Boosting

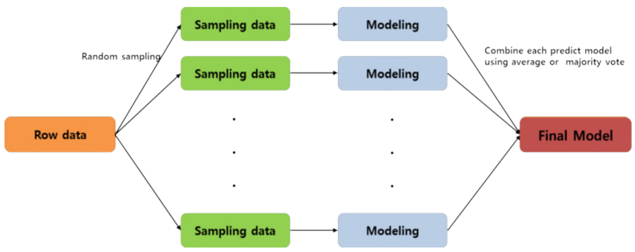
Gradiant Boosting Algorithm

Bagging - 여러 Bootstrap 자료 결합하여 예측

∴ Bootstrap 자료를 단순 복원임의추출법

복원 샘플링으로 분산 줄이고 예측력 향상

과대적합, 분산이 크고 편의가 작은 모형 사용 적합



한번에

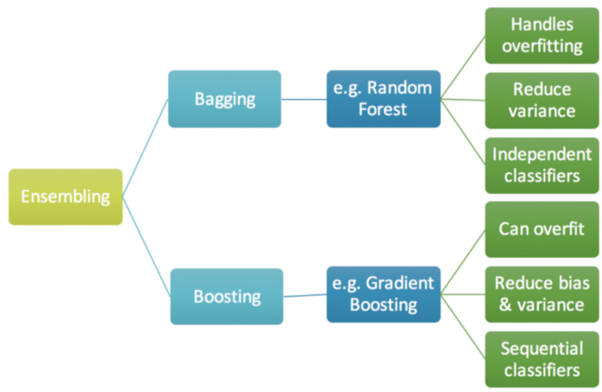
순차적으로

·Boosting - 잘못 분류된 개체에 관심

약한 예측 모형 결합 → 강한 예측모형

∴오분류에 높은 가중치

예측모형 정확도 상승



비교	Bagging	Boosting
특징	병렬 앙상블 모델 (각 모델은 서로 독립적)	연속 앙상블 (이전 모델의 오류를 고려)
목적	Variance 감소	Bias 감소
적합한 상황	복잡한 모델 (High Variance, Low Bias)	Low Variance, High Bias 모델
대표 알고리즘	Random Forest	Gradient Boosting, AdaBoost
Sampling	Randomg Sampling	Random Sampling with weight on error

참고 : https://www.slideshare.net/freepsw/boosting-bagging-vs-boosting

Boosting - ex) Gradient, adaptive

약한 분류기 결합 → 강한 분류기

순차적

GBM 이해 - Residual fitting

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma).$$ → Baseline model

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*: → negative gradient = pseudo-residual

   $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

   $$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   이 과정은 생략되기도 함

   4. Update the model:

   $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

   매우 작은값

3. Output $F_M(x)$.

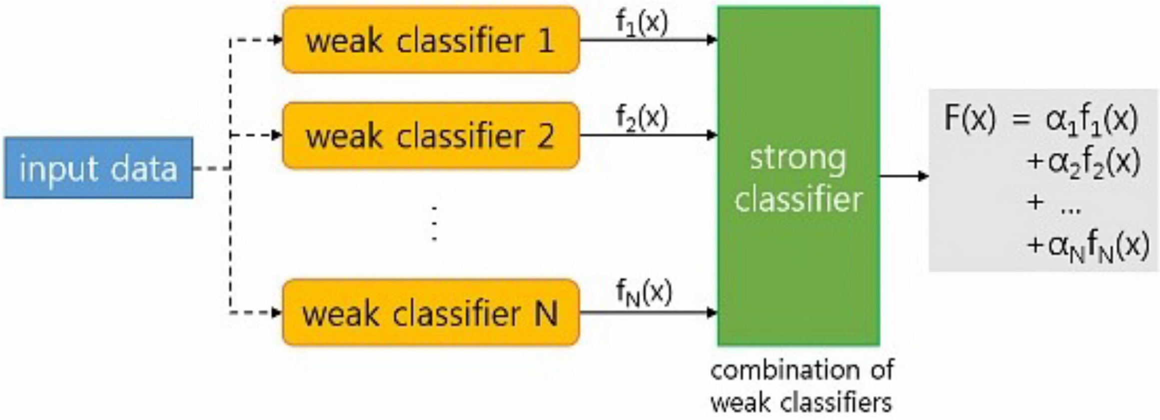
1. t_x Gradient f_x

2. t_{x+1} Gradient update f_{x+1}

3. t_{x+2} 〃 f_{x+2}

$$H_x = \sum_{i=1}^{n} \alpha_i f_i$$

약한 분류기 조합하여 강한 분류기)



AdaBoost is an algorithm for constructing a "strong" classifier as linear combination

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

of "simple" "weak" classifiers $h_t(x)$.

$h_t(x)$ ... "weak" or basis classifier, hypothesis, "feature"

$H(x) = sign(f(x))$ ... "strong" or final classifier/hypothesis

Adaboost - adaptive + boosting · 상호보완, 순차적

· 앞서 분류기가 잘못 분류한 결과를 다음 분류기 학습시 사용

How it works?

→ 아마 $w_{n+1} = w_n - r\frac{\partial L}{\partial w}$

1. 모든 feature에 대해 성능평가, 각 feature마다 weighted error 계산
2. 분류 성능 가장 좋은 feature을 해당 round의 weak classifier
3. 해당 weak classifier의 weight를 구한다. - 에러가 작으면 중요도↑
4. training sample의 weight를 업데이트

   잘못 분류된 sample의 weight 증가 → 해당 샘플 중요도 증가

   → 다음에 잘못분류 더 영향 신경쓰음

5. t=t+1 반복

최종 output $H(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_T h_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

weight

training sample

target label (no, yes)

Given: $\boxed{(x_1, y_1), \ldots, (x_m, y_m)}$ where $\boxed{x_i \in \mathscr{X}}$, $\boxed{y_i \in \{-1, +1\}}$.

Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.

D : weight of training samples (the probability of being selected for training the component classifier)

$\boxed{\text{For } t = 1, \ldots, T:}$ **T** : iteration round

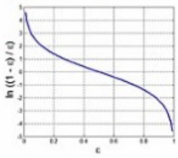
- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: $\boxed{\text{select } h_t \text{ with low weighted error:}}$

ε : error rate of weak classifier

$$\sum_{i : h_t(x_i) \neq y_i} D_t(i) \qquad \varepsilon_t = \Pr_{i \sim D_t}\left[\underline{h_t(x_i) \neq y_i}\right].$$

1 if misclassified,
0 if properly classified

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$. **α** : weight (importance) of weak classifier



- Update, for $i = 1, \ldots, m$:

update weights of training samples

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

$$\sum_i D_{t+1}(i) = 1$$

Output the final hypothesis:

H : strong classifier

$$\boxed{H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)}.$$