Machine Learning

CS 486/686
Introduction to AI
University of Waterloo

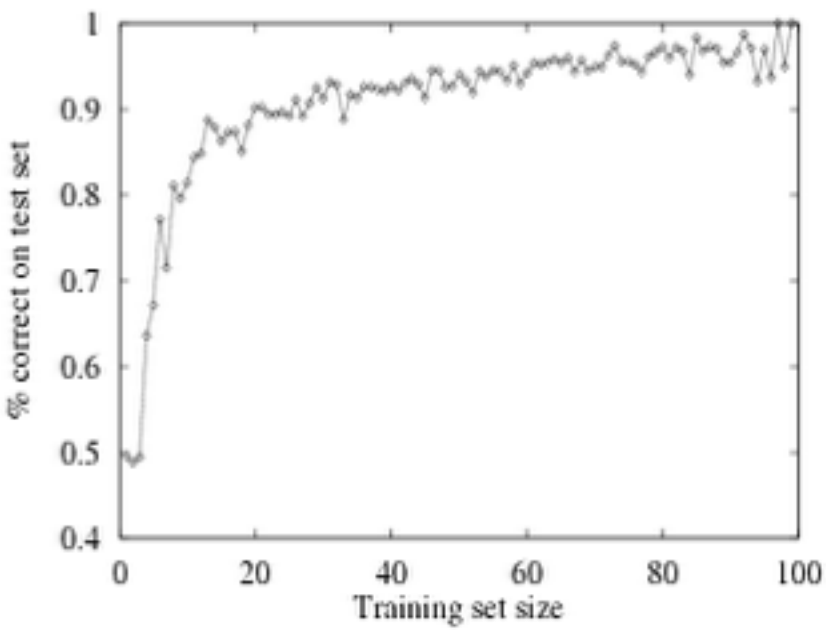
Assessing Performance of a Learning Algorithm

- A learning algorithm is **good** if it produces a hypothesis that does a good job of predicting classifications of unseen examples
- There are theoretical guarantees (learning theory)
- Can also test this

Assessing Performance of a Learning Algorithm

- Test set
  - Collect a large set of examples
  - Divide them into 2 disjoint sets: training set and test set
  - Apply learning algorithm to the training set to get h
  - Measure percentage of examples in the test set that are correctly classified by h

Learning Curves



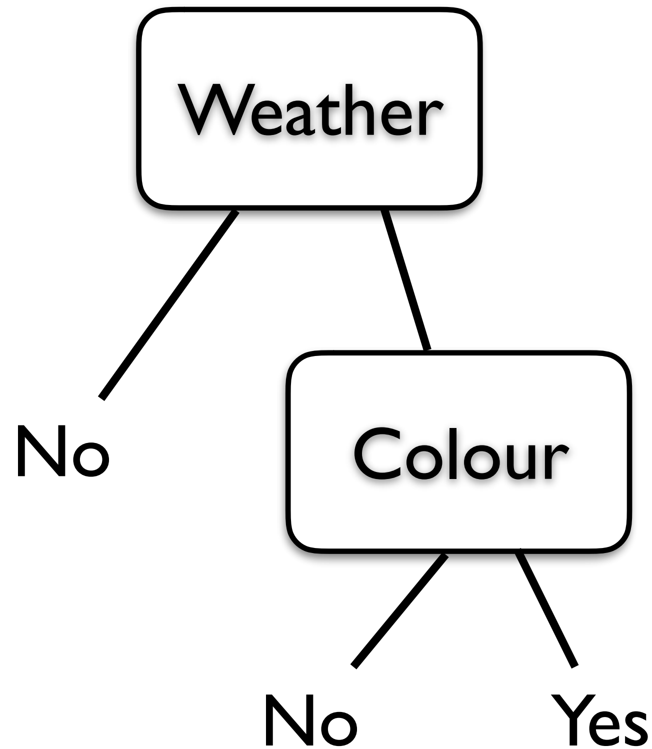
As the training set grows, accuracy increases

Overfitting

- Why might a consistent hypothesis have a high error rate on a test set?
- Overfitting
  - Finding patterns in the data where there is no actual pattern

Overfitting

Training Data			f(x)
red	sunny	3	0
blue	sunny	6	1
red	sunny	1	0
blue	sunny	6	1
red	rain	2	0
blue	rain	1	0
red	rain	3	0
blue	rain	2	0
red	rain	5	0
blue	rain	4	0



Overfitting

- Given a hypothesis space H, a hypothesis h in H is said to overfit the training data if there exists some alternative hypothesis h' in H such that h has smaller error than h' on the training examples, but h' has smaller error than h over the entire distribution of instances
  - h in H overfits if there exists h' in H such that $error_{Tr}(h) < error_{Tr}(h')$ but $error_{Te}(h') < error_{Te}(h)$

- Overfitting has been found to decrease accuracy of decision trees by 10-25%

Avoiding Overfitting

- Pruning
  - Assume there is no pattern in the data (null hypothesis)
    - Attribute is irrelevant and so info gain would be 0 for an infinitely large sample
  - Compute probability that (under null hypothesis) a sample size p+n would exhibit observed deviation

$$\hat{p}_i = p\frac{p_i + n_i}{p + n} \qquad \hat{n}_i = n\frac{p_i + n_i}{p + n}$$

$$D = \sum_{i=1}^{v} \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

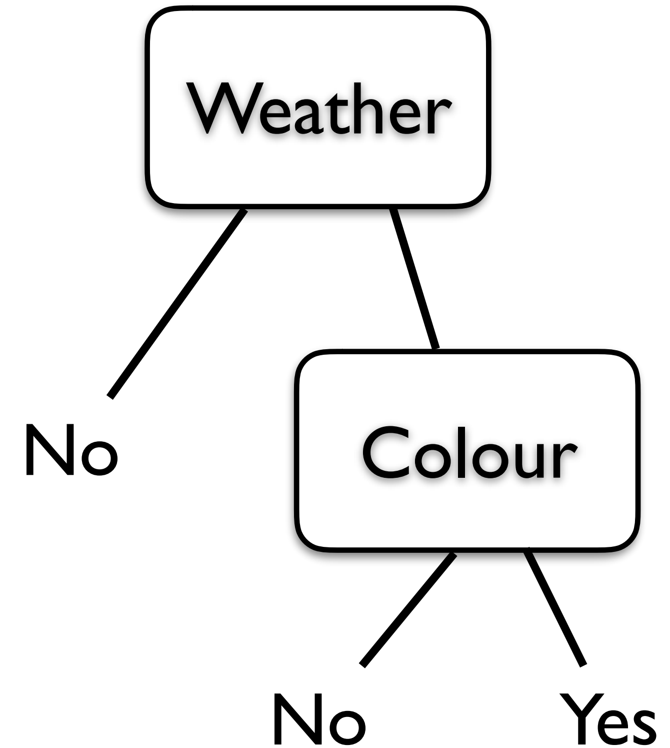
$$\text{compare to } \chi^2 \text{ table}$$

Overfitting

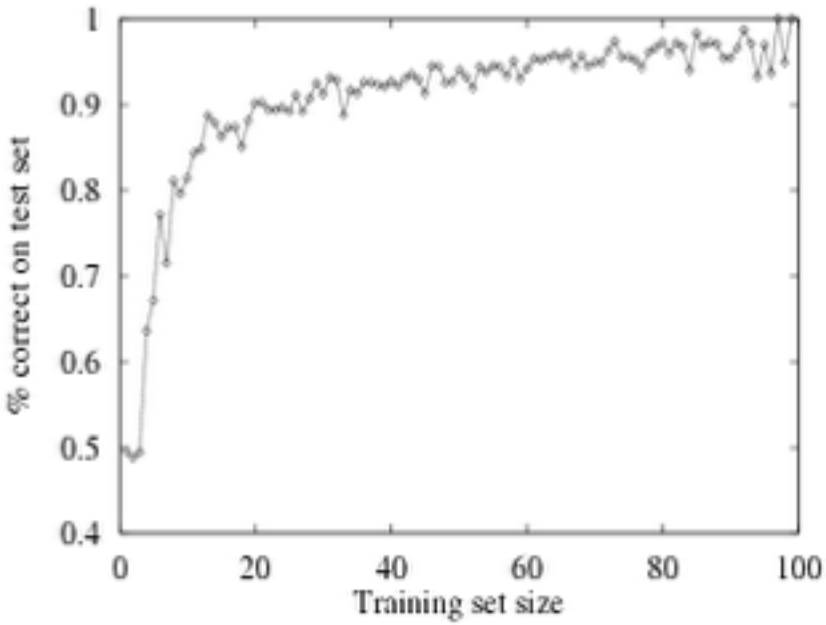
Training Data f(x)

red	sunny	3	0
blue	sunny	6	1
red	sunny	1	0
blue	sunny	6	1
red	rain	2	0
blue	rain	1	0
red	rain	3	0
blue	rain	2	0
red	rain	5	0
blue	rain	4	0

Weather can be pruned



Learning Curves



As the training set grows, accuracy increases

No Peeking at the Test Set!

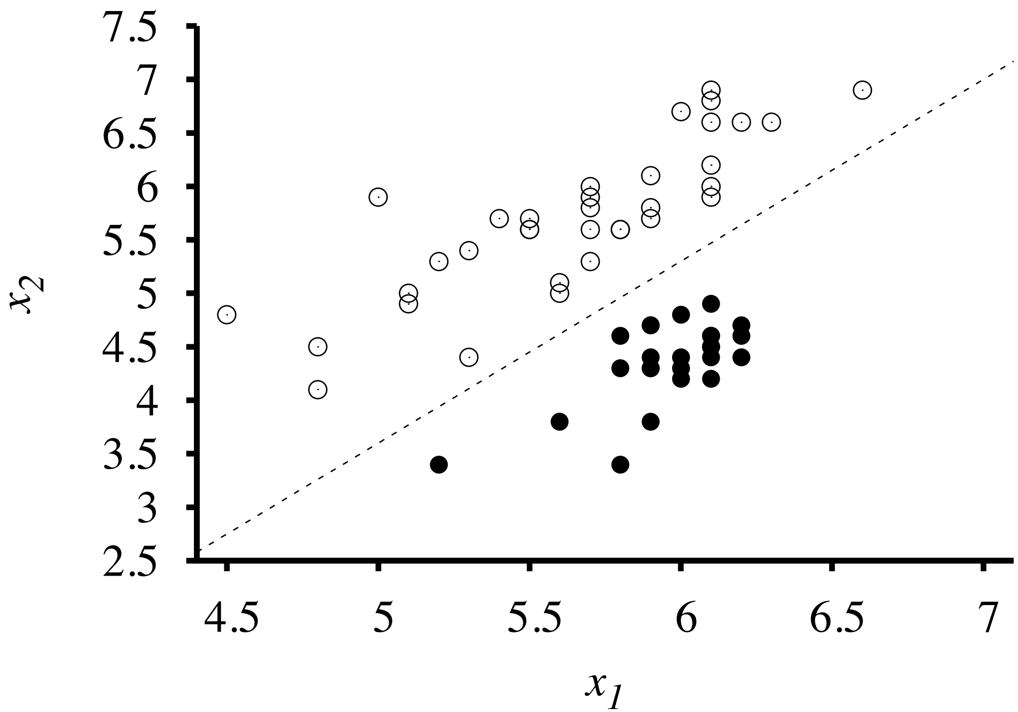
- A learning algorithm should not be allowed to see the test set data before the hypothesis is tested on it
  - ***No Peeking!!***
- Every time you want to compare performance of a hypothesis on a test set ***you should use a new test set***!

Cross Validation

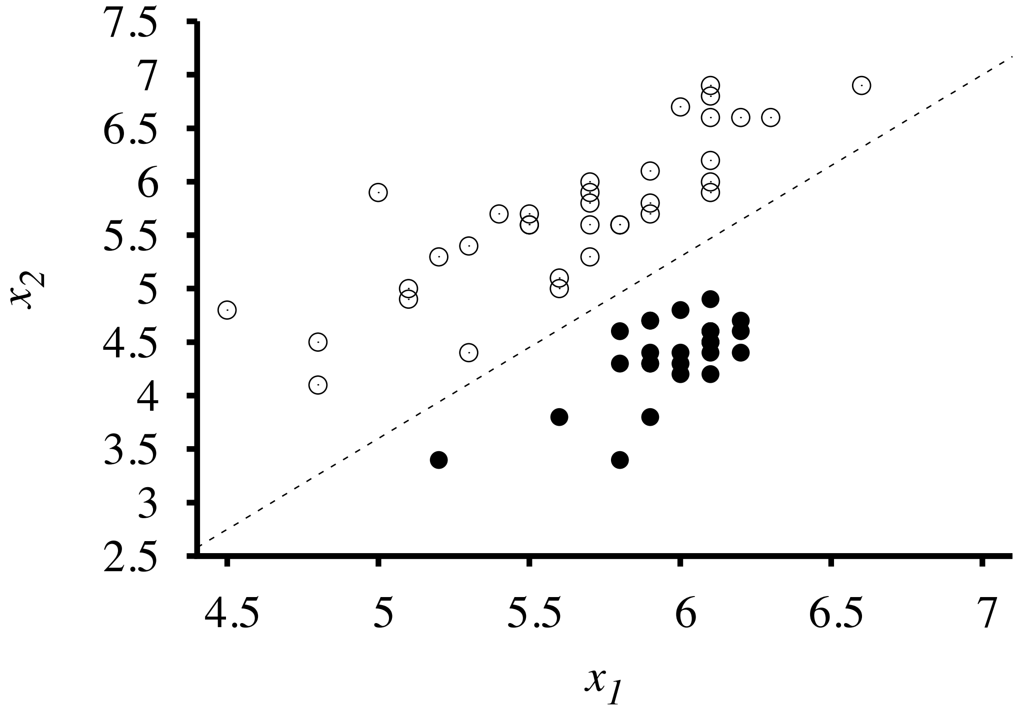
- Split the training set into two parts, one for training and one for choosing the hypothesis with highest accuracy
  - K-fold cross validation means you run k experiments, each time putting aside 1/k of the data to test on
  - Leave-one-out cross validation

Linear Threshold Classifiers

Imagine you have data of the form(**x**,f(**x**)) where **x** in $\boldsymbol{R}^n$ and f(**x**)=0 or 1



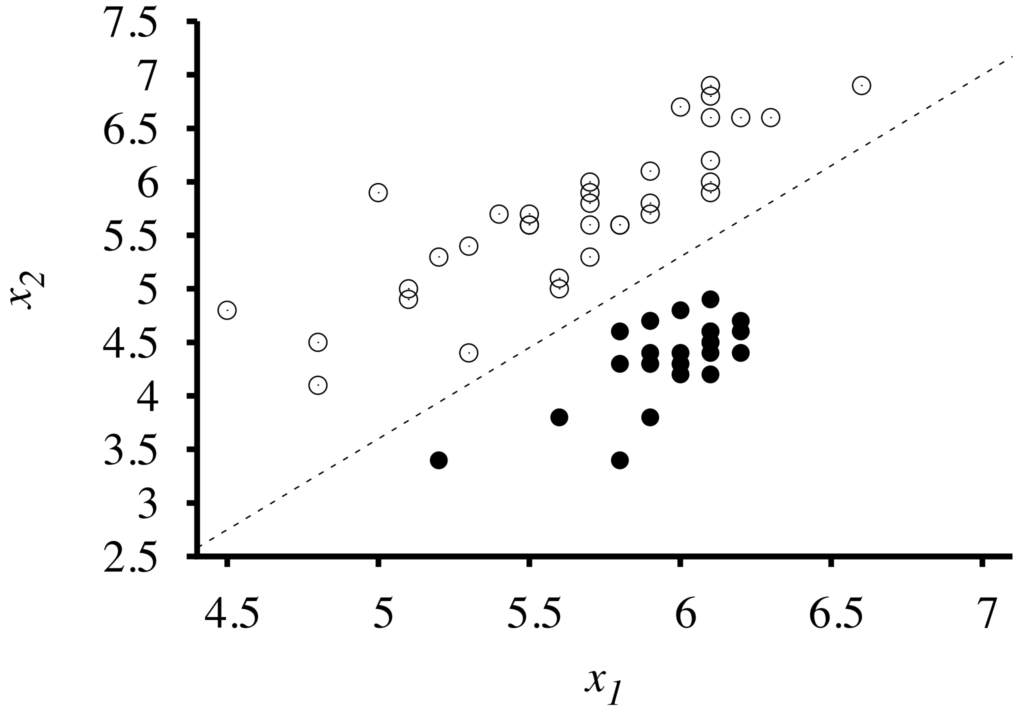
Linear Threshold Classifiers



$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{w} \cdot \mathbf{x} = w_0 + w_1 x_1 + w_2 x_2$$

Linear Threshold Classifiers



$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Learning problem: Find weights, w, to minimize loss

$$\text{Loss}(h_{\mathbf{w}}) = L_2(y, h_{\mathbf{w}}(\mathbf{x})) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2$$

$$w_i \leftarrow w_i + \alpha(y - h_{\mathbf{w}}(\mathbf{x})) \cdot x_i$$

Ensemble learning

- So far our learning methods have had the following general approach
  - Choose a single hypothesis from the hypothesis space
  - Use this hypothesis to make predictions

- Maybe we can do better by using a lot of hypothesis from the hypothesis space and combine their predictions

Ensemble Learning

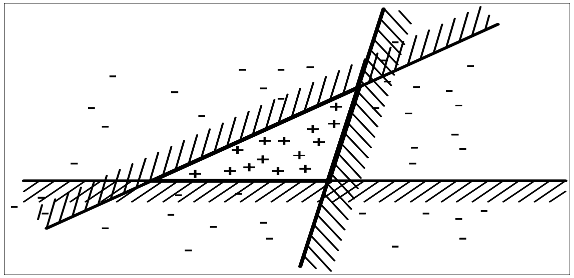
- Analogies
  - Elections
  - Committees
- Intuitions:
  - Individuals may make mistakes
    - The majority may be less likely to make a mistake
  - Individuals have partial information
    - Committees pool expertise

Ensemble expressiveness

- Using ensembles can also enlarge the hypothesis space
  - Ensemble as hypothesis
  - Set of all ensembles as hypothesis space

Original hypothesis space: linear threshold hypothesis

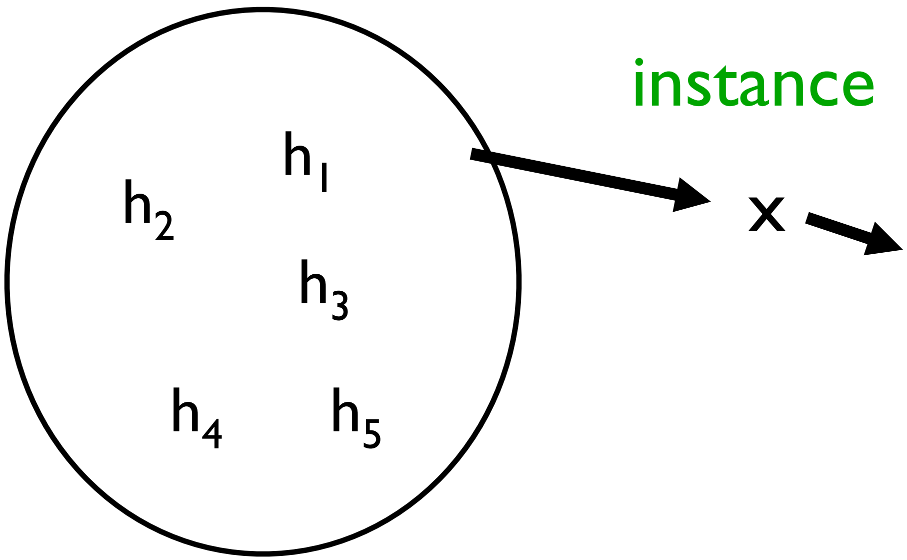
- Simple, efficient learning algorithms but not particularly expressive



Bagging

- Majority voting:

instance



classification

$\text{Majority}(h_1(x), h_2(x), h_3(x), h_4(x), h_5(x))$

For the classification to be wrong, at least 3 out of 5 hypothesis have to be wrong

Ensemble of hypothesis

Bagging

- Assumptions:
  - Each h_i makes an error with probability p
  - Hypotheses are independent

- Majority voting of n hypotheses
  - Probability k make an error?
  - Probability majority make an error?

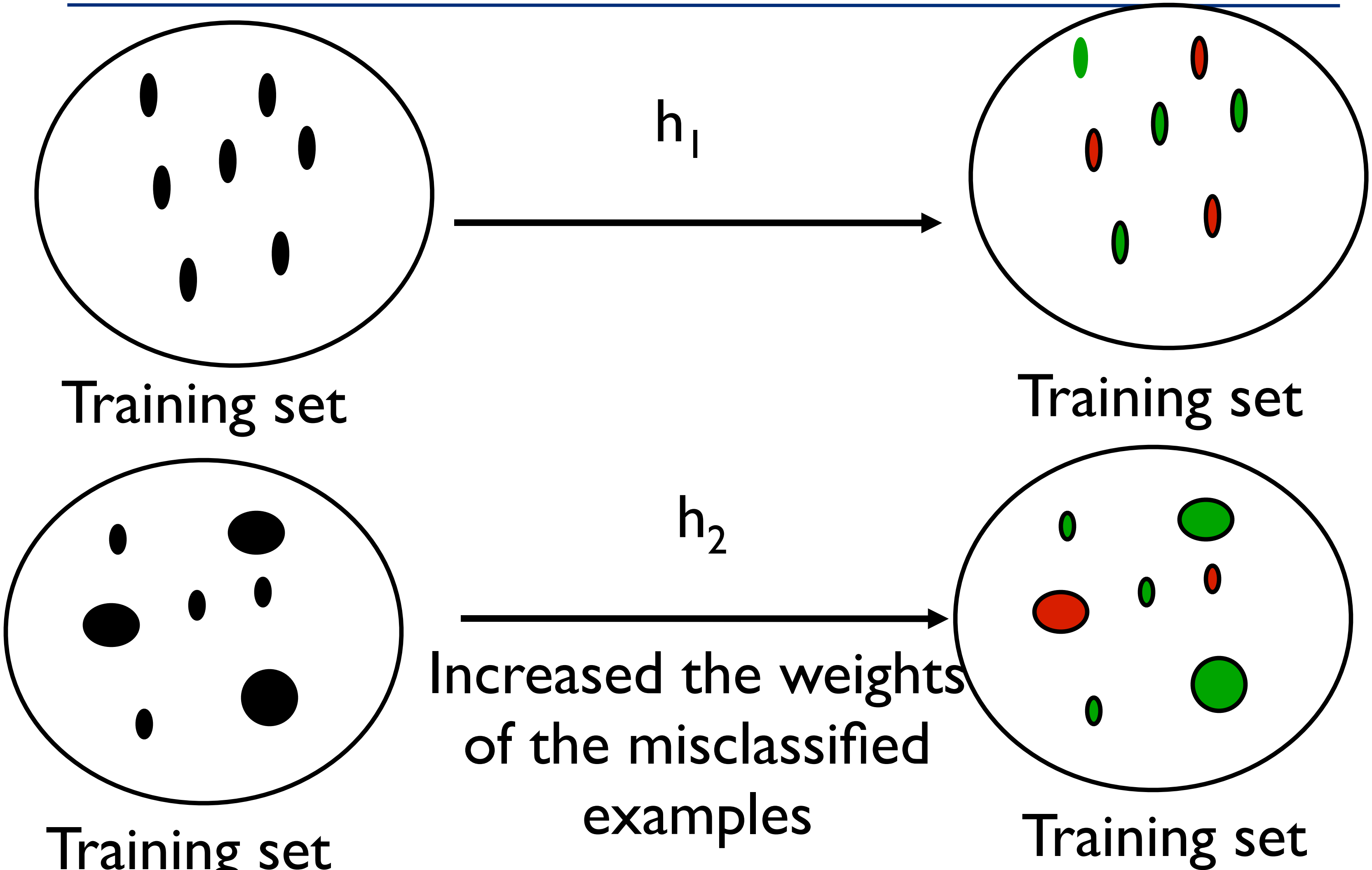
Weighted Majority

- In practice
  - Hypotheses are rarely independent
  - Some hypotheses have less errors than others
- Weighted majority
  - Intuition
    - Decrease weights of correlated hypotheses
    - Increase weights of good hypotheses

Boosting

- Boosting is the most commonly used form of ensemble learning
  - Very simple idea, but very powerful
    - Computes a weighted majority
    - Operates on a weighted training set

Boosting



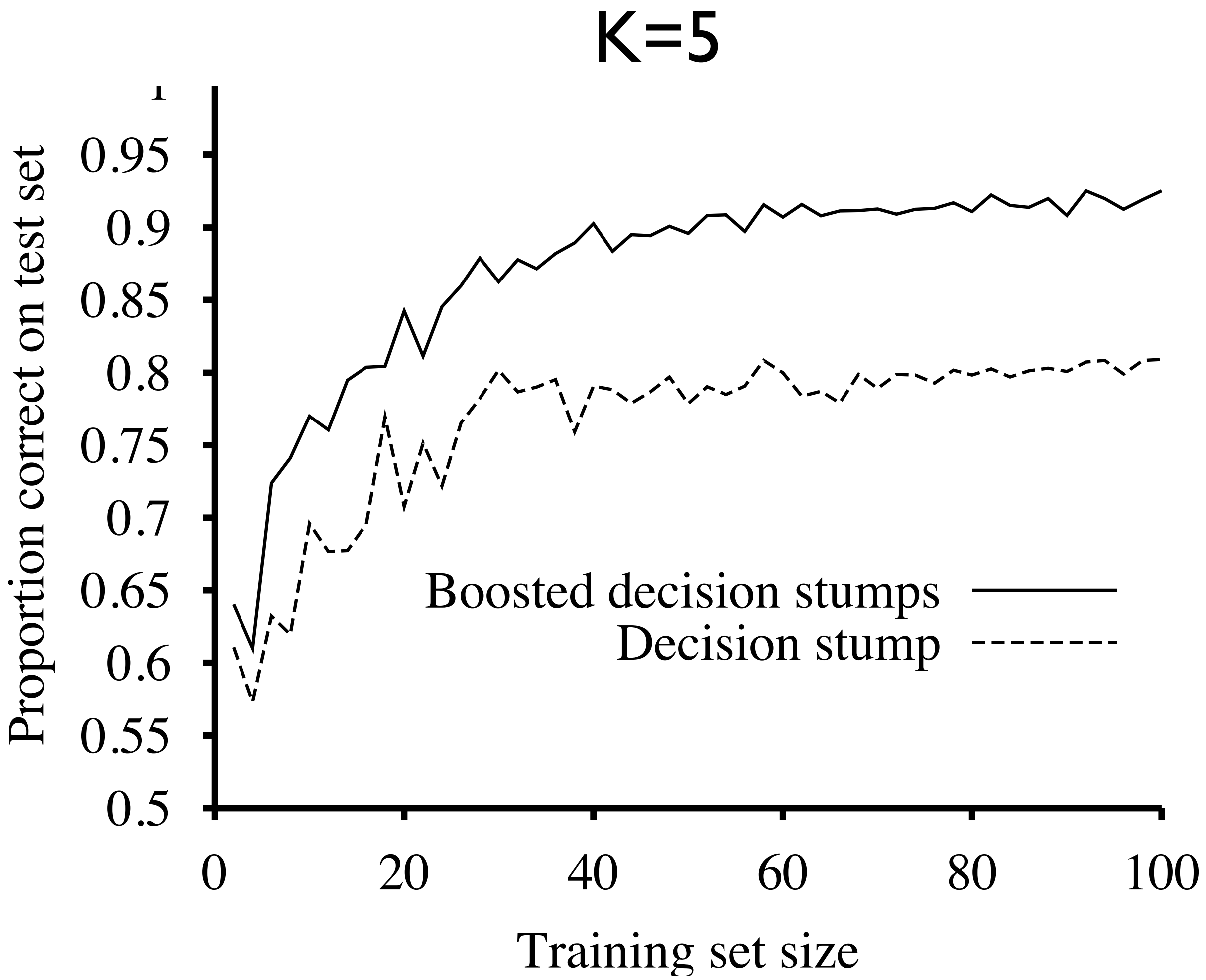
h_1
Training set
Training set
h_2
Increased the weights of the misclassified examples
Training set
Training set

AdaBoost

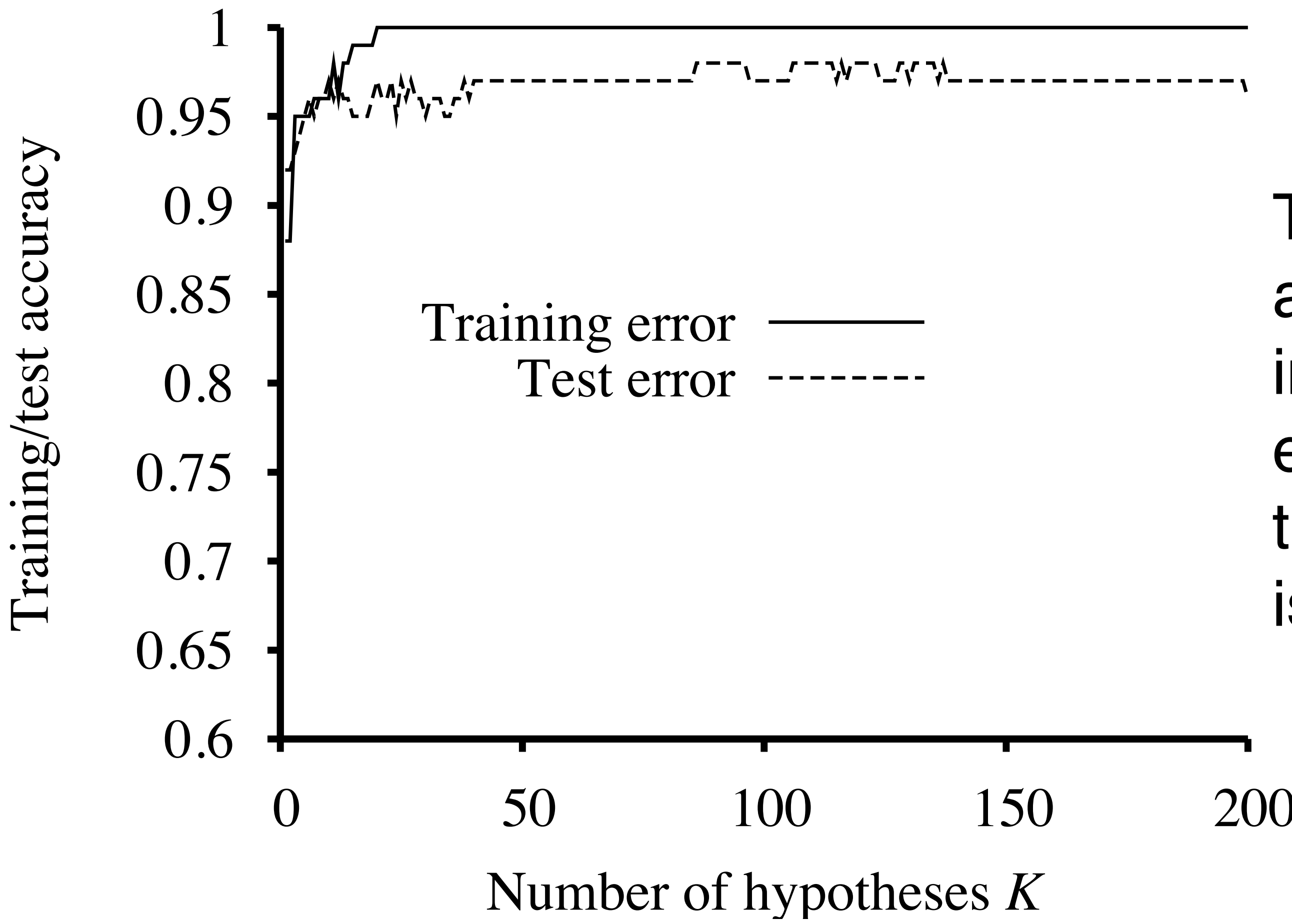
function ADABOOST(*examples*, L, K) **returns** a weighted-majority hypothesis
  inputs: *examples*, set of N labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$
      L, a learning algorithm
      K, the number of hypotheses in the ensemble
  local variables: $\mathbf{w}$, a vector of N example weights, initially $1/N$
      $\mathbf{h}$, a vector of K hypotheses
      $\mathbf{z}$, a vector of K hypothesis weights

  for $k = 1$ **to** K **do**
    $\mathbf{h}[k] \leftarrow L(examples, \mathbf{w})$
    $error \leftarrow 0$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \mathbf{w}[j]$
    for $j = 1$ **to** N **do**
      if $\mathbf{h}[k](x_j) = y_j$ **then** $\mathbf{w}[j] \leftarrow \mathbf{w}[j] \cdot error/(1 - error)$
    $\mathbf{w} \leftarrow$ NORMALIZE($\mathbf{w}$)
    $\mathbf{z}[k] \leftarrow \log\,(1 - error)/error$
  return WEIGHTED-MAJORITY($\mathbf{h}, \mathbf{z}$)

Boosting


K=5
Proportion correct on test set
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
20
40
60
80
100
Training set size
Boosted decision stumps
Decision stump

Boosting



Test set accuracy still improves slightly even after training accuracy is equal to 1

Boosting

- Many variations of boosting
  - ADABOOST is a specific boosting algorithm
  - Takes a weak learner L (classifies slightly better than just random guessing)
  - Returns a hypothesis that classifies training data with 100% accuracy (for large enough M)

Robert Schapire and Yoav Freund
Kanellakis Award for 2004

Boosting Paradigm

- Advantages
  - No need to learn a perfect hypothesis
  - Can boost any weak learning algorithm
  - Easy to program
  - Good generalization
- When we have a bunch of hypotheses, boosting provides a principled approach to combine them
  - Useful for sensor fusion, combining experts…