Ensemble methods *make dumb learner smart*

- Combine weak learners (regression):

$$H(x_i) = \sum_{j=1}^{T} h_j(x_i)$$

  - Random forest: dumb learners are random trees.
  - Bagging: generates new training sets by sampling.
  - Boosting: adaptively generates new training sets.

☐ Bagging, random forests, boosting are all ensemble methods.

☐ AdaBoost will not lead to *overfitting*, which means the testing error will decrease monotonously during the training time.

☐ The *Random Forests* method fits many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.

Adaboost

proposed by Freund & Schapire' 95 (Gödel prize)

Given: $(\boldsymbol{x}_1, \cdots, y_1), \cdots, (\boldsymbol{x}_m, y_m)$, where $\boldsymbol{x}_i \in X, y_i \in \{-1, +1\}$

Input

Initialize: $D^i(1) = \frac{1}{m}$.

Initially equal weights

For $t \in \{1, \cdots, T\}$:

- Train weak learner using distribution $D(t)$.

  Naive bayes, decision tree

- Get weak classifier $h_t : X \to \{+1, -1\}$
- Choose $\alpha_t \in \mathbb{R}^+$

  Magic number, introduce later

- Update:

  Increase weight if wrong on point i, $y_i h_t(\mathbf{x}_i) = -1 < 0$

$$D^i(t+1) = \frac{D^i(t+1)}{Z_t} \begin{cases} e^{-\alpha_t}, \text{if } y_i = h_t(\boldsymbol{x}_i) \\ e^{\alpha_t}, \text{if } y_i \neq h_t(\boldsymbol{x}_i) \end{cases}$$

$$= \frac{D^i(t) \exp\{-\alpha y_i h_t(\boldsymbol{x}_i)\}}{Z_i}$$

where Z_t is a normalization factor:

$$Z_t = \sum_i D^i(t) \exp\{-\alpha y_i h_t(\boldsymbol{x}_i)\}$$

Normalization factor: $\sum D^i(t+1) = 1$

Adaboost

proposed by Freund & Schapire' 95 (Gödel prize)

Given: $(\boldsymbol{x}_1, \cdots, y_1), \cdots, (\boldsymbol{x}_m, y_m)$, where $\boldsymbol{x}_i \in X, y_i \in \{-1, +1\}$

Input

Initialize: $D^i(1) = \frac{1}{m}$.

Initially equal weights

For $t \in \{1, \cdots, T\}$:

- Train weak learner using distribution $D(t)$.

Naive bayes, decision tree

- Get weak classifier $h_t : X \to \{+1, -1\}$
- Choose $\alpha_t \in \mathbb{R}^+$

Magic number, introduce later

- Update:

Increase weight if wrong on point i, $y_i h_t(\mathbf{x}_i) = -1 < 0$

$$D^i(t+1) = \frac{D^i(t) \exp\{-\alpha y_i h_t(\boldsymbol{x}_i)\}}{Z_i}$$

Out put the ***final classifier***:

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

Choosing weights for weak classifiers

- Weight (for points) update

$$D^i(t+1) = \frac{D^i(t)\exp\{-\alpha_t y_i h_t(\boldsymbol{x}_i)\}}{Z_i}$$

$$\boxed{\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)}$$

Freund & Schapire'95

- Weighted training error:

$$\epsilon_t = \mathbb{E}_{i\sim D(t)}[\mathbb{1}(y_i \neq h_t(\boldsymbol{x}_t))] = \sum_{i=1}^{m} D^i(t)\mathbb{1}(y_i \neq h_t(\boldsymbol{x}_t))$$

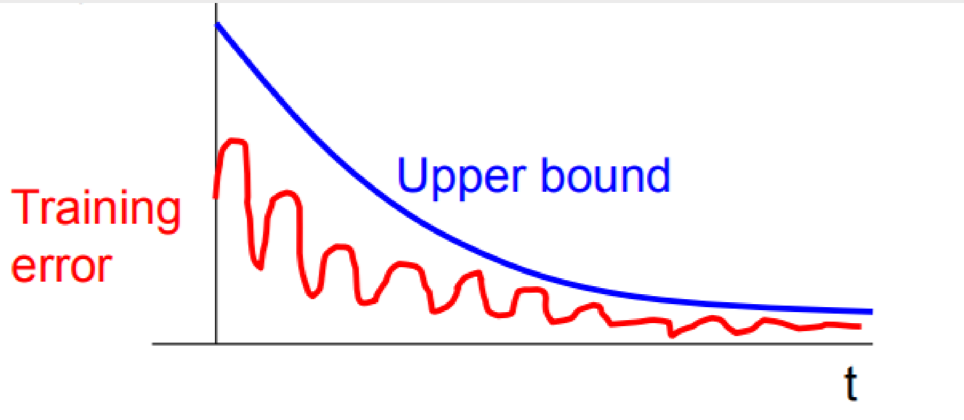
- Recall weaker learner assumption, $\varepsilon_t < 0.5, \alpha_t > 0$

Analyzing training error

- If each weak learner is slightly better than ***random guessing*** (training error $\varepsilon_t < 0.5$) , then ***training error*** of AdaBoost decays ***exponentially fast*** in number of rounds T.

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}(y_t \neq H(\boldsymbol{x}_t)) \leq \exp\left(-2\sum_{t=1}^{T}(\frac{1}{2}-\epsilon_t)^2\right)$$

Training error



☐ Suppose in every AdaBoost iteration we get a *weak* classifier C_m with margin $\gamma > 0$, we can achieve *ZERO* error on training dataset after finite iterations.

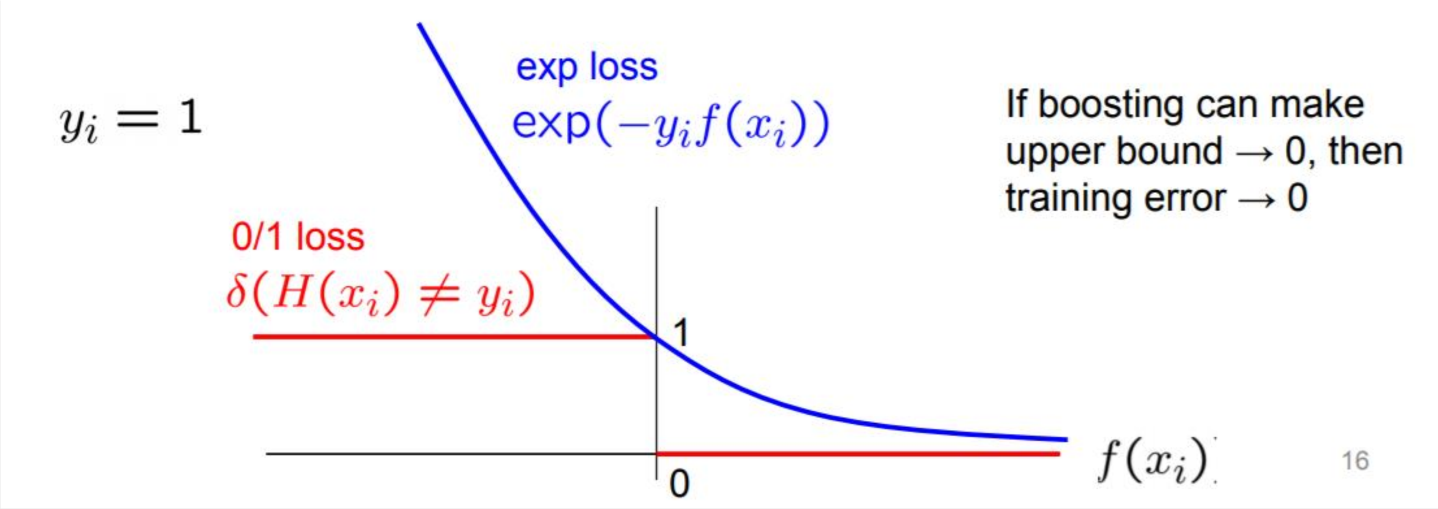
Analyzing training error

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\underline{\mathbb{1}(y_t \neq H(\boldsymbol{x}_t))} \leq \frac{1}{m}\sum_{i=1}^{m}\underline{\exp\{-y_i f(\boldsymbol{x}_i)\}}$$

$\boxed{\text{Convex upper bound}}$

where $f(\boldsymbol{x}) = \sum_{i=1}^{T}\alpha_t h_t(\boldsymbol{x}); H(\boldsymbol{x}) = sign(f(\boldsymbol{x}))$



Analyzing training error

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}(y_t \neq H(\boldsymbol{x}_t)) \leq \underline{\frac{1}{m}\sum_{i=1}^{m}\exp\{-y_i f(\boldsymbol{x}_i)\} = \prod_{t=1}^{T} Z_t}$$

where

$$f(\boldsymbol{x}) = \sum_{i=1}^{T}\alpha_t h_t(\boldsymbol{x}); H(\boldsymbol{x}) = sign(f(\boldsymbol{x}))$$

Proof. **Using weight update rule**

$$D^i(1) = \frac{1}{m}$$

$$D^i(2) = \frac{1}{m}\frac{e^{-\alpha_1 y_i h_1(\boldsymbol{x}_i)}}{Z_1}$$

$$D^i(3) = \frac{1}{m}\frac{e^{-\alpha_1 y_i h_1(\boldsymbol{x}_i)}e^{-\alpha_2 y_i h_2(\boldsymbol{x}_i)}}{Z_1 Z_2}$$

$$D^i(T+1) = \frac{1}{m}\frac{\exp\{-y_i f(\boldsymbol{x}_i)\}}{\prod_t Z_t}$$

Weights summing up to 1

$$\sum_i D^i(T+1) = 1$$

Analyzing training error

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}(y_t \neq H(\boldsymbol{x}_t)) \leq \frac{1}{m}\sum_{i=1}^{m}\exp\{-y_i f(\boldsymbol{x}_i)\} = \prod_{t=1}^{T} Z_t$$

where

$$f(\boldsymbol{x}) = \sum_{i=1}^{T}\alpha_t h_t(\boldsymbol{x}); H(\boldsymbol{x}) = sign(f(\boldsymbol{x}))$$

- **Previous analysis is independent with how we choose α_t and Z_t**
  - We can tighten this bound greedily, by choosing α_t and h_t on each iteration to minimize z_t

What α_t to choose for classifier h_t?

Choosing α_t and h_t on each iteration to minimize Z_t

$$Z_t = \sum_i D^i(t) \exp\{-\alpha y_i h_t(\boldsymbol{x}_i)\}$$

For boolean function, this is accomplished by **[Freund & Schapire '97]:**

$$\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}$$

Proof:

$$\begin{aligned} Z_t &= \sum_{i: y_i \neq h_t(\boldsymbol{x}_t)} D^i(t) \exp\{\alpha_t\} + \sum_{i: y_i = h_t(\boldsymbol{x}_t)} D^i(t) \exp\{-\alpha_t\} \\ &= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} \end{aligned}$$

$$\frac{\partial Z_t}{\partial \alpha_t} = \epsilon_t e^{\alpha_t} - (1-\epsilon_t) e^{-\alpha_t} \Rightarrow e^{2\alpha_t} = \frac{1-\epsilon_t}{\epsilon_t}$$

What α_t to choose for classifier h_t?

Choosing α_t and h_t on each iteration to minimize Z_t

$$Z_t = \sum_i D^i(t) \exp\{-\alpha y_i h_t(\boldsymbol{x}_i)\}$$

For boolean function, this is accomplished by **[Freund & Schapire '97]:**

$$\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}$$

Proof:

$$\begin{aligned} Z_t &= \sum_{i: y_i \neq h_t(\boldsymbol{x}_t)} D^i(t) \exp\{\alpha_t\} + \sum_{i: y_i = h_t(\boldsymbol{x}_t)} D^i(t) \exp\{-\alpha_t\} \\ &= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-2(1-\epsilon_t)^2} \end{aligned}$$

Dumb classifiers made smart!!!

- Training error

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{1}(y_i \neq H(\boldsymbol{x}_i)) \leq \prod_{t=1}^{T} Z_t \leq \prod_{t}^{T}\sqrt{1-(1-2\epsilon_t)^2}$$
$$\leq \exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

If each classifier is (at least slightly) ***better than random guess***, Adaboost will achieve zero training error ***exponentially fast*** (in terms of rounds T)!!

4 [20 pts] Boosting

Consider a variant of AdaBoost in which the combined classifier is replaced by a classifier $\tilde{H}$ whose predictions are randomized; specifically, suppose, for any $\mathbf{x}$, that $\tilde{H}$ predicts +1 with probability:

$$Pr[\tilde{H}(\mathbf{x}) = +1] = \frac{e^{F(\mathbf{x})}}{e^{F(\mathbf{x})} + e^{-F(\mathbf{x})}}, \quad F(\mathbf{x}) := \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

where h_t is the t-th classifier and α_t is the weight of h_t. Prove that

$$\frac{1}{n} \sum_{i=1}^{n} Pr[\tilde{H}(\mathbf{x}_i) \neq y_i] \leq \frac{1}{2} \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2}$$

where $\gamma_t = 1/2 - \epsilon_t$, ϵ_t is the error of the t-th base classifier h_t and $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$ are data points.

Hint: Use the equality $\mathbb{1}[y_i = +1]e^{-F(\mathbf{x}_i)} + \mathbb{1}[y_i = -1]e^{F(\mathbf{x}_i)} = e^{-y_i F(\mathbf{x}_i)}$.

$$
\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} Pr[\tilde{H}(\mathbf{x}_i) \neq y_i] &= \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbb{1}[y_i = +1]e^{-F(\mathbf{x}_i)} + \mathbb{1}[y_i = -1]e^{F(\mathbf{x}_i)}}{e^{F(\mathbf{x}_i)} + e^{-F(\mathbf{x}_i)}} \\
&= \frac{1}{n}\sum_{i=1}^{n} \frac{e^{-y_i F(\mathbf{x}_i)}}{e^{F(\mathbf{x}_i)} + e^{-F(\mathbf{x}_i)}} \\
&\leq= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2} e^{-y_i F(\mathbf{x}_i)}
\end{aligned}
$$