ONLINE MASTERS IN **DATA SCIENCE**

DSC 255 - MACHINE LEARNING FUNDAMENTALS

GENERALIZATION IN BOOSTING

SANJOY DASGUPTA, PROFESSOR

UC San Diego
COMPUTER SCIENCE & ENGINEERING
HALICIOĞLU DATA SCIENCE INSTITUTE

AdaBoost

Data set $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})$, labels $y^{(i)} \in \{-1, +1\}$.

1. Initialize $D_1(i) = 1/n$ for all $i = 1, 2, \dots, n$

2. For $t = 1, 2, \dots, T$:
   - Give D_t to weak learner, get back some $h_t : \mathcal{X} \longrightarrow [-1, 1]$
   - Compute h_t's margin of correctness:

$$r_t = \sum_{i=1}^{n} D_t(i)\, y^{(i)} h_t(x^{(i)}) \in [-1, 1]$$

$$\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$$

   - Update weights: $D_{t+1}(i) \propto D_t(i)\, exp(-\alpha_t y^{(i)} h_t(x^{(i)}))$

3. Final classifier: $H(x) = \mathrm{sign}(\sum_{t=1}^{T} \alpha_t h_t(x))$

The Surprising Power of Weak Learning

Suppose that on each round t, the weak learner returns a rule h_t whose error on the time-t weighted data distribution is $\leq 1/2 - \gamma$.

Then, after T rounds, the training error of the combined rule

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

Is at most $e^{-\gamma^2 T/2}$.

The Surprising Power of Weak Learning

Suppose that on each round t, the weak learner returns a rule h_t whose error on the time-t weighted data distribution is $\leq 1/2 - \gamma$.

Then, after T rounds, the training error of the combined rule

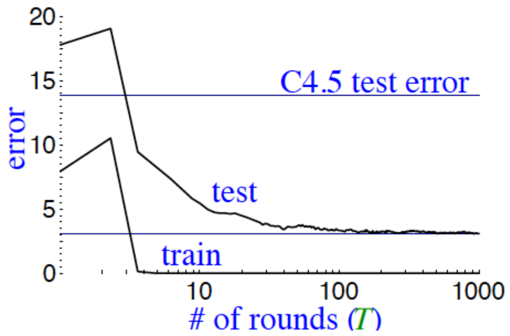
$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Is at most $e^{-\gamma^2 T/2}$.

Presumably, there will come a time T at which further reductions in training error are simply overfitting and will cause test error to rise?

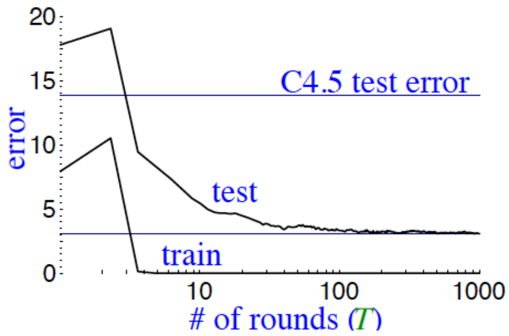
Overfitting?

Freund and Schapire: boosting decision trees for "letter" dataset.



Overfitting?

Freund and Schapire: boosting decision trees for "letter" dataset.



- After 1000 rounds: total size is over 2 million nodes
- Test error keeps dropping even after training error is zero:

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

Looking at the Margin

Final classifier with weights normalized to sum to 1:

$$H(x) = sign\underbrace{\left(\frac{\sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}\right)}_{\text{call this } f(x)}$$

Looking at the Margin

Final classifier with weights normalized to sum to 1:

$$H(x) = sign\underbrace{\left(\frac{\sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}\right)}_{\text{call this } f(x)}$$

Margin of this classifier on data point $(x, y) \in \mathcal{X} \times \{-1, 1\}$:

(fraction of votes correct) - (fraction incorrect) = $yf(x) \in [-1, 1]$.

Looking at the Margin

Final classifier with weights normalized to sum to 1:

$$H(x) = sign\underbrace{\left(\frac{\sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}\right)}_{\text{call this } f(x)}$$

Margin of this classifier on data point $(x, y) \in \mathcal{X} \times \{-1, 1\}$:

(fraction of votes correct) - (fraction incorrect) = $yf(x) \in [-1, 1]$.

- Intuitively and mathematically: the larger a classifier's margins on the training data, the better its generalization.

Looking at the Margin

Final classifier with weights normalized to sum to 1:

$$H(x) = sign\underbrace{\left(\frac{\sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}\right)}_{\text{call this } f(x)}$$

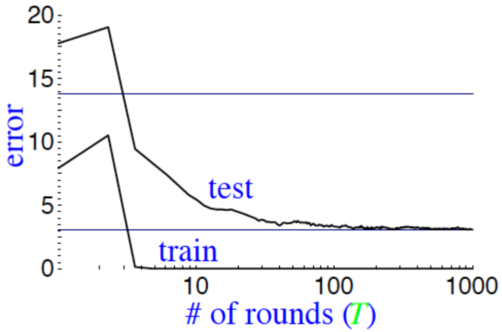
Margin of this classifier on data point $(x, y) \in \mathcal{X} \times \{-1, 1\}$:

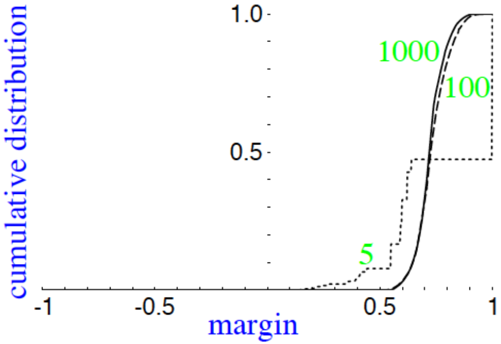
(fraction of votes correct) - (fraction incorrect) = $yf(x) \in [-1, 1]$.

- Intuitively and mathematically: the larger a classifier's margins on the training data, the better its generalization.
- Adaboost seems to increase the margins on the training points even after training error has gone to zero.

Example Revisited

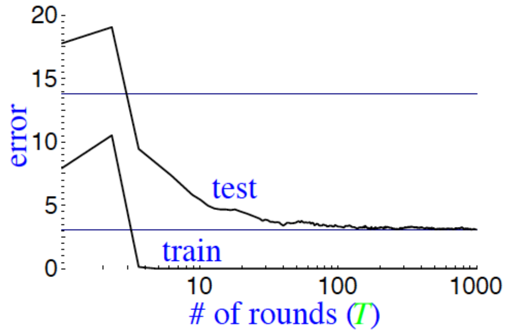
Cumulative distribution of margins of training examples:

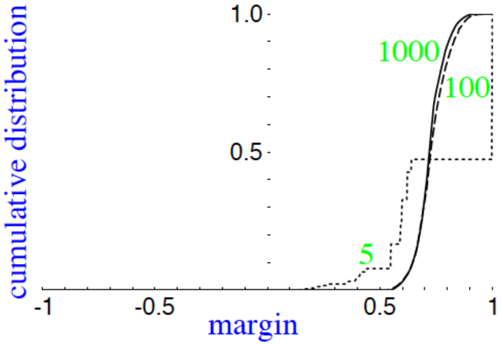




Example Revisited

Cumulative distribution of margins of training examples:





 # rounds

	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins $\leq$ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Another View of Boosting

Let $\mathcal{H}$ denote the set of base classifiers $\mathcal{X} \longrightarrow \{-1, 1\}$.

For instance, $\mathcal{H} = \{\text{decision stumps}\}$.

Another View of Boosting

Let $\mathcal{H}$ denote the set of base classifiers $\mathcal{X} \longrightarrow \{-1, 1\}$.

For instance, $\mathcal{H} = \{\text{decision stumps}\}$.

Representation $\phi(x)$ in which each $h \in \mathcal{H}$ is a feature:

$$\phi(x) = (h(x) : h \in \mathcal{H})$$

Boosting returns a linear classifier in this enhanced space:

$$H(x) = sign\underbrace{\left(\sum_t \alpha_t h_t(x)\right)}_{\text{call this } f(x)}.$$

Another View of Boosting

Let $\mathcal{H}$ denote the set of base classifiers $\mathcal{X} \longrightarrow \{-1, 1\}$.

For instance, $\mathcal{H} = \{\text{decision stumps}\}$.

Representation $\phi(x)$ in which each $h \in \mathcal{H}$ is a feature:

$$\phi(x) = (h(x) : h \in \mathcal{H})$$

Boosting returns a linear classifier in this enhanced space:

$$H(x) = sign\underbrace{\left(\sum_t \alpha_t h_t(x)\right)}_{\text{call this } f(x)}$$

What kind of linear classifier does boosting return?
Is it optimizing some loss function?

Minimizing Exponential Loss

Boosting looks for the linear classifier f that minimizes the exponential loss:

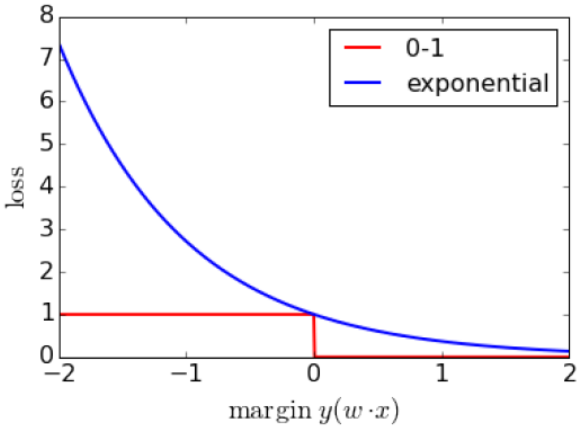
$$\frac{1}{n}\sum_{i=1}^{n} e^{-y^{(i)} f(x^{(i)})}.$$

Minimizing Exponential Loss

Boosting looks for the linear classifier f that minimizes the exponential loss:

$$\frac{1}{n}\sum_{i=1}^{n} e^{-y^{(i)} f(x^{(i)})}.$$

This loss function is a convex upper bound on $0-1$ loss:

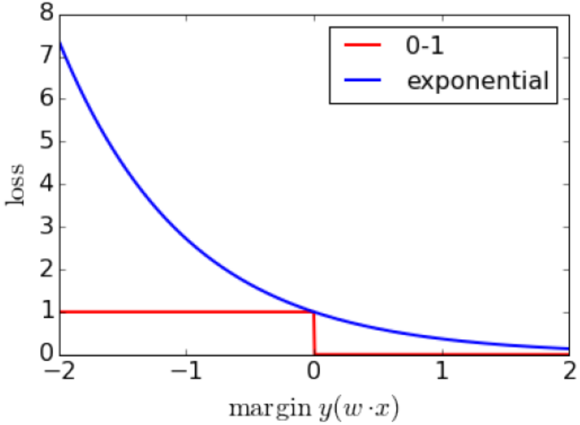


Minimizing Exponential Loss

Boosting looks for the linear classifier f that minimizes the exponential loss:

$$\frac{1}{n}\sum_{i=1}^{n} e^{-y^{(i)} f(x^{(i)})}.$$

This loss function is a convex upper bound on $0-1$ loss:



Loss minimization by coordinate descent.