COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

Ensemble learning

Reminders/Comments

- To find a dataset for Assignment 4, consider
  - http://archive.ics.uci.edu/ml/
  - https://www.kaggle.com/
- Assignment 3 almost graded
- Quiz 2 in one week
- Review class next Monday

Though question

- Why do we consider the errors to be normally distributed, and get confidence intervals using normal distributions? Can we/should we use other distributions?
- Central limit theorem: for iid RVs {x1, …, xn} with mean mu and finite variance sigma^2, Sn = 1/n sum x_i is close to being normally distribute with with mean mu and variance sigma^2/n
- How quickly does the above RV converge?
  - Some results have finite sample convergence rate (e.g., O(1/sqrt(n)));
  - 30 is the usual rule of thumb for "enough" samples

Distribution-independent convergence rates

- Assume you have n instances sampled i.i.d.
- With probability 1-delta,
- l true_mean - sample_mean l < sqrt(1 / 2n) sqrt(ln(1/delta))
- This is called Hoeffding's inequality
- Useful also for statistical learning theory: true expected error bounded by the sum of
  - training error
  - model class complexity
  - O(sqrt(1/n)), from the Hoeffding bound

Collections of models

- Have mostly discussed learning one single "best" model
  - best linear regression model
  - best neural network model
- Can we take advantage of multiple learned models?

Rationale

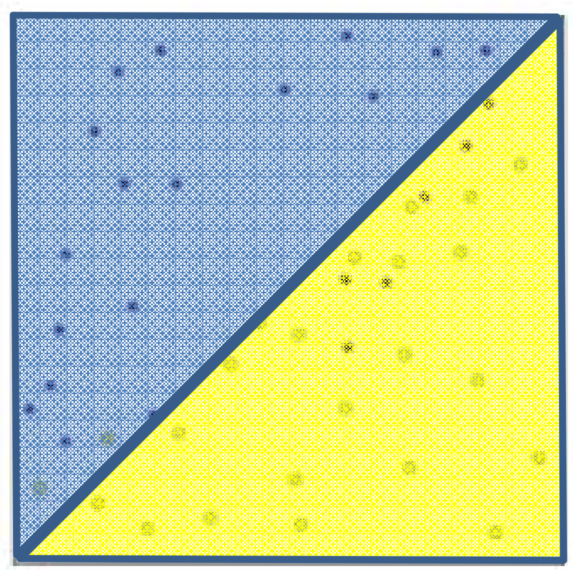
- There is no algorithm that is always the most accurate
- Different learners can use different
  - Algorithms (e.g., logistic regression or SVMs)
  - Parameters (e.g., regularization parameters)
  - Representations (e.g., polynomial basis or kernels)
  - Training sets (e.g., two different random subsamples of data)
- The problem: how to combine them

Ensembles

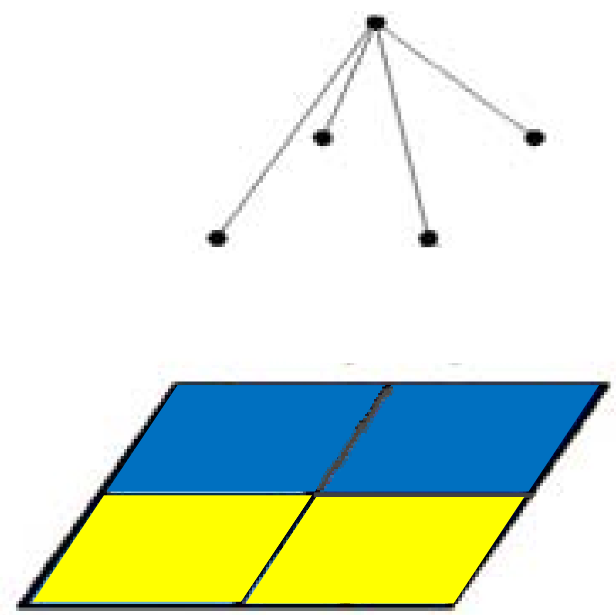
- Can a set of **weak learners** create a single strong learner?
- Answer: yes! See seminal paper: “The Strength of Weak Learnability” Schapire, 1990
- Why do we care?
  - can be easier to specify weak learners e.g., shallow decision trees, set of neural networks with smaller number of layers, etc.
  - fighting the bias-variance trade-off

Weak learners

- **Weak learners:** naive Bayes, logistic regression, decision stumps (or shallow decision trees)



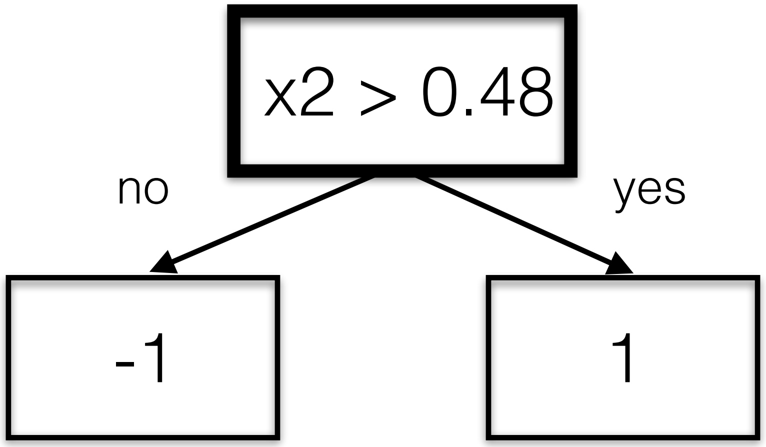
logistic regression



decision stump

*some material from slides by Eric Xing: http://www.cs.cmu.edu/~epxing/Class/10701-11f/Lecture/lecture22.pdf

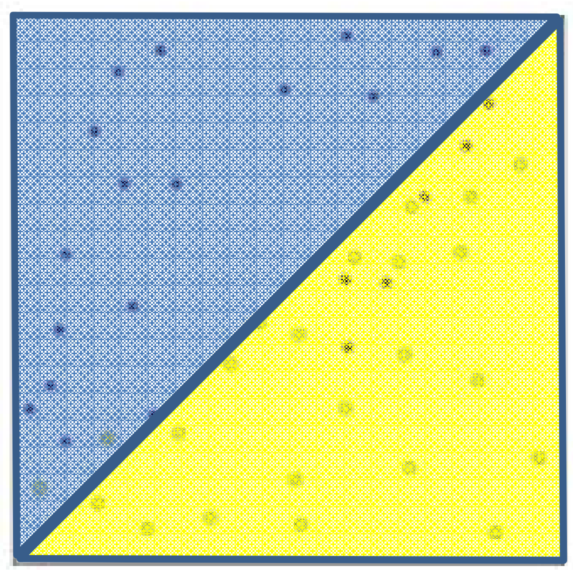
Example of a decision stump

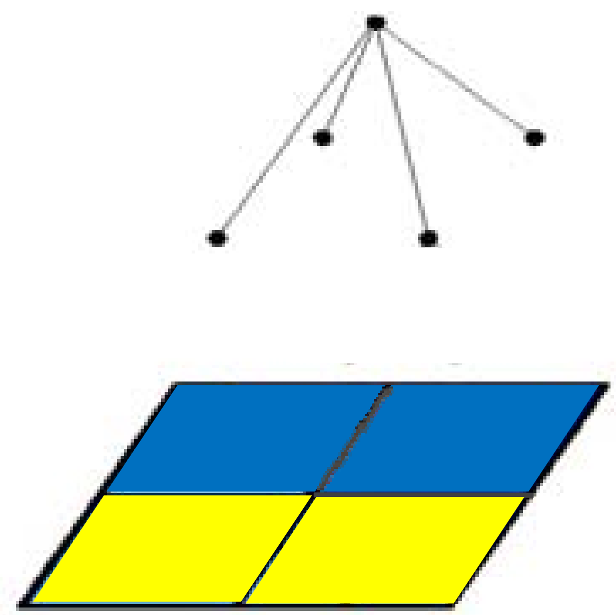


Decision tree provides more splits;
decision stump is a one level decision tree

Weak learners

- **Weak learners:** naive Bayes, logistic regression, decision stumps (or shallow decision trees)





Are good ☺ - Low variance, don't usually overfit

Are bad ☹ - High bias, can't solve hard learning problems

*some material from slides by Eric Xing: http://www.cs.cmu.edu/~epxing/Class/10701-11f/Lecture/lecture22.pdf

Bias-variance tradeoff

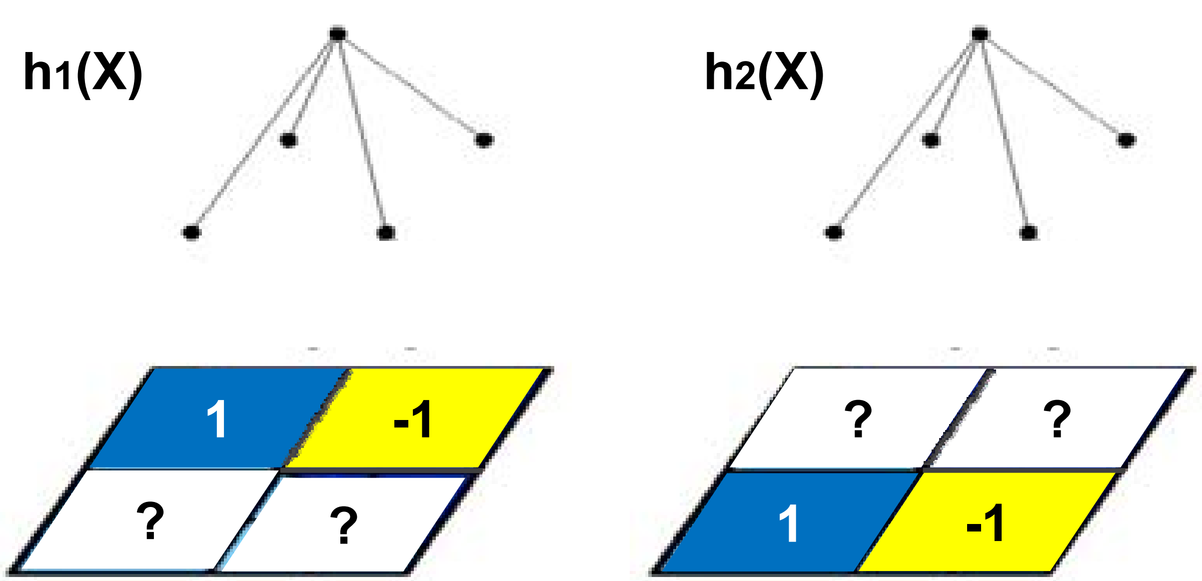
- We encountered this trade-off for weights in linear regression
- Regularizing introduced bias, but reduced variance

$$\mathrm{MSE}(\hat{\theta}) = \mathrm{Var}(\hat{\theta}) + \left(\mathrm{Bias}(\hat{\theta}, \theta)\right)^2.$$

Voting (Ensemble methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**

- **Output class:** (Weighted) vote of each classifier
  Classifiers that are most “sure” will vote with more conviction
  Classifiers will be most “sure” about a particular part of the space. On average, do better than single classifier!



H: X → Y (-1,1)

$H(X) = h_1(X)+h_2(X)$

$H(X) = \text{sign}(\sum_t \alpha_t h_t(X))$

weights

Voting (Ensemble methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**

- **Output class:** (Weighted) vote of each classifier
  Classifiers that are most "sure" will vote with more conviction
  Classifiers will be most "sure" about a particular part of the space On average, do better than single classifier!

- **But how do you**
  force classifiers h_t to learn about different parts of the input space? weight the votes of different classifiers? α_t

Boosting [Schapire 89]

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote

- On each iteration *t*:
  - weight each training example by how incorrectly it was classified
  - Learn a weak hypothesis – h_t
  - Obtain a strength for this hypothesis – α_t

- Final classifier: $\boxed{\mathbf{H(X) = sign(\sum \alpha_t \, h_t(X))}}$

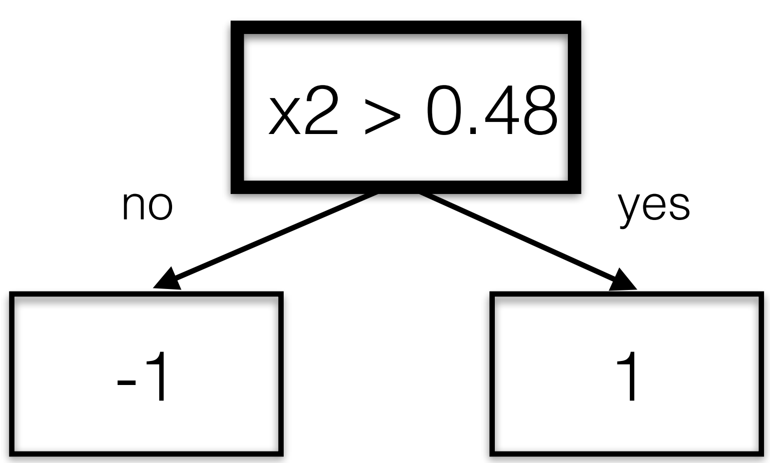
Combination of classifiers

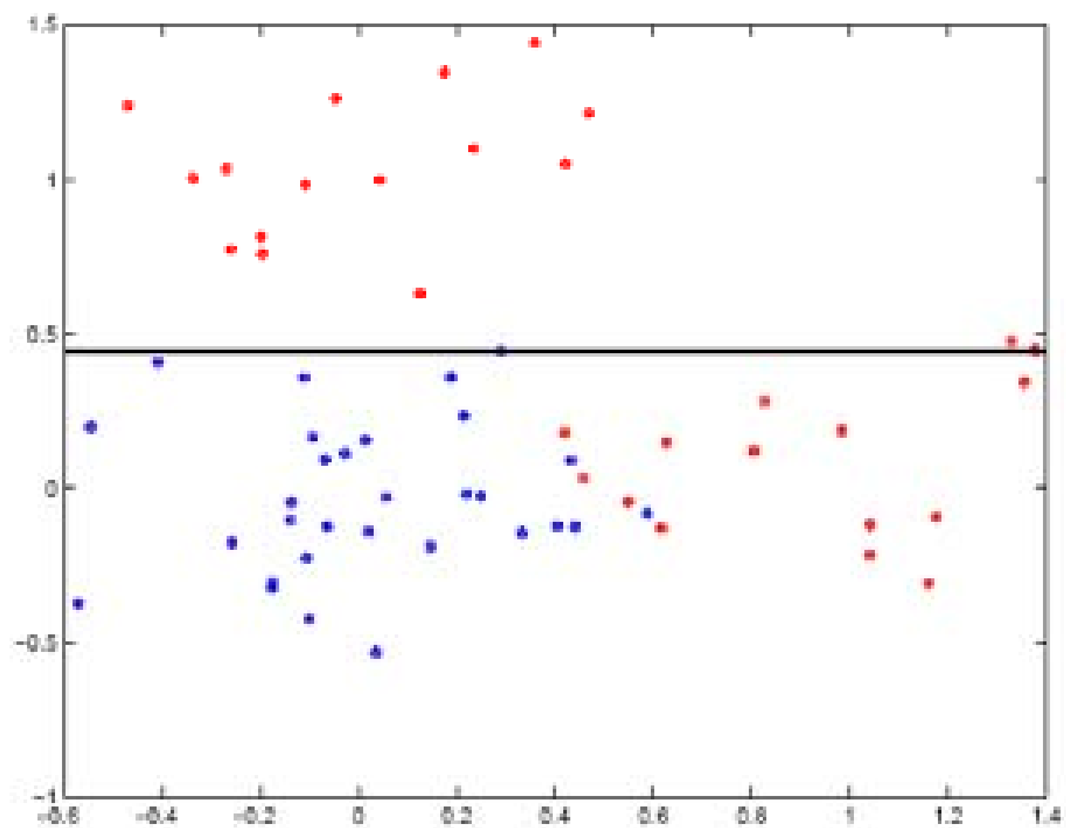
- Suppose we have a family of component classifiers (generating ±1 labels) such as decision stumps:

$$h(x;\theta) = \operatorname{sign}\left(wx_k + b\right)$$

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

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





w = 1, k = 2, b = 0.48

Combination of classifiers

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

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

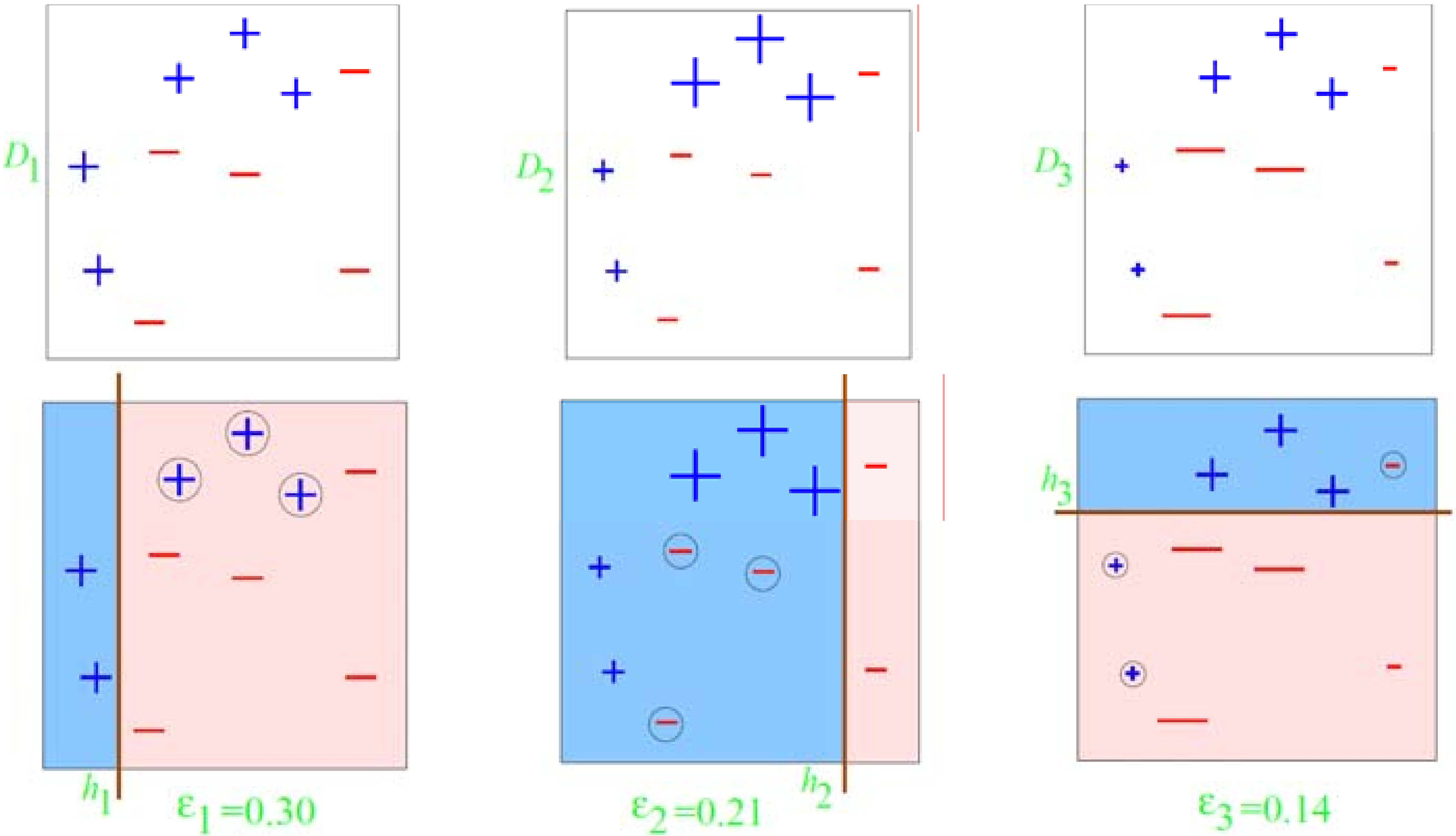
  where the "votes" $\{\alpha_i\}$ emphasize component classifiers that make more reliable predictions than others

Recall

- On each iteration *t*:
  - weight each training example by how incorrectly it was classified
  - Learn a weak hypothesis – h_t
  - Obtain a strength for this hypothesis – α_t

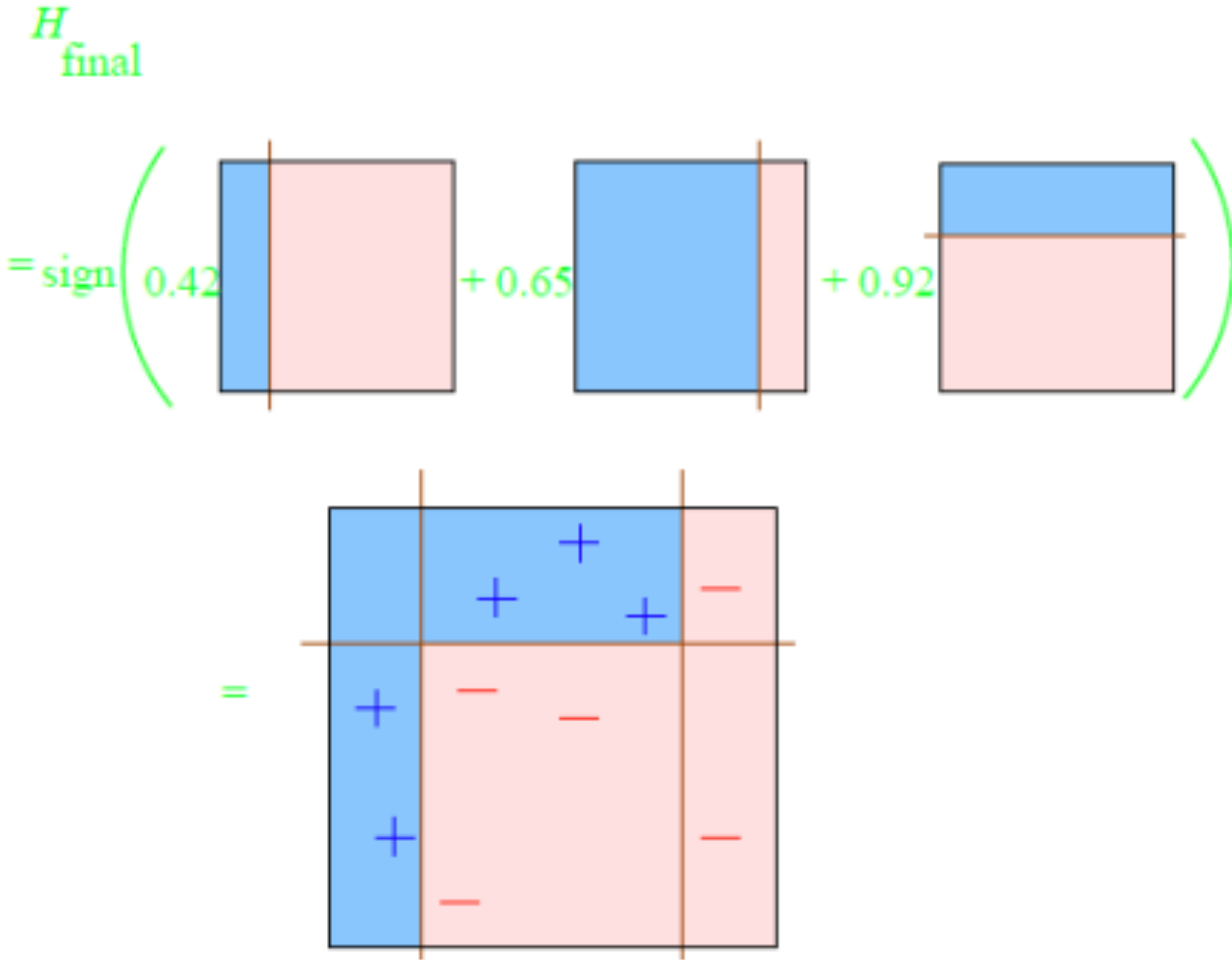
Boosting example with decision stumps

d = 2
n = 10



$\alpha_1 = 0.42$ $\alpha_2 = 0.65$ $\alpha_3 = 0.92$

Boosting example with decision stumps



AdaBoost

- **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

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

$$\varepsilon_k = \sum_{i=1}^{n} W_i^{k-1} I(y_i \neq h(\mathbf{x}_i;\theta_k^*)) \Big/ \sum_{i=1}^{n} W_i^{k-1}$$

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

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

  - stronger classifier gets more votes

epsilon small,
(1-epsilon)/epsilon is big
epsilon = 0.5 (random),
alpha = 0

- Update the weights on the training examples:

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

$$W_i^k = \exp(-y_i f(x_i))$$

Base learners

- Weak learners used in practice:
  - Decision stumps
  - 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

Exercise

- How can we modify logistic regression with kernel features to use different weights for each example?

- Can we modify naive Bayes to use different weights for each example? How might we go about checking this?

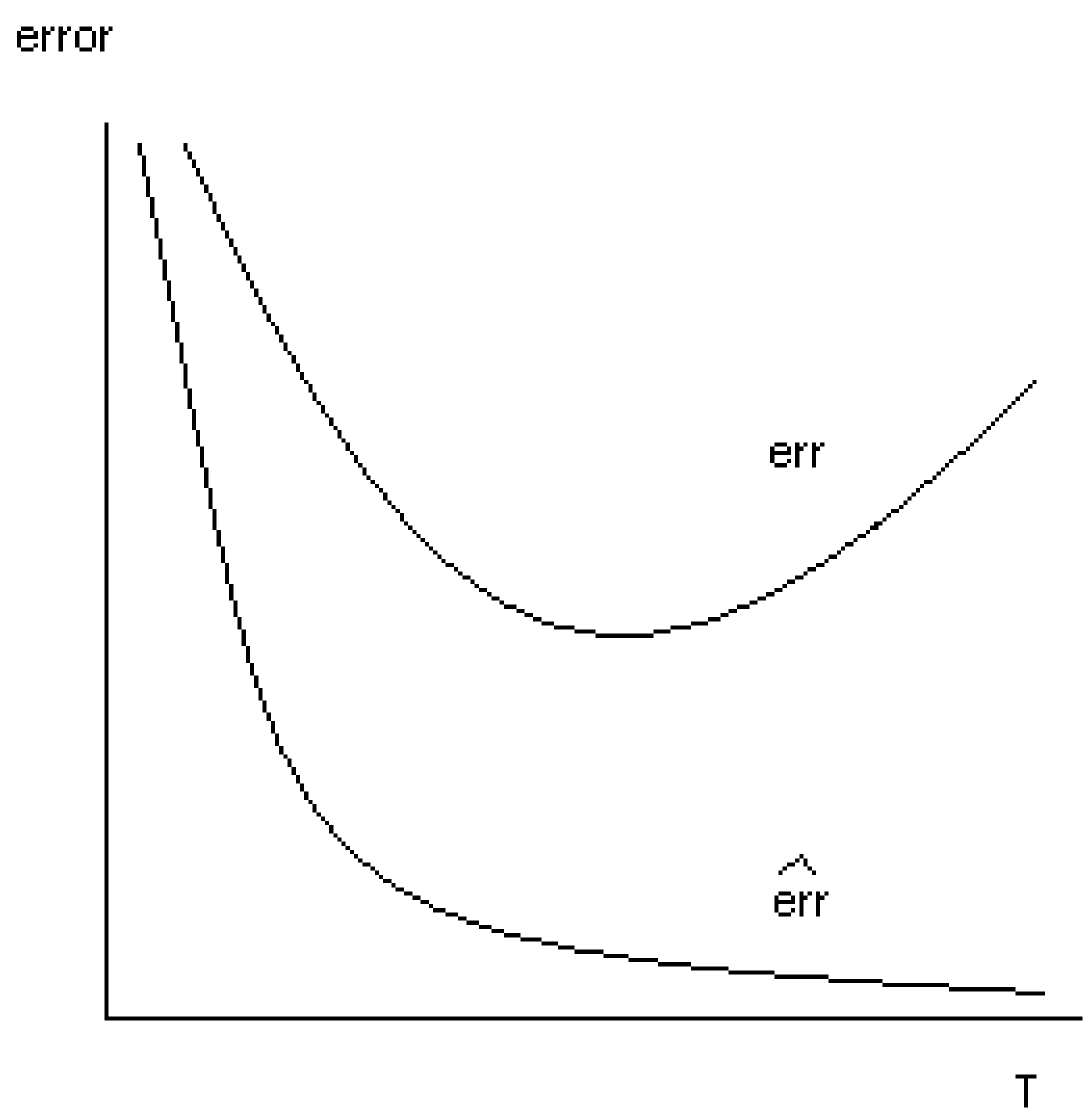
Generalization error bounds for Adaboost

$$error_{true}(H) \quad \leq \quad error_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

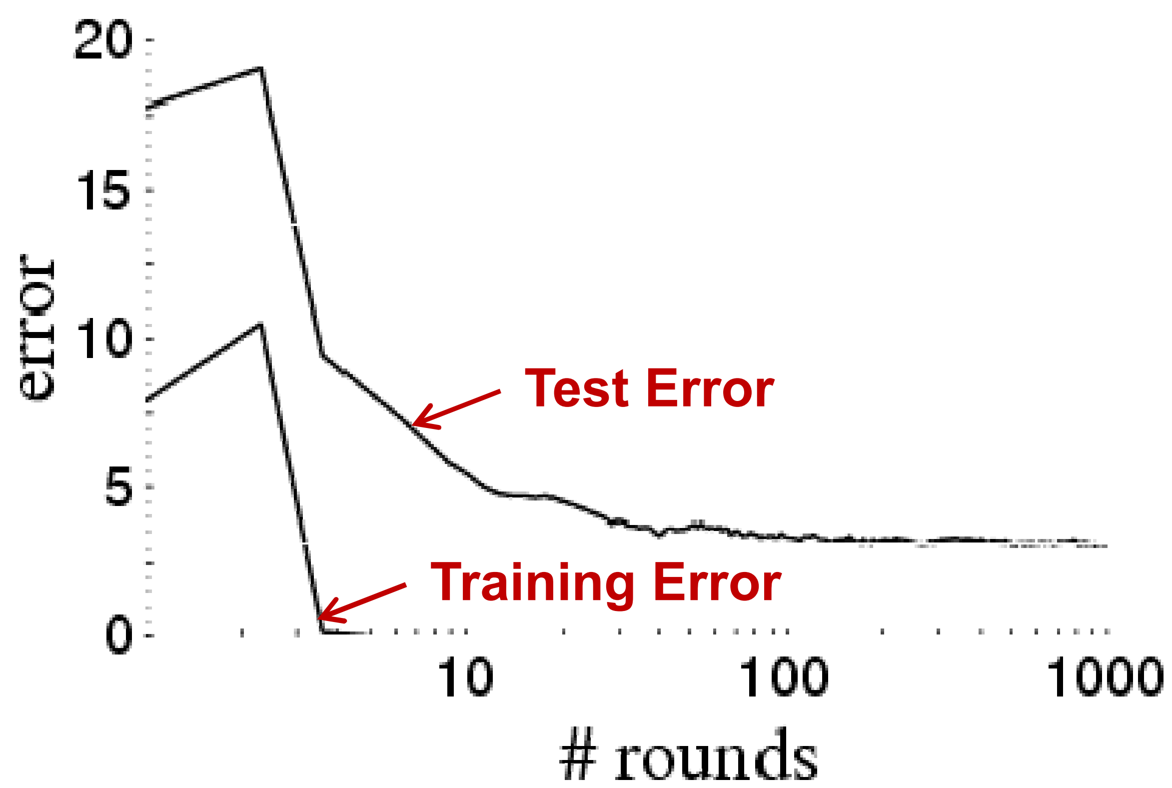
	bias	variance	
tradeoff	large	small	T small
	small	large	T large

- T – number of boosting rounds
- d – VC dimension of weak learner, measures complexity of classifier
- m – number of training examples

Expected Adaboost behavior due to overfitting



Adaboost in practice



- Boosting often, **but not always**
  - Robust to overfitting
  - Test set error decreases even after training error is zero

Why does this seem to contradict the generalization bound?

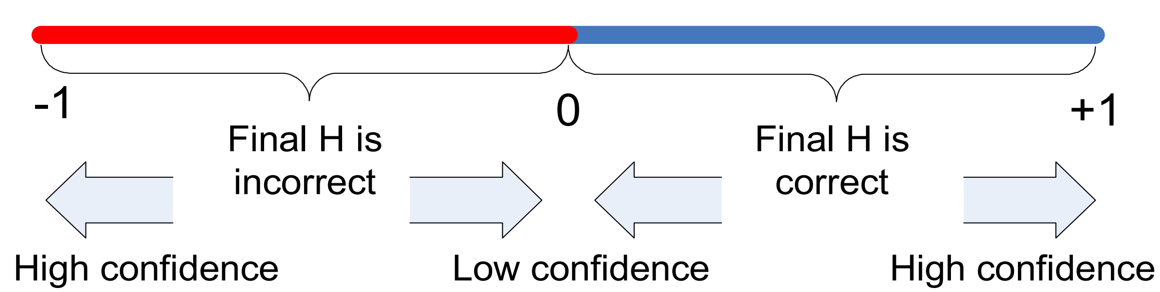
Intuition

- Even when training error becomes zero, the confidence in the hypotheses continues to increase
- Large margin in training (increase in confidence) reduces the generalization error (rather than causing overfitting)
- Quantify with margin bound, to measure confidence of a hypothesis: when a vote is taken, the more predictors agreeing, the more confident you are in your prediction

Margin

$$
\begin{aligned}
\mathrm{margin}(x,y) &= yf(x) \\
&= y\sum_t a_t h_t(x) \\
&= \sum_t a_t y h_t(x) \\
&= \sum_{t:h_t(x)=y} a_t - \sum_{t:h_t(x)\neq y} a_t
\end{aligned}
$$

where y is the correct label of instance x, and a_t is a normalized version of α_t such that $\alpha_t \geq 0$ and $\sum_t a_t = 1$. The expression $\sum_{t:h_t(x)=y} a_t$ stands for the weighted fraction of correct votes, and $\sum_{t:h_t(x)\neq y} a_t$ stands for the weighted fraction of incorrect votes. Margin is a number between -1 and 1 as shown in Figure 4.



* from http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0305.pdf

Margins and Adaboost

- AdaBoost increases the margins
- Assume gamma such that error on step t for weak learner on step t is < 1/2 - gamma (better than random)
- 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!!!

The number of boosting rounds

General Boosting

- “Boosting algorithms as gradient descent”, Mason et al, 2000
- Adaboost is only one of many choices, with exponential loss
- Other examples and comparison: see “Cost-sensitive boosting algorithms: Do we really need them?” Nikolaou et al., 2016
- Main idea: given some loss L, (implicit) set of hypotheses and a weak learning algorithm,
  - generate hypothesis ht that point in a descent direction
  - assign weight relative to how much pointing in descent direction

Boosting and logistic regression

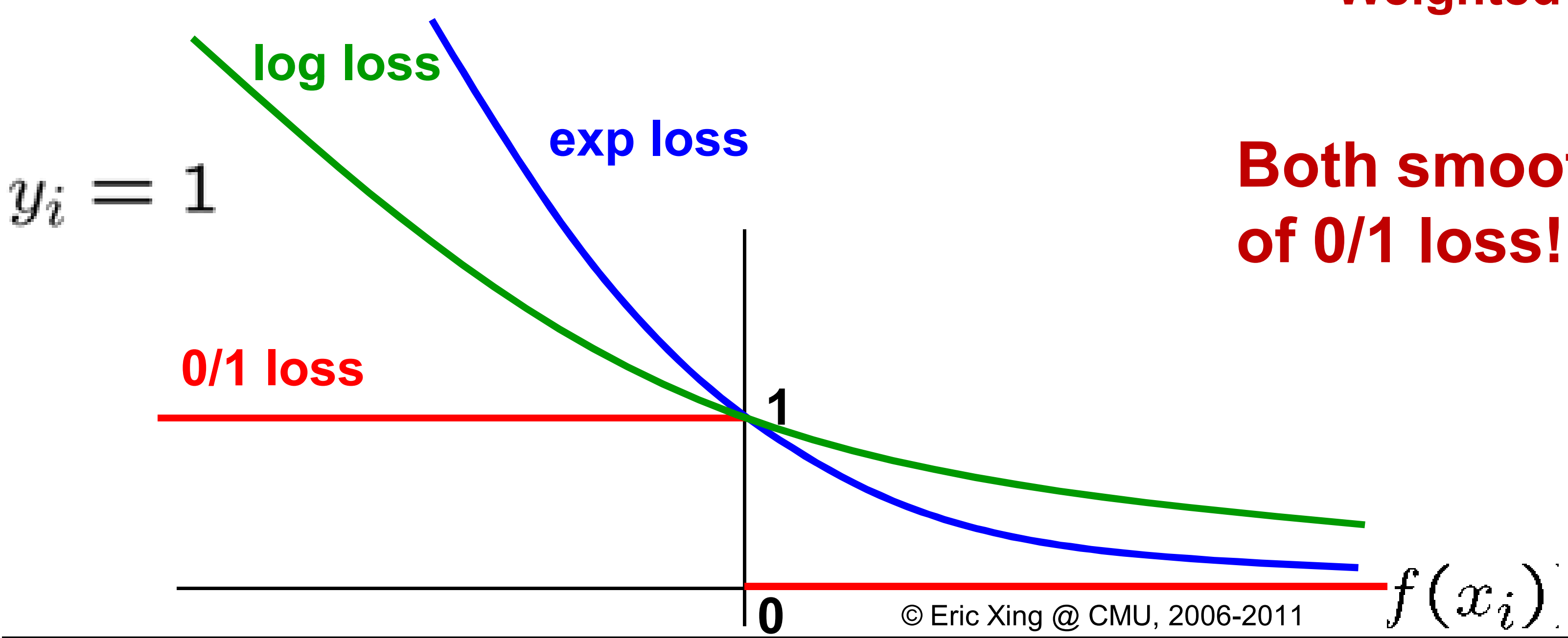
Logistic regression equivalent to minimizing log loss

$$\sum_{i=1}^{m} \ln(1 + \exp(-y_i f(x_i))) \qquad f(x) = w_0 + \sum_j w_j x_j$$

Boosting minimizes similar loss function!!

$$\frac{1}{m}\sum_{i=1}^{m} \exp(-y_i f(x_i)) = \prod_t Z_t \qquad f(x) = \sum_t \alpha_t h_t(x)$$

Weighted average of weak learners



Both smooth approximations of 0/1 loss!