Machine Learning

CS 486/686: Introduction to Artificial Intelligence

What is Machine Learning

Definition:

A computer program is said to **learn** from **experience** E with respect to some class of **tasks** T and **performance measures** P, if its performance at tasks in T, as measured by P, improves with experience E.

[T Mitchell, 1997]

Examples

- A checkers learning problem
  - T: playing checkers
  - P: percent of games won against an opponenet
  - E: playing practice games against itself
- Handwriting recognition
  - T: recognize and classify handwritten words within images
  - P: percent of words correctly classified
  - E: database of handwritten words with given classifications

Examples

- Autonomous driving:
  - T: driving on a public four-lane highway using vision sensors
  - P: average distance traveled before an error was made (as judged by human overseer)
  - E: sequence of images and steering commands recorded while observing a human driver

Types of Learning

Supervised (Inductive) Learning

- Learn a function from examples of its inputs and outputs
- Example scenario: Handwriting recognition
- Techniques:
  - Decision trees, SVM, Neural Nets,…

Types of Learning

Unsupervised learning

- Learn patterns in the input when no specific output is given
- Example scenario: Cluster web log data to discover groups of similar access patterns
- Techniques: Clustering

Type of Learning

Reinforcement learning

- Agents learn from feedback (rewards and punishments)
- Example scenario: Game-playing agent
- Techniques: TD-learning, Q-learning,…

Representation

- Representation of learned information is important
  - Determines how the learning algorithm will work
- Common representations:
  - Linear weighted polynomials
  - Propositional logic
  - First order logic
  - Bayes nets
  - ...

Supervised Learning

- Given a **training set** of examples of the form (x,f(x))
  - x is the input, f(x) is the output
- Return a function h that approximates f
  - h is the hypothesis

attribute

Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

x

f(x)

Hypothesis Representation

- We need a hypothesis representation for the problem
  - A reasonable candidate for our example is a conjunction of constraints
  - Vector of 6 constraints specifying the values of the 6 attributes
    - ? to denote that any value is acceptable
    - Specify a single required value (e.g. Warm)
    - 0 to specify that no value is acceptable
  - If some instance satisfies all constraints of hypothesis h, then h classifies x as a positive example (h(x)=1)
  - h=<?,Cold,High,?,?,?> represents a hypothesis that someone enjoys her favorite sport only on cold days with high humidity

Hypothesis Space

- Most general hypothesis
  - <?,?,?,?,?,?> (every day is a positive example)
- Most specific hypothesis
  - <0,0,0,0,0,0> (no day is a positive example)
- **Hypothesis space,** H
  - Set of all possible hypothesis that the learner may consider regarding the target concept
  - Can think of learning as a search through a hypothesis space

Learning Problem

Our goal is to find a **good** hypothesis

Why is this hard?

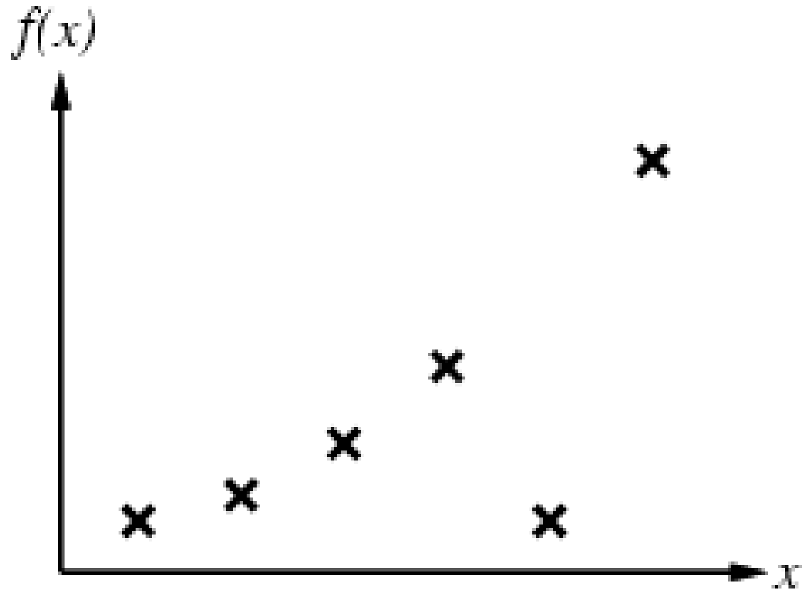
Inductive Learning Hypothesis

Any hypothesis found to approximate the target function well ***over a sufficiently large set of training examples*** will also approximate the target function well over any unobserved examples

Supervised Learning

Construct/adjust h to agree with f on training set

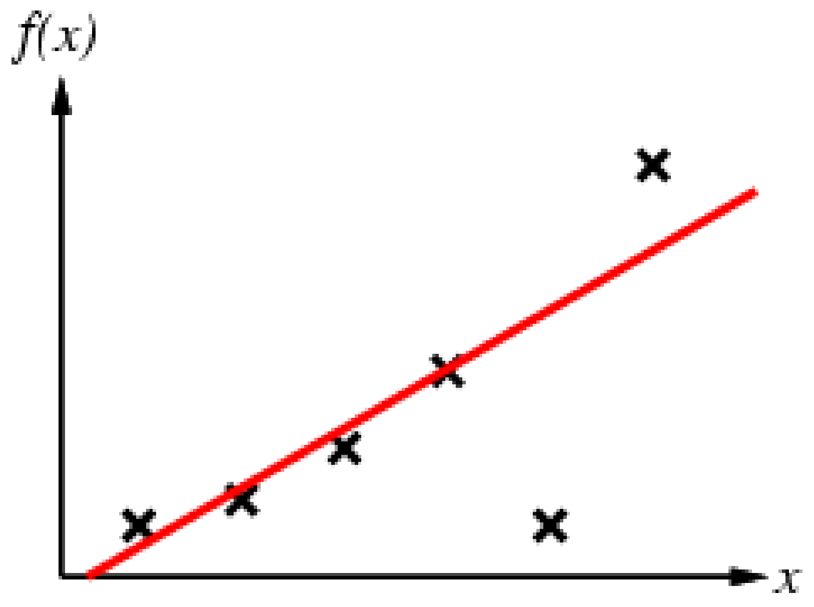
- h is **consistent** if it agrees with f on all examples
- e.g. curve fitting



Supervised Learning

Construct/adjust h to agree with f on training set

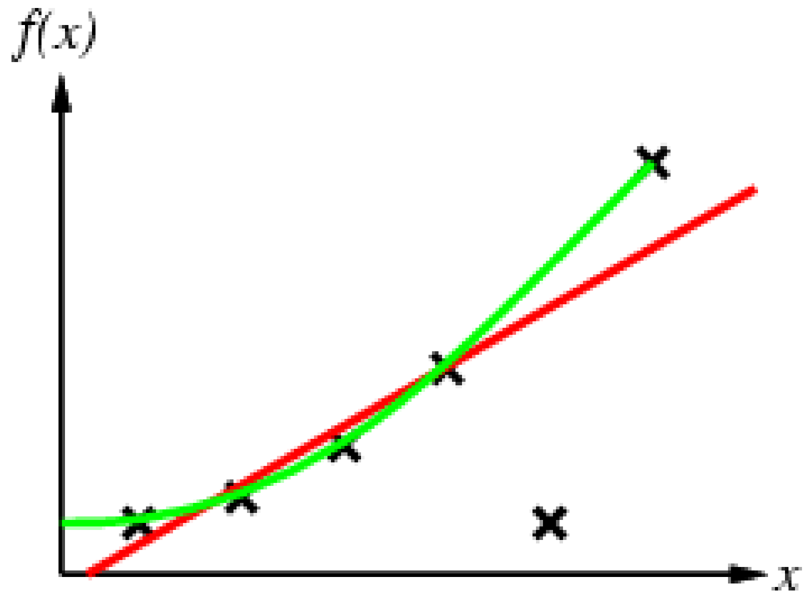
- h is **consistent** if it agrees with f on all examples
- e.g. curve fitting



Supervised Learning

Construct/adjust h to agree with f on training set

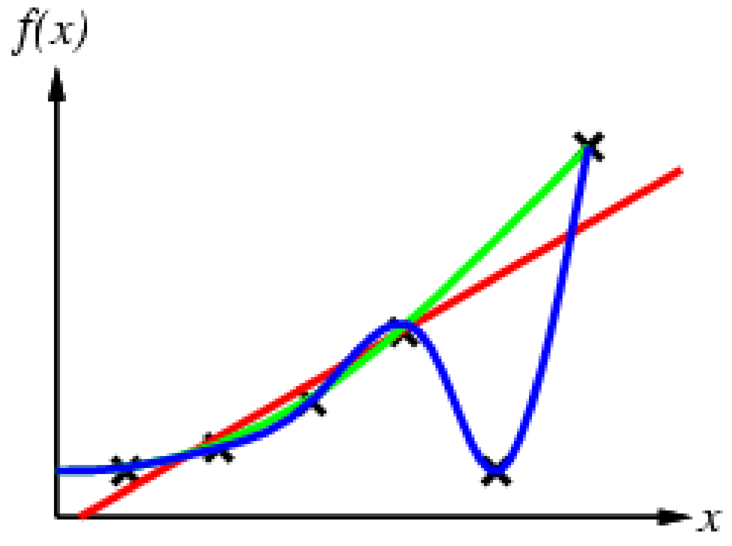
- h is **consistent** if it agrees with f on all examples
- e.g. curve fitting



Supervised Learning

Construct/adjust h to agree with f on training set

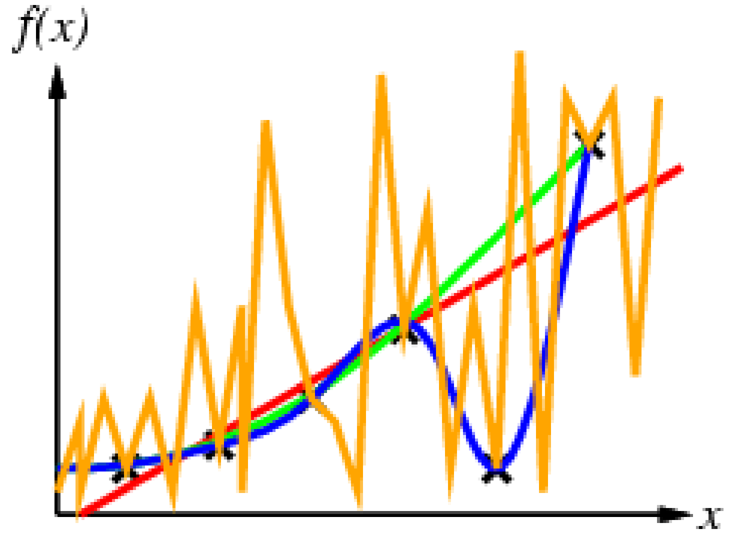
- h is **consistent** if it agrees with f on all examples
- e.g. curve fitting



Supervised Learning

Construct/adjust h to agree with f on training set

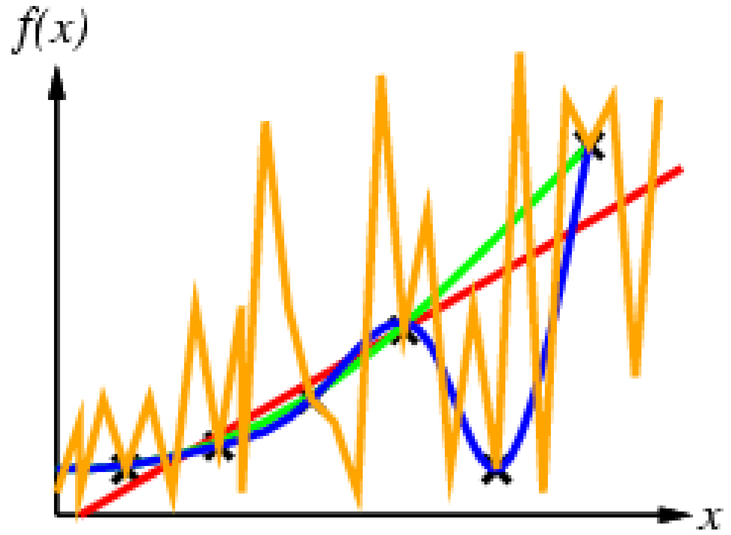
- h is **consistent** if it agrees with f on all examples
- e.g. curve fitting



Supervised Learning

Construct/adjust h to agree with f on training set

- h is **consistent** if it agrees with f on all examples
- e.g. curve fitting



Ockham’s Razor: Prefer the simplest hypothesis consistent with the data

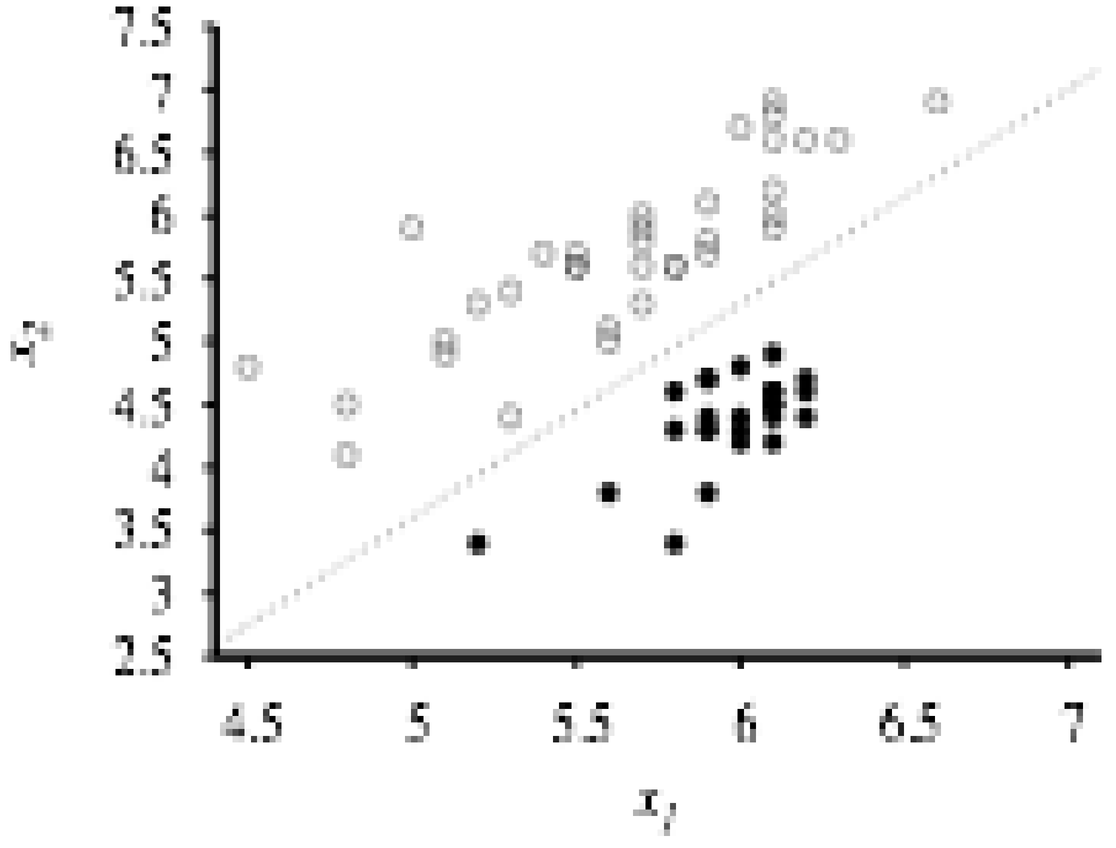
Supervised Learning

- Possibility of finding a single consistent hypothesis depends on the hypothesis space
  - **Realizable**: hypothesis space contains the true function

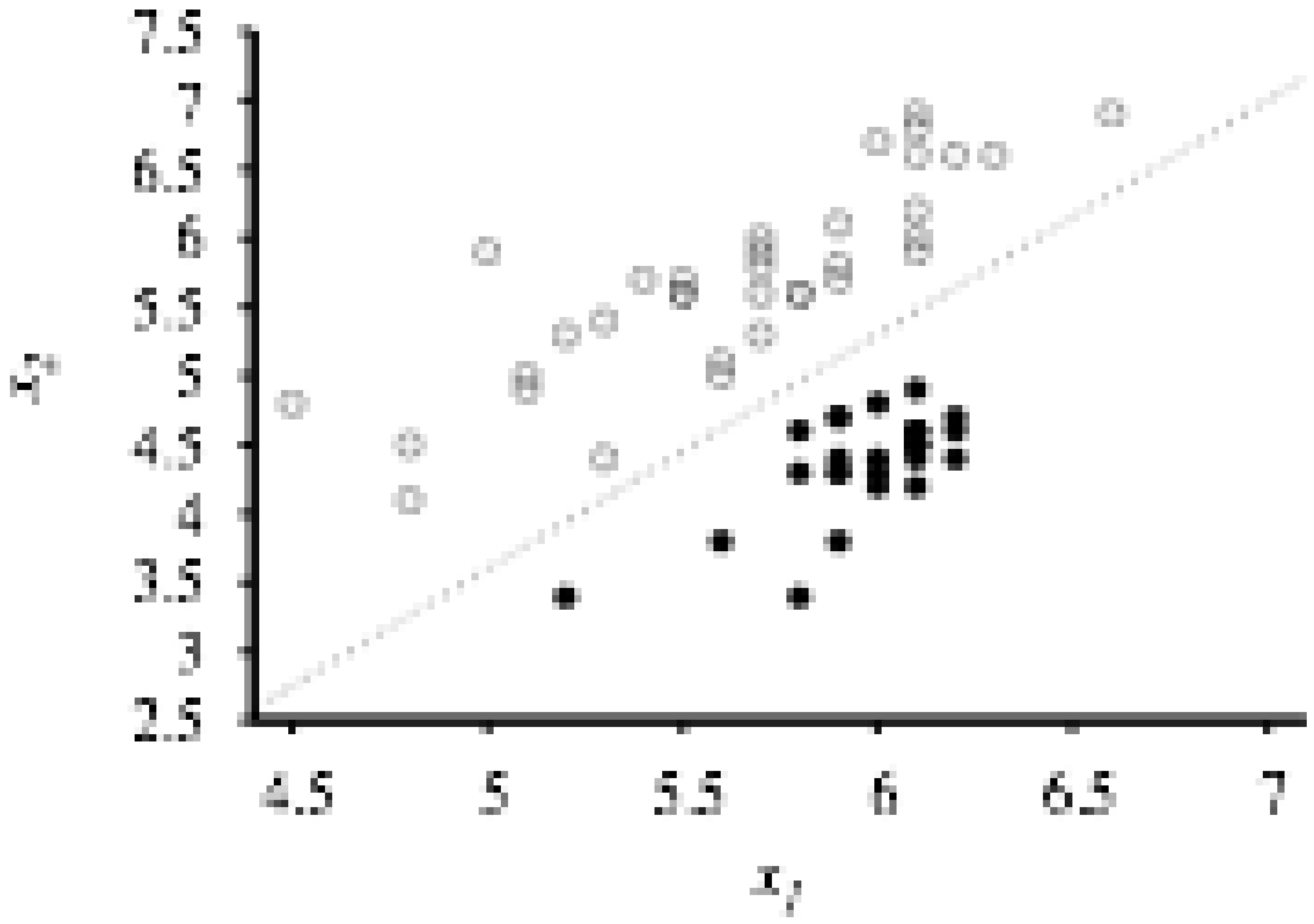
- Can use large hypothesis space (e.g. space of all Turing machines)
  - Tradeoff between expressiveness and complexity of finding a simple consistent hypothesis

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



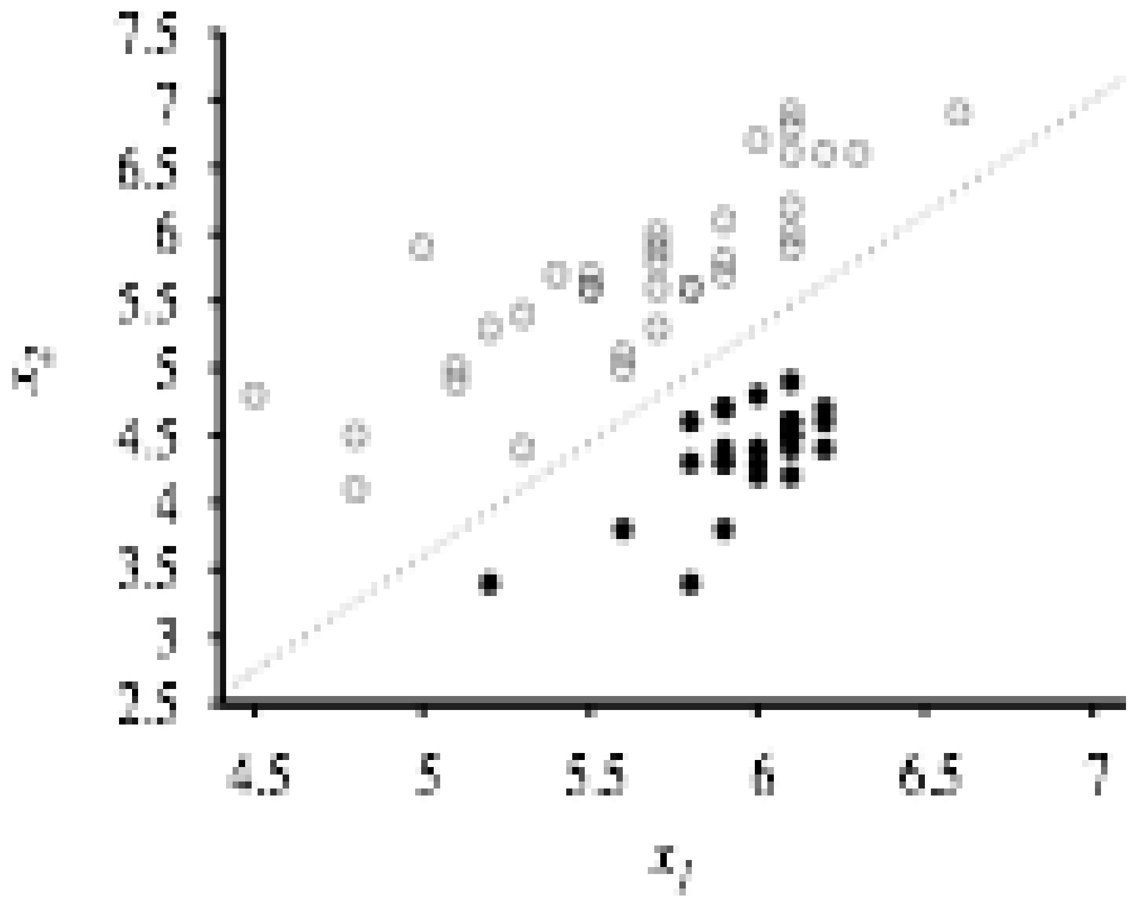
Linear Threshold Classifiers



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

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

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

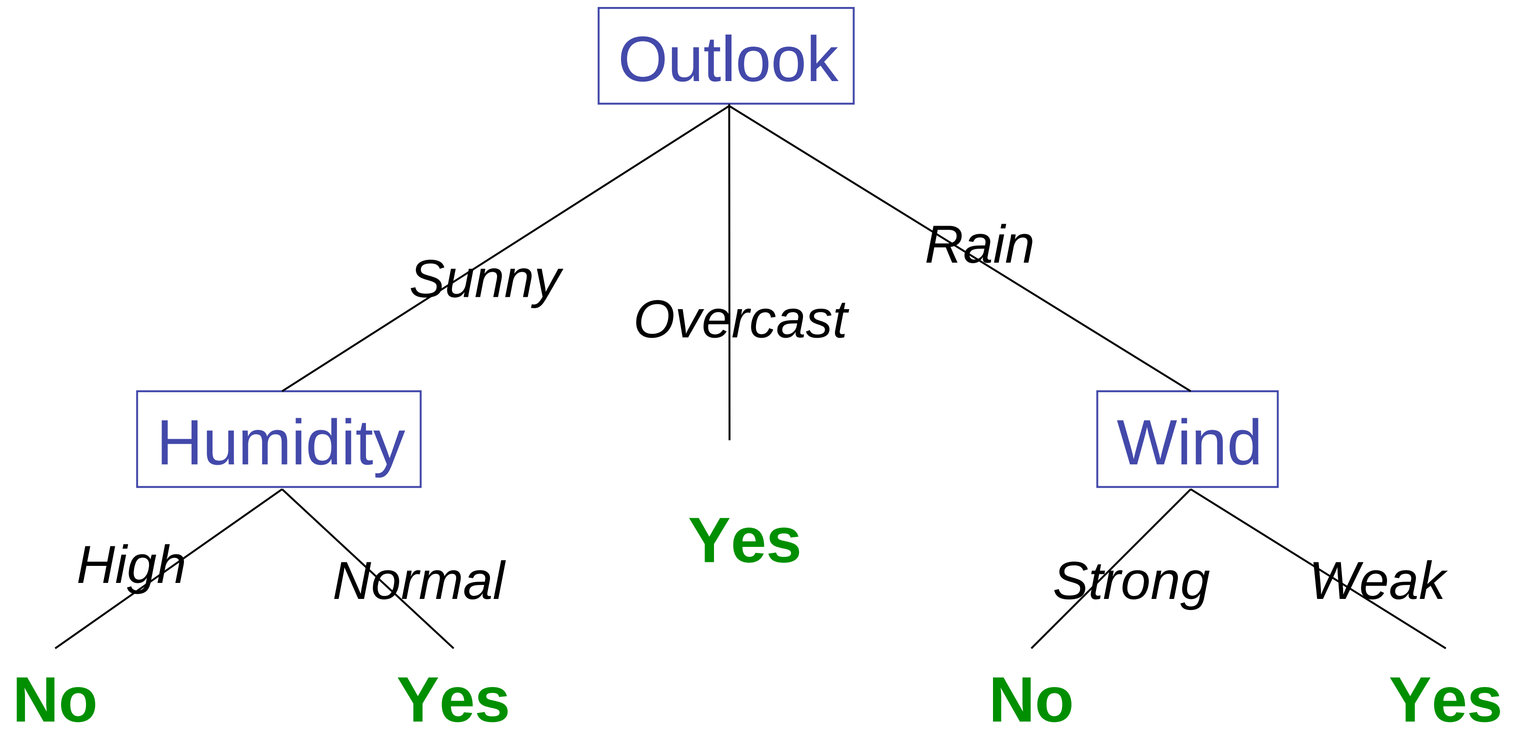
$$\text{Loss}(h_{\mathbf{w}}) = L_2(y, h_{\mathbf{w}}(\mathbf{x})) = \sum_{j=1}^{N}(y_j - h_{\mathbf{w}}(\mathbf{x}_j))^2$$

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

Decision Trees

- Decision trees classify instances by sorting them down the tree from root to leaf
  - Nodes correspond with a test of some attribute
  - Each branch corresponds to some value an attribute can take
- Classification algorithm
  - Start at root, test attribute specified by root
  - Move down the branch corresponding to value of the attribute
  - Continue until you reach leaf (classification)

Decision Tree



An instance

<Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong>

Classification: No

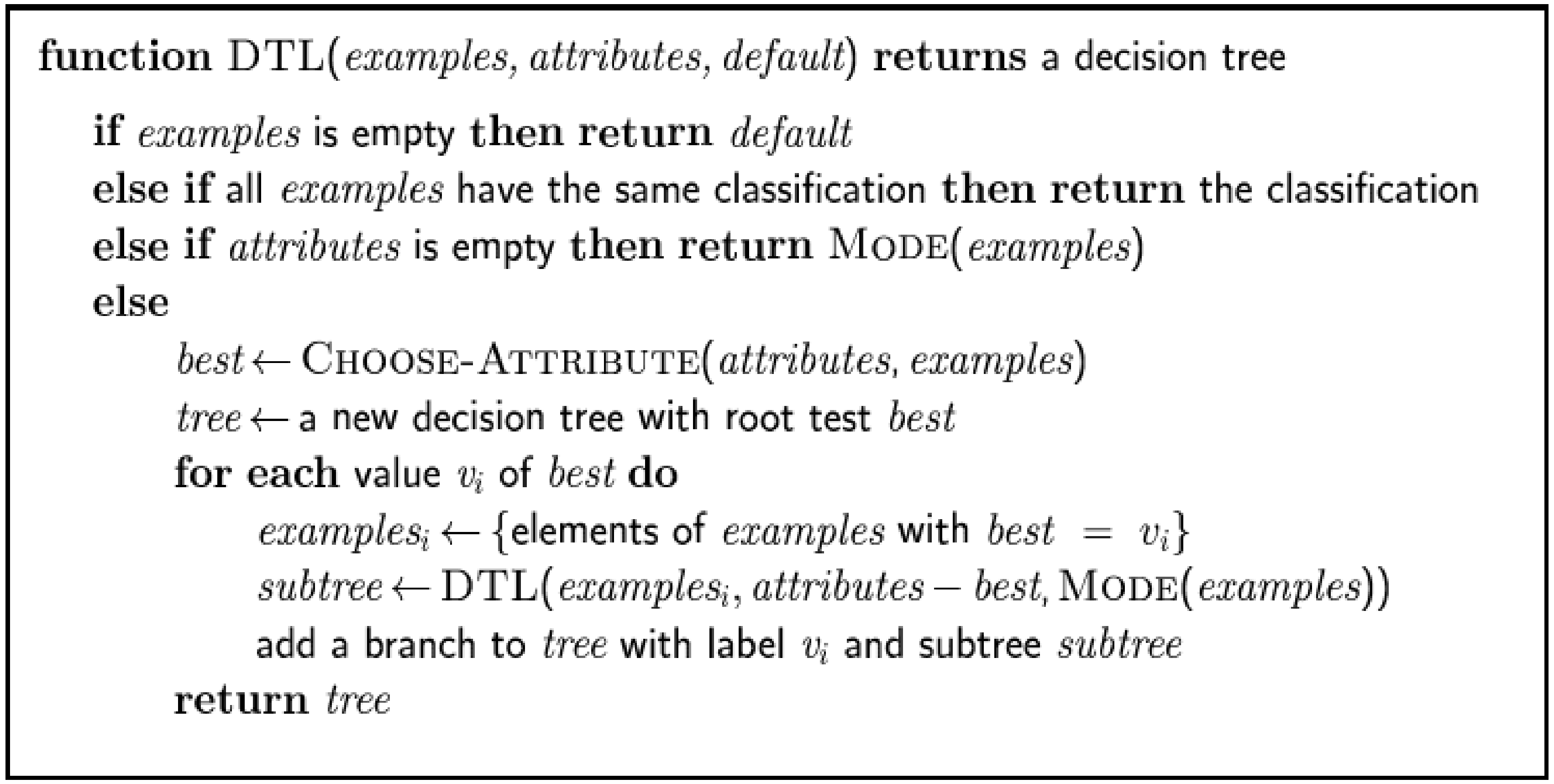
Note: Decision trees represent disjunctions of conjunctions of constraints on attribute values

Decision-Tree Representation

- Decision trees are fully expressive within the class of propositional languages
- Any Boolean function can be written as a decision tree
- No representation is efficient for all functions

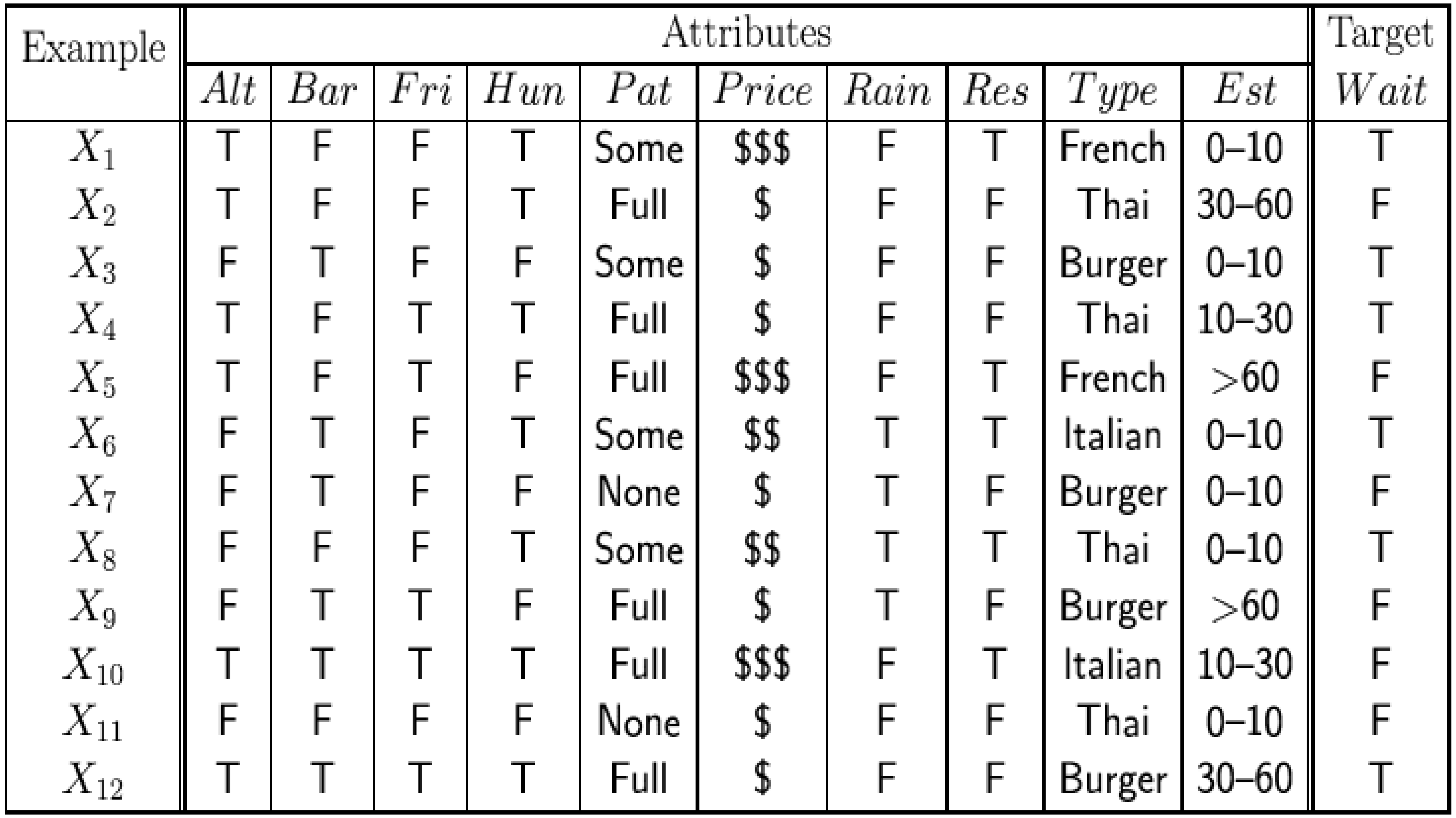
Inducing a Decision Tree

- **Aim**: Find a small tree consistent with the training examples
- **Idea**: (recursively) choose “most significant” attribute as root of (sub)tree



```
function DTL($examples$, $attributes$, $default$) returns a decision tree
    if $examples$ is empty then return $default$
    else if all $examples$ have the same classification then return the classification
    else if $attributes$ is empty then return MODE($examples$)
    else
        $best \leftarrow$ CHOOSE-ATTRIBUTE($attributes$, $examples$)
        $tree \leftarrow$ a new decision tree with root test $best$
        for each value $v_i$ of $best$ do
            $examples_i \leftarrow$ {elements of $examples$ with $best = v_i$}
            $subtree \leftarrow$ DTL($examples_i$, $attributes - best$, MODE($examples$))
            add a branch to $tree$ with label $v_i$ and subtree $subtree$
        return $tree$
```

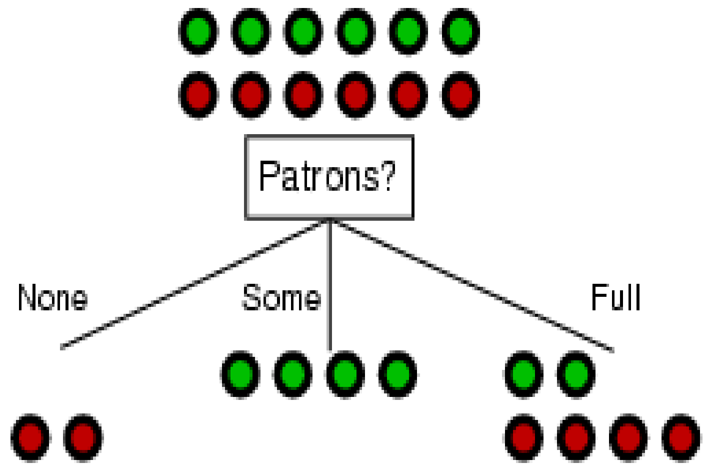
Example: Restaurant

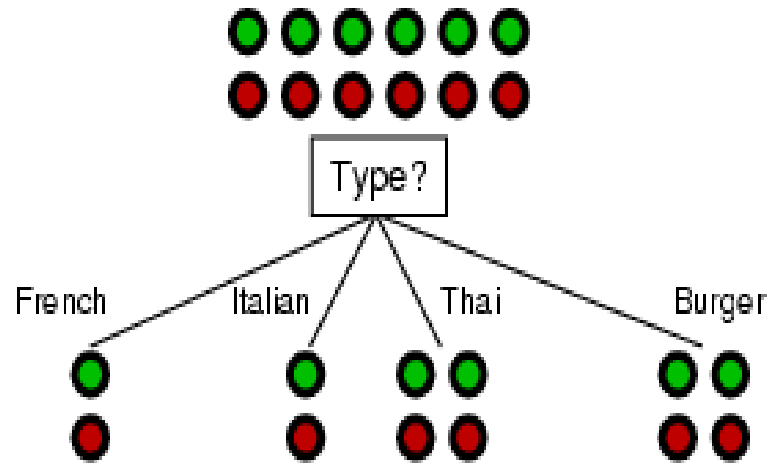


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

Choosing an Attribute

A good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”





Using Information Theory

- Information content (Entropy):

  $I(P(v_1),\ldots,P(v_2))=\sum -P(v_i)\log_2 P(v_i)$

- For a training set containing p positive examples and n negative examples

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right)=-\frac{p}{p+n}\log_2\frac{p}{p+n}-\frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain

- Chosen attribute A divides the training set E into subsets $E_1, \ldots, E_v$ according to their values for A, where A has v distinct values

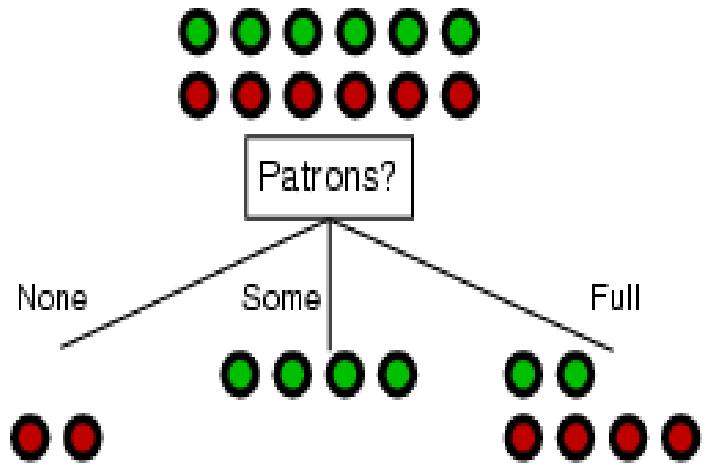
$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

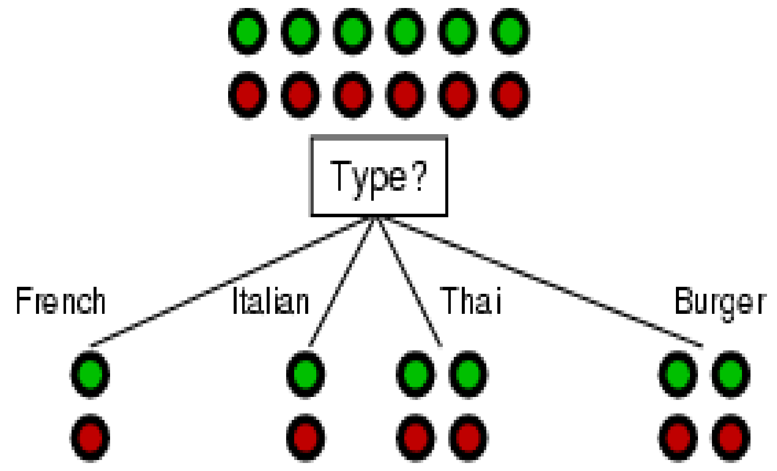
- Information Gain (IG) or reduction in entropy from the attribute test:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

Choosing an Attribute

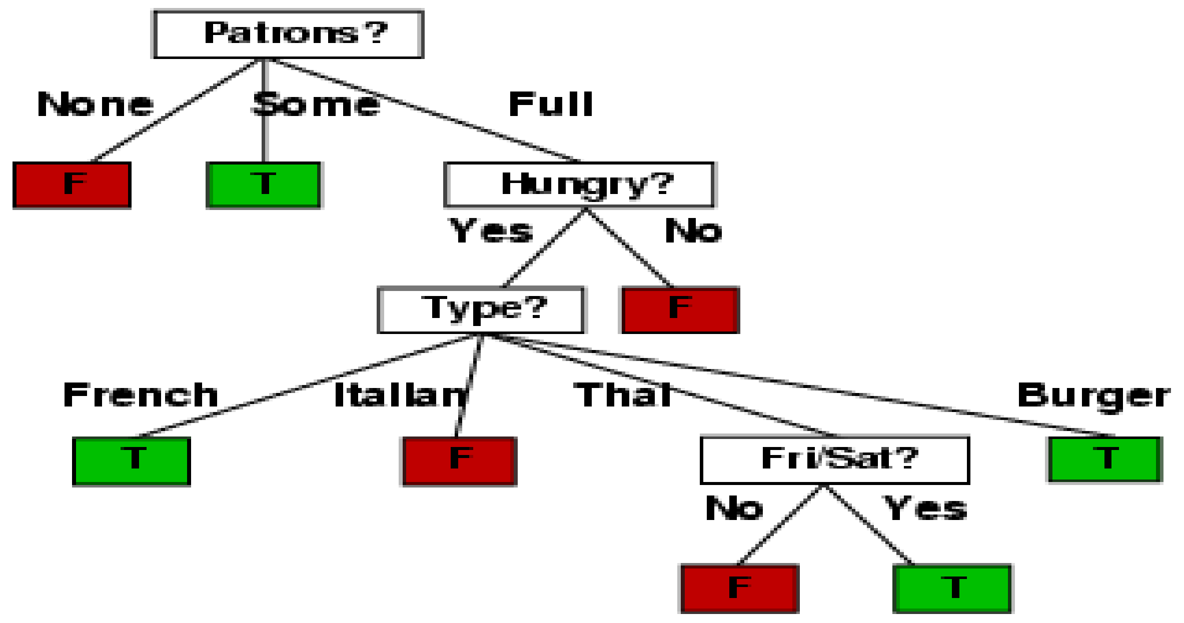
A good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





Decision Tree Example

- Decision tree learned from 12 examples



- Substantially simpler than “true” tree
  - A more complex hypothesis isn’t justified by the small amount of data

Assessing Performance of a Learning Algorithm

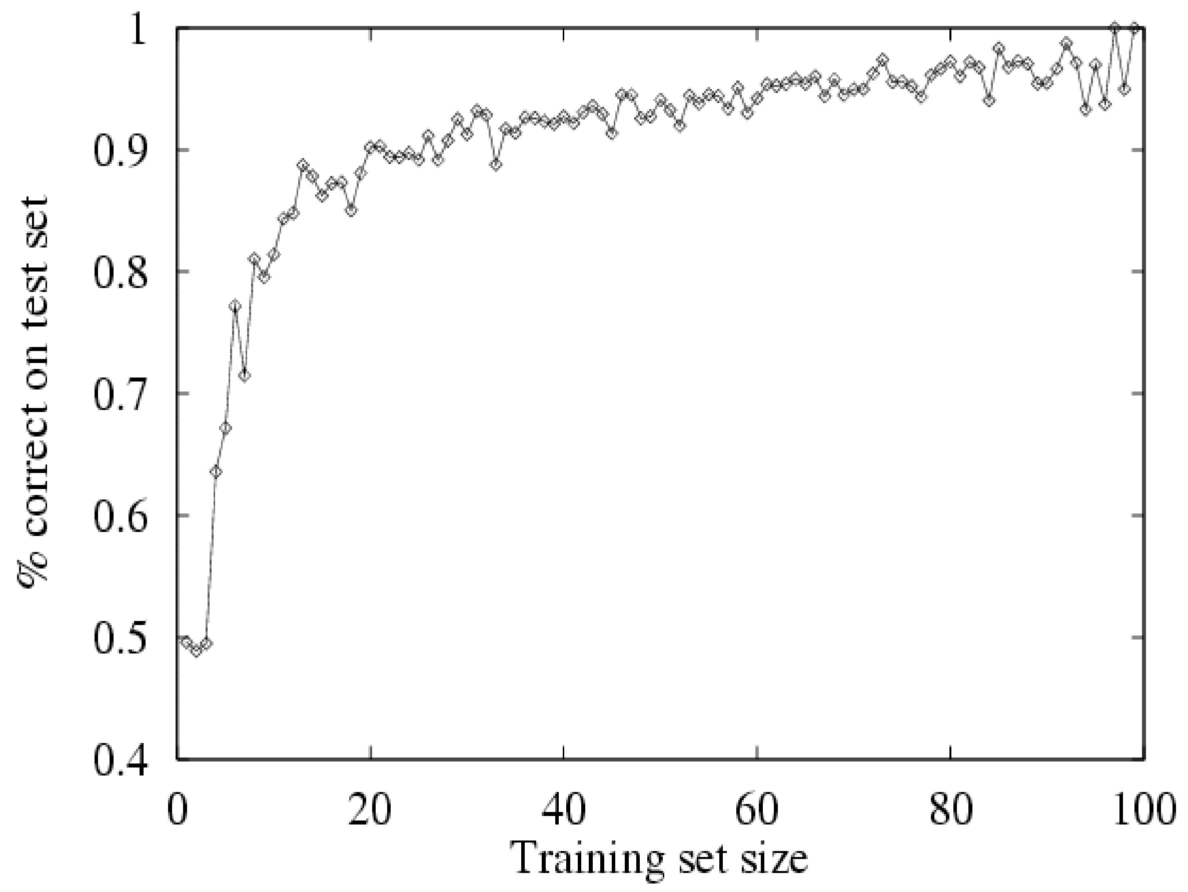
- A learning algorithm is **good** if it produces a hypothesis that does a good job of predicting classifications of unseen examples
- There are theoretical guarantees (learning theory)
- Can also test this

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

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***!

Overfitting

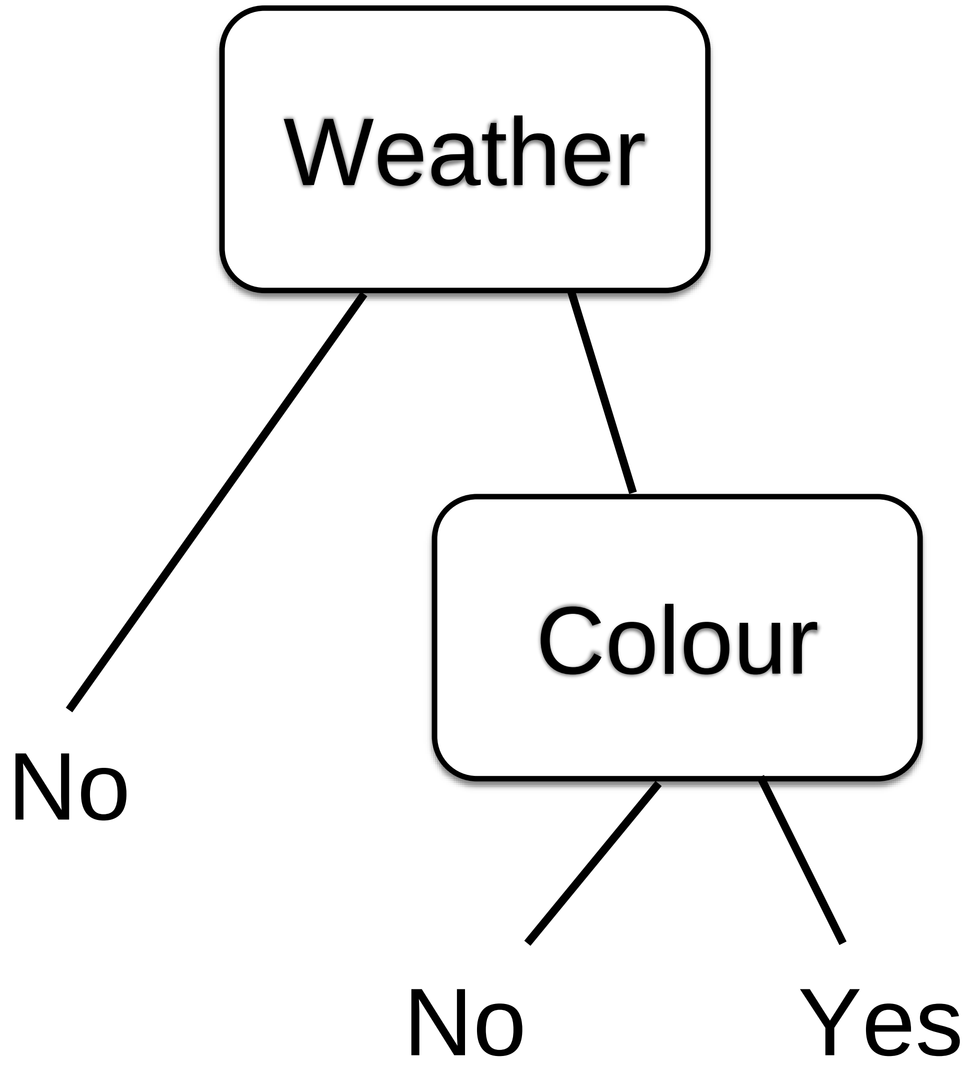
Why might a consistent hypothesis have a high error rate on a test set?

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 $\text{error}_{Tr}(h) < \text{error}_{Tr}(h')$ but $\text{error}_{Te}(h') < \text{error}_{Te}(h)$

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

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

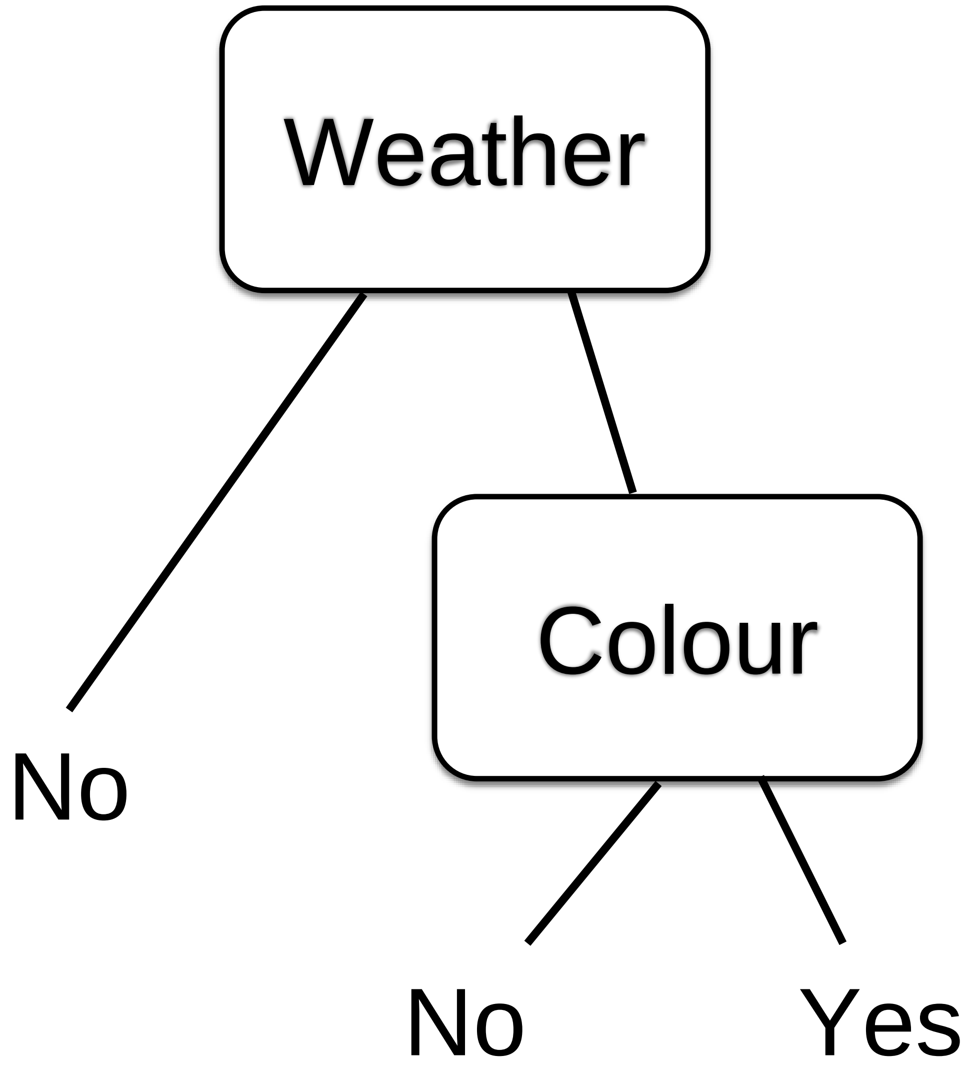
$$D = \sum_{i=1}^{v} \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$

compare to χ^2 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



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*

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

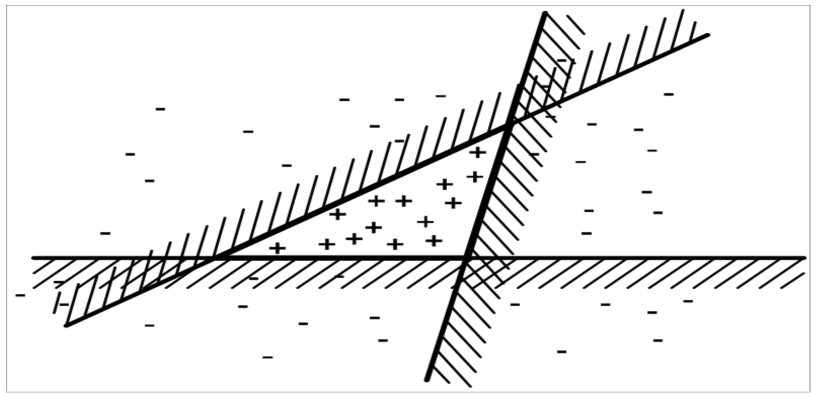
Using ensembles can also enlarge the hypothesis space

- Ensemble as hypothesis
- Set of all ensembles as hypothesis space

Original hypothesis space: Linear threshold hypothesis

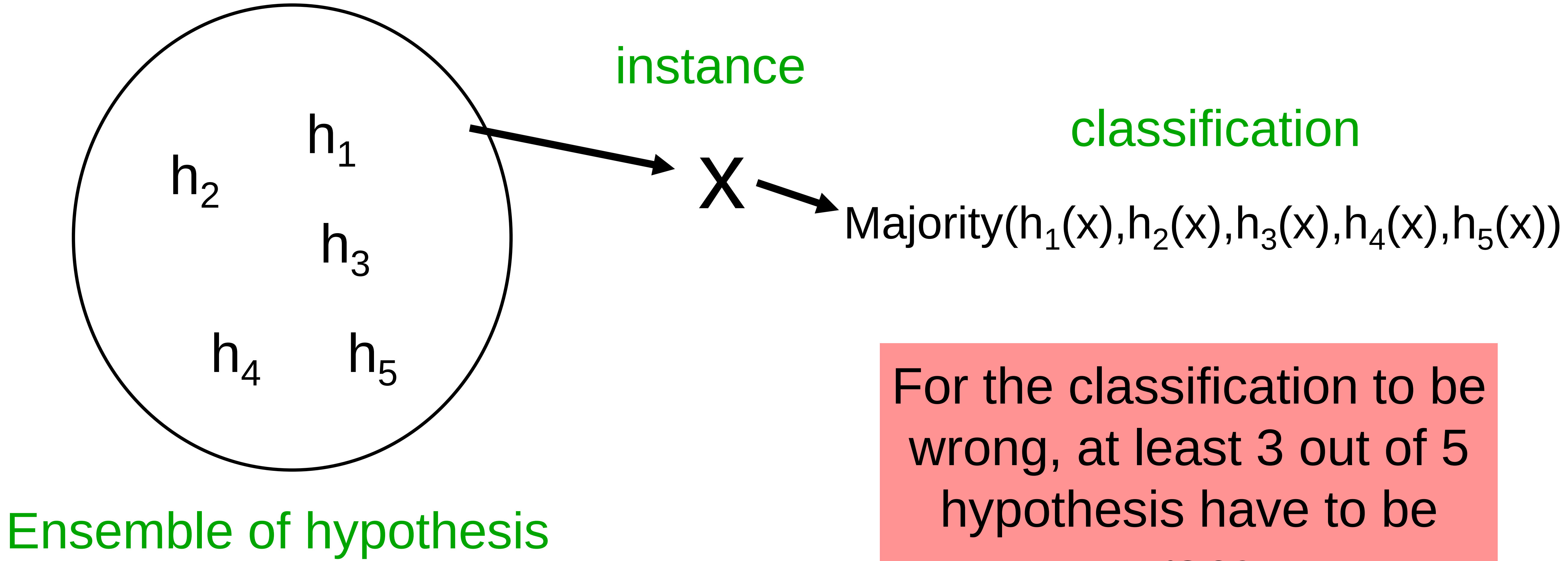
- Simple, efficient learning algorithms but not particularly expressive

Ensemble allows us to learn more expressive hypothesis



Bagging

Majority voting:



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

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

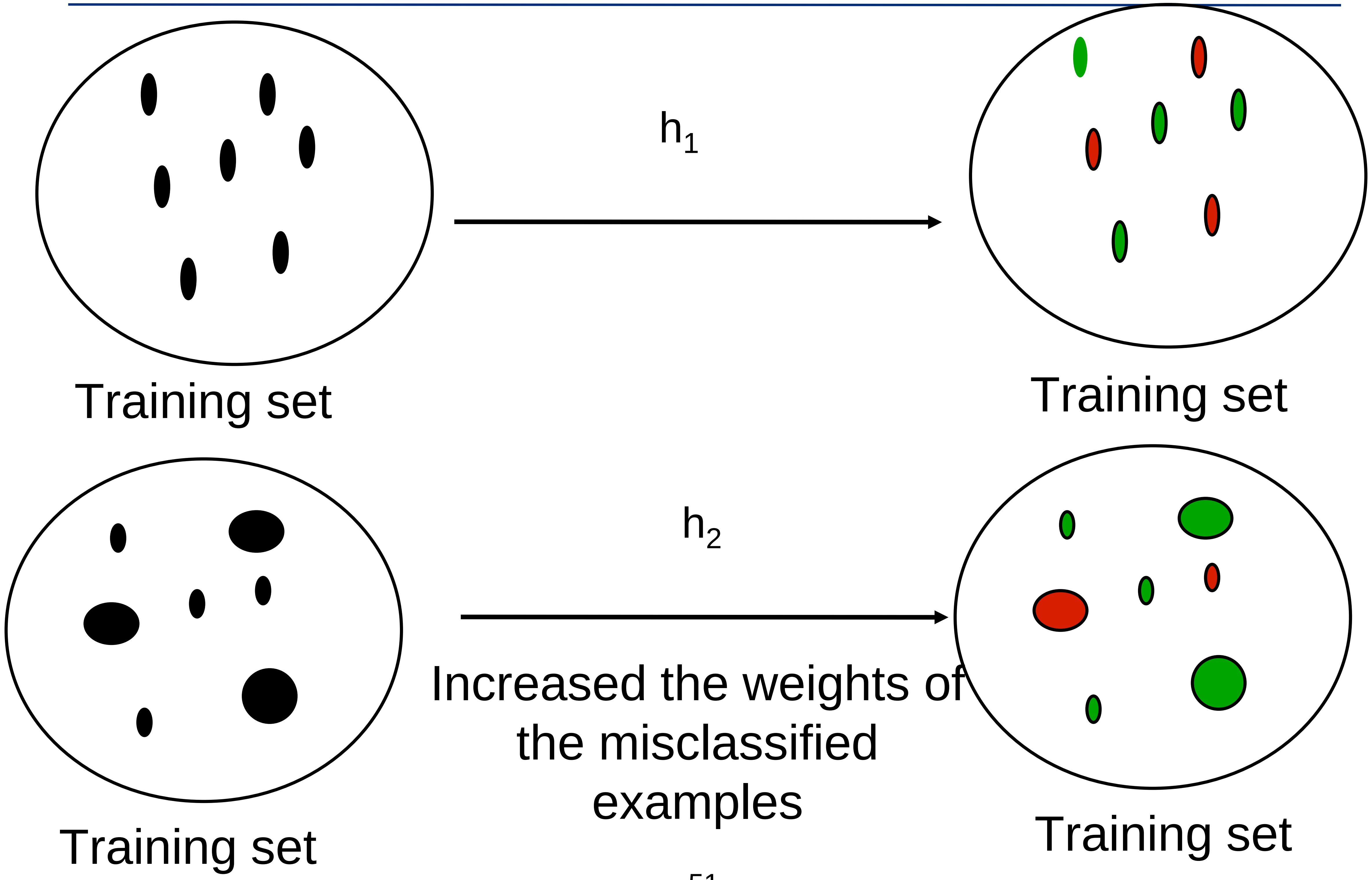
- In practice
  - Hypotheses are rarely independent
  - Some hypotheses have less errors than others
- Weighted majority
  - Intuition:
    - Decrease weights of correlated hypotheses
    - Increase weights of good hypotheses

Boosting

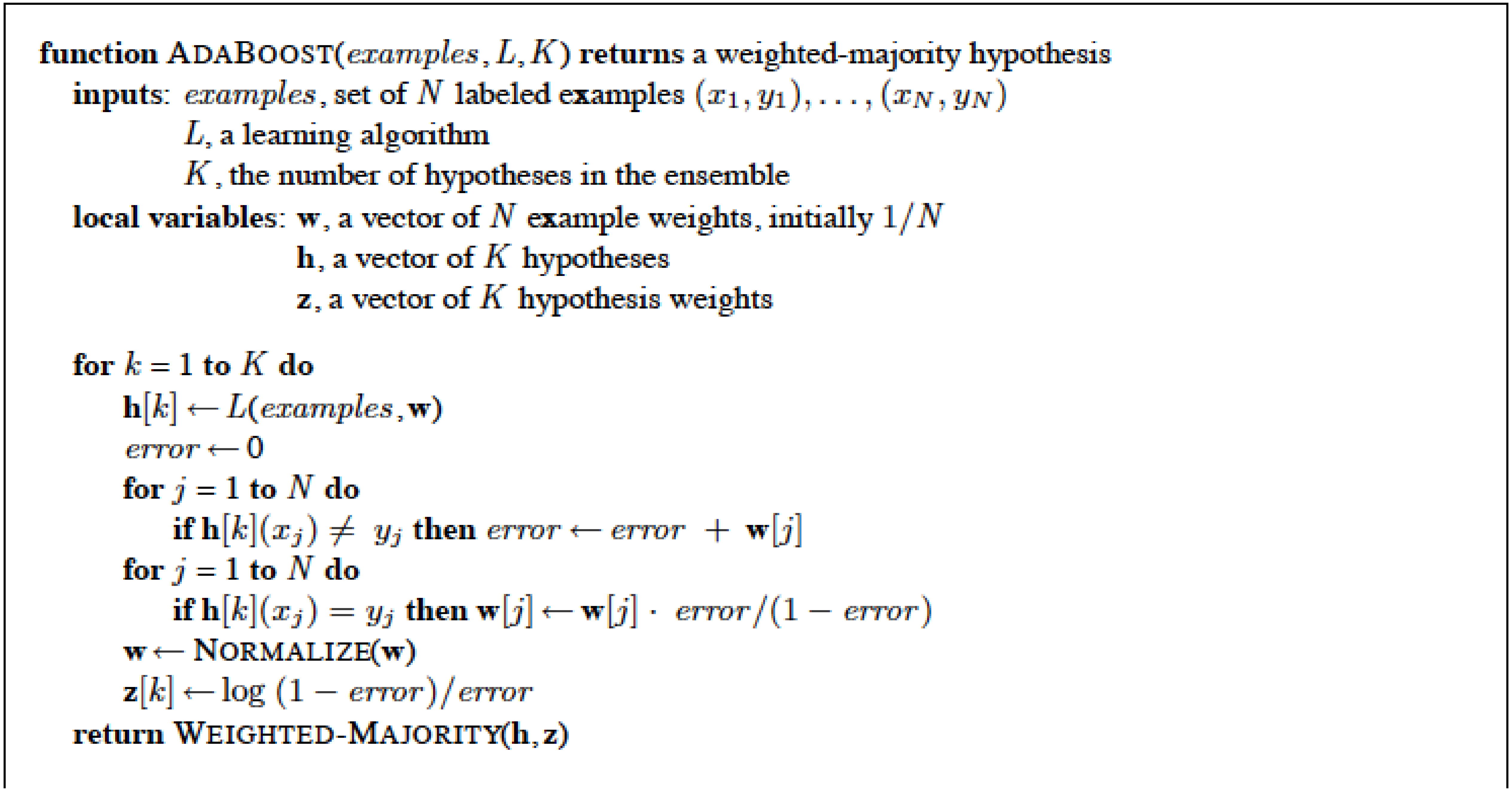
Boosting is the most commonly used form of ensemble learning

- Computes a weighted majority
- Operates on a weighted training set

Boosting


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

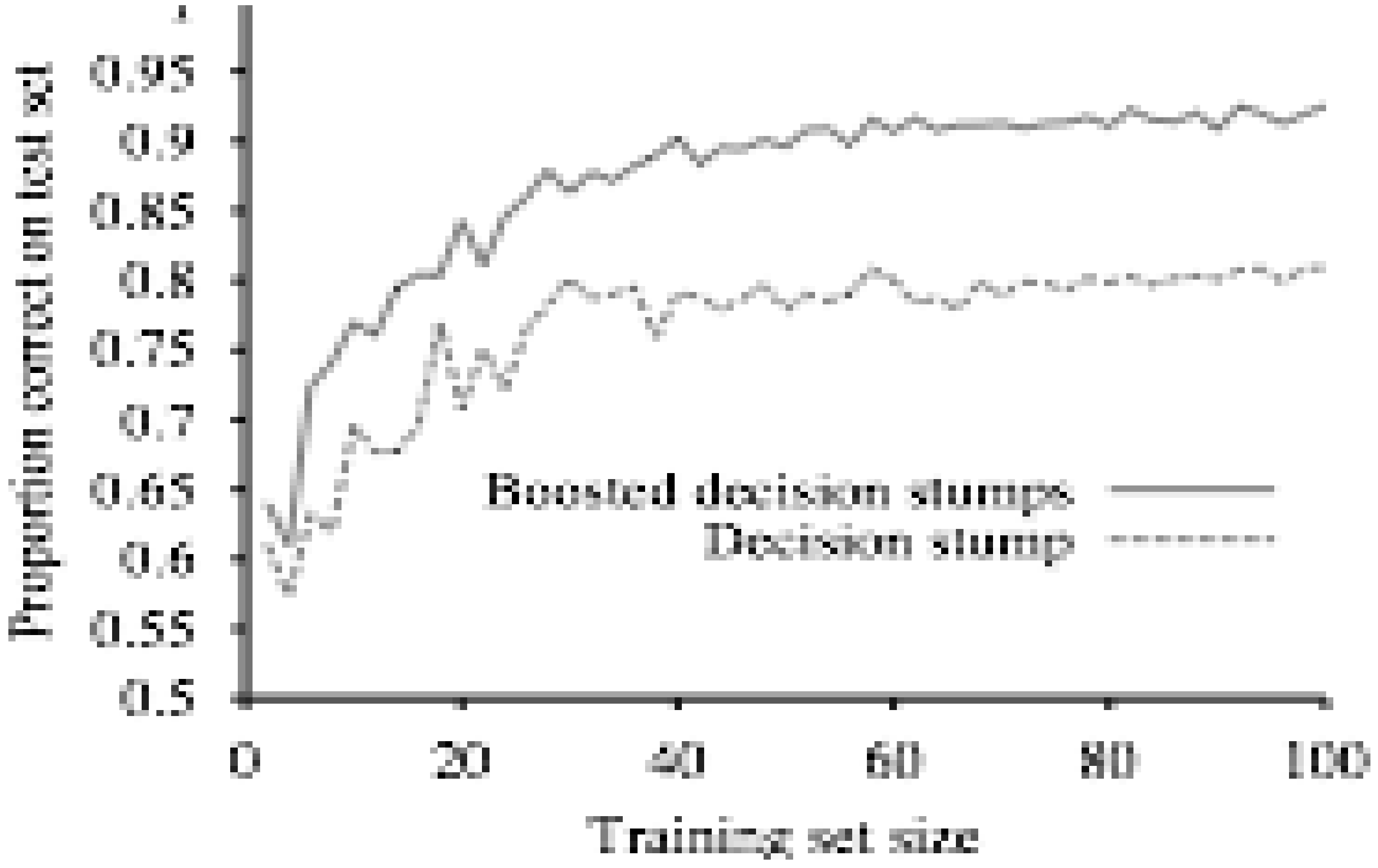
AdaBoost



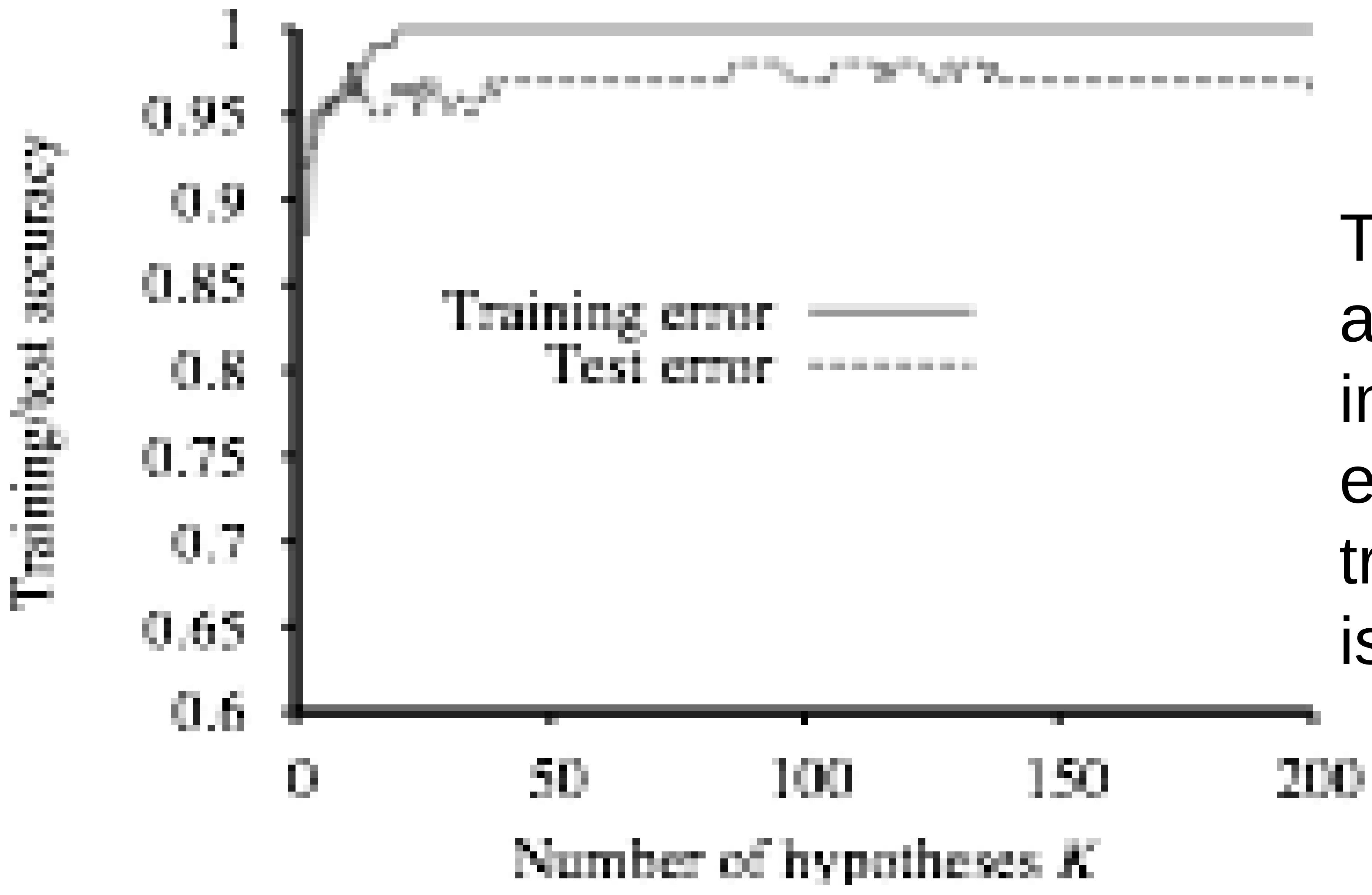
```
function ADABOOST($examples$, $L$, $K$) returns a weighted-majority hypothesis
  inputs: $examples$, set of $N$ labeled examples $(x_1, y_1), \ldots, (x_N, y_N)$
          $L$, a learning algorithm
          $K$, the number of hypotheses in the ensemble
  local variables: $\mathbf{w}$, a vector of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