Ensemble Learning

- Combine (weak) base learners to obtain a stronger one
  - base learners should be as diverse as possible
- Bias-Variance Decomposition
  - High variance low bias: tree model
  - Low variance high bias: linear model

Bagging

- **reduce variance**
- Suppose we can repeatedly sample D from $P(D)$, then let $f(x) = \frac{1}{T}\sum_{t=1}^{T} f(x, D_t)$
  - $f(x) \to \bar{f}(x)$ when $T \to \infty$, then variance $\to 0$
- However, we only have one $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

Bootstrap

- **Bootstrapping** (自举) is a way that uses **random sampling with replacement**.
  i. Sample n points from D with replacement
  ii. Repeat i. for T times, get $\hat{D}_1, \ldots, \hat{D}_T$
  iii. $\hat{f}(x) = \frac{1}{T}\sum_{t=1}^{T} f(x, \hat{D}_t)$ as the bagged model
- 一共n个样本，有放回地每次选一个，一共选n次，样本在这n次中一次也没有被选到的概率为$(1-\frac{1}{n})^n \to \frac{1}{e} \approx 36.8\% (n \to \infty)$，因此有大约36.8%的数据不会被采样到
  - We can use the remaining data as **hold-out validation set (out-of-bag set)**
- Bootstrap has no theoretical guarantee. But empirically it can reduce test error greatly

Random Forest

- the most successful bagging model

1. For t in $[1, \ldots, T]$
   i. Sample n points $\hat{D}_t$ from D with replacement
   ii. Sample $d' < d$ features $F' \in F$ without replacement
   iii. Build a full decision tree on $\hat{D}_t, F'$ (can do some pruning to minimize out-of-bag error)
2. End for
3. Average all trees

- Simple, easy to implement, very powerful

Boosting

- **reduce bias**
- for linear models or tree models with certain height

Additive Model

- a general paradigm for *Boosting*

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

1. Define $g_t(x) = \sum_{i=1}^{t} \alpha_i f_i(x)$
2. At step t, keep $g_{t-1}(x)$ fixed, then learn α_t, $f_t(x)$ by minimizing $\sum_{i=1}^{n} L(y_i, g_{t-1}(x_i) + \alpha_t f_t(x_i))$
3. Finally, $g(x) = g_T(x)$

Adaboost

General Form

- $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}, y \in \{-1, 1\}, x \in \mathbb{R}^d$
- Input: D, a weak learning algorithm A

1. Initialize sample weights $w_i^{(1)} = \bar{w}_i^{(1)} = \frac{1}{n}, \forall i \in \{1, \ldots, n\}$
2. For t in $[1, \ldots, T]$
    i. use D and $\{w_i^{(1)}\}$ to train A, get a model $f_t(x) : \mathbb{R}^d \to \{-1, 1\}$
    ii. evaluate $f_t(x)$'s weighted classification error e_t on D

$$e_t = \sum_{i=1}^{n} w_i^{(t)} \cdot 1(f_t(x_i) \neq y_i)$$

    iii. compute a weight α_t for $f_t(x)$

$$\alpha_t = \frac{1}{2} \log \frac{1 - e_t}{e_t}$$

    $e_t \searrow, \alpha_t \nearrow$. If $e_t < 0.5, \alpha_t > 0$; if $e_t > 0.5, \alpha_t < 0$ (如果这个模型错误率大于0.5，显然直接结果取反就好了)
    iv. update sample weights $\bar{w}_i^{t+1} = \bar{w}_i^t \cdot \exp(-\alpha_t y_i f_t(x_i))$
        - those $y_i \neq \text{sign}(x_i f_t(x_i))$ get larger weights
    v. normalization

$$w_i^{t+1} = \frac{\bar{w}_i^{t+1}}{\sum_{j=1}^{n} \bar{w}_j^{t+1}}$$

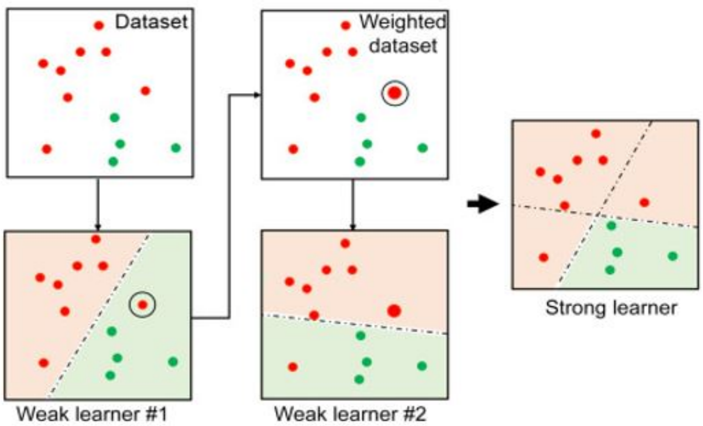
    by normalization, we have:

$$\sum_{i=1}^{n} w_i^{t+1} = 1$$

3. Combine T classifiers linearly

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

- $g(x) \geq 0$, predict $+1$, otherwise predict -1



    - 可以看出，被分错的样本权重增加了，而且多个分类器叠加后就能实现非线性划分的效果

Adaboost Derivation

- Adaboost is an Additive model using exponential loss
    - $L(y, f(x)) = \exp(-y f(x))$
    - In Adaboost, $L(y, f(x)) = \{e, \frac{1}{e}\}$
- At step t, we already have

$$g_{t-1}(x) = \sum_{j=1}^{t-1} \alpha_j f_j(x)$$

We need to optimize

$$\min_{\alpha_t, f_t} \sum_{i=1}^{n} \exp\Big(-y_i\big(g_{t-1}(x_i) + \alpha_t f_t(x_i)\big)\Big)$$

Define

$$\begin{aligned} &\exp\big(-y_i g_{t-1}(x_i)\big) \\ =&\prod_{j=1}^{t-1} \exp\big(-y_i \alpha_j f_j(x_i)\big) \\ =&\bar{w}_i^{(t-1)} \cdot \exp\big(-y_i \alpha_{t-1} f_{t-1}(x_i)\big) \end{aligned}$$

to be $\bar{w}_i^{(t)}$, then we need to optimize

$$\min_{\alpha_t} \min_{f_t} \sum_{i=1}^{n} w_i^{(t)} \cdot \exp\big(-y_i \alpha_t f_t(x_i)\big)$$

- In Adaboost, we just train f_t using its **original loss** and sample weights $w_i^{(t)}$, which means that we fix α_t and optimize f_t
- $\bar{w}_i^{(t)}$ 和 $w_i^{(t)}$只差一个常数

- After solving f_t, we need to optimize

$$\begin{aligned} &\min_{\alpha_t} \sum_{i=1}^{n} w_i^{(t)} \exp\big(-y_i \alpha_t f_t(x_i)\big) \\ \Leftrightarrow &\min_{\alpha_t} \sum_{y_i = f_t(x_i)} w_i^{(t)} \exp\big(-y_i \alpha_t f_t(x_i)\big) + \sum_{y_i \neq f_t(x_i)} w_i^{(t)} \exp\big(-y_i \alpha_t f_t(x_i)\big) \\ \Leftrightarrow &\min_{\alpha_t} \sum_{y_i = f_t(x_i)} w_i^{(t)} \exp(-\alpha_t) + \sum_{y_i \neq f_t(x_i)} w_i^{(t)} \exp(\alpha_t) \\ \Leftrightarrow &\min_{\alpha_t} \exp(-\alpha_t) \sum_{i=1}^{n} w_i^{(t)} + \big(\exp(\alpha_t) - \exp(-\alpha_t)\big) \sum_{y_i \neq f_t(x_i)} w_i^{(t)} \\ \Leftrightarrow &\min_{\alpha_t} \exp(-\alpha_t) + \big(\exp(\alpha_t) - \exp(-\alpha_t)\big) e_t \end{aligned}$$

  - Take gradient with α_t

$$-\exp(-\alpha_t) + \big(\exp(\alpha_t) + \exp(-\alpha_t)\big) e_t = 0$$

  and we can get

$$\alpha_t = \frac{1}{2} \log \frac{1 - e_t}{e_t}$$

Gradient Boosting Model

- For regression, no need for α_t, because α_t can be absorbed into f_t (因为都是实数)
- For squared loss, we need to optimize

$$\min_{f_t} \big(y_i - g_{t-1}(x_i) - f_t(x_i)\big)^2$$

  - we call $r_i^{(t)} = y_i - g_{t-1}(x_i)$ **residual error**
  - we iteratively train a new model f_t to fit the residuals $\{(x_1, r_1^{(t)}), \ldots, (x_n, r_n^{(t)})\}$
- Boosting Tree: Squared loss & Additive model & f_t is a tree
- Gradient Boosting Model
  - instead of fitting residuals, fit the negative gradient of loss, that is, let

$$r_i^{(t)} = -\frac{\partial L(y_i, \hat{y})}{\partial \hat{y}}\bigg|_{\hat{y} = g_{t-1}(x_i)}$$

  - essentially, use new models $\alpha_t f_t$ to perform *gradient descent*
  - for squared loss: $L(y_i, \hat{y}) = \frac{1}{2}(y_i - \hat{y})^2$

$$-\frac{\partial L(y_i, \hat{y})}{\partial \hat{y}} = y_i - \hat{y}$$

XGBoost

- perform second-order optimization & regularization (of tree complexity) & parallelization techniques