Adaboost

Idea: training $f_2(x)$ on the new training set that fails $f_1(x)$

Way:

$$\varepsilon_1 = \frac{\sum_n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1} \qquad Z_1 = \sum_n u_1^n \qquad \underline{\varepsilon_1 < 0.5}$$

always

ε_1: the error rate of $f_1(x)$ on training set

Changing the example weights from u_1^n to u_2^n such that:

$$\varepsilon_2 = \frac{\sum_n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5 \quad \text{means random}$$

$$\begin{cases} \text{misclassified} & u_2^n \leftarrow u_1^n \times d_1 \quad \text{increase} \\ \text{correctly} & u_2^n \leftarrow u_1^n \div d_1 \quad \text{decrease.} \end{cases}$$

$$\Rightarrow \sum_{f_1(x^n) = \hat{y}^n} u_1 / d = \sum_{f_1(x^n) \neq \hat{y}^n} u_1 \times d$$

$$\Rightarrow Z_1(1-\varepsilon_1)/d_1 = Z_1 \varepsilon_1 d_1$$

$$\Rightarrow d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1}$$

After we obtain a set of functions: $f_1(x), \ldots, f_t(x), \cdots$

Aggregate them.

$$H(x) = \text{sign}\Big(\underbrace{\sum_{t=1}^{T} \alpha_t f_t(x)}_{g(x)}\Big) \qquad \alpha_t = \ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times \exp(-\hat{y}^n f_t(x^n) \alpha_t)$$

T increase, $H(x)$ achieves smaller error rate

Math time:

$$\text{Training data error rate} = \frac{1}{N} \sum \delta(H(x^n) \neq \hat{y}^n)$$

$$= \frac{1}{N} \delta(\hat{y}^n g(x) < 0) \leqslant \frac{1}{N} \sum_n \exp(-\hat{y}^n g(x))$$

$$= \frac{1}{N} Z_{T+1}$$

$$\begin{cases} u_1^n = 1 \\ u_{t+1}^n = u_t^n \times \exp(-\hat{y}^n f_t(x^n)\alpha_t) \end{cases}$$

$$\Rightarrow u_{T+1}^n = \prod_{t=1}^{T} \exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$Z_{T+1} = \sum_n \prod_{t=1}^{T} \exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$= \sum_n \exp\Big(-\hat{y}^n \underbrace{\sum_{t=1}^{T} (f_t(x^n)\alpha_t)}_{g(x)}\Big)$$