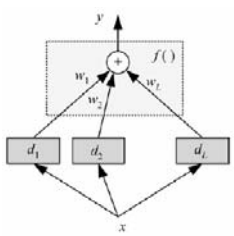
Machine Learning

10-701/15-781, Fall 2006

Boosting



Eric Xing

Lecture 9, October 10, 2006

Reading: Chap. 14.3, C.B book

Rationale: Combination of methods

- There is no algorithm that is always the most accurate
- We can select simple “weak” classification or regression methods and combine them into a single “strong” method
- Different learners use different
  - **Algorithms**
  - **Hyperparameters**
  - **Representations (Modalities)**
  - **Training sets**
  - **Subproblems**
- The problem: how to combine them

Some early algorithms

- Boosting by filtering (Schapire 1990)
  - Run weak learner on differently filtered example sets
  - Combine weak hypotheses
  - Requires knowledge on the performance of weak learner
- Boosting by majority (Freund 1995)
  - Run weak learner on weighted example set
  - Combine weak hypotheses linearly
  - Requires knowledge on the performance of weak learner
- Bagging (Breiman 1996)
  - Run weak learner on bootstrap replicates of the training set
  - Average weak hypotheses
  - Reduces variance

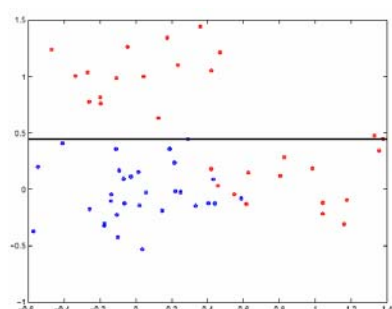
Combination of classifiers

- Suppose we have a family of component classifiers (generating ±1 labels) such as decision stumps:

$$h(x;\theta) = \text{sign}(wx_k + b)$$

where $\theta = \{k, w, b\}$

- Each decision stump pays attention to only a single component of the input vector



Combination of classifiers con'd

- We'd like to combine the simple classifiers additively so that the final classifier is the sign of

$$\hat{h}(\mathbf{x}) = \alpha_1 h(\mathbf{x};\theta_1) + \ldots + \alpha_m h(\mathbf{x};\theta_m)$$

  where the "votes" $\{\alpha_i\}$ emphasize component classifiers that make more reliable predictions than others

- Important issues:
  - what is the criterion that we are optimizing? (measure of loss)
  - we would like to estimate each new component classifier in the same manner (modularity)

Measurement of error

- Loss function:

$$\lambda(y, h(\mathbf{x})) \qquad (\text{e.g. } I(y \neq h(\mathbf{x})))$$

- Generalization error:

$$L(h) = E[\lambda(y, h(\mathbf{x}))]$$

- Objective: find ***h*** with minimum *generalization* error

- Main boosting idea: minimize the *empirical* error:

$$\hat{L}(h) = \frac{1}{N}\sum_{n=1}^{N} \lambda(y_n, h(\mathbf{x}_n))$$

Exponential Loss

- One possible measure of empirical loss is

$$\sum_{i=1}^{n} \exp\{-y_i \hat{h}_m(\mathbf{x}_i)\} \qquad \hat{h}(\mathbf{x}) = \alpha_1 h(\mathbf{x};\theta_1) + \ldots + \alpha_m h(\mathbf{x};\theta_m)$$

$$= \sum_{i=1}^{n} \exp\{-y_i \hat{h}_{m-1}(\mathbf{x}_i) - y_i a_m h(\mathbf{x}_i;\theta_m)\}$$

$$= \sum_{i=1}^{n} \exp\{-y_i \hat{h}_{m-1}(\mathbf{x}_i)\}\exp\{-y_i a_m h(\mathbf{x}_i;\theta_m)\}$$

$$= \sum_{i=1}^{n} W_i^{m-1} \exp\{-y_i a_m h(\mathbf{x}_i;\theta_m)\}$$

  - The combined classifier based on m − 1 iterations defines a weighted loss criterion for the next simple classifier to add
  - each training sample is weighted by its "classifiability" (or difficulty) seen by the classifier we have built so far

Linearization of loss function

- We can simplify a bit the estimation criterion for the new component classifiers (assuming α is small)

$$\exp\{-y_i a_m h(\mathbf{x}_i;\theta_m)\} \approx 1 - y_i a_m h(\mathbf{x}_i;\theta_m)$$

- Now our empirical loss criterion reduces to

$$\sum_{i=1}^{n} \exp\{-y_i \hat{h}_m(\mathbf{x}_i)\}$$

$$\approx \sum_{i=1}^{n} W_i^{m-1}(1 - y_i a_m h(\mathbf{x}_i;\theta_m))$$

$$= \sum_{i=1}^{n} W_i^{m-1} - a_m \sum_{i=1}^{n} W_i^{m-1} y_i h(\mathbf{x}_i;\theta_m)$$

- We could choose a new component classifier to optimize this weighted agreement

A possible algorithm

- At stage m we find θ^* that maximize (or at least give a sufficiently high) weighted agreement:

$$\sum_{i=1}^{n} W_i^{m-1} y_i h(\mathbf{x}_i; \theta_m^*)$$

  - each sample is weighted by its "difficulty" under the previously combined $m-1$ classifiers,
  - more "difficult" samples received heavier attention as they dominates the total loss
- Then we go back and find the "votes" α_m* associated with the new classifier by minimizing the **original** weighted (exponential) loss

$$\sum_{i=1}^{n} W_i^{m-1} \exp\{-y_i a_m h(\mathbf{x}_i; \theta_m)\}$$

Boosting

- We have basically derived a Boosting algorithm that sequentially adds new component classifiers, each trained on reweighted training examples
  - each component classifier is presented with a slightly different problem
- AdaBoost preliminaries:
  - we work with *normalized weights* W_i on the training examples, initially uniform ($W_i = 1/n$)
  - the weight reflect the "*degree of difficulty*" of each datum on the latest classifier

The AdaBoost algorithm

- At the kth iteration we find (any) classifier $h(\mathbf{x};\ \theta_k^*)$ for which the weighted classification error:

$$\varepsilon_k = 0.5 - \frac{1}{2}\left(\sum_{i=1}^{n} W_i^{k-1} y_i h(\mathbf{x}_i;\theta_k^*)\right)$$

  is better than change.
  - This is meant to be "easy" --- weak classifier
- Determine how many “votes” to assign to the new component classifier:

$$\alpha_k = 0.5\log\left((1-\varepsilon_k)/\varepsilon_k\right)$$

  - stronger classifier gets more votes
- Update the weights on the training examples:

$$W_i^k = W_i^{k-1}\exp\{-y_i a_k h(\mathbf{x}_i;\theta_k)\}$$

The AdaBoost algorithm cont’d

- The final classifier after m boosting iterations is given by the sign of

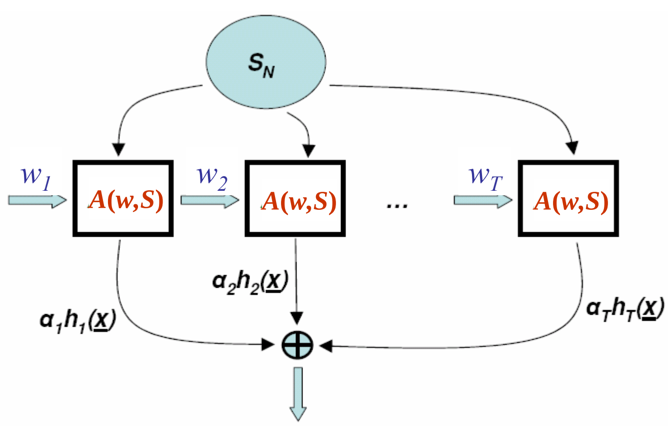
$$\hat{h}(\mathbf{x}) = \frac{\alpha_1 h(\mathbf{x};\theta_1) + \ldots + \alpha_m h(\mathbf{x};\theta_m)}{\alpha_1 + \ldots + \alpha_m}$$

  - the votes here are normalized for convenience

AdaBoost: summary

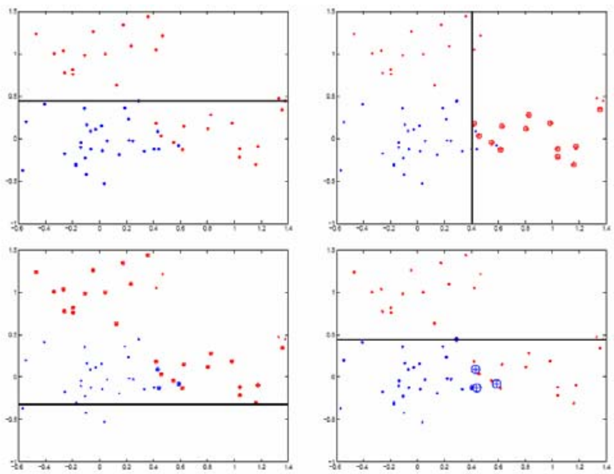
- **Input:**
  - N examples $S_N = \{(x_1,y_1),\ldots,(x_N,y_N)\}$
  - a weak base learner $h = h(x,\theta)$
- **Initialize:** equal example weights $w_i = 1/N$ for all $i = 1..N$
- **Iterate for** $t = 1\ldots T$**:**
  1. train base learner according to weighted example set (w_t,x) and obtain hypothesis $h_t = h(x,\theta_t)$
  2. compute hypothesis error ε_t
  3. compute hypothesis weight α_t
  4. update example weights for next iteration w_{t+1}
- **Output:** final hypothesis as a linear combination of h_t

AdaBoost: dataflow diagram

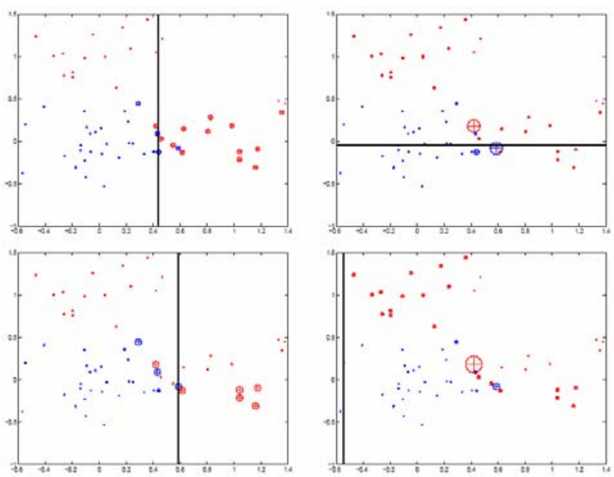


$$f_T(\underline{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\sum_{r=1}^{T} \alpha_r} h_t(\underline{x})$$

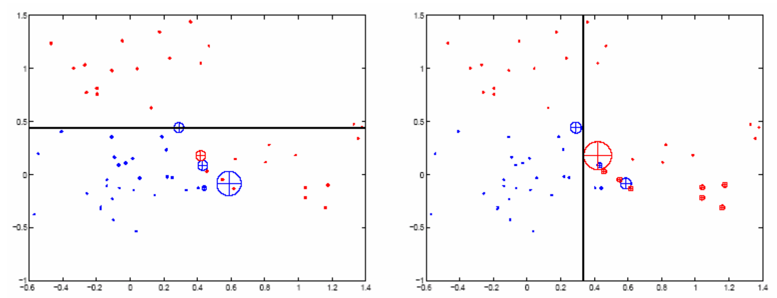
Boosting: examples



Boosting: example cont’d



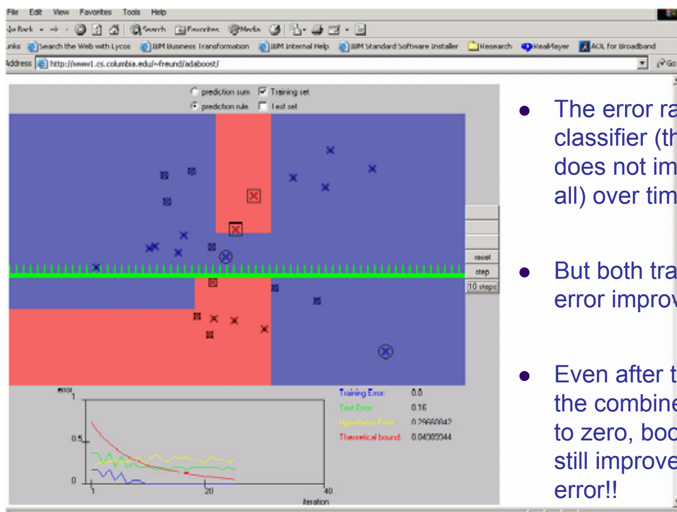
Boosting: example cont'd



Base Learners

- Weak learners used in practice:
  - Decision stumps (axis parallel splits)
  - Decision trees (e.g. C4.5 by Quinlan 1996)
  - Multi-layer neural networks
  - Radial basis function networks
- Can base learners operate on weighted examples?
  - In many cases they can be modified to accept weights along with the examples
  - In general, we can sample the examples (with replacement) according to the distribution defined by the weights

Boosting performance



- The error rate of component classifier (the decision stumps) does not improve much (if at all) over time
- But both training and testing error improve over time!
- Even after the training error of the combined classifier goes to zero, boosting iterations can still improve the generalization error!!

Why it is working?

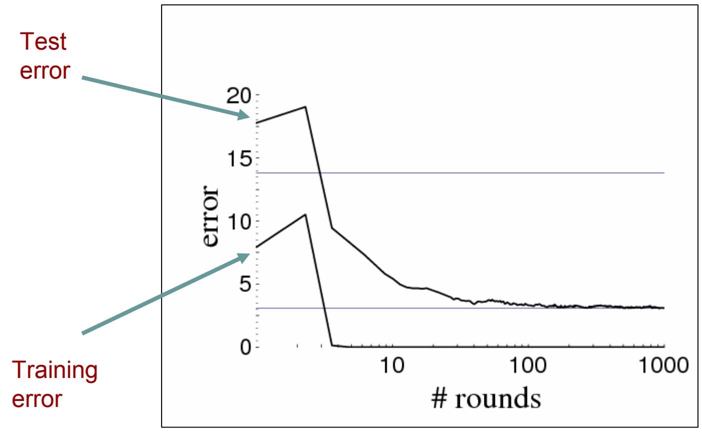
- You will need some learning theory (to be covered in the next two lectures) to understand this fully, but for now let's just go over some high level ideas
- Generalization Error:

With high probability, Generalization error is less than:

$$\hat{\Pr}\left[H(x) \neq y\right] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

As T goes up, our bound becomes worse,
Boosting should overfit!

Experiments



The Boosting Approach to Machine Learning, by Robert E. Schapire

Training Margins

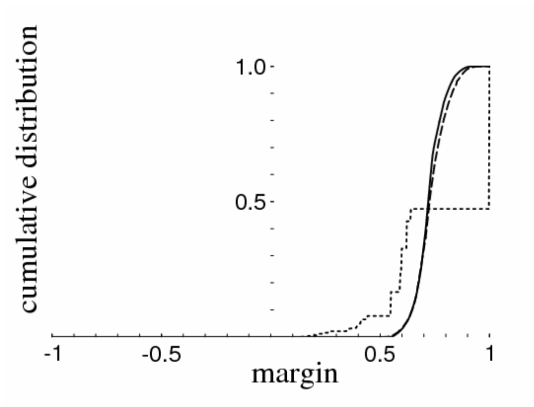
- When a vote is taken, the more predictors agreeing, the more confident you are in your prediction.
- Margin for example:

$$\text{margin}_h(\mathbf{x}_i, y_i) = y_i \left[\frac{\alpha_1 h(\mathbf{x}_i;\theta_1) + \ldots + \alpha_m h(\mathbf{x}_i;\theta_m)}{\alpha_1 + \ldots + \alpha_m} \right]$$

The margin lies in [−1, 1] and is negative for all misclassified examples.

- Successive boosting iterations improve the majority vote or margin for the training examples

More Experiments



The Boosting Approach to Machine Learning, by Robert E. Schapire

A Margin Bound

- For any γ, the generalization error is less than:

$$\Pr\left(\text{margin}_h(\mathbf{x}, y) \leq \gamma\right) + O\left(\sqrt{\frac{d}{m\gamma^2}}\right)$$

Robert E. Schapire, Yoav Freund, Peter Bartlett and Wee Sun Lee. **Boosting the margin: A new explanation for the effectiveness of voting methods**. *The Annals of Statistics*, 26(5):1651-1686, 1998.

- It does not depend on T!!!

Summary

- Boosting takes a weak learner and converts it to a strong
- one
- Works by asymptotically minimizing the empirical error
- Effectively maximizes the margin of the combined hypothesis