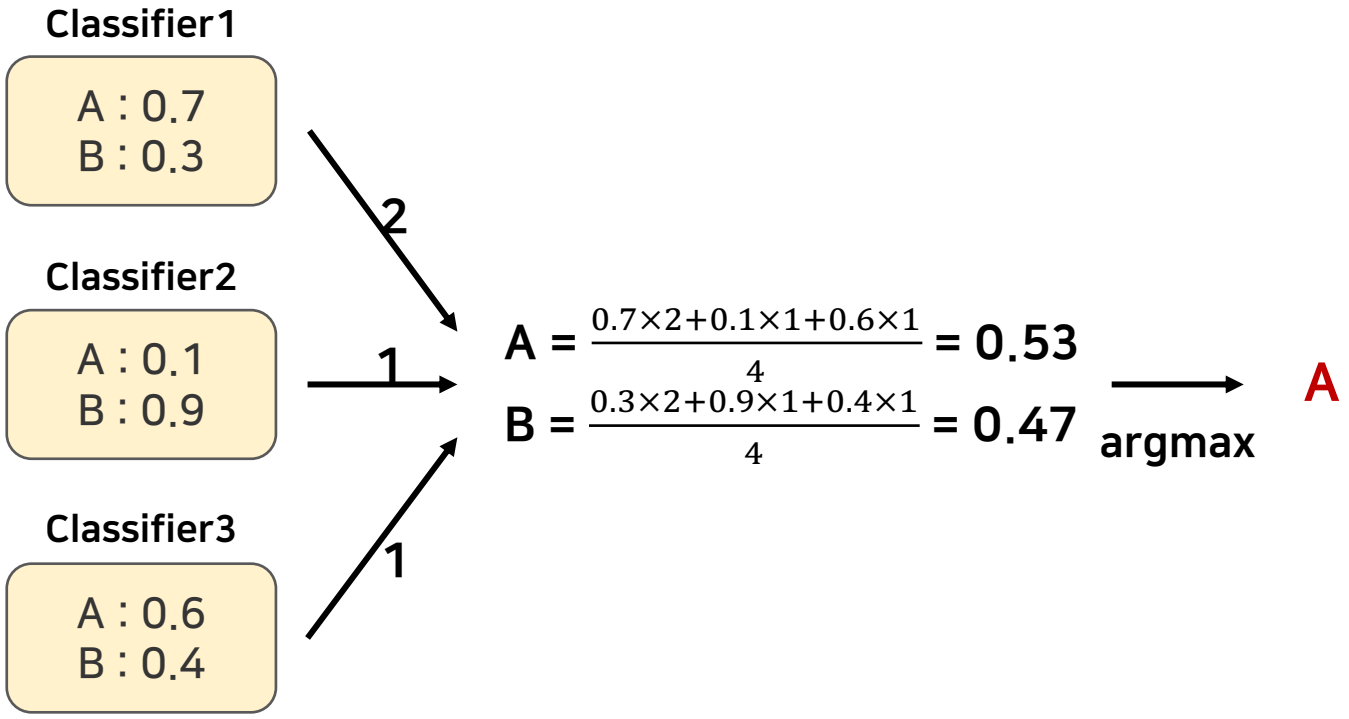
2. Voting
- Hard Voting



Hard Voting

- 다수결의 원칙과 비슷
- 다수의 분류기가 결정한 값을 최종 예측값으로 선정

2. Voting
- Soft Voting



Soft Voting

- 가중치 투표
- 클래스 확률을 평균하여 결정

03 ,
Bagging

3. Bagging
- Bootstrap Aggregating

Bagging = Bootstrap Aggregating

Bootstrap

- n개의 sample 에서 m번 복원추출
- 분산을 줄이는 효과

OOB (Out-Of-Bag) 평가

- 생략된 sample을 이용해 validation 이나 cross validation 에 사용! (36.8%)

질문 2 부트스트래핑 과정 중에서 n개의 샘플에 대해 n번의 샘플링을 하는데, n이 무한대로 커진다면 한 번도 추출되지 않는 데이터의 수는 얼마나 될까요?

난이도 ★★★

분석·해답

하나의 샘플이 한 번의 샘플링 과정에서 추출되지 않을 확률은 $\left(1-\frac{1}{n}\right)$입니다. 그리고 n번의 샘플링에서 한 번도 뽑히지 않을 확률은 $\left(1-\frac{1}{n}\right)^n$이 됩니다. n이 무한대로 커질 경우 확률은 $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n$이 됩니다.

$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$이기 때문에 다음의 식과 같이 됩니다.

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n = \lim_{n\to\infty}\frac{1}{\left(1+\frac{1}{n-1}\right)^n}$$

$$= \frac{1}{\lim_{n\to\infty}\left(1+\frac{1}{n-1}\right)^{n-1}} \cdot \frac{1}{\lim_{n\to\infty}\left(1+\frac{1}{n-1}\right)}$$

$$= \frac{1}{e} \approx 0.368 \qquad (2.17)$$

따라서 샘플 수가 매우 클 경우에 약 36.8%*의 샘플이 한 번도 추출되지 않게 됩니다.

출처 : 데이터 과학자와 데이터 엔지니어를 위한 인터뷰 문답집

3. Bagging
- Bagging

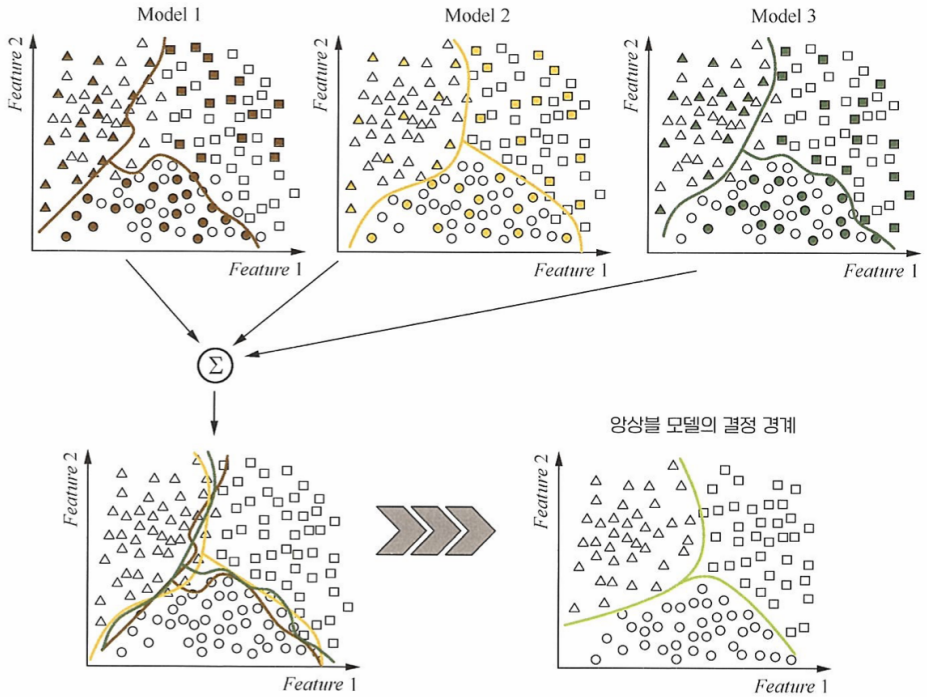


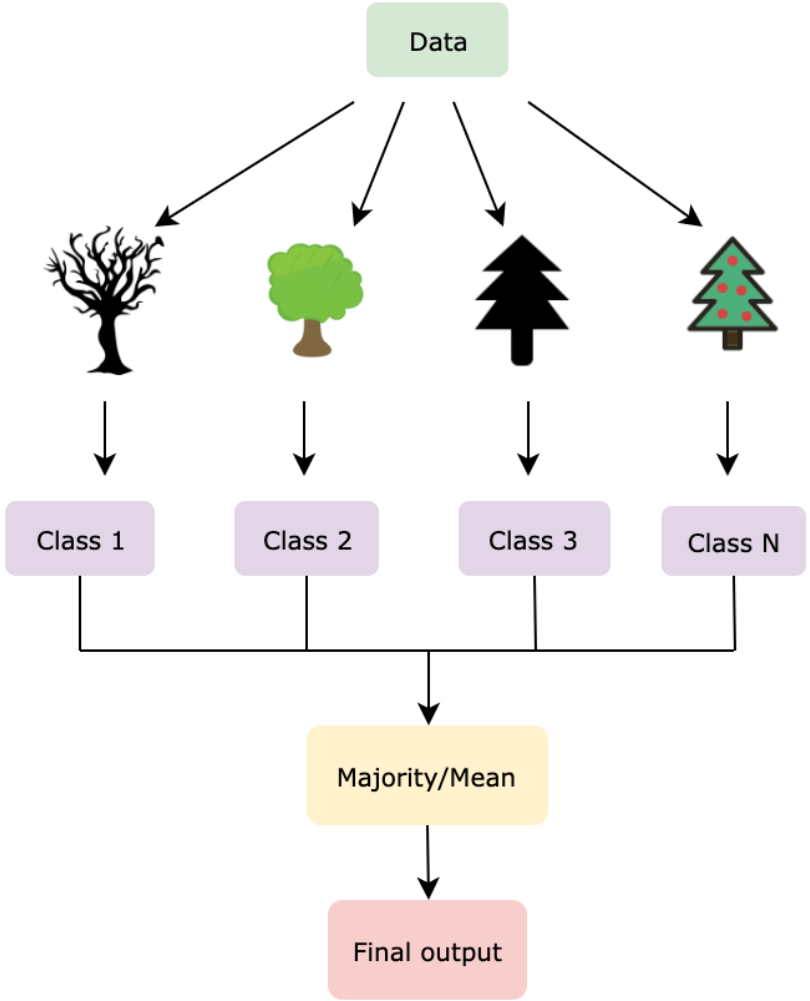
그림 12.3 배깅(bagging) 알고리즘의 예시

PROCESS

1. 각 분류기는 훈련 세트에서 추출한 랜덤한 부분 집합 사용
2. 개별 분류기가 부트스트랩 샘플에 학습 되면
3. 다수결 투표를 사용하여 예측

- 개별적으로 보면 결정경계가 모두 구불구불해 과적합 경향이 나타남
- 앙상블 후에 모델의 결정 경계는 각각 독립적인 모델에 비해 평활 smooth 해짐
- 앙상블의 가중 투표 방법이 분산을 감소시켰기 때문!

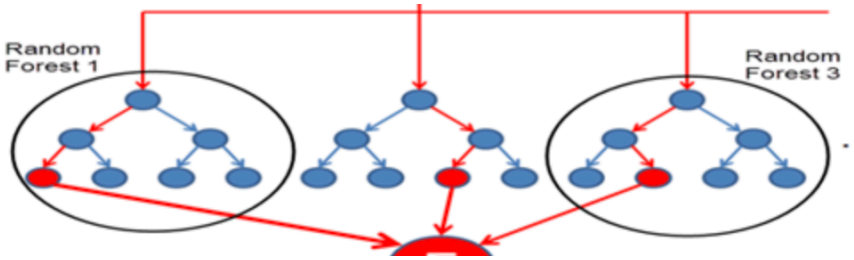
3. Bagging
- Random Forest



"나무가 모여서 숲을 이룬다. "

결정 트리 (Decision Tree) 가 모여서 랜덤 포레스트를 구성!

개별 결정 트리를 학습할 때 랜덤하게 특성의 부분집합을 선택하는, Bagging 의 특별한 경우



모델마다 다른 Feature 선택

04 ,
Boosting

4. Boosting

- Boosting

분류하기 어려운 훈련 샘플에 초점!

- 잘못 분류된 훈련 샘플을 그 다음 학습기가 학습
- 이전 층 기초 분류기가 잘못 분류한 샘플에, **더 큰 가중치** 부여
- 직렬적 serial 방식 사용, 각 기초 분류기 사이에 의존 관계 존재
- Bias & Variance 둘 다 감소 가능

ex. 부스팅 과정은 **사람 학습 과정**과 유사! 🧐

우리가 새로운 지식을 습득하는 과정은 늘 반복적입니다. 처음 학습할 때는 일부분의 지식을 기억하는 동시에 어떤 부분에서는 **실수**를 합니다. 하지만 두 번째 학습에서는 이전에 실수한 부분에 대해 같은 실수를 범하지 않기 위해서 **공부를 더 많이** 하게 됩니다. 이러한 순환은 계속되는데, 일반적으로 **같은 실수를 반복하지 않을 정도까지** 학습을 반복하게 됩니다.

4. Boosting

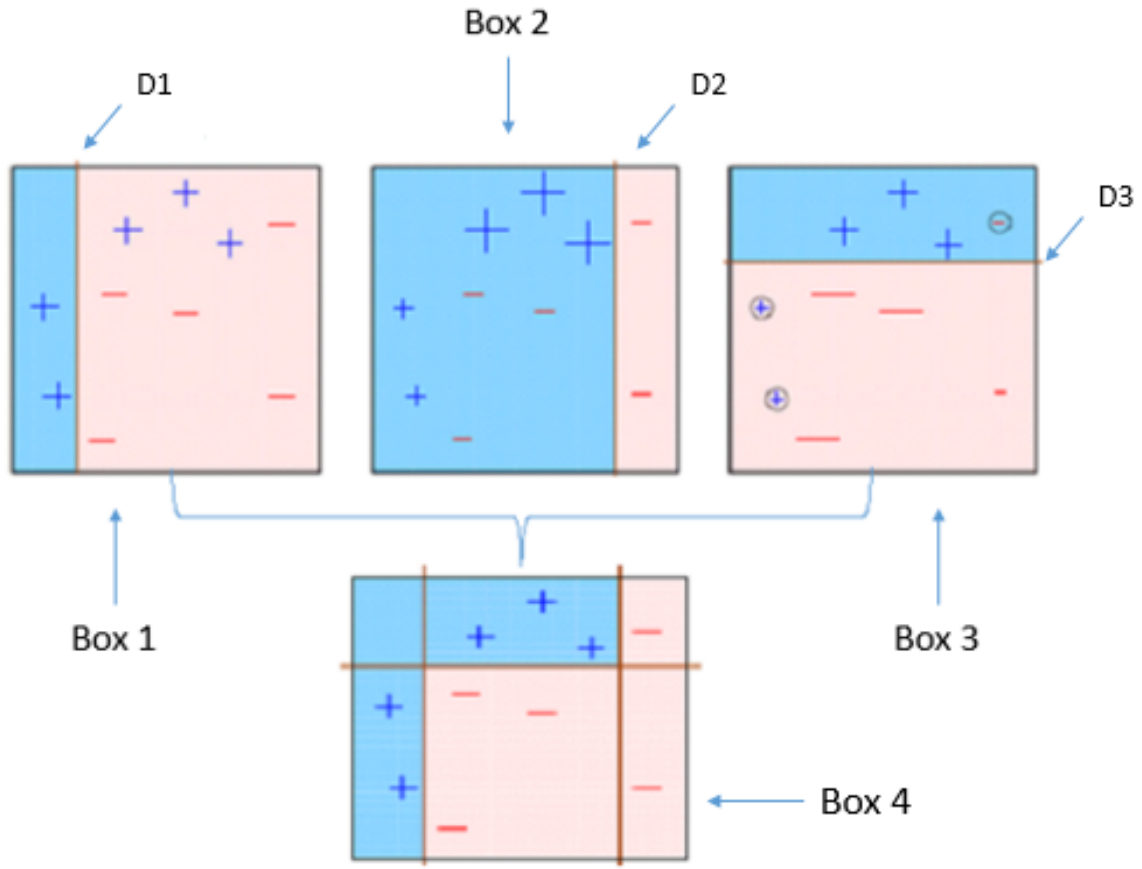
- Boosting

Boosting의 초창기 방법

1. 훈련 세트 D 에서 **중복을 허용하지 않고** 랜덤한 부분 집합 d1 을 뽑아 약한 학습기 C1 을 훈련시킴
2. 훈련 세트에서 중복을 허용하지 않고 두 번째 랜덤한 부분 집합 d2 를 뽑고,
   이전에 잘못 분류된 샘플의 **50%를 더해서** 약한 학습기 C2 를 훈련시킴
3. 훈련 세트 D 에서 C1 과 C2 에서 잘못 분류한 훈련 샘플 d3 를 찾아
   세 번째 약한 학습기인 C3를 훈련시킴
4. 약한 학습기 C1, C2, C3 를 다수결 투표로 연결

4. Boosting

- AdaBoost



AdaBoost

- 훈련 시 **훈련 세트 전체** 사용
- 훈련 샘플은 반복될 때 마다 가중치 부여
- 이 앙상블은 이전 학습기의

  실수를 학습하는 강력한 분류기를 만듦

4. Boosting

- AdaBoost

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N.$ 가중치 벡터 w를 동일한 가중치로 설정
2. For $m = 1$ to M: m번 부스팅 반복
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i. 약한 학습기 훈련 및 레이블 예측
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$ 가중치 적용된 에러율 계산
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$. 학습된 가중치 계산
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N.$ 가중치 업데이트 (+ 정규화)
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right].$ 최종 예측 계산

4. Boosting

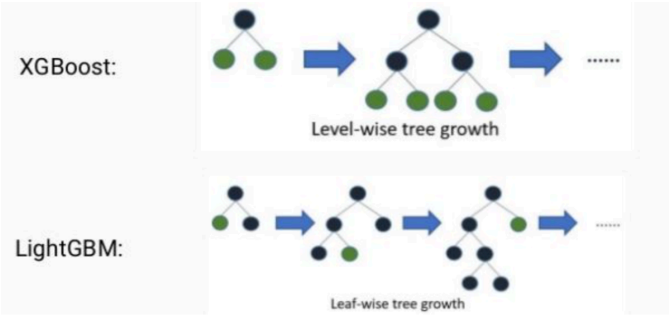
- Boosting

Gradient Boost

- 가중치 업데이트로 경사하강법 (Gradient Descent) 이용
- 잔차 (y-F(x)) 를 target 으로 하고, 이를 최소화 하는 방향으로 모델 학습

XGBoost

- GBM 기반
- Tree 를 만들 때 병렬처리를 가능하게 해서 GBM의 속도를 개선하기 위해 만들어짐
- Greedy-algorithm 을 사용한 자동 가지치기 가능 -> 오버피팅에 강함



LightGBM

- 한 쪽으로만 성장 (<-> XGBoost : 균형 트리 모델)
- 균형을 맞추지 않고, 최대 손실 값을 가지는 리프노드를 지속적으로 한쪽만 분할하면서 트리의 깊이가 깊어지고 비대칭적인 규칙 트리 생성

감사합니다 😊