Boosting

CSci 5525: Machine Learning

Instructor: Arindam Banerjee

October 9, 2013

Weak Learning

- A weak learner predicts "slightly better" than random

Weak Learning

- A weak learner predicts "slightly better" than random
- The PAC setting

Weak Learning

- A weak learner predicts "slightly better" than random
- The PAC setting
  - Let $\mathcal{X}$ be an instance space, $c : \mathcal{X} \mapsto \{0, 1\}$ be a target concept, $\mathcal{H}$ be a hypothesis space ($h : \mathcal{X} \mapsto \{0, 1\}$)

Weak Learning

- A weak learner predicts "slightly better" than random
- The PAC setting
  - Let $\mathcal{X}$ be an instance space, $c : \mathcal{X} \mapsto \{0, 1\}$ be a target concept, $\mathcal{H}$ be a hypothesis space ($h : \mathcal{X} \mapsto \{0, 1\}$)
  - Let D be a fixed (but unknown) distribution on $\mathcal{X}$

Weak Learning

- A weak learner predicts "slightly better" than random
- The PAC setting
  - Let $\mathcal{X}$ be an instance space, $c : \mathcal{X} \mapsto \{0, 1\}$ be a target concept, $\mathcal{H}$ be a hypothesis space ($h : \mathcal{X} \mapsto \{0, 1\}$)
  - Let D be a fixed (but unknown) distribution on $\mathcal{X}$
- An algorithm, after training on $(x_i, c(x_i)), [i]_1^m$, selects $h \in \mathcal{H}$ such that

$$P_{x \sim D}[h(x) \neq c(x)] \leq \frac{1}{2} - \gamma$$

Weak Learning

- A weak learner predicts "slightly better" than random
- The PAC setting
  - Let $\mathcal{X}$ be an instance space, $c : \mathcal{X} \mapsto \{0,1\}$ be a target concept, $\mathcal{H}$ be a hypothesis space ($h : \mathcal{X} \mapsto \{0,1\}$)
  - Let D be a fixed (but unknown) distribution on $\mathcal{X}$
- An algorithm, after training on $(x_i, c(x_i)), [i]_1^m$, selects $h \in \mathcal{H}$ such that

$$P_{x \sim D}[h(x) \neq c(x)] \;\leq\; \frac{1}{2} - \gamma$$

- The algorithm is called a γ-weak learner

Weak Learning

- A weak learner predicts "slightly better" than random
- The PAC setting
  - Let $\mathcal{X}$ be an instance space, $c : \mathcal{X} \mapsto \{0, 1\}$ be a target concept, $\mathcal{H}$ be a hypothesis space $(h : \mathcal{X} \mapsto \{0, 1\})$
  - Let D be a fixed (but unknown) distribution on $\mathcal{X}$
- An algorithm, after training on $(x_i, c(x_i)), [i]_1^m$, selects $h \in \mathcal{H}$ such that

$$P_{x \sim D}[h(x) \neq c(x)] \leq \frac{1}{2} - \gamma$$

- The algorithm is called a γ-weak learner
- We assume the existence of such a learner

The Boosting Model

- Boosting converts a weak learner to a strong learner

The Boosting Model

- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds

The Boosting Model

- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds
  - Booster constructs D_t on X, the train set

The Boosting Model

- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds
  - Booster constructs D_t on X, the train set
  - Weak learner produces a hypothesis $h_t \in \mathcal{H}$ so that $P_{x \sim D_t}[h_t(x) \neq c(x)] \leq \frac{1}{2} - \gamma_t$

The Boosting Model

- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds
  - Booster constructs D_t on X, the train set
  - Weak learner produces a hypothesis $h_t \in \mathcal{H}$ so that $P_{x \sim D_t}[h_t(x) \neq c(x)] \leq \frac{1}{2} - \gamma_t$
  - After T rounds, the weak hypotheses $h_t, [t]_1^T$ are combined into a final hypothesis h_{final}

The Boosting Model

- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds
  - Booster constructs D_t on X, the train set
  - Weak learner produces a hypothesis $h_t \in \mathcal{H}$ so that $P_{x \sim D_t}[h_t(x) \neq c(x)] \leq \frac{1}{2} - \gamma_t$
  - After T rounds, the weak hypotheses $h_t, [t]_1^T$ are combined into a final hypothesis h_{final}
- We need procedures

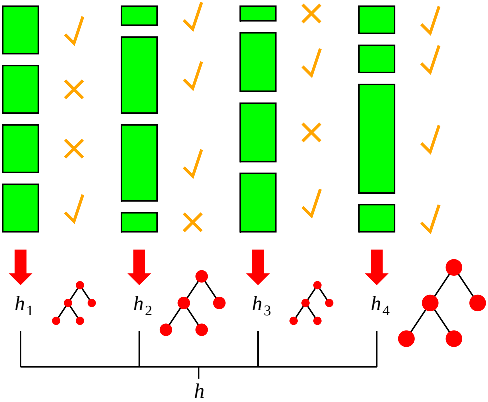
The Boosting Model

- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds
  - Booster constructs D_t on X, the train set
  - Weak learner produces a hypothesis $h_t \in \mathcal{H}$ so that $P_{x \sim D_t}[h_t(x) \neq c(x)] \leq \frac{1}{2} - \gamma_t$
  - After T rounds, the weak hypotheses $h_t, [t]_1^T$ are combined into a final hypothesis h_{final}
- We need procedures
  - for obtaining D_t at each step

The Boosting Model

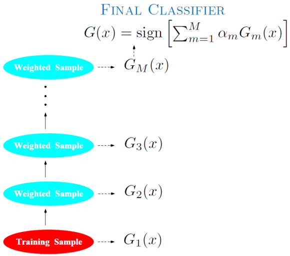
- Boosting converts a weak learner to a strong learner
- Boosting proceeds in rounds
  - Booster constructs D_t on X, the train set
  - Weak learner produces a hypothesis $h_t \in \mathcal{H}$ so that $P_{x \sim D_t}[h_t(x) \neq c(x)] \leq \frac{1}{2} - \gamma_t$
  - After T rounds, the weak hypotheses $h_t, [t]_1^T$ are combined into a final hypothesis h_{final}
- We need procedures
  - for obtaining D_t at each step
  - for combining the weak hypotheses

Boosting Algorithms



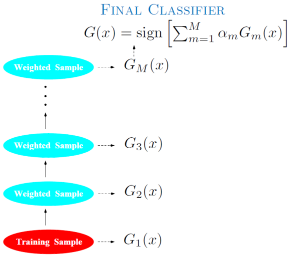
- Weight decreased on correct samples
- Weight increased on incorrect samples

Adaboost Training



- Weight on $(\mathbf{x}_i, y_i)$ is $D_t(i) = w_t(i)$, learn classifier $G_t(\mathbf{x})$

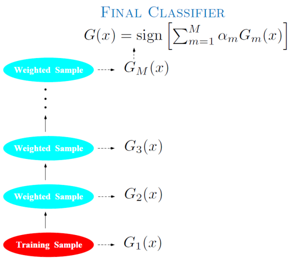
Adaboost Training



- Weight on $(\mathbf{x}_i, y_i)$ is $D_t(i) = w_t(i)$, learn classifier $G_t(\mathbf{x})$
- The error rate

$$\epsilon_t = P_{\mathbf{x} \sim w_t}[G_t(\mathbf{x}) \neq y] = \frac{\sum_{i=1}^{N} w_t(i) \mathbb{I}(y_i \neq G_t(\mathbf{x}_i))}{\sum_{i=1}^{N} w_t(i)}$$

Adaboost Training



- Weight on $(\mathbf{x}_i, y_i)$ is $D_t(i) = w_t(i)$, learn classifier $G_t(\mathbf{x})$
- The error rate

$$\epsilon_t = P_{\mathbf{x}\sim w_t}[G_t(\mathbf{x}) \neq y] = \frac{\sum_{i=1}^{N} w_t(i)\mathbb{I}(y_i \neq G_t(\mathbf{x}_i))}{\sum_{i=1}^{N} w_t(i)}$$

- The combined classifier

$$g(\mathbf{x}) = \text{sign}\left[\sum_{t=1}^{T} \alpha_t G_t(\mathbf{x})\right]$$

Adaboost Algorithm

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Algorithm: Initialize $w_1(i) = 1/N$
For $t = 1, \ldots, T$

- Train a weak learner using distribution w_t

Adaboost Algorithm

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Algorithm: Initialize $w_1(i) = 1/N$
For $t = 1, \ldots, T$

- Train a weak learner using distribution w_t
- Get weak hypothesis G_t with error $\epsilon_t = \Pr_{\mathbf{x} \sim w_t}[G_t(\mathbf{x}) \neq y]$

Adaboost Algorithm

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Algorithm: Initialize $w_1(i) = 1/N$
For $t = 1, \ldots, T$

- Train a weak learner using distribution w_t
- Get weak hypothesis G_t with error $\epsilon_t = \Pr_{\mathbf{x} \sim w_t}[G_t(\mathbf{x}) \neq y]$
- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

Adaboost Algorithm

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Algorithm: Initialize $w_1(i) = 1/N$

For $t = 1, \ldots, T$

- Train a weak learner using distribution w_t
- Get weak hypothesis G_t with error $\epsilon_t = \Pr_{\mathbf{x} \sim w_t}[G_t(\mathbf{x}) \neq y]$
- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- Update

$$w_{t+1}(i) = \frac{w_t(i) \exp(-\alpha_t y_i G_t(\mathbf{x}_i))}{Z_t}$$

where Z_t is the normalization factor

Adaboost Algorithm

Input: Training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Algorithm: Initialize $w_1(i) = 1/N$
For $t = 1, \ldots, T$

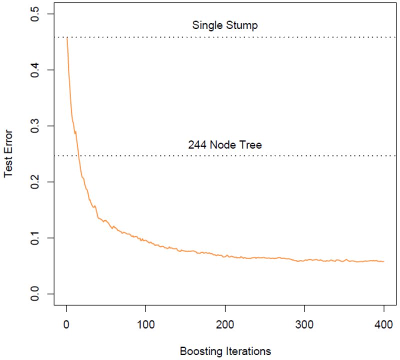
- Train a weak learner using distribution w_t
- Get weak hypothesis G_t with error $\epsilon_t = \Pr_{\mathbf{x} \sim w_t}[G_t(\mathbf{x}) \neq y]$
- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
- Update

$$w_{t+1}(i) = \frac{w_t(i) \exp(-\alpha_t y_i G_t(\mathbf{x}_i))}{Z_t}$$

where Z_t is the normalization factor

Output: $g(\mathbf{x}) = \text{sign} \left[\sum_{t=1}^{T} \alpha_t G_t(\mathbf{x}) \right]$

Adaboost Example

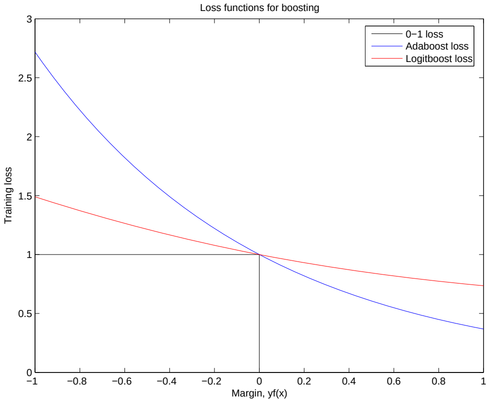


- $X_1, \ldots, X_{10}$ are univariate independent Gaussians

$$Y = \begin{cases} 1 & \text{if } \sum_j X_j^2 > \chi_{10}^2(0.5) \\ -1 & \text{otherwise} \end{cases}$$

- $\chi_{10}^2(0.5) = 9.34$, median of chi-squared r.v. with 10 degrees of freedom

Training Error, Margin Error



The 0-1 training set loss with convex upper bounds: exponential loss and logistic loss

The Training Error

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

The Training Error

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i)\neq y_i)\leq\frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i))=\prod_{t=1}^{T} Z_t$$

- Training error can be reduced most rapidly by choosing α_t that minimizes

$$Z_t=\sum_{i=1}^{N} w_t(i)\exp(-\alpha_t y_i G_t(\mathbf{x}_i))$$

The Training Error

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

- Training error can be reduced most rapidly by choosing α_t that minimizes

$$Z_t = \sum_{i=1}^{N} w_t(i)\exp(-\alpha_t y_i G_t(\mathbf{x}_i))$$

- Adaboost chooses the optimal α_t

The Training Error

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i)\neq y_i)\leq\frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i))=\prod_{t=1}^{T} Z_t$$

- Training error can be reduced most rapidly by choosing α_t that minimizes

$$Z_t=\sum_{i=1}^{N} w_t(i)\exp(-\alpha_t y_i G_t(\mathbf{x}_i))$$

- Adaboost chooses the optimal α_t
    - Margin is (sort of) maximized

The Training Error

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

- Training error can be reduced most rapidly by choosing α_t that minimizes

$$Z_t = \sum_{i=1}^{N} w_t(i)\exp(-\alpha_t y_i G_t(\mathbf{x}_i))$$

- Adaboost chooses the optimal α_t
  - Margin is (sort of) maximized
- Other boosting algorithms minimize other upper bounds

Obtaining α_t

- For a given $G_t(\mathbf{x})$, goal is to minimize

$$f(\alpha) = \sum_{i=1}^{N} w_t(i) \exp(-\alpha y_i G_t(\mathbf{x}_i)) = e^{-\alpha} \sum_{y_i = G_t(\mathbf{x}_i)} w_t(i) + e^{\alpha} \sum_{y_i \neq G(\mathbf{x}_i)} w_t(i)$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_t(i) \mathbb{1}(y_i \neq G_t(\mathbf{x}_i)) + e^{-\alpha} \sum_{i=1}^{n} w_t(i)$$

Obtaining α_t

- For a given $G_t(\mathbf{x})$, goal is to minimize

$$f(\alpha) = \sum_{i=1}^{N} w_t(i) \exp(-\alpha y_i G_t(\mathbf{x}_i)) = e^{-\alpha} \sum_{y_i = G_t(\mathbf{x}_i)} w_t(i) + e^{\alpha} \sum_{y_i \neq G(\mathbf{x}_i)} w_t(i)$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_t(i) \mathbb{1}(y_i \neq G_t(\mathbf{x}_i)) + e^{-\alpha} \sum_{i=1}^{n} w_t(i)$$

- Minimization over α gives

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

Obtaining α_t

- For a given $G_t(\mathbf{x})$, goal is to minimize

$$f(\alpha) = \sum_{i=1}^{N} w_t(i) \exp(-\alpha y_i G_t(\mathbf{x}_i)) = e^{-\alpha} \sum_{y_i = G_t(\mathbf{x}_i)} w_t(i) + e^{\alpha} \sum_{y_i \neq G(\mathbf{x}_i)} w_t(i)$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_t(i) \mathbb{1}(y_i \neq G_t(\mathbf{x}_i)) + e^{-\alpha} \sum_{i=1}^{n} w_t(i)$$

- Minimization over α gives

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

- The error rate is given by

$$\epsilon_t = \frac{\sum_{i=1}^{N} w_t(i) \mathbb{1}(y_i \neq G_t(\mathbf{x}_i))}{\sum_{t=1}^{N} w_t(i)}$$

The Training Error (Contd)

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

The Training Error (Contd)

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(g(\mathbf{x}_i)\neq y_i)\leq\frac{1}{N}\sum_{i=1}^{N}\exp(-y_i G(\mathbf{x}_i))=\prod_{t=1}^{T} Z_t$$

- For round t, with $\epsilon_t = 1/2 - \gamma_t/2$

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-\gamma_t^2}$$

The Training Error (Contd)

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

- For round t, with $\epsilon_t = 1/2 - \gamma_t/2$

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-\gamma_t^2}$$

- Then the total training error

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \prod_{t=1}^{T} \sqrt{1-\gamma_t^2} \leq \exp(-\frac{1}{2}\sum_{t} \gamma_t^2)$$

The Training Error (Contd)

- The training error of the final classifier is bounded

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i G(\mathbf{x}_i)) = \prod_{t=1}^{T} Z_t$$

- For round t, with $\epsilon_t = 1/2 - \gamma_t/2$

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-\gamma_t^2}$$

- Then the total training error

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(\mathbf{x}_i) \neq y_i) \leq \prod_{t=1}^{T} \sqrt{1-\gamma_t^2} \leq \exp(-\frac{1}{2}\sum_t \gamma_t^2)$$

- A more careful argument gives the margin error

$$\frac{1}{m}\sum_i \mathbb{I}_{[y_i h(\mathbf{x}_i) \leq \theta]} \leq \prod_{t=1}^{T} (1-\gamma_t)^{\frac{1-\theta}{2}} (1+\gamma_t)^{\frac{1+\theta}{2}}$$

Boosting as Entropy Projection

For a general α we have

$$D_{t+1}^{\alpha}(i) = \frac{D_t(i)\exp(-\alpha y_i h_t(\mathbf{x}_i))}{Z_t(\alpha)}$$

where $Z_t(\alpha) = \sum_{i=1}^{m} \exp(-\alpha y_i h_t(\mathbf{x}_i))$.

Boosting as Entropy Projection

For a general α we have

$$D_{t+1}^{\alpha}(i) = \frac{D_t(i)\exp(-\alpha y_i h_t(\mathbf{x}_i))}{Z_t(\alpha)}$$

where $Z_t(\alpha) = \sum_{i=1}^{m} \exp(-\alpha y_i h_t(\mathbf{x}_i))$.
Then

$$\min_{D_{t+1}, E_{D_{t+1}}[yh(\mathbf{x})]=0} KL(D_{t+1}, D_t) = \max_{\alpha} \left(-\ln Z_t(\alpha)\right) .$$

Boosting as Entropy Projection

For a general α we have

$$D_{t+1}^{\alpha}(i) = \frac{D_t(i)\exp(-\alpha y_i h_t(\mathbf{x}_i))}{Z_t(\alpha)}$$

where $Z_t(\alpha) = \sum_{i=1}^{m} \exp(-\alpha y_i h_t(\mathbf{x}_i))$.
Then

$$\min_{D_{t+1}, E_{D_{t+1}}[yh(\mathbf{x})]=0} KL(D_{t+1}, D_t) = \max_{\alpha} \left(-\ln Z_t(\alpha)\right) .$$

Further

$$\operatorname{argmin}_{D_{t+1}, E_{D_{t+1}}[yh(\mathbf{x})]=0} KL(D_{t+1}, D_t) = D_{t+1}^{\alpha_t} ,$$

which is the update Adaboost uses.

Additive Models

- An additive model

$$H_T(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x}) \ .$$

Additive Models

- An additive model
$$H_T(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x}) \ .$$
- Fit the t-th component to minimize the "residual" error

Additive Models

- An additive model
$$H_T(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x}) \ .$$
- Fit the t-th component to minimize the "residual" error
- Error is measured by $C(yh(\mathbf{x}))$, an upper bound on $[y \neq h(\mathbf{x})]$

Additive Models

- An additive model

$$H_T(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x}) \ .$$

- Fit the t-th component to minimize the "residual" error
- Error is measured by $C(yh(\mathbf{x}))$, an upper bound on $[y \neq h(\mathbf{x})]$
- For Adaboost,

$$C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$$

Additive Models

- An additive model

$$H_T(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x}) \ .$$

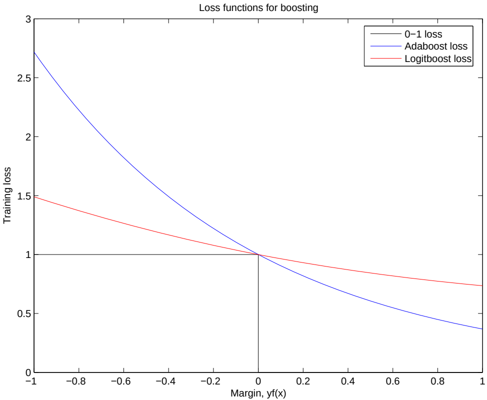
- Fit the t-th component to minimize the "residual" error
- Error is measured by $C(yh(\mathbf{x}))$, an upper bound on $[y \neq h(\mathbf{x})]$
- For Adaboost,

$$C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$$

- For Logitboost,

$$C(yh(\mathbf{x})) = \log(1 + \exp(-yh(\mathbf{x})))$$

Upper Bounds on Training Errors



The 0-1 training set loss with convex upper bounds: exponential loss and logistic loss

Adaboost as Gradient Descent

- Margin Cost function $C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$

Adaboost as Gradient Descent

- Margin Cost function $C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$

Adaboost as Gradient Descent

- Margin Cost function $C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$
- Next classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$

Adaboost as Gradient Descent

- Margin Cost function $C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$
- Next classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$
- The cost function to be minimized

$$\sum_{i=1}^{N} C(y_i[H_{t-1}(\mathbf{x}_i) + \alpha_t h_t(\mathbf{x}_i)]) = \sum_{i=1}^{N} u_t(i) \exp(-y_i \alpha_t h_t(\mathbf{x}_i))$$

where $u_t(i) = \exp(-y_i H_{t-1}(\mathbf{x}_i))$

Adaboost as Gradient Descent

- Margin Cost function $C(yh(\mathbf{x})) = \exp(-yh(\mathbf{x}))$
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$
- Next classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$
- The cost function to be minimized

$$\sum_{i=1}^{N} C(y_i[H_{t-1}(\mathbf{x}_i) + \alpha_t h_t(\mathbf{x}_i)]) = \sum_{i=1}^{N} u_t(i) \exp(-y_i \alpha_t h_t(\mathbf{x}_i))$$

where $u_t(i) = \exp(-y_i H_{t-1}(\mathbf{x}_i))$

- The optimum solution

$$\alpha_t = \frac{1}{2} \ln \left(\frac{\sum_{i:h_t(\mathbf{x}_i)=y_i} u_t(i)}{\sum_{i:h_t(\mathbf{x}_i)\neq y_i} u_t(i)} \right) = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$$

Gradient Descent in Function Space

- Choose any appropriate margin cost function C

Gradient Descent in Function Space

- Choose any appropriate margin cost function C
- Goal is to minimize margin cost on the training set

Gradient Descent in Function Space

- Choose any appropriate margin cost function C
- Goal is to minimize margin cost on the training set
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$

Gradient Descent in Function Space

- Choose any appropriate margin cost function C
- Goal is to minimize margin cost on the training set
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$
- Next classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$

Gradient Descent in Function Space

- Choose any appropriate margin cost function C
- Goal is to minimize margin cost on the training set
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$
- Next classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$
- Choose α_t to minimize

$$C(H_t(\mathbf{x})) = C(H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})) \ .$$

Gradient Descent in Function Space

- Choose any appropriate margin cost function C
- Goal is to minimize margin cost on the training set
- Current classifier $H_{t-1}(\mathbf{x})$, next weak learner $h_t(\mathbf{x})$
- Next classifier $H_t(\mathbf{x}) = H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})$
- Choose α_t to minimize

$$C(H_t(\mathbf{x})) = C(H_{t-1}(\mathbf{x}) + \alpha_t h_t(\mathbf{x})) \ .$$

- "Anyboost" class of algorithms

Margin Bounds: Why Boosting works

- Test set error decreases even after train set error is 0

Margin Bounds: Why Boosting works

- Test set error decreases even after train set error is 0
- No simple “bias-variance” explanation

Margin Bounds: Why Boosting works

- Test set error decreases even after train set error is 0
- No simple "bias-variance" explanation

Bounds on the error rate of the boosted classifier depends on

- the number of training examples

Margin Bounds: Why Boosting works

- Test set error decreases even after train set error is 0
- No simple "bias-variance" explanation

Bounds on the error rate of the boosted classifier depends on

- the number of training examples
- the margin achieved on the train set

Margin Bounds: Why Boosting works

- Test set error decreases even after train set error is 0
- No simple "bias-variance" explanation

Bounds on the error rate of the boosted classifier depends on

- the number of training examples
- the margin achieved on the train set
- the complexity of the weak learners

Margin Bounds: Why Boosting works

- Test set error decreases even after train set error is 0
- No simple "bias-variance" explanation

Bounds on the error rate of the boosted classifier depends on

- the number of training examples
- the margin achieved on the train set
- the complexity of the weak learners

It does not depend on the number of classifiers combined

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \;\leq\; \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \;\leq\; \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1 - \delta)$

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \;\leq\; \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1-\delta)$
  - Bound holds for all $f \in F$, a class of functions

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \; \leq \; \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1 - \delta)$
  - Bound holds for all $f \in F$, a class of functions
  - $C_n(F)$ is some measure of complexity of F

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \ \leq \ \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1 - \delta)$
  - Bound holds for all $f \in F$, a class of functions
  - $C_n(F)$ is some measure of complexity of F
- Boosting analysis uses Rademacher complexity $R_n(F)$

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \;\leq\; \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1 - \delta)$
  - Bound holds for all $f \in F$, a class of functions
  - $C_n(F)$ is some measure of complexity of F
- Boosting analysis uses Rademacher complexity $R_n(F)$
  - Best alignment with randomly chosen class labels

$$R_n(F) = E\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \rho_i f(x_i)\right]$$

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \;\leq\; \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1 - \delta)$
  - Bound holds for all $f \in F$, a class of functions
  - $C_n(F)$ is some measure of complexity of F
- Boosting analysis uses Rademacher complexity $R_n(F)$
  - Best alignment with randomly chosen class labels

$$R_n(F) = E\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \rho_i f(x_i)\right]$$

  - Rademacher variables $\rho_i \in \{-1, +1\}$ with prob $\{1/2, 1/2\}$

Why Boosting Works: Rademacher Complexity

- Margin bounds on error rate based on n samples

$$\Pr[yf(\mathbf{x}) \leq 0] \ \leq \ \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{C_n(F)}{\theta}\right) + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

  - Holds with probability $(1 - \delta)$
  - Bound holds for all $f \in F$, a class of functions
  - $C_n(F)$ is some measure of complexity of F
- Boosting analysis uses Rademacher complexity $R_n(F)$
  - Best alignment with randomly chosen class labels

$$R_n(F) = E\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \rho_i f(x_i)\right]$$

  - Rademacher variables $\rho_i \in \{-1, +1\}$ with prob $\{1/2, 1/2\}$
  - Expectation is over samples $S = \{x_i\}$ and $B = \{\rho_i\}$

Margin Bound for Boosting

- F: Binary classifiers of Rademacher complexity $R_n(F)$

Margin Bound for Boosting

- F: Binary classifiers of Rademacher complexity $R_n(F)$
- Interestingly, $R_n(\text{co}_k(F)) = R_n(F), \ \forall k$, where

$$\text{co}_k(F) = \left\{ f : f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i f_i(\mathbf{x}), f_i \in F, \alpha_i \geq 0, \sum_i \alpha_{i=1}^{k} = 1 \right\}$$

Margin Bound for Boosting

- F: Binary classifiers of Rademacher complexity $R_n(F)$
- Interestingly, $R_n(\text{co}_k(F)) = R_n(F)$, $\forall k$, where

$$\text{co}_k(F) = \left\{ f : f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i f_i(\mathbf{x}), f_i \in F, \alpha_i \geq 0, \sum_i \alpha_{i=1}^k = 1 \right\}$$

- Complexity does not change with convex combination

Margin Bound for Boosting

- F: Binary classifiers of Rademacher complexity $R_n(F)$
- Interestingly, $R_n(\text{co}_k(F)) = R_n(F)$, $\forall k$, where

$$\text{co}_k(F) = \left\{ f : f(\mathbf{x}) = \sum_{i=1}^{k} \alpha_i f_i(\mathbf{x}), f_i \in F, \alpha_i \geq 0, \sum_i \alpha_{i=1}^k = 1 \right\}$$

- Complexity does not change with convex combination
- With probability at least $(1-\delta)$ over the draw of train set S of size N, $\forall f \in \text{co}_k(\mathcal{F}), \theta > 0$ we have

$$\Pr[yf(\mathbf{x}) \leq 0] \;\leq\; \Pr_S[yf(\mathbf{x}) \leq \theta] + \frac{4R_n(F)}{\theta} + O\left(\sqrt{\frac{\log \frac{1}{\delta}}{n}}\right)$$

Maximizing the Margin

- Do boosting algorithms maximize the margin?

Maximizing the Margin

- Do boosting algorithms maximize the margin?
    - Minimize upper bound on training error

Maximizing the Margin

- Do boosting algorithms maximize the margin?
  - Minimize upper bound on training error
  - Often get large margin in the process

Maximizing the Margin

- Do boosting algorithms maximize the margin?
  - Minimize upper bound on training error
  - Often get large margin in the process
  - Do not explicitly maximize the margin

Maximizing the Margin

- Do boosting algorithms maximize the margin?
  - Minimize upper bound on training error
  - Often get large margin in the process
  - Do not explicitly maximize the margin
- Explicit margin maximization

Maximizing the Margin

- Do boosting algorithms maximize the margin?
  - Minimize upper bound on training error
  - Often get large margin in the process
  - Do not explicitly maximize the margin
- Explicit margin maximization
  - Classical Boost: Minimize bound on $\Pr_S[yh(\mathbf{x}) \leq 0]$

Maximizing the Margin

- Do boosting algorithms maximize the margin?
  - Minimize upper bound on training error
  - Often get large margin in the process
  - Do not explicitly maximize the margin
- Explicit margin maximization
  - Classical Boost: Minimize bound on $\Pr_S[yh(\mathbf{x}) \leq 0]$
  - Margin Boost: Minimize (bound on) $\Pr_S[yh(\mathbf{x}) \leq \theta]$