

Ensemble

앙상블이란?

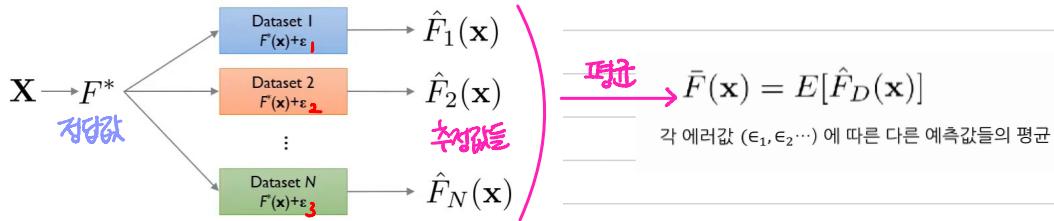
서로 다른 알고리즘들이 적절하게 결합되어 단일 모델보다 더 좋은 성능을 낸다.

1. Bias & Variance decomposition

$$y = F^*(x) + \epsilon, \epsilon \stackrel{iid}{\sim} N(0, \sigma^2)$$

자연 발생하는 에러
독립적이고, 동일한 분포부터 만들어짐

정답값
(target function)



x 를 F^* 에 대입하면 항상 같은 값을 내지만 ϵ 에 따라 dataset이 달라짐.
이를 통해 F 를 추정하고, 추정된 F 와 F^* 는 동일해야 good!

x_o 에서의 MSE

$$\begin{aligned} Err(x_o) &= E(y - \hat{F}(x) | x=x_o)^2 \\ &= E(F^*(x_o) + \epsilon - \hat{F}(x_o))^2 \\ &= E((F^*(x_o) - \bar{F}(x_o)) + \epsilon)^2 \\ &= E((F^*(x_o) - \hat{F}(x_o))^2 + 2\epsilon(F^*(x_o) - \hat{F}(x_o)) + \epsilon^2) \xrightarrow{\text{0}} \\ &= E(F^*(x_o) - \hat{F}(x_o))^2 + \sigma^2 \\ &= E(F^*(x_o) - \bar{F}(x_o) + \bar{F}(x_o) - \hat{F}(x_o))^2 + \sigma^2 \\ &= E(F^*(x_o) - \bar{F}(x_o))^2 + E(\bar{F}(x_o) - \hat{F}(x_o))^2 + \sigma^2 \\ &= (F^*(x_o) - \bar{F}(x_o))^2 + E(\bar{F}(x_o) - \hat{F}(x_o))^2 + \sigma^2 \\ &= \text{Bias}^2(\hat{F}(x_o)) + \text{Var}(\hat{F}(x_o)) + \sigma^2 \end{aligned}$$

Bias : ϵ 으로 noise를 바꾸기며 모델링을 했을 때, 모델의 평균적인 결과가 정답에 얼마나 가까울지.
Variance : ϵ 을 바꾸기며 모델링을 했을 때 개별적인 모델링이 평균과 얼마나 큼 차이를 보이는지.

→ MSE을 이용해 분해를 하면 bias와 variance, σ^2 을 표현함
(loss Function)

→ 앙상블의 목적 : 분산을 줄이자 (boosting), 편차를 줄이자 (bagging)

2. Bagging

- low bias, high variance에 잘 맞음
- 모든 지도 학습 알고리즘 사용 가능

bootstrap

원래 데이터에서 복원추출한 데이터셋

Original Dataset	Bootstrap 1	Bootstrap 2	Bootstrap B
x ¹ y ¹	x ³ y ³	x ¹ y ⁷	x ⁸ y ⁹
x ² y ²	x ⁶ y ⁶	x ¹ y ¹	x ⁵ y ⁵
x ³ y ³	x ² y ²	x ¹⁰ y ¹⁰	x ² y ²
x ⁴ y ⁴	x ¹⁰ y ¹⁰	x ¹ y ¹	x ⁴ y ⁴
x ⁵ y ⁵	x ⁸ y ⁸	x ⁸ y ⁸	x ⁷ y ⁷
x ⁶ y ⁶	x ⁷ y ⁷	x ⁵ y ⁵	x ² y ²
x ⁷ y ⁷	x ⁷ y ⁷	x ⁴ y ⁴	x ³ y ³
x ⁸ y ⁸	x ³ y ³	x ⁶ y ⁶	x ¹⁰ y ¹⁰
x ⁹ y ⁹	x ² y ²	x ¹ y ¹	x ⁸ y ⁸
x ¹⁰ y ¹⁰	x ⁷ y ⁷	x ⁹ y ⁹	x ² y ²

1) Majority voting (hard voting)

: 다수결에 의해 (0과 1 중 어느것의 개수가 더 많은지에 따라)

$$\hat{y}_{\text{ensemble}} = \arg \max_i \left(\sum_{j=1}^n \delta(\hat{y}_j = i), i \in \{0, 1\} \right)$$

↑ 개수

2) Weighted Voting (soft voting)

: 가중치를 부여해 확률 확대값에 따라

$$\hat{y}_{\text{ensemble}} = \arg \max_i \left(\frac{\sum_{j=1}^n (\text{Trn Acc}_j) \cdot \delta(\hat{y}_j = i)}{\sum_{j=1}^n (\text{Trn Acc}_j)}, i \in \{0, 1\} \right)$$

↑ 예측 정확도

3) Stacking

: 각 모델의 예측 확률을 구한 후,
이를 input으로 다시 모델링 진행 (Meta-Classifier),
그 결과가 최종 예측값이 됨

3. Random Forest

- 배깅의 한 종류 + decision tree 기반
- 배깅 (복원추출하여 각 bootstrap으로 training)
- 변수 랜덤하게 선택

변수 중요도 산출

- i) 원래 데이터셋의 OOB 예측 구하기 $\rightarrow e_i$
- ii) 특정 변수 x_i 가 바뀐 데이터셋의 OOB 예측 구하기 $\rightarrow p_i$

$\Rightarrow e_i$ 와 p_i 차이가 크면 x_i 는 중요 변수 ($p_i > e_i$)

- m번재 tree에서 x_i 변수에 대한 OOB error 차이 : $d_i^m = p_i^m - e_i^m$

- 변수 중요도 : $V_i = \frac{d_i}{S_i}$ $\leftarrow d_i$ 의 평균 $\leftarrow S_i$ 의 표편

OOB error (Out Of Bag error)

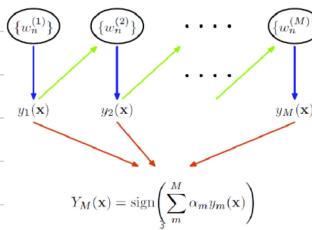
\rightarrow 개별 학습 데이터셋 구성시 bootstrap 되지 않은 개체들을 검증용으로 사용

\rightarrow 이 값을 test 데이터로 삼으면, test error가 계산됨

4. Boosting - AdaBoost

Boosting

- 약한 학습기에 집중
- 특정 데이터셋 모델링 \rightarrow missclassified 된 데이터 찾기 \rightarrow 이 데이터가 다음 학습에서 봉황 확률이 높아지도록 Sampling



알고리즘

Algorithm 2 AdaBoost

```

Input: Required ensemble size T 분류기 개수 ①
Input: Training set S = {(x1, y1), (x2, y2), ..., (xN, yN)}, where yi ∈ {-1, +1}
Define a uniform distribution D1(i) over elements of S.
for t = 1 to T do ②
    Train a model ht using distribution Dt.
    Calculate εt = PDt(ht(x) ≠ y) ③
    If εt ≥ 0.5 break 주의 X
    Set αt = ½ ln (1 - εt) / εt ④
    Update Dt+1(i) = Dt(i) exp(-αtyiht(xi)) / Zt 정답에 맞췄을 경우
    where Zt is a normalization factor so that Dt+1 is a valid distribution.
end for
For a new testing point (x', y'),
H(x') = sign (sumt=1T αt ht(x')) ⑤
-정답을 맞췄을 경우
-정답을 맞쳤을 경우

```

① 이진분류 모델 가정

③ 에러가 0.5이상이면 의미 없다고 판단

④ 가중치 계산. 에러가 0.5에 가까울수록 가중치 0, 에러가 0에 가까울수록 무한대로 증가

⑤ 다음부분에 대한 확률 계산

-정답을 맞췄을 경우

-정답을 맞쳤을 경우

5. Boosting - GBM

- 경사하강법 + Boosting

- 예측이 틀린 데이터를 수정하지 않는다.

- 예측 값과 실제 값의 차이 (잔차)에 대해 다음 모델이 보완할 수 있도록 모델 구성

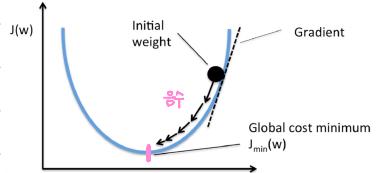
경사 하강법 (Gradient Boosting)

$$\min L = \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2 \rightarrow \text{최적화}$$

$$\frac{\partial L}{\partial f(x_i)} = f(x_i) - y_i, \quad y_i - f(x_i) = -\frac{\partial L}{\partial f(x_i)}$$

잔차

잔차는 loss function의 negative gradient와 같다



6. XGBoost

- GBM의 최적화 버전

Split finding Algorithm

