Lecture 8: Ensemble classifiers. Bagging. Boosting

- Idea of boosting
- AdaBoost algorithm (Freund and Schapire)
- Why does boosting work?

Recall: Bias and variance

- For regression problems, the expected error can be decomposed as:
$$\text{Bias}^2 + \text{Variance} + \text{Noise}$$
- Bias is typically caused by the hypothesis class being too simple, and hence not able to represent the true function (*underfitting*)
- Variance is typically caused by the hypothesis class being too large (*overfitting*)
- There is often a trade-off between bias and variance
- A similar but more involved decomposition of the error can be done for classification problems (using the 0-1 loss error function)

Measuring bias and variance in practice

- Recall that bias and variance are both defined as expectations:

$$Bias(\mathbf{x}) = E_P[f(\mathbf{x}) - \bar{h}(\mathbf{x})]$$

$$Var(\mathbf{x}) = E_P[(h(\mathbf{x}) - \bar{h}(\mathbf{x}))^2]$$

- To get expected values we *simulated* multiple data sets, by drawing with samples with replacement from the original data set
- This gives a set of hypothesis, whose predictions can be *averaged* together

Bootstrap replicates

- Given data set D, construct a *bootstrap replicate* of D, called D_b, which has the same number of examples, by drawing samples from D *with replacement*
- Use the learning algorithm to construct a hypothesis h_b by training on D_b
- Compute the prediction of h_b on each of the *remaining* points, from the set $T_b = D - D_b$
- This process is repeated B times, where B is typically a few hundred
- If D is very large, the replicates should contain $m < |D|$ points (still drawn with replacement)

Estimating bias and variance

- For each point, we have a set of estimates $h_1(\mathbf{x}), \ldots h_K(\mathbf{x})$, with $K \leq B$
- The average prediction, determined empirically, is:

$$\bar{h}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^{K} h_k(\mathbf{x})$$

- We will estimate the bias as:

$$y - \bar{h}(\mathbf{x})$$

- We estimate the variance as:

$$\frac{1}{K-1} \sum_{k=1}^{K} (\bar{h}(\mathbf{x}) - h_k(\mathbf{x}))^2$$

Approximations

- Bootstrap replicates are not real data
- We typically ignore the noise
- If we had multiple points with the same $\mathbf{x}$ value, we can estimate the noise
- Alternatively, we can do an estimation using ”similar points”, if this appropriate

Bagging: Bootstrap aggregation

- If we did all the work to get the hypotheses h_b, why not use all of them to make a prediction?
- All hypotheses can have a vote, in the classification case, and we pick the majority class
- For regression, we can average all the predictions
- Which hypotheses classes would benefit most from this approach?

Estimated bias and variance of bagging

- According with our way of estimating variance and bias, bagging eliminates variance altogether!
- In practice, bagging tends to reduce variance and increase bias
- Hence, the main benefit is for “unstable” learners, i.e., learners with high variance.
- This includes complex hypotheses classes, e.g. decision trees (even unpruned), neural networks, nearest-neighbor-type methods

Ensemble learning in general

- Ensemble learning algorithms work by running a *base learning algorithm* multiple times, then *combining* the predictions of the different hypotheses obtained using some form of voting
- One approach is to construct several classifiers *independently*, then combine their predictions. Examples include:
  - Bagging
  - Randomizing the test selection in decision trees
  - Using a different subset of input features to train different neural nets
- A second approach is to *coordinate* the construction of the hypotheses in the ensemble.

Additive models

- In an ensemble, the output on any instance is computed by averaging the outputs of several hypotheses, possibly with a different weighting.
- Hence, we should choose the individual hypotheses and their weight in such a way as to provide a good fit
- This suggests that instead of constructing the hypotheses independently, we should construct them such that new hypotheses focus on instances that are problematic for existing hypotheses.
- **Boosting** is an algorithm implementing this idea

Main idea of boosting

Component classifiers should concentrate more on difficult examples

- Examine the training set
- Derive some rough "rule of thumb"
- *Re-weight* the examples of the training set, concentrating on "hard" cases for the previous rule
- Derive a second rule of thumb
- And so on... (repeat this T times)
- *Combine* the rules of thumb into a single, accurate predictor

Questions:

- How do we re-weight the examples?
- How do we combine the rules into a single classifier?

Notation

- Assume that examples are drawn independently from some probability distribution P on the set of possible data $\mathcal{D}$
- Notation: $J_P(h)$ is the expected error of h when data is drawn from P:

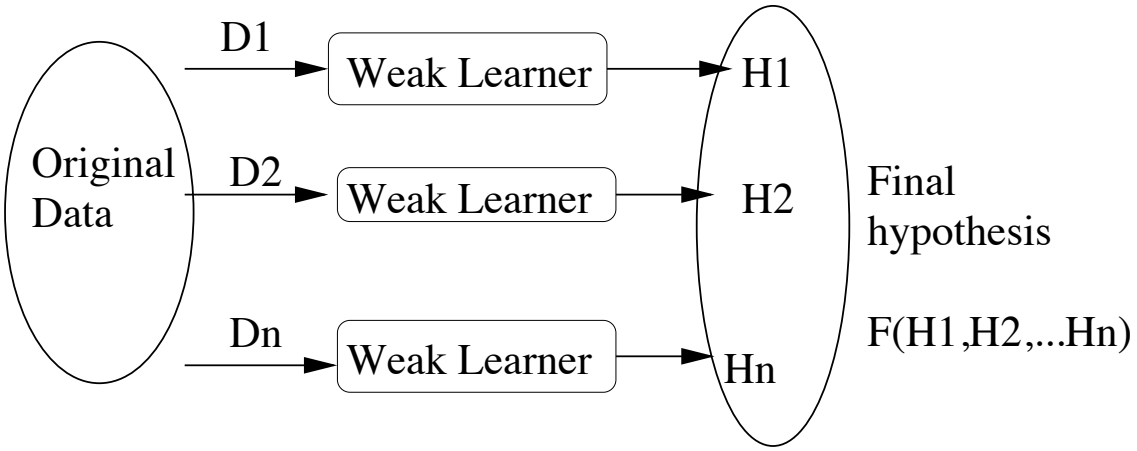
$$J_P(h) = \sum_{\langle \mathbf{x}, y \rangle} J(h(\mathbf{x}), y) P(\langle \mathbf{x}, y \rangle)$$

where $J(h(\mathbf{x}), y)$ could be squared error, or $0/1$ loss

Weak learners

- Assume we have some "weak" binary classifiers (e.g., decision stumps: $x_i > t$)
- "Weak" means $J_P(h) < 1/2 - \gamma$ where $\gamma > 0$ (i.e., the true error of the classifier is better than random).

Boosting classifier



AdaBoost (Freund & Schapire, 1995)

1. Input N training examples $\{(\mathbf{x_1}, y_1), \ldots (\mathbf{x_N}, y_N)\}$, where $\mathbf{x_i}$ are the inputs and y_i is the desired class label
2. Let $D_1(\mathbf{x_i}) = \frac{1}{N}$ (we start with a uniform distribution)
3. Repeat T times:
   (a) Construct D_{t+1} from D_t (details in a moment)
   (b) Train a new hypothesis h_{t+1} on distribution D_{t+1}
4. Construct the final hypothesis:

$$h_f(\mathbf{x}) = \text{sign}\left(\sum_t \alpha_t h_t(\mathbf{x})\right),$$

Constructing the new distribution

We want data on which we make mistakes to be emphasized:

$$D_{t+1}(\mathbf{x_i}) = \frac{1}{Z_t} D_t(\mathbf{x_i}) \times \begin{cases} \beta_t, & \text{if } h_t(\mathbf{x_i}) = y_i \\ 1, & \text{otherwise} \end{cases} \quad \text{where}$$

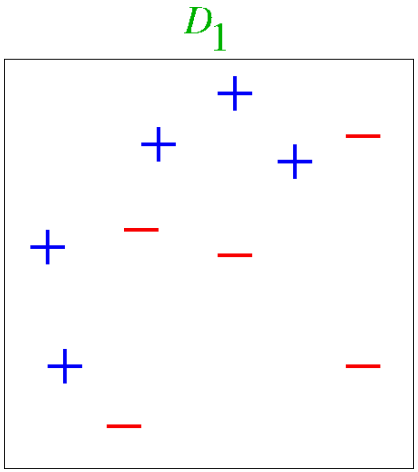
$$\beta_t = \frac{J_{D_t}(h_t)}{1 - J_{D_t}(h_t)}$$

and Z_t is a normalization factor (set such that the probabilities $D_{t+1}(x_i)$ sum to 1).

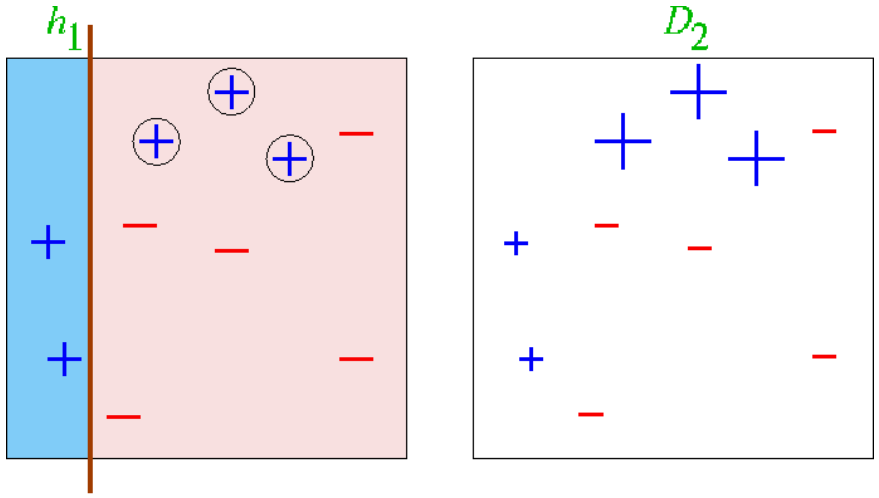
Construct the final hypothesis:

$$h_f(\mathbf{x}) = \text{sign}\left(\sum_t \alpha_t h_t(\mathbf{x})\right), \text{ where } \alpha_t = \log(1/\beta_t)$$

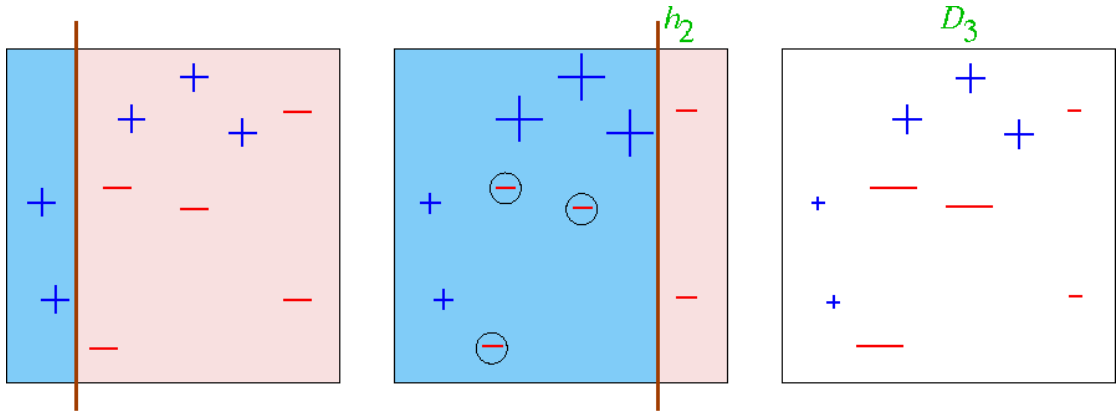
Toy example



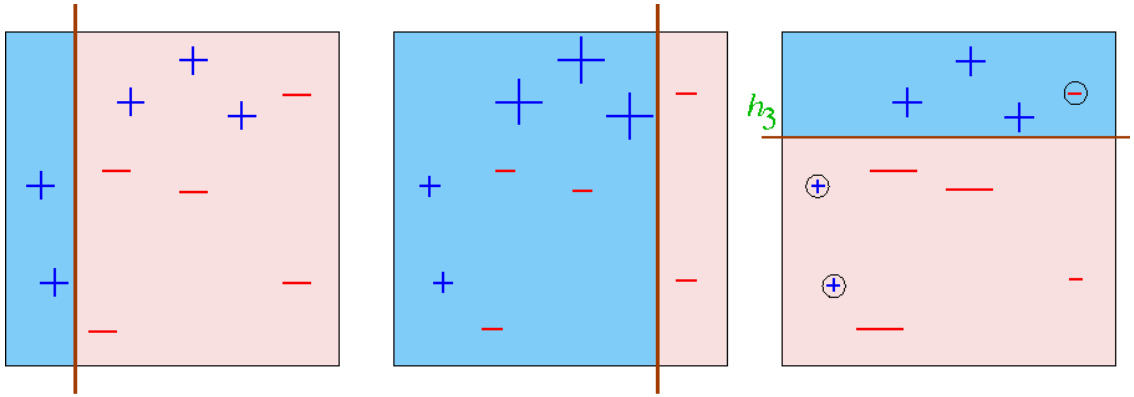
Toy example: First step



Toy example: Second step



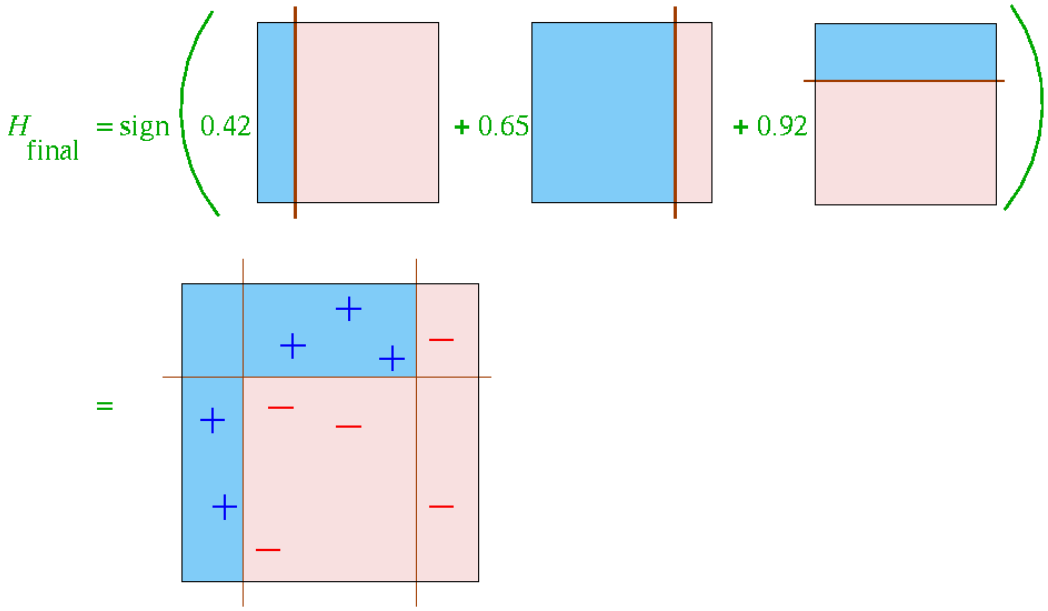
Toy example: Third step



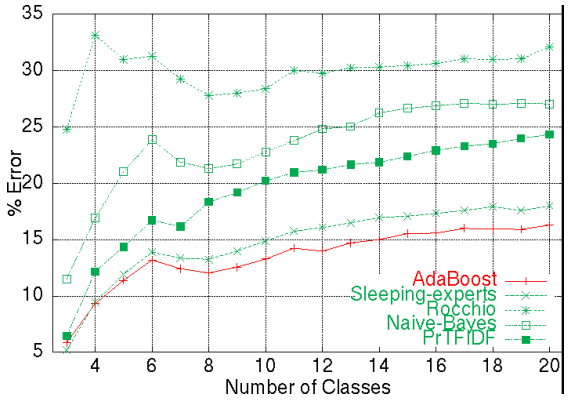
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

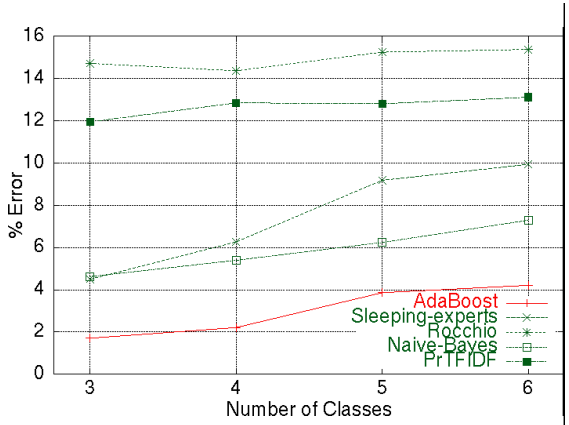
Toy example: Final hypothesis



Real data set: Text Categorization

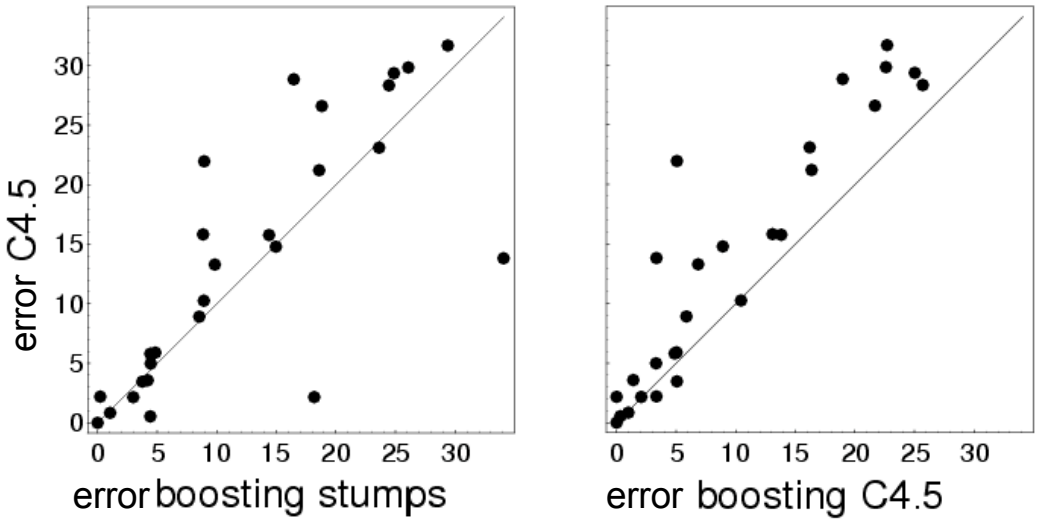


database: AP

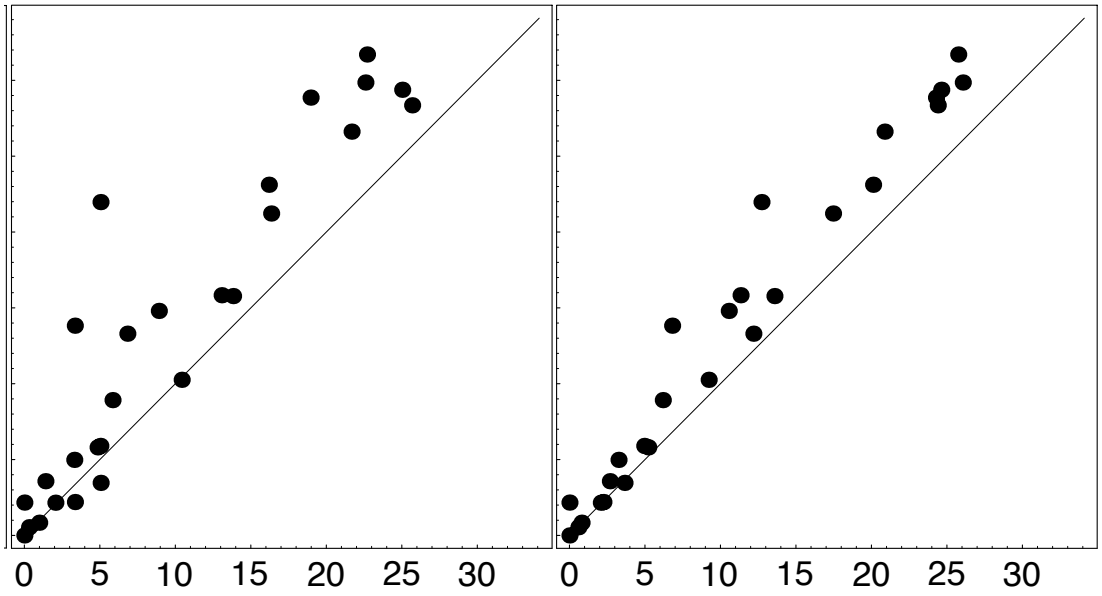


database: Reuters

Boosting empirical evaluation



Bagging vs. Boosting



boosting C4.5 bagging C4.5

Parallel of bagging and boosting

- Bagging is typically faster, but may get a smaller error reduction (not by much)
- Bagging works well with "reasonable" classifiers
- Boosting works with very simple classifiers

  E.g., Boostexter - text classification using decision stumps based on single words
- Boosting may have a problem if a lot of the data is mislabeled, because it will focus on those examples a lot, leading to overfitting.

Why does boosting work?

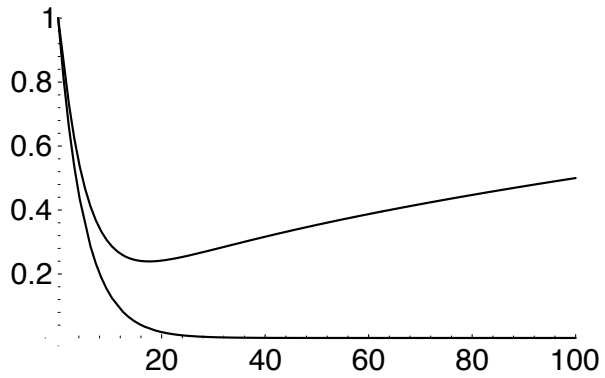
- Weak learners have high bias
- By combining them, we get more expressive classifiers
- Hence, boosting is a *bias-reduction technique*
- What happens as we run boosting longer?

Why does boosting work?

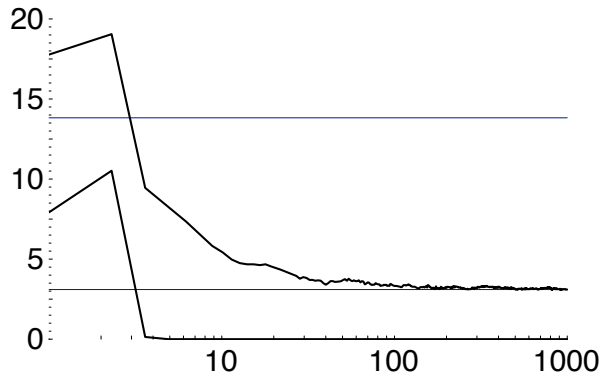
- Weak learners have high bias
- By combining them, we get more expressive classifiers
- Hence, boosting is a *bias-reduction technique*
- What happens as we run boosting longer?
  Intuitively, we get more and more complex hypotheses
- How would you expect bias and variance to evolve over time?

A naive (but reasonable) analysis of generalization error

- Expect the training error to continue to drop (until it reaches 0)
- Expect the test error to *increase* as we get more voters, and h_f becomes too complex.



Actual typical run of AdaBoost



- Test error *does not increase* even after 1000 runs! (more than 2 million decision nodes!)
- Test error *continues to drop* even after training error reaches 0!

These are consistent results through many sets of experiments!

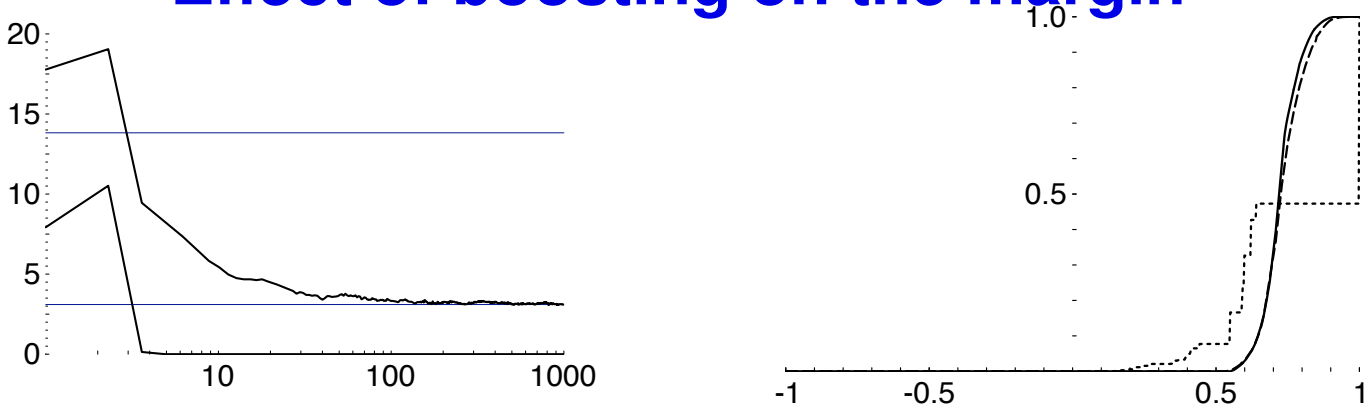
Classification margin

- Boosting constructs hypotheses of the form $h_f(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$
- The classification of an example is correct if $\text{sign}(f(\mathbf{x})) = y$
- The *margin* of a training example is defined as:

$$\text{margin}(f(\mathbf{x}), y) = y \cdot f(\mathbf{x})$$

- The margin tells us how close the decision boundary is to the data point
- The *minimum margin* over the data set gives an idea of how close the training points are to the decision boundary
- A higher margin on the training set should yield a lower generalization error
- Intuitively, increasing the margin is similar to lowering the variance

Effect of boosting on the margin



- Between rounds 5 and 10 there is no training error reduction
- But there is a *significant shift in margin distribution*!
- There is a formal proof that boosting increases the margin
- Next time: classifiers that explicitly aim to construct a large margin.

Summary

- Ensemble methods combine several hypotheses into one prediction
- They work better than the best individual hypothesis from the same class because they reduce bias or variance (or both)
- Bagging is mainly a variance-reduction technique, useful for complex hypotheses
- Main idea is to sample the data repeatedly, train several classifiers and average their predictions.
- Boosting focuses on harder examples, and gives a weighted vote to the hypotheses.
- Boosting works by reducing bias and increasing classification margin.