4 Boosting

Gradient Boosting is an iterative functional gradient descent algorithm that optimizes a risk function over function space. Let $\mathcal{X}$ be the input space, $\mathcal{Y}$ be the output space. Suppose we wish to find $g=\sum_{t=1}^{T}\alpha_t h_t$ as an ensemble of weak prediction models $h_t \in H$ that minimizes $\hat{\mathcal{R}}_S(g)$, where $\hat{\mathcal{R}}_S$ is an empirical risk function on Span(H) that depends on labeled sample $S=((x_i,y_i))_{i=1}^{m} \in (\mathcal{X}\times\mathcal{Y})^m$. We first initialize $g_1=0$ and for each iteration $t=1,2,...,T$ update $g_{t+1}=g_t+\alpha_t h_t$, where

$$h_t \in \operatorname*{argmin}_{h\in H} \frac{\partial}{\partial\alpha}\hat{\mathcal{R}}_S(g_t+\alpha h)\bigg|_{\alpha=0}, \quad \alpha_t \in \operatorname*{argmin}_{\alpha\in\mathbb{R}} \hat{\mathcal{R}}_S(g_t+\alpha h_t)$$

4.1 AdaBoost

In binary classification problem $\mathcal{Y}=\{\pm 1\}$, AdaBoost aims to minimize the empirical **exponential risk** $\hat{\mathcal{R}}_S^{exp}(g)=\frac{1}{m}\sum_{i=1}^{m} e^{-y_i g(x_i)}$ over $g \in \text{Span}(H)$ for which $H \subset \{\pm 1\}^{\mathcal{X}}$ is a hypothesis set of weak classifiers. Following the procedure of gradient boosting, it first initialize $g_1=0$ and for each iteration $t=1,2,...,T$ update $g_{t+1}=g_t+\alpha_t h_t$, where

$$\begin{aligned}
h_t \in \operatorname*{argmin}_{h\in H} \frac{\partial}{\partial\alpha}\hat{\mathcal{R}}_S^{exp}(g_t+\alpha h)\bigg|_{\alpha=0} &= \operatorname*{argmin}_{h\in H} \frac{\partial}{\partial\alpha}\sum_{i=1}^{m} e^{-y_i g_t(x_i)-\alpha y_i h(x_i)}\bigg|_{\alpha=0} \\
&= \operatorname*{argmin}_{h\in H} -\sum_{i=1}^{m} e^{-y_i g_t(x_i)} y_i h(x_i) = \operatorname*{argmin}_{h\in H} -Z_t \mathbb{E}_{i\sim D_t}[y_i h(x_i)] \\
&= \operatorname*{argmin}_{h\in H} Z_t(2\mathbb{P}_{i\sim D_t}[h(x_i)\neq y_i]-1) = \operatorname*{argmin}_{h\in H} \mathbb{P}_{i\sim D_t}[h(x_i)\neq y_i],
\end{aligned}$$

$$\begin{aligned}
\alpha_t \in \operatorname*{argmin}_{\alpha\in\mathbb{R}} \hat{\mathcal{R}}_S^{exp}(g_t+\alpha h_t) &= \operatorname*{argmin}_{\alpha\in\mathbb{R}} \sum_{i=1}^{m} e^{-y_i g_t(x_i)-\alpha y_i h_t(x_i)} \\
&= \operatorname*{argmin}_{\alpha\in\mathbb{R}} Z_t \mathbb{E}_{i\sim D_t}\left[e^{-\alpha y_i h_t(x_i)}\right] = \operatorname*{argmin}_{\alpha\in\mathbb{R}} Z_t\left(\epsilon_t e^{\alpha}+(1-\epsilon_t)e^{-\alpha}\right) = \left\{\log\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}\right\}
\end{aligned}$$

where $Z_t=\sum_{i=1}^{m} e^{-y_i g_t(x_i)}$, D_t is a probability distribution on $[\![1,m]\!]$ given by $D_t(i)=e^{-y_i g_t(x_i)}/Z_t$, and $\epsilon_t=\mathbb{P}_{i\sim D_t}[h_t(x_i)\neq y_i]$ is the error of h_t on training sample weighted by the distribution D_t. Note that $Z_1=m$ and

$$Z_{t+1}=\sum_{i=1}^{m} e^{-y_i g_{t+1}(x_i)} = \sum_{i=1}^{m} e^{-y_i g_t(x_i)-\alpha_t y_i h_t(x_i)} = Z_t \mathbb{E}_{i\sim D_t}\left[e^{-\alpha_t y_i h_t(x_i)}\right]$$

Denote

$$\gamma_t=\mathbb{E}_{i\sim D_t}\left[e^{-\alpha_t y_i h_t(x_i)}\right]=\epsilon_t e^{\alpha_t}+(1-\epsilon_t)e^{-\alpha_t}=2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Then $Z_{t+1}=\gamma_t Z_t$, and

$$\begin{aligned}
D_{t+1}(i) &= Z_{t+1}^{-1} e^{-y_i g_{t+1}(x_i)} = Z_{t+1}^{-1} e^{-y_i g_t(x_i)-\alpha_t y_i h_t(x_i)} = Z_{t+1}^{-1} Z_t D_t(i) e^{-\alpha_t y_i h_t(x_i)} \\
&= \gamma_t^{-1} D_t(i) e^{-\alpha_t y_i h_t(x_i)}
\end{aligned}$$

This leads to Algorithm.1.

Algorithm 1 AdaBoost

procedure AdaBoost($S = ((x_i, y_i))_{i=1}^m$)
    for $i \leftarrow 1$ to m **do**
        $D_1(i) \leftarrow \frac{1}{m}$
    end for
    for $t \leftarrow 1$ to T **do**
        $h_t \leftarrow$ base classifier in H with small error $\epsilon_t = \mathbb{P}_{i \sim D_t}[h_t(x_i) \neq y_i]$.
        $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
        $\gamma_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$ ▷ normalization factor
        for $i \leftarrow 1$ to m **do**
            $D_{t+1}(i) \leftarrow \gamma_t^{-1} D_t(i) \exp(-\alpha_t y_t h_t(x_i))$
        end for
    end for
    $g \leftarrow \sum_{t=1}^T \alpha_t h_t$
    return g
end procedure