Machine Learning

CS 486/686: Introduction to AI

Outline

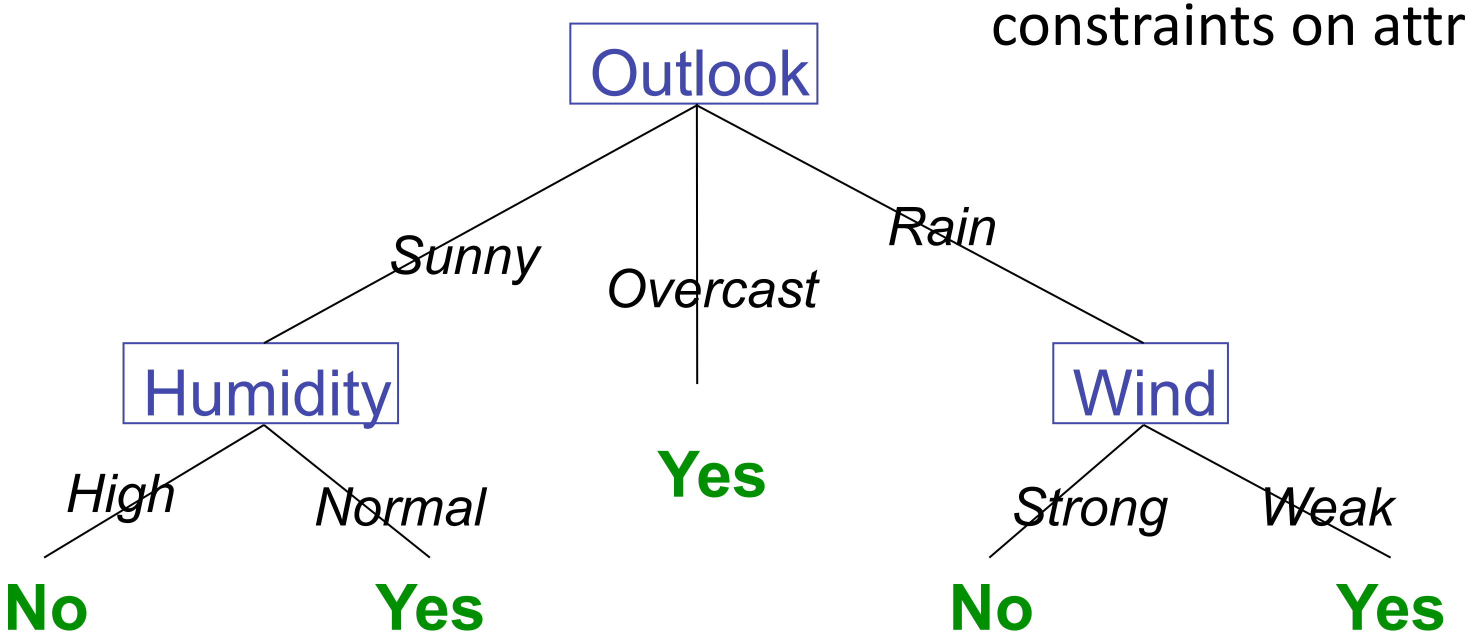
- Decision Trees
- Overfitting
- Cross Validation
- Ensembles

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 Trees

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



An instance

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

Classification: No

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

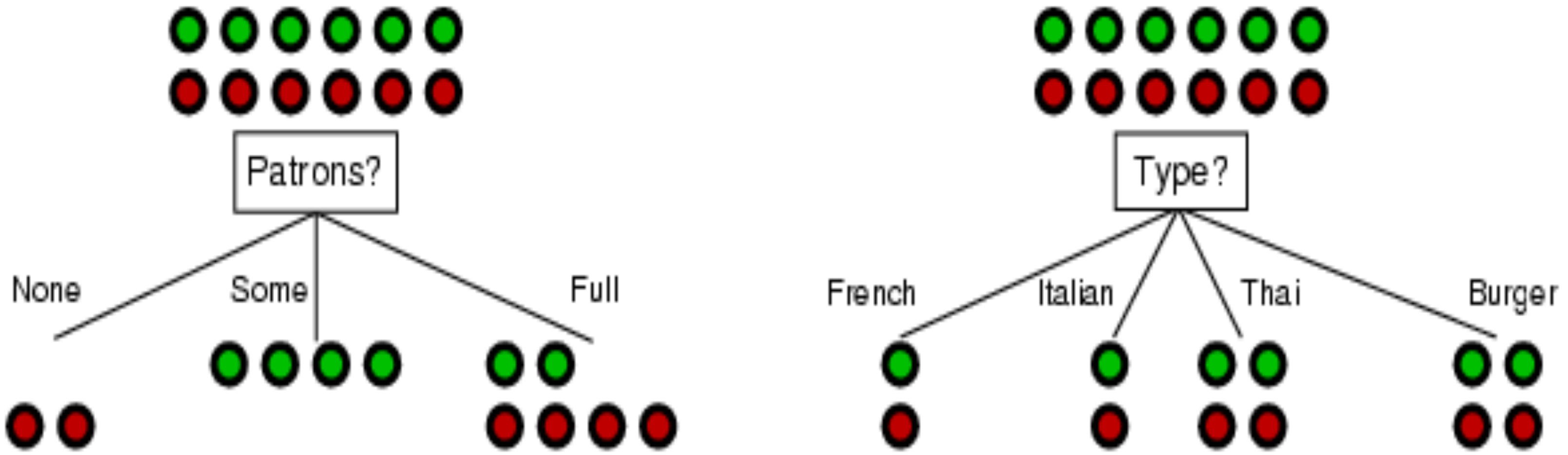
```
function DTL(examples, attributes, default) returns a decision tree

  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 ← CHOOSE-ATTRIBUTE(attributes, examples)
      tree ← a new decision tree with root test best
      for each value v_i of best do
          examples_i ← {elements of examples with best = v_i}
          subtree ← 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

- There are different ways of selecting attributes, but generally a "good attribute" splits the training examples appropriately



Using Information Theory

Information content (Entropy):

$$I(P(v_1), \ldots, P(v_n) = \sum_{i=1}^{n}(-P(V_i)\log_2 P(V_i))$$

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

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

Information Gain

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

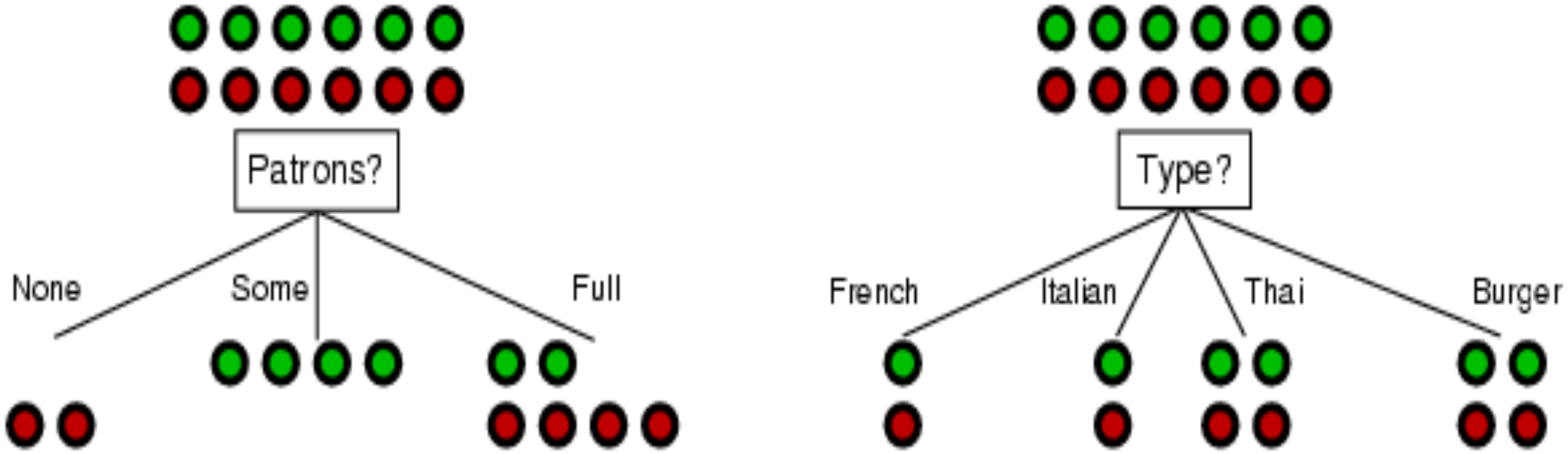
$$remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

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

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

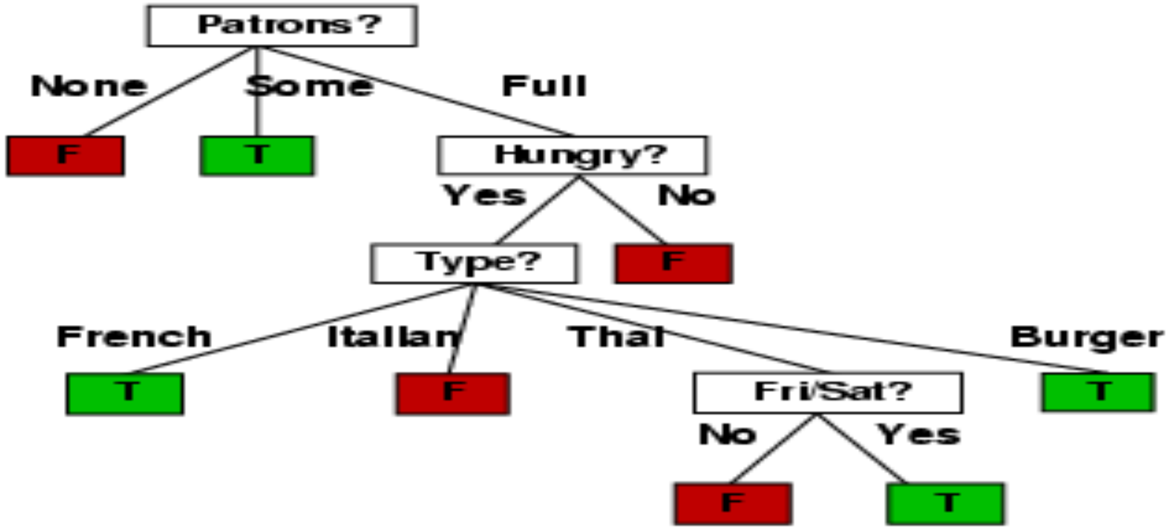
Choosing an Attribute

- There are different ways of selecting attributes, but generally a “good attribute” splits the training examples appropriately



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

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)

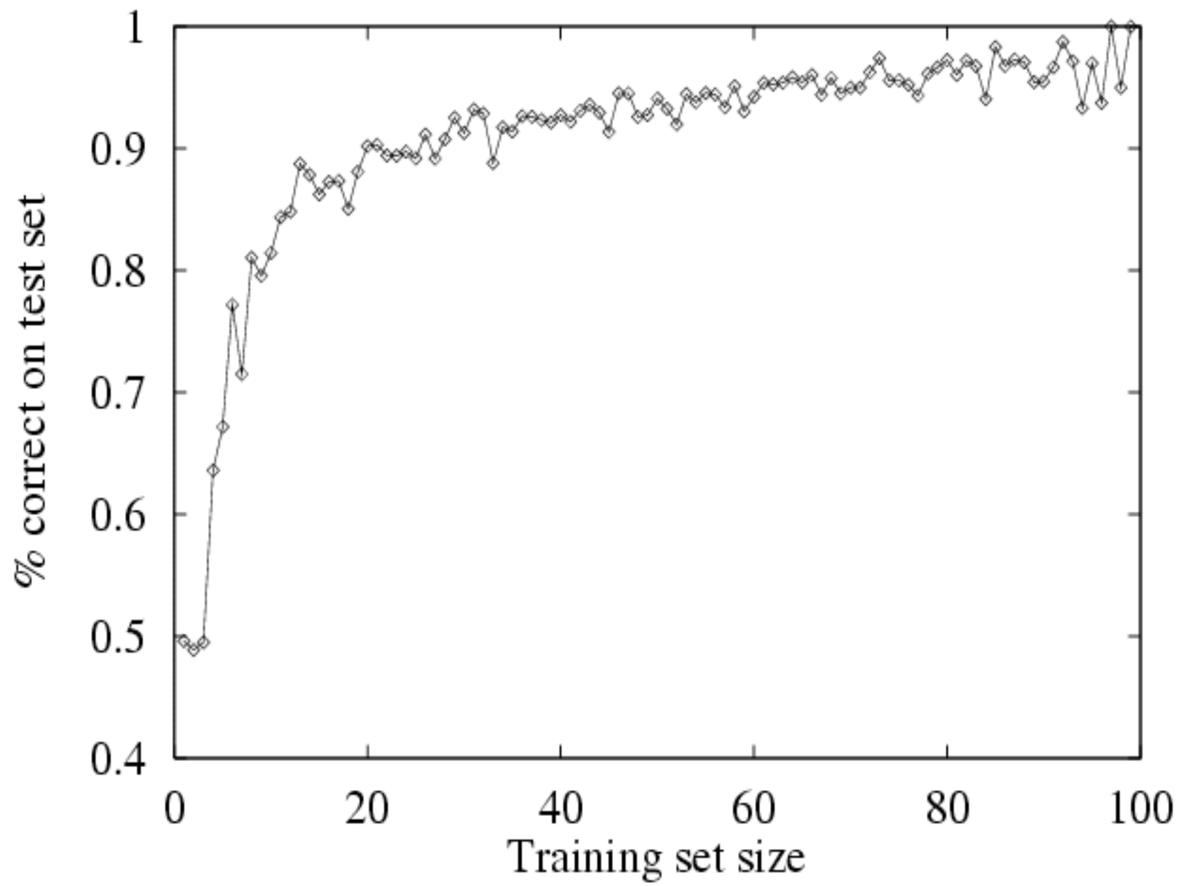
Can also test this

Assessing Performance

Test set

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

Learning Curve



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

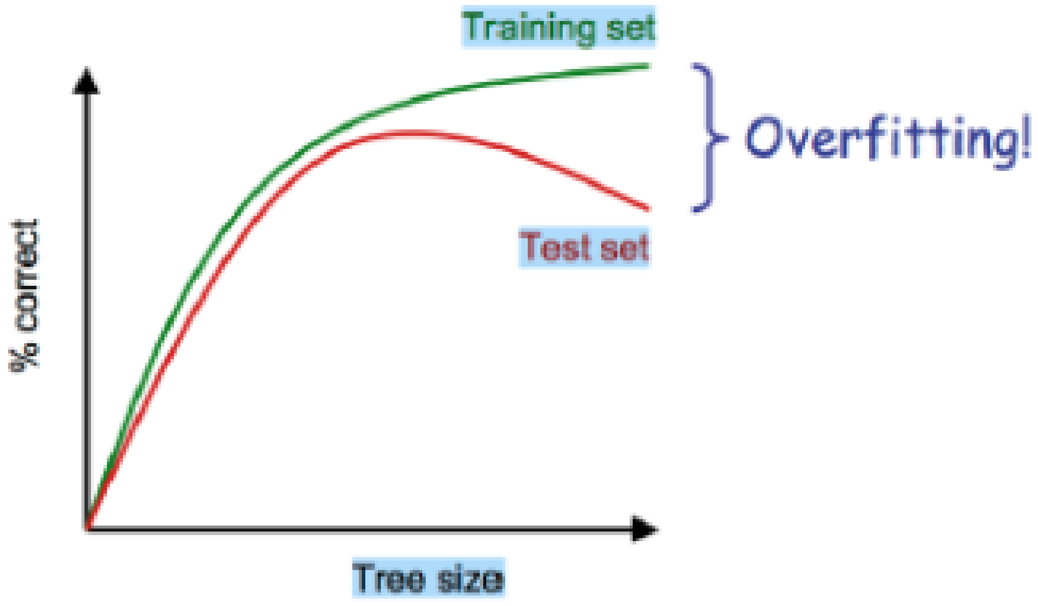
Why might a consistent hypothesis have a high error rate on a test set?

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

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



Overfitting

Test errors caused by

- **Bias**: the error due to the algorithm finding an imperfect model
  - Representation bias: model is too simple
  - Search bias: not enough search
- **Variance**: error due to lack of data
- **Noise**: error due to data depending on features not modeled or because the process generating data was inherently stochastic
- **Bias-Variance Trade-Off**:
  - Complicated model, not enough data (low bias, high variance)
  - Simple model, lots of data (high bias, low variance)

Avoiding Overfitting

- Regularization: Prefer small decision trees over large ones so add a complexity penalty to the stopping criteria (stop early)

- Pseudocounts: Add some data based on prior knowledge

- Cross validation

Cross Validation

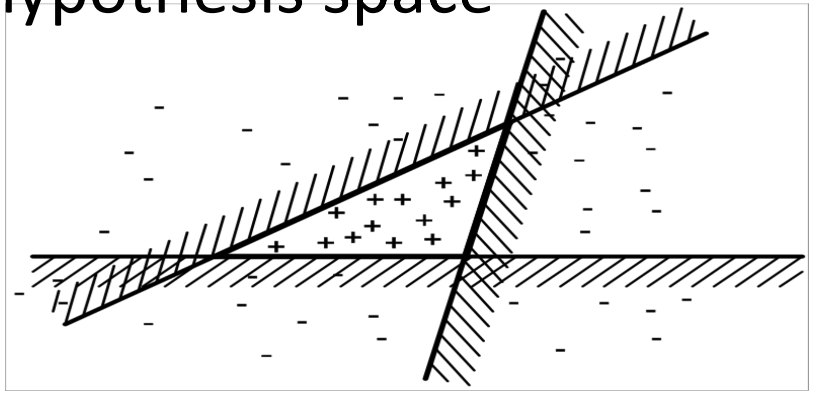
- Split your **training set** into a training and a validation set
- Use the validation set as a "***pretend***" test set
- Optimize the hypothesis/classifier/etc to perform well on the validation set, not the training set
- Can do this multiple times with different validation sets
    - K-fold validation, leave-one-out validation
- When measuring actual performance, report performance on **test set**

Ensembles

- So far we have discussed learning as though it followed a single general approach
  - Choose a single hypothesis from the hypothesis spae
  - Use it to make predictions/classifications
- What happens if we want to use many hypothesis and combine their predictions?

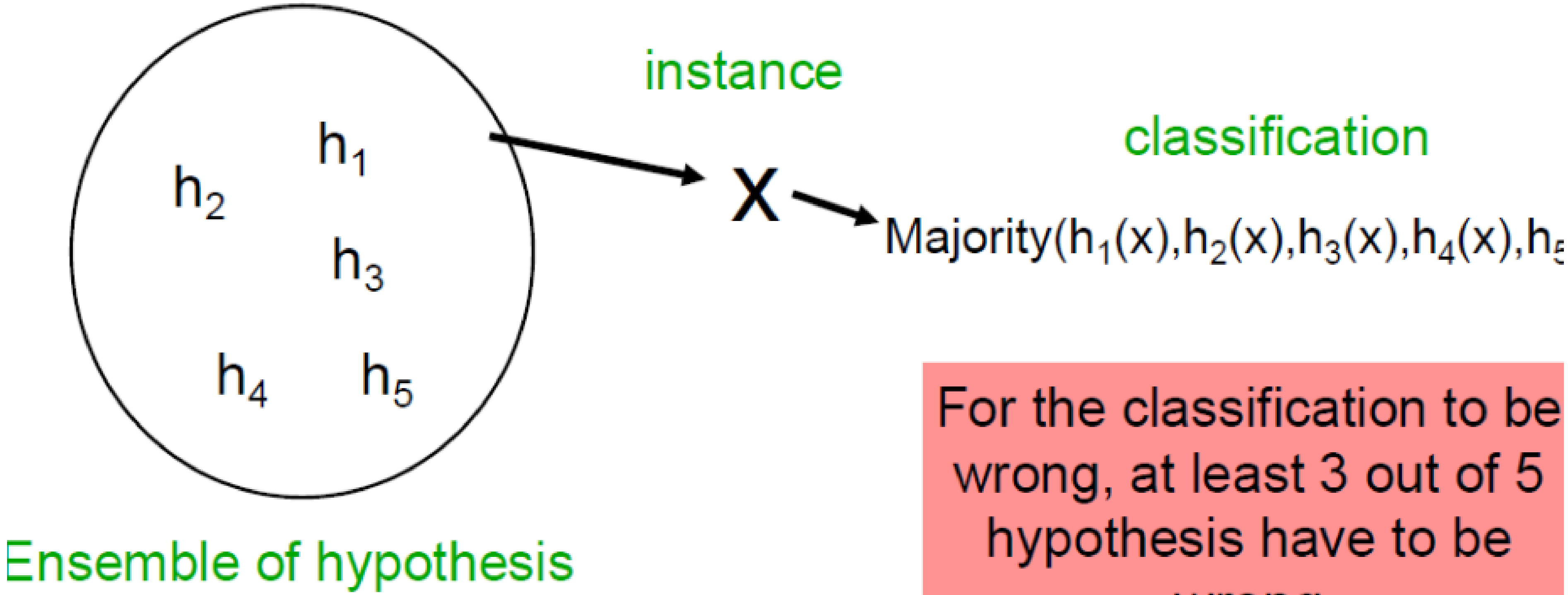
Ensembles

- Analogies
  - Elections, committees
- Intuition
  - Individuals may make mistakes, but the majority may be less likely to make a mistake
  - Individuals have partial information but committees can pool their expertise
- Using ensembles can also enlarge the hypothesis space



Ensembles: Bagging

Majority voting:



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

Ensembles: Bagging

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

- Majority voting of n hypothesis
  - Probability k made an error?
  - Probability that the majority made an error?

Bagging Example: Random Forests

- Do K times
  - Randomly sample (with replacement) subsets of your training data
  - Randomly sample subsets of features (feature bagging)
  - Learn a decision tree using the subset of features

Classify using a majority vote of the k trees in your forest.

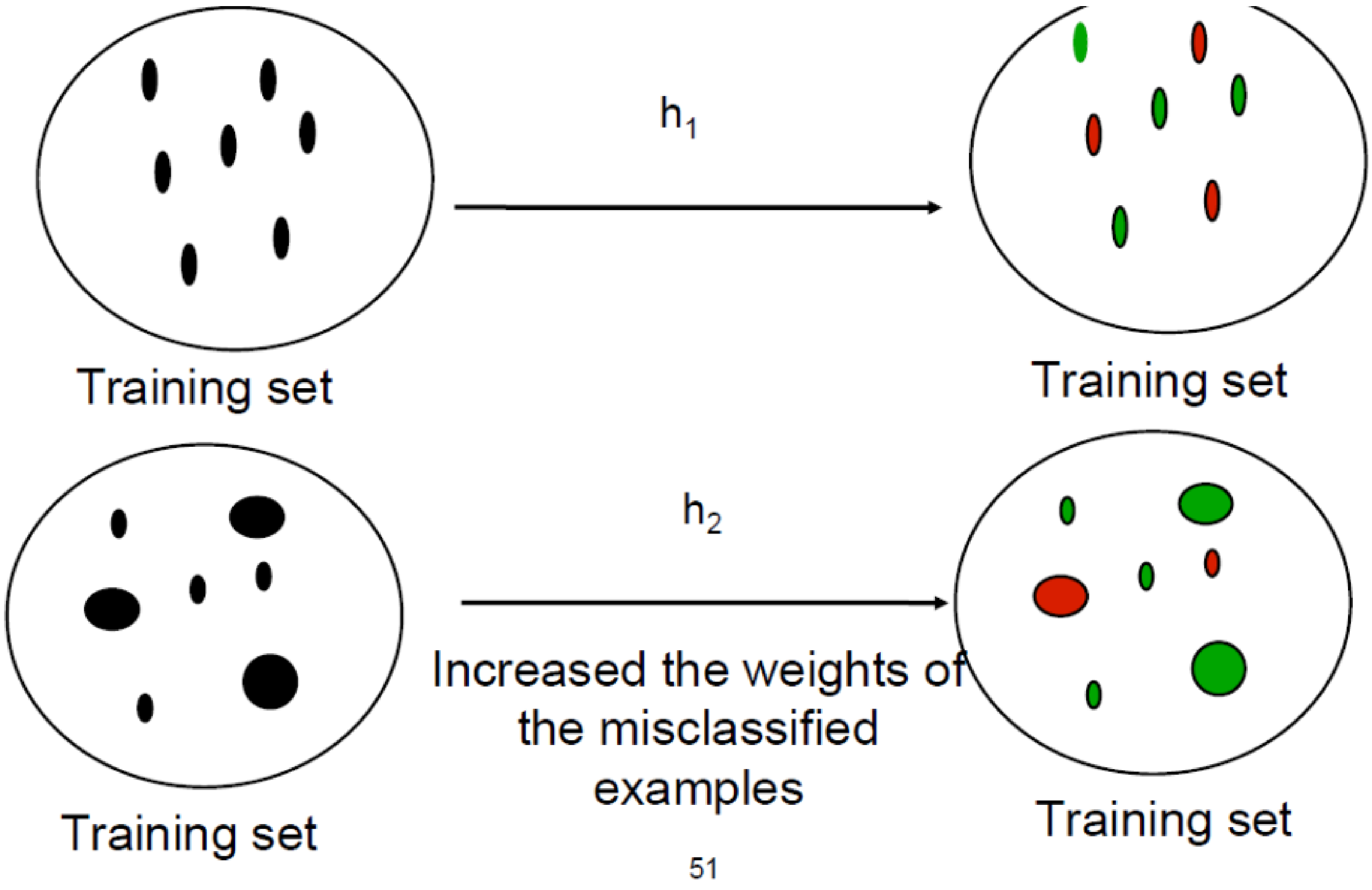
Ensembles: Boosting

- In practice hypothesis are rarely independent and some are more error-prone than others

- Weight majority:
  - Decrease weights of correlated hypothesis
  - Increase weights of good hypothesis

- Boosting

Ensembles: Boosting

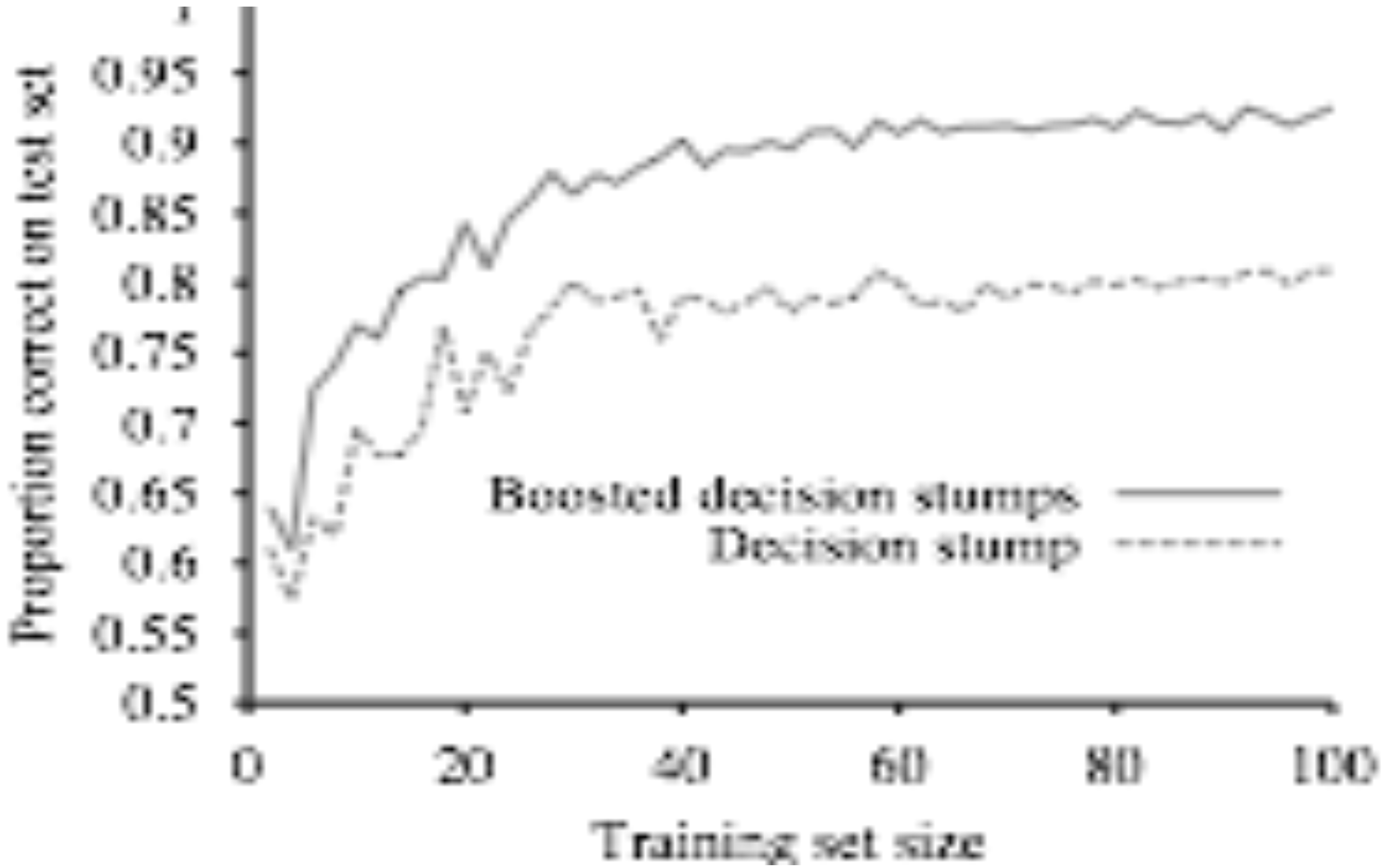


Ensembles: AdaBoost

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($\mathbf{h}, \mathbf{z}$)

Ensembles: Boosting



K=5

Ensembles: Boosting

- Many variations of boosting (AdaBoost is a specific boosting algorithm)
  - Takes a weak learning L (classifies slightly better than just random guessing) and returns a hypothesis that classifies training data with 100% accuracy

Robert Schapire and Yoav Freund
Kanellakis Award for 2004

What You Should Know

- How to learn a decision tree
- Overfitting and its causes
- Cross validation
- Ensembles (bagging and boosting)