

CS195-5 : Introduction to Machine Learning

Lecture 29

Greg Shakhnarovich

November 22, 2006

Announcements

- Project proposals due today! e-mail to me in PDF/PS format.

Review

$$y = \sum_{j=0}^d f_j(\mathbf{x}; \mathbf{w}), \quad f_j(\mathbf{x}; \mathbf{w}) = w_j \phi_j(\mathbf{x})$$

- Forward stepwise regression: at step t , fit

$$\theta_t = \operatorname{argmin}_{\theta} \sum_{i=1}^N \left(y_i - \sum_{j=1}^{t-1} f(\mathbf{x}_i; \theta_j) - f(\mathbf{x}_i; \theta) \right)^2.$$

until no significant improvement in empirical loss.

- Boosting
 - Details today

Combining classifiers

- A similar idea: combine classifiers $h_1(\mathbf{x}), \dots, h_m(\mathbf{x})$

$$H(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x}),$$

- α_j is the *vote* assigned to classifier h_j ;
assume $y = \pm 1$.
 - Votes should be higher for more reliable classifiers.
- Prediction with the ensemble classifier:

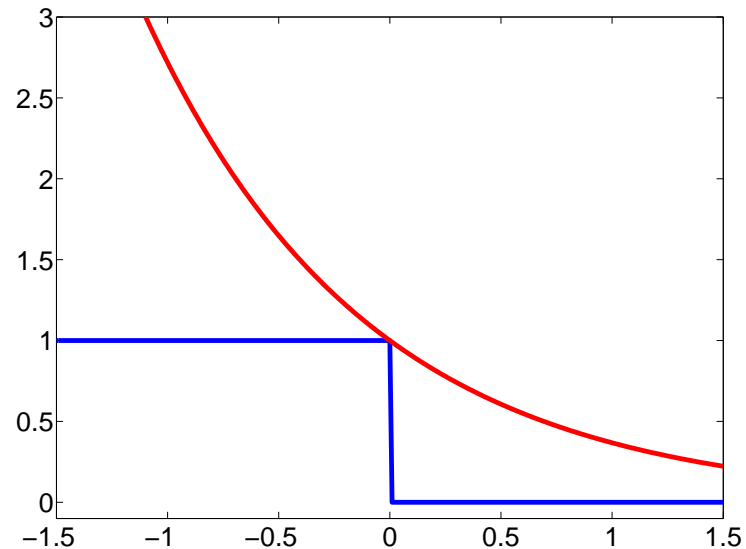
$$\hat{y}(\mathbf{x}) = \text{sign } H(\mathbf{x}).$$

Exponential loss function

- We will use the *exponential loss* to measure the quality of the classifier:

$$L(H(\mathbf{x}), y) = e^{-y \cdot H(\mathbf{x})}$$

$$\begin{aligned} L_N(H) &= \sum_{i=1}^N L(H(\mathbf{x}_i), y_i) \\ &= \sum_{i=1}^N e^{-y_i \cdot H(\mathbf{x}_i)} \end{aligned}$$



- Differentiable approximation (bound) of 0/1 loss
 - Easy to optimize!
- Other options are possible.

Ensemble loss

- Denote $H_m(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x})$
- Suppose we add $\alpha_m \cdot h_m(\mathbf{x})$ to H_{m-1} :

$$L_N(H_m) = \sum_{i=1}^N e^{-y_i \cdot [H_{m-1}(\mathbf{x}_i) + \alpha_m h_m(\mathbf{x}_i)]}$$

Ensemble loss

- Denote $H_m(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x})$
- Suppose we add $\alpha_m \cdot h_m(\mathbf{x})$ to H_{m-1} :

$$\begin{aligned} L_N(H_m) &= \sum_{i=1}^N e^{-y_i \cdot [H_{m-1}(\mathbf{x}_i) + \alpha_m h_m(\mathbf{x}_i)]} \\ &= \sum_{i=1}^N e^{-y_i H_{m-1}(\mathbf{x}_i) - \alpha_m y_i h_m(\mathbf{x}_i)} \end{aligned}$$

Ensemble loss

- Denote $H_m(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x})$
- Suppose we add $\alpha_m \cdot h_m(\mathbf{x})$ to H_{m-1} :

$$\begin{aligned} L_N(H_m) &= \sum_{i=1}^N e^{-y_i \cdot [H_{m-1}(\mathbf{x}_i) + \alpha_m h_m(\mathbf{x}_i)]} \\ &= \sum_{i=1}^N e^{-y_i H_{m-1}(\mathbf{x}_i) - \alpha_m y_i h_m(\mathbf{x}_i)} \\ &= \sum_{i=1}^N e^{-y_i H_{m-1}(\mathbf{x}_i)} \cdot e^{-\alpha_m y_i h_m(\mathbf{x}_i)} \end{aligned}$$

- Define $W_i^{(m-1)} = e^{-y_i H_{m-1}(\mathbf{x}_i)}$

Weighted loss

- Exponential loss after m -th iteration:

$$L_N(H_m) = \sum_{i=1}^N \underbrace{e^{-y_i H_{m-1}(\mathbf{x}_i)}}_{W_i^{(m-1)}} \underbrace{e^{-y_i \alpha_m h_m(\mathbf{x}_i)}}_{\text{need to optimize}}$$

- $W_i^{(m-1)}$ captures the “history” of classification of \mathbf{x}_i by H_{m-1} .
- Optimization: choose α_m , $h_m = h(\mathbf{x}; \theta_m)$ that optimize the (weighted) exponential loss at iteration m .
 - Remember: this means optimizing an upper bound on classification error.

Optimizing weak learner

$$\sum_{i=1}^N W_i^{(m-1)} e^{-\alpha_m y_i h_m(\mathbf{x}_i)} = e^{-\alpha_m} \sum_{i: y_i = h_m(\mathbf{x}_i)} W_i^{(m-1)} + e^{\alpha_m} \sum_{i: y_i \neq h_m(\mathbf{x}_i)} W_i^{(m-1)}$$

- For any $\alpha_m > 0$, minimizing this \Rightarrow *minimizing weighted training error*
- We can normalize the weights:

$$W_i^{(m-1)} = \frac{e^{-y_i H_{m-1}(\mathbf{x}_i)}}{\sum_{j=1}^N e^{-y_j H_{m-1}(\mathbf{x}_j)}}$$

Optimizing votes

- The weighted error of h_m :

$$\epsilon_m = \frac{1}{2} \left(1 - \sum_{i=1}^N W_i^{(m-1)} y_i h_m(\mathbf{x}_i) \right)$$

- Given h_m and its ϵ_m , set α_m that minimizes the exponential loss:

$$\alpha_m = \frac{1}{2} \log \frac{1 - \epsilon_m}{\epsilon_m}$$

- As long as $\epsilon_m < \frac{1}{2}$, $\alpha_m > 0$.

AdaBoost

1. Initialize weights: $W_i^{(0)} = 1/N$

2. Iterate for $m = 1, \dots, M$:

- Find (any) “weak” classifier h_m that attains weighted error

$$\epsilon_m = \frac{1}{2} \left(1 - \sum_{i=1}^N W_i^{(m-1)} y_i h_m(\mathbf{x}_i) \right) < \frac{1}{2}$$

- Let $\alpha_m = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$.

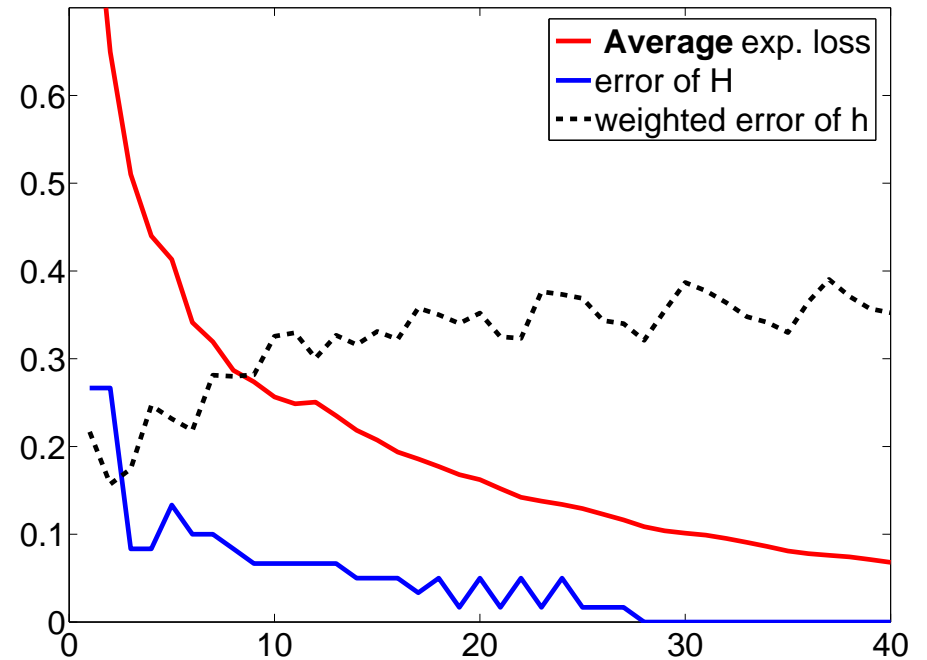
- Update the weights and normalize so that $\sum_i W_i^{(m)} = 1$:

$$W_i^{(m)} = \frac{1}{Z} W_i^{(m-1)} e^{-\alpha_m y_i h_m(\mathbf{x}_i)},$$

3. The combined classifier: $\text{sign} \left(\sum_{m=1}^M \alpha_m h_m(\mathbf{x}) \right)$

AdaBoost: typical behavior

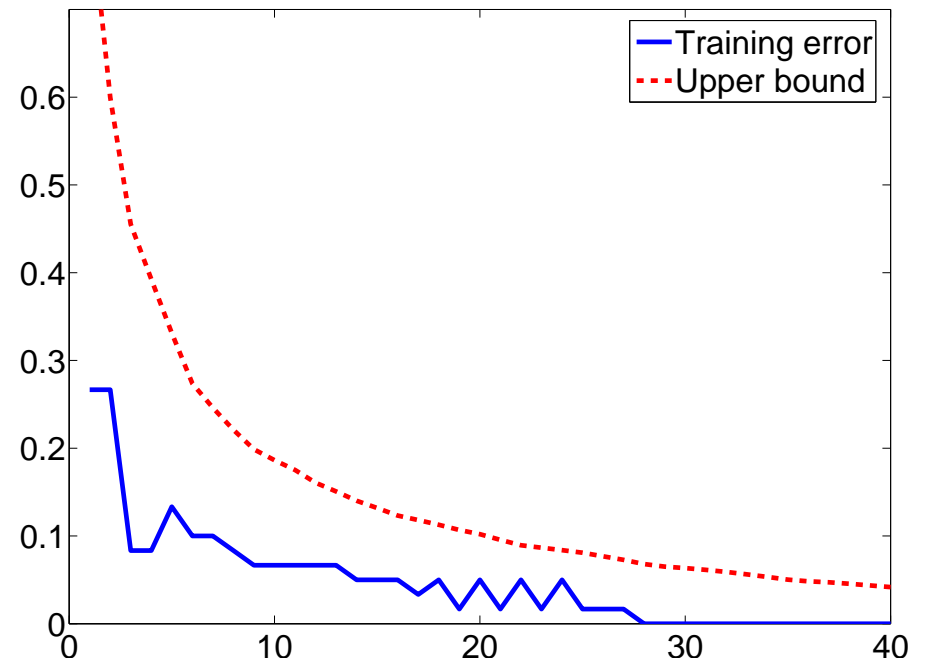
- Training error of H goes down;
- Weighted error ϵ_m goes up;
 \Rightarrow votes α_m go down.
- Exponential loss goes strictly down.



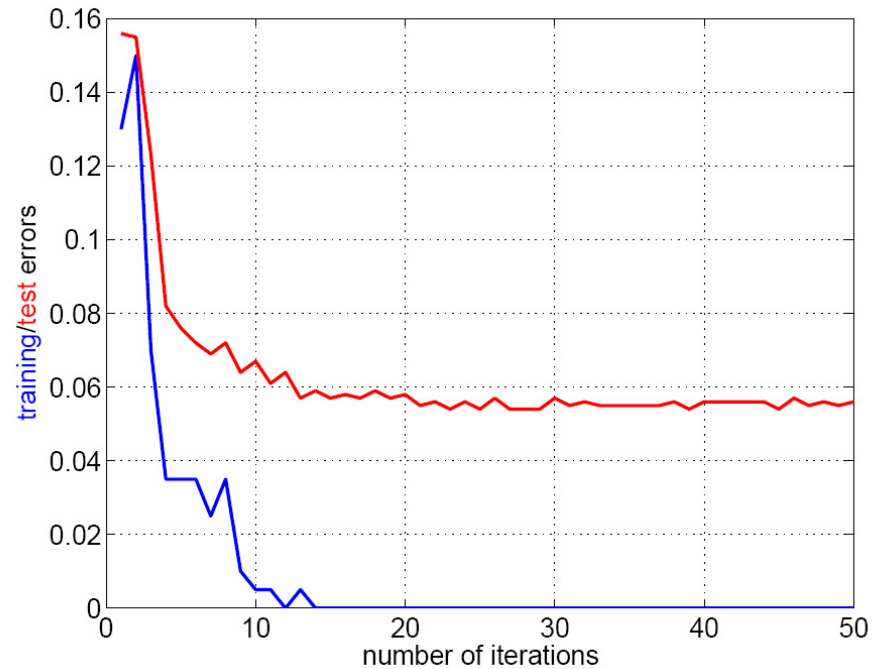
AdaBoost behavior: training error

- Can show (PS6): the training error in m -th iteration is bounded

$$\text{err}(H_m) \leq \prod_{j=1}^m 2\sqrt{\epsilon_j(1 - \epsilon_j)}.$$



AdaBoost behavior: test error



- Typical behavior: test error can still decrease after training error is flat (even zero).

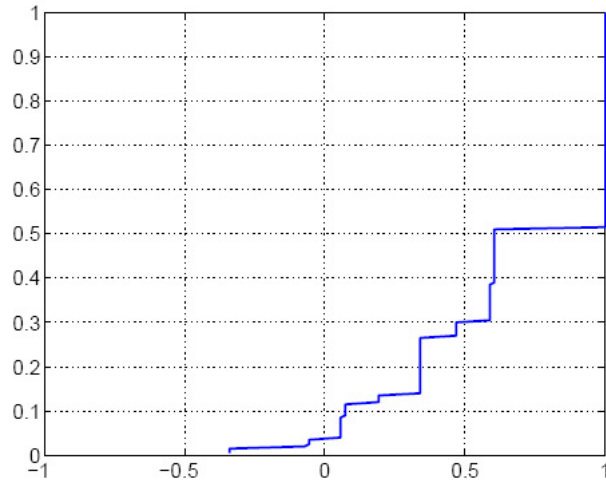
Boosting the margin

- We can define the *margin* of an example

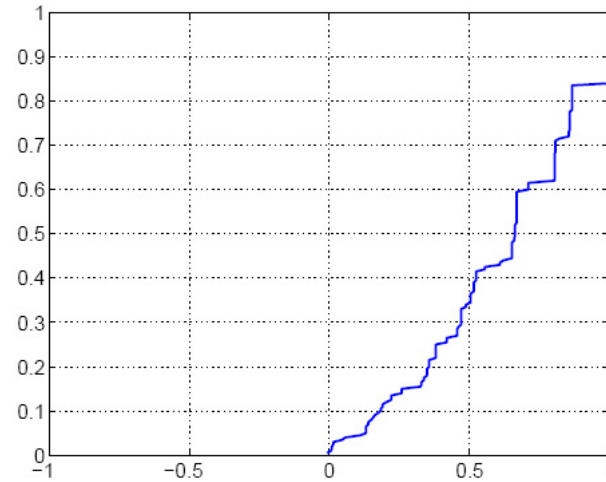
$$\gamma(\mathbf{x}_i) = y_i \cdot \frac{\alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x})}{\alpha_1 + \dots + \alpha_m}$$

- $\gamma(\mathbf{x}_i) \in [-1, 1]$, positive iff $H(\mathbf{x}_i) = y_i$.
- Iterations of AdaBoost *increase* the margin of training examples!
 - Even for correct classification can further improve confidence.

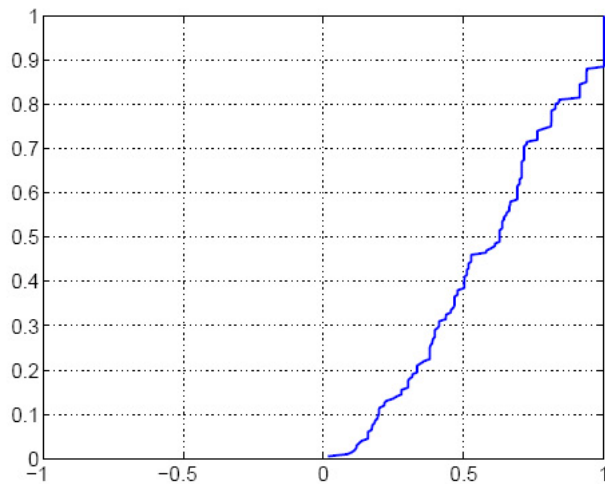
Cumulative distribution of $\gamma(\mathbf{x}_i)$



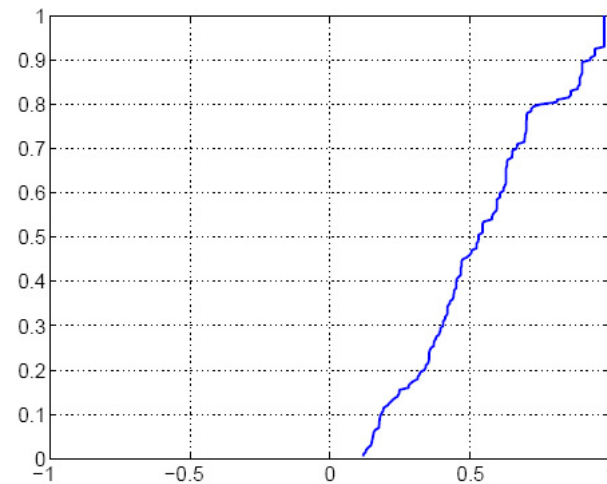
4 iterations



10 iterations



20 iterations



50 iterations

Next time

Mixtures of experts;
Markov models.