

# CMSC 671

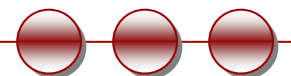
## Fall 2010

Tue 11/16/10

# Machine Learning: Decision Trees

## Chapter 18.1-18.3; 18.8, 18.9

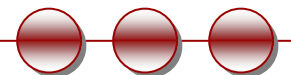
Prof. Laura Zavala, [laura.zavala@umbc.edu](mailto:laura.zavala@umbc.edu), ITE 373, 410-455-8775



# What is learning?

---

- “Learning denotes changes in a system that ... enable a system to do the same task more efficiently the next time.” –Herbert Simon
- “Learning is constructing or modifying representations of what is being experienced.”  
–Ryszard Michalski
- “Learning is making useful changes in our minds.”  
–Marvin Minsky



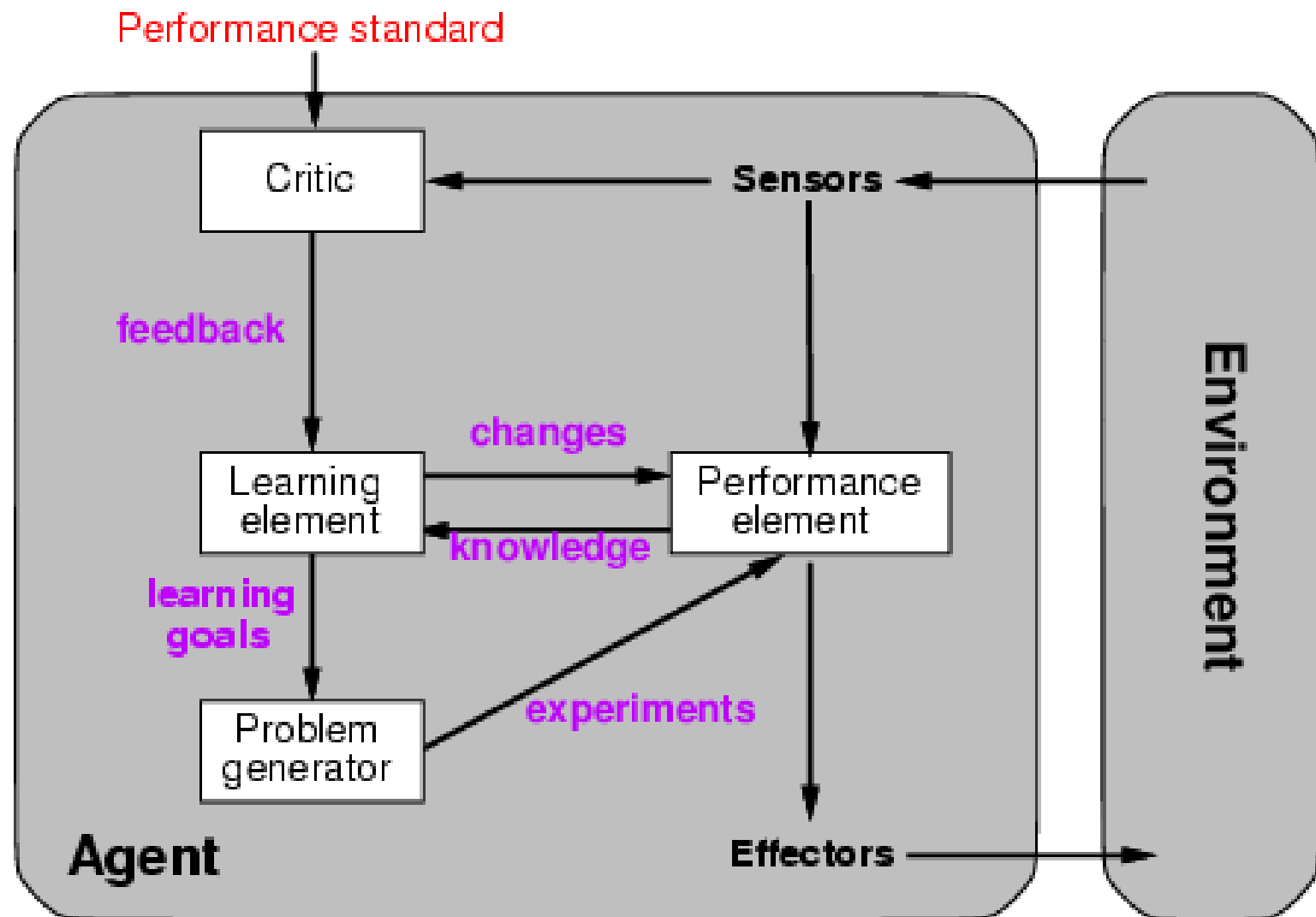
# Why study learning?

---

- Understand and improve efficiency of human learning
  - Use to improve methods for teaching and tutoring people (e.g., better computer-aided instruction)
- Discover new things or structure previously unknown
  - Examples: data mining, scientific discovery
- Fill in skeletal or incomplete specifications about a domain
  - Large, complex AI systems can't be completely built by hand and require dynamic updating to incorporate new information
  - Learning new characteristics expands the domain or expertise and lessens the “brittleness” of the system
- Build agents that can adapt to users, other agents, and their environment



# A general model of learning agents



# Different ways of learning

## ■ Supervised versus unsupervised learning

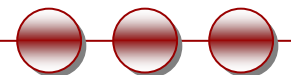
- Learn an unknown function  $f(X) = Y$ , where  $X$  is an input example and  $Y$  is the desired output.
- **Supervised learning** implies we are given a **training set** of  $(X, Y)$  pairs by a “teacher”
- **Unsupervised learning** means we are only given the  $X$ s and no explicit feedback function.  
Clustering: Detecting potential useful clusters of input examples.

## ■ Reinforcement learning

- Feedback (positive or negative reward) given at the end of a sequence of steps

## ■ Semi-supervised learning

- A continuum between supervised and unsupervised learning.
- Few labeled examples (training set)



# Supervised concept learning

- Given a training set of positive and negative examples of a concept
- Construct a description that will accurately classify whether future examples are positive or negative
- That is, learn some good estimate of function  $f$  given a training set  $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$  where each  $y_i$  is either + (positive) or - (negative), or a probability distribution over +/-

# Inductive learning framework

- Raw input data from sensors are typically preprocessed to obtain a **feature vector**,  $X$ , that adequately describes all of the relevant features for classifying examples
- Each  $x$  is a list of (attribute, value) pairs. For example,  
 $X = [\text{Person:Sue, EyeColor:Brown, Age:Young, Sex:Female}]$
- The number of attributes (a.k.a. features) is fixed (positive, finite)
- Each attribute has a fixed, finite number of possible values (or could be continuous)
- Each example is interpreted as a point in an  $n$ -dimensional **feature space**, where  $n$  is the number of attributes

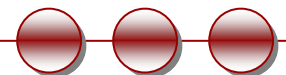
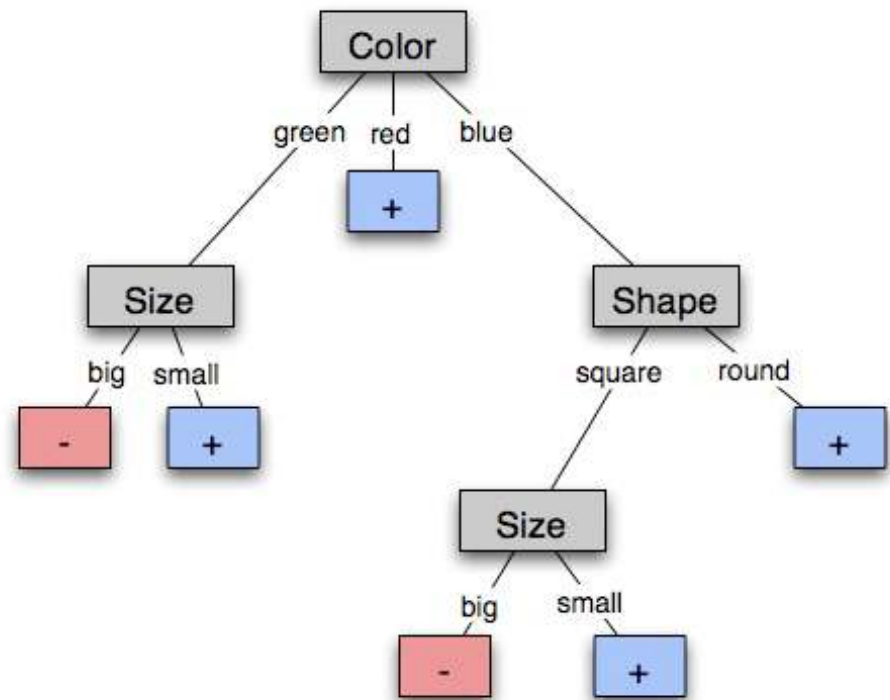
# Inductive learning as search

- Instance space  $I$  defines the language for the training and test instances
  - Typically, but not always, each instance  $i \in I$  is a feature vector
  - Features are sometimes called attributes or variables
  - $I: V_1 \times V_2 \times \dots \times V_k, i = (v_1, v_2, \dots, v_k)$
- Class variable  $C$  gives an instance's class (to be predicted)
- Model space  $M$  defines the possible classifiers
  - $M: I \rightarrow C, M = \{m_1, \dots, m_n\}$  (possibly infinite)
  - Model space is sometimes, but not always, defined in terms of the same features as the instance space
- Training data can be used to direct the search for a good (consistent, complete, simple) hypothesis in the model space

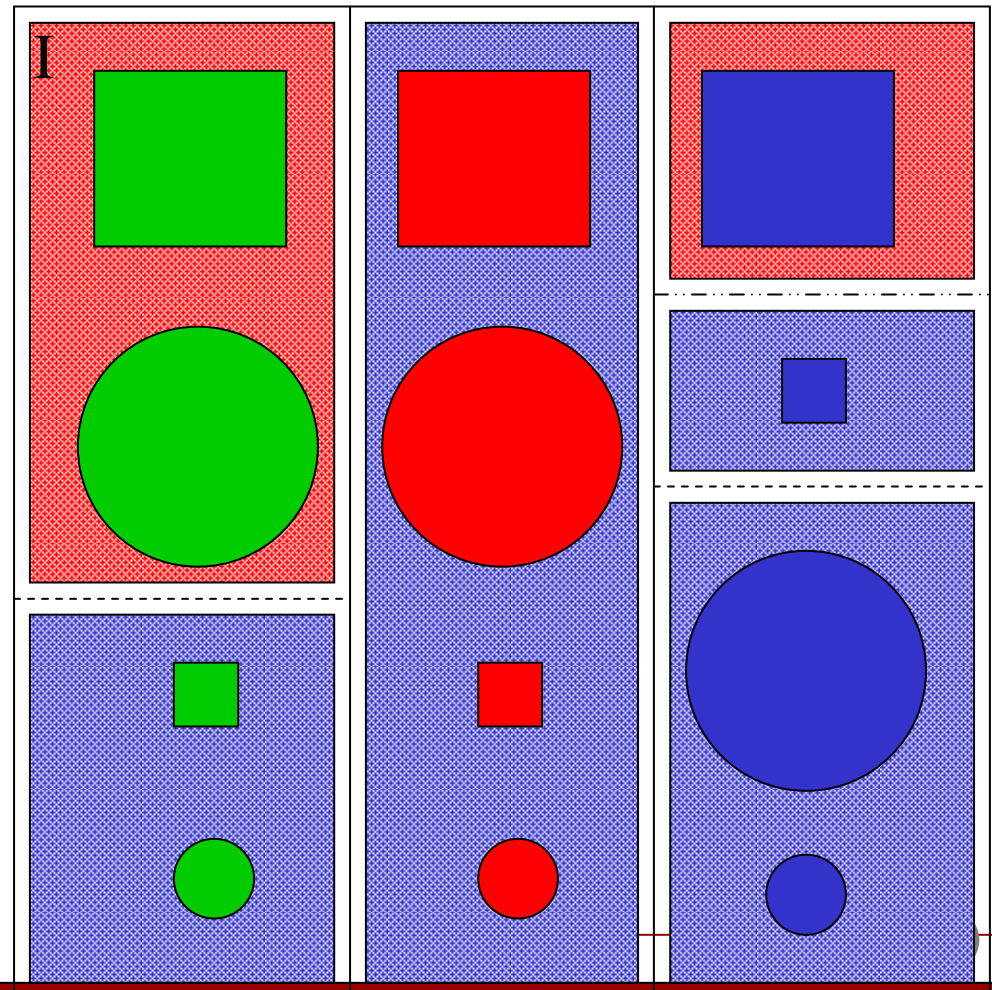
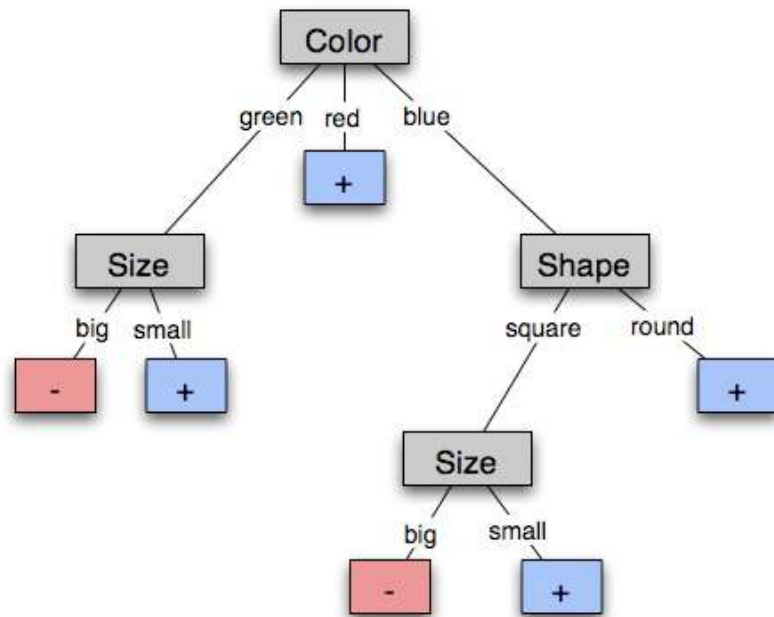


# Learning decision trees

- Goal: Build a **decision tree** to classify examples as positive or negative instances of a concept using supervised learning from a training set
- A **decision tree** is a tree where
  - each non-leaf node has associated with it an attribute (feature)
  - each leaf node has associated with it a classification (+ or -)
  - each arc has associated with it one of the possible values of the attribute at the node from which the arc is directed
- Generalization: allow for  $>2$  classes
  - e.g., for stocks, classify into {sell, hold, buy}

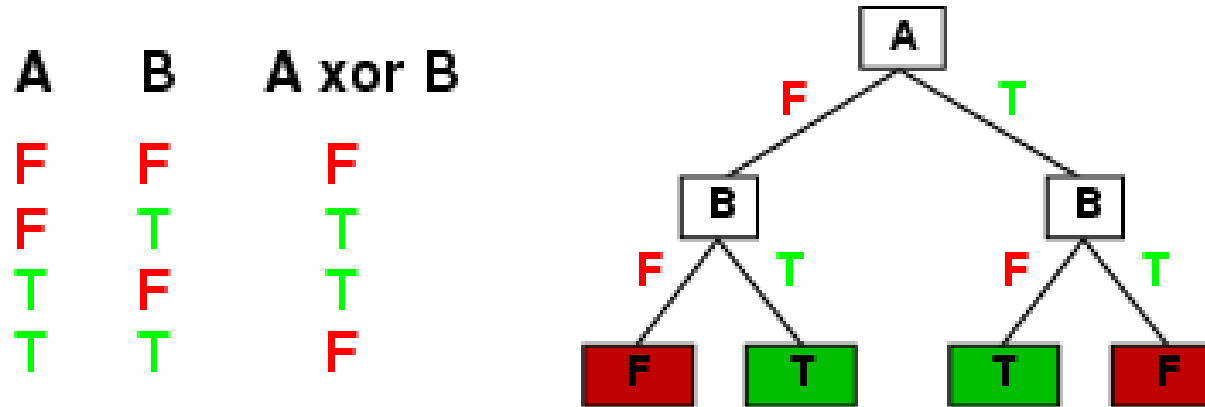


# Decision tree-induced partition – example



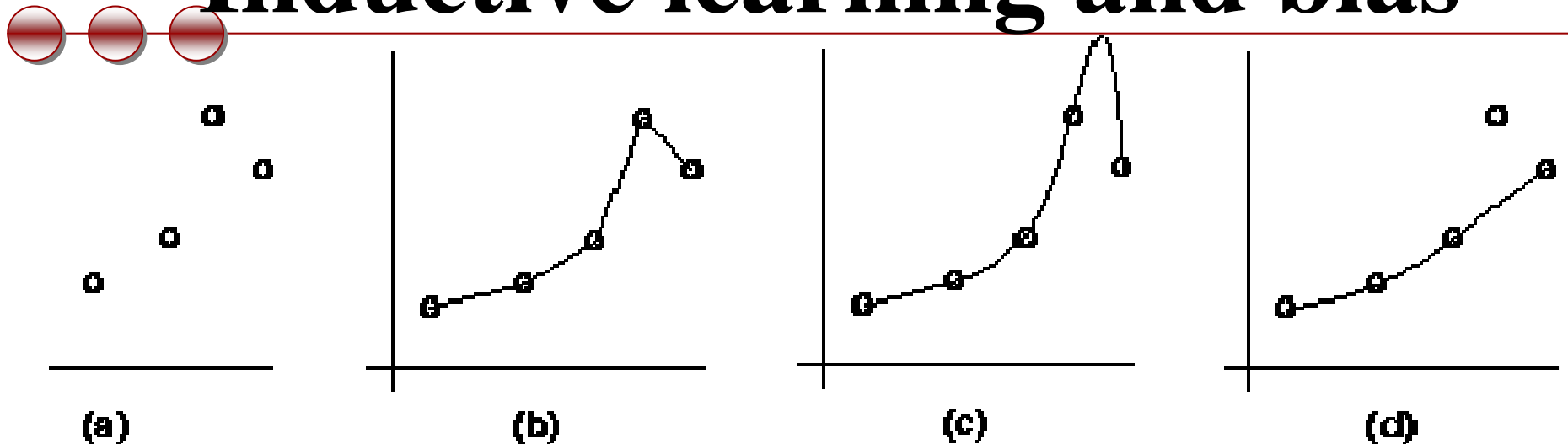
# Expressiveness

- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row  $\rightarrow$  path to leaf:



- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless  $f$  nondeterministic in  $x$ ) but it probably won't generalize to new examples
- We prefer to find more **compact** decision trees

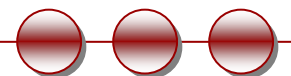
# Inductive learning and bias



- Suppose that we want to learn a function  $f(x) = y$  and we are given some sample  $(x,y)$  pairs, as in figure (a)
- There are several hypotheses we could make about this function, e.g.: (b), (c) and (d)
- A preference for one over the others reveals the **bias** of our learning technique, e.g.:
  - prefer piece-wise functions (b)
  - prefer a smooth function (c)
  - prefer a simple function and treat outliers as noise (d)

# Preference bias: Ockham's Razor

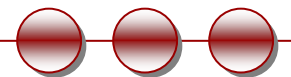
- A.k.a. Occam's Razor, Law of Economy, or Law of Parsimony
- Principle stated by William of Ockham (1285-1347/49), a scholastic, that
  - “*non sunt multiplicanda entia praeter necessitatem*”
  - or, entities are not to be multiplied beyond necessity
- **The simplest consistent explanation is the best**
- Therefore, the **smallest decision tree** that correctly classifies all of the training examples is best.



# R&N's restaurant domain

---

- Develop a decision tree to model decision a patron makes when deciding whether or not to wait for a table at a restaurant
- Two classes: wait, leave (Yes, No) (T, F)
- Ten attributes: Alternative available? Bar in restaurant? Is it Friday? Are we hungry? How full is the restaurant? How expensive? Is it raining? Do we have a reservation? What type of restaurant is it? What's the purported waiting time?
- Training set of 12 examples
- ~ 7000 possible cases

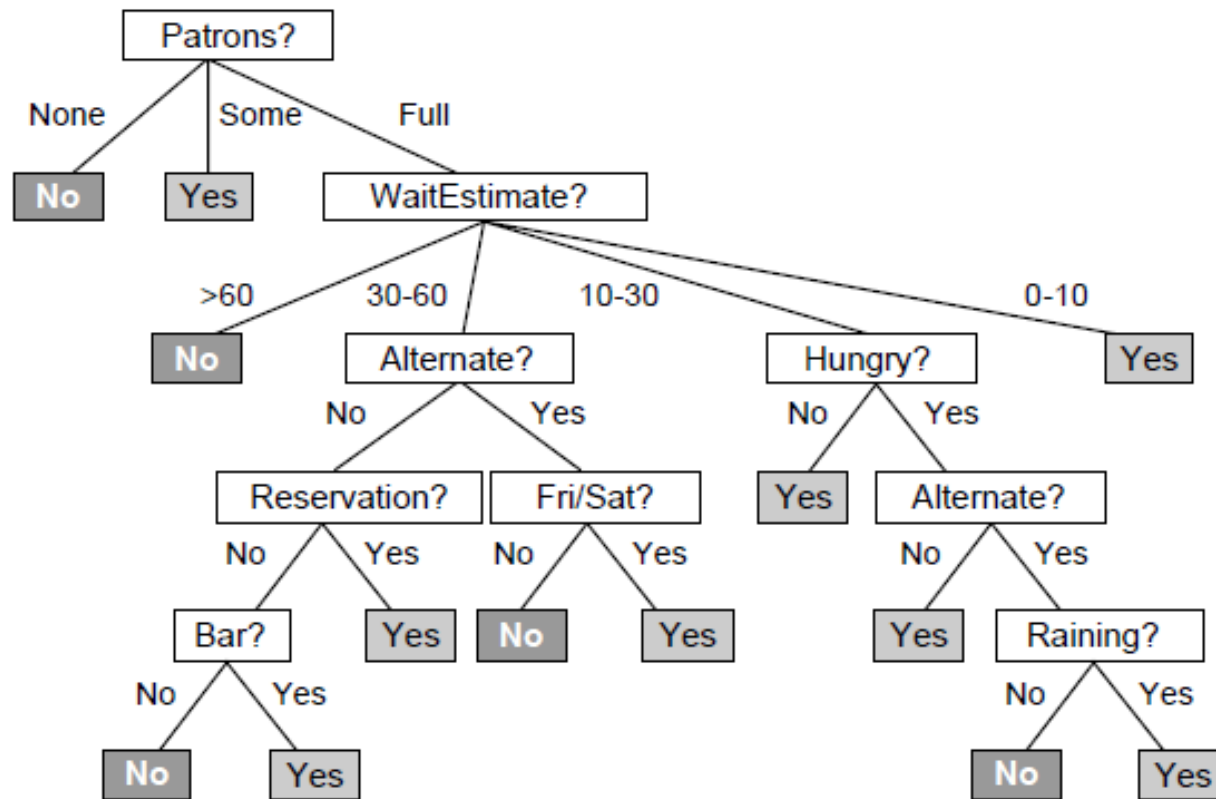


# Attribute-based representations

Example	Attributes										Target Wait
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	
$X_1$	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

- Examples described by **attribute values** (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table
- **Classification** of examples is **positive** (T) or **negative** (F)
- Serves as a training set

# Induced decision tree





# Decision Tree Learning: ID3, C4.5

- A greedy algorithm for decision tree construction developed by Ross Quinlan circa 1987
- Top-down construction of decision tree by recursively selecting “best attribute” to use at the current node in tree
  - Once attribute is selected for current node, generate child nodes, one for each possible value of selected attribute
  - Partition examples using the possible values of this attribute, and assign these subsets of the examples to the appropriate child node
  - Repeat for each child node until all examples associated with a node are either all positive or all negative

# R&N Algorithm

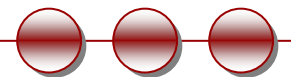
```
function DECISION-TREE-LEARNING(examples, attributes, parent_examples) returns a
tree

if examples is empty then return PLURALITY-VALUE(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return PLURALITY-VALUE(examples)
else
   $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$ 
  tree  $\leftarrow$  a new decision tree with root test A
  for each value  $v_k$  of A do
     $\text{exs} \leftarrow \{e : e \in \text{examples} \text{ and } e.A = v_k\}$ 
    subtree  $\leftarrow$  DECISION-TREE-LEARNING(exs, attributes - A, examples)
    add a branch to tree with label ( $A = v_k$ ) and subtree subtree
  return tree
```

# Choosing the best attribute

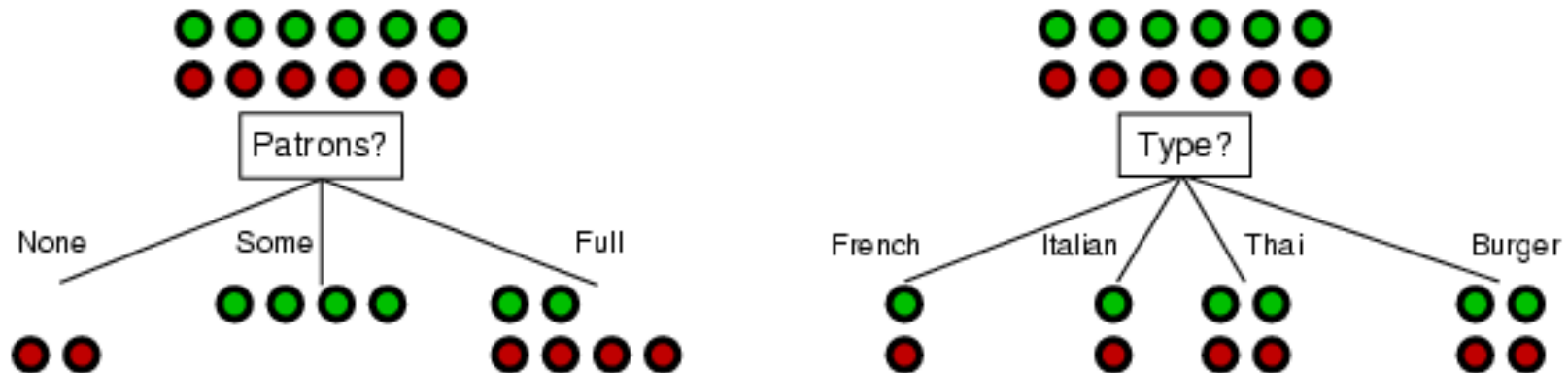


- Key problem: choosing which attribute to split a given set of examples
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*—i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 and C4.5 algorithms use the Max-Gain method of selecting the best attribute

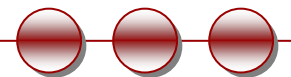


# Choosing an attribute

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

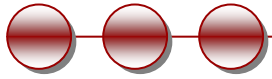


Which is better: *Patrons?* or *Type?*

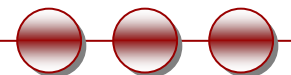


# Information theory 101

---



- Information theory sprang almost fully formed from the seminal work of Claude E. Shannon at Bell Labs
  - A Mathematical Theory of Communication, *Bell System Technical Journal*, 1948.
- Intuitions
  - Common words (a, the, dog) are shorter than less common ones (parliamentarian, foreshadowing)
  - In Morse code, common (probable) letters have shorter encodings
- Information is measured in minimum number of bits needed to store or send some information
- Wikipedia: The measure of data, known as information entropy, is usually expressed by the average number of bits needed for storage or communication.



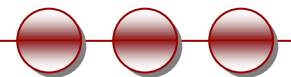
# Information theory 101

- Information is measured in bits
- Information conveyed by message depends on its probability
- With  $n$  equally probable possible *messages*, the probability  $p$  of each is  $1/n$
- Information conveyed by message is  $\log_2(n)$ 
  - e.g., with 16 messages, then  $\log_2(16) = 4$  and we need 4 bits to identify/send each message
- Given probability distribution for  $n$  messages  $P = (p_1, p_2, \dots, p_n)$ , the information conveyed by distribution (aka *entropy* of  $P$ ) is:

$$I(P) = -(p_1 * \log_2(p_1) + p_2 * \log_2(p_2) + \dots + p_n * \log_2(p_n))$$

probability of msg 2

info in msg 2



# Information theory II

- Entropy is the average number of bits/message needed to represent a stream of messages
- Information conveyed by distribution (a.k.a. *entropy* of P):  
$$I(P) = -(p_1 * \log_2 (p_1) + p_2 * \log_2 (p_2) + .. + p_n * \log_2 (p_n))$$
- Examples:
  - If P is (0.5, 0.5) then  $I(P) = 1$  → entropy of a fair coin flip
  - If P is (0.67, 0.33) then  $I(P) = 0.92$
  - If P is (0.99, 0.01) then  $I(P) = 0.08$
  - If P is (1, 0) then  $I(P) = 0$

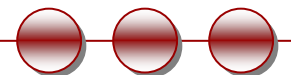
# Entropy as measure of homogeneity of examples

- Entropy used to characterize the (im)purity of an arbitrary collection of examples.
- Given a collection  $S$  (e.g. the table with 12 examples for the restaurant domain), containing positive and negative examples of some target concept, the entropy of  $S$  relative to its boolean classification is:

$$I(S) = -(p_+ * \log_2 (p_+) + p_- * \log_2 (p_-))$$

Entropy([6+, 6-]) = 1 → entropy of the restaurant dataset

Entropy([9+, 5-]) = 0.940





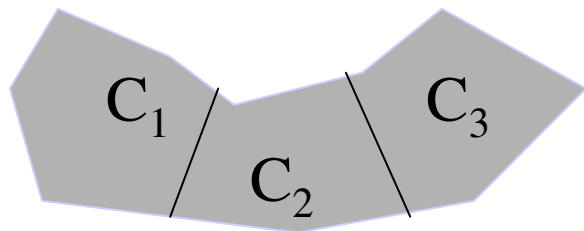
# Information for classification

If a set  $S$  of records is partitioned into disjoint exhaustive classes  $(C_1, C_2, \dots, C_k)$  on the basis of the value of the class attribute, then information needed to identify class of an element of  $T$  is:

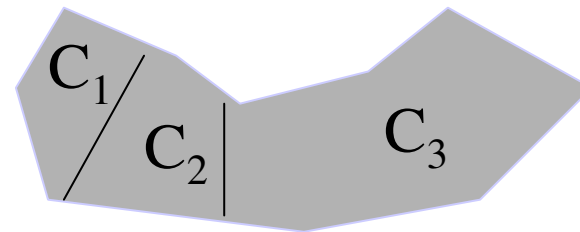
$$\text{Info}(S) = I(P)$$

where  $P$  is the probability distribution of partition  $(C_1, C_2, \dots, C_k)$ :

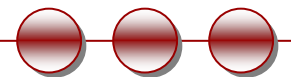
$$P = (|C_1|/|S|, |C_2|/|S|, \dots, |C_k|/|S|)$$



High information



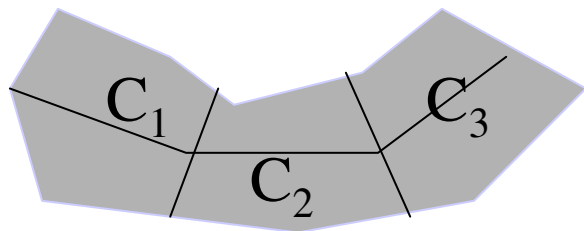
Low information



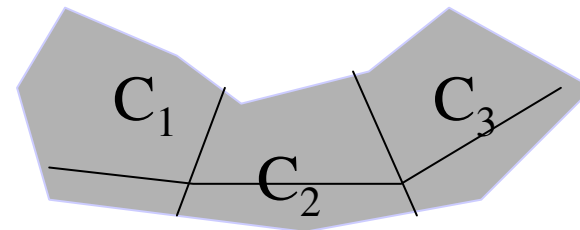
# Information for classification II

If we partition  $S$  w.r.t attribute  $A$  into sets  $\{S_1, S_2, \dots, S_n\}$  then the information needed to identify the class of an element of  $S$  becomes the weighted average of the information needed to identify the class of an element of  $S_i$ , i.e. the weighted average of  $\text{Info}(S_i)$ :

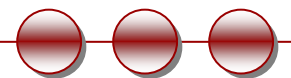
$$\text{Info}(A, S) = \sum |S_i|/|S| * \text{Info}(S_i)$$



High information



Low information



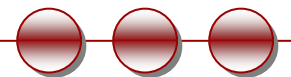
# Information gain

- A chosen attribute  $A$  divides the training set  $E$  into subsets  $E_1, \dots, E_v$  according to their values for  $A$ , where  $A$  has  $v$  distinct values.
- Consider the quantity  $IG(S,A)$ , the Information Gain of an attribute  $A$  relative to a collection of examples  $S$ , defined as

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

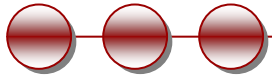
- This represents the difference between
  - $I(S)$  – the entropy of the original collection  $S$
  - $\text{Remainder}(A)$  - expected value of the entropy after  $S$  is partitioned using attribute  $A$
- This is the **gain in information due to attribute  $A$** 
  - Expected reduction in Entropy
  - $IG(S,A)$  or simply  $IG(A)$ :

$$IG(S,A) = I(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times I(S_v) \quad IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder}(A)$$

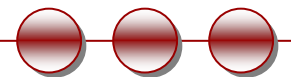


# Information gain

---



- Use to rank attributes and build DT (decision tree) where each node uses attribute with **greatest gain** of those not yet considered (in path from root)
- The intent of this ordering is to:
  - Create small DTs so records can be identified with few questions
  - Match a hoped-for minimality of the process represented by the records being considered (Occam's Razor)



# Information gain

For the training set, S:

$$p = n = 6,$$

$$I(6/12, 6/12) = 1 \text{ bit}$$

Consider the attributes *Patrons* and *Type* (and others too):

$$IG(\textit{Patrons}) = 1 - \left[ \frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}$$

$$IG(\textit{Type}) = 1 - \left[ \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

*Patrons* has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

# How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

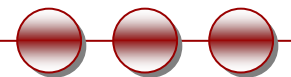
- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

# Extensions of ID3

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

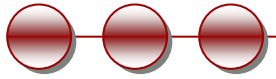
# Real-valued data

- Select a set of thresholds defining intervals
- Each interval becomes a discrete value of the attribute
- Use some simple heuristics...
  - always divide into quartiles
- Use domain knowledge...
  - divide age into infant (0-2), toddler (3 - 5), school-aged (5-8)
- Or treat this as another learning problem
  - Try a range of ways to discretize the continuous variable and see which yield “better results” w.r.t. some metric
  - E.g., try midpoint between every pair of values





# Noisy data



Many kinds of “noise” can occur in the examples:

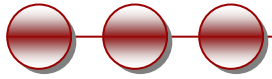
- Two examples have same attribute/value pairs, but different classifications
- Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
- Missing attribute values
- The classification is wrong (e.g., + instead of -) because of some error
- Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome



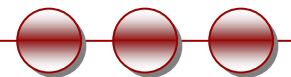
# Overfitting

- Irrelevant attributes, can result in *overfitting* the training example data
- If hypothesis space has many dimensions (large number of attributes), we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features
- If we have too little training data, even a reasonable hypothesis space will ‘overfit’

# Overfitting



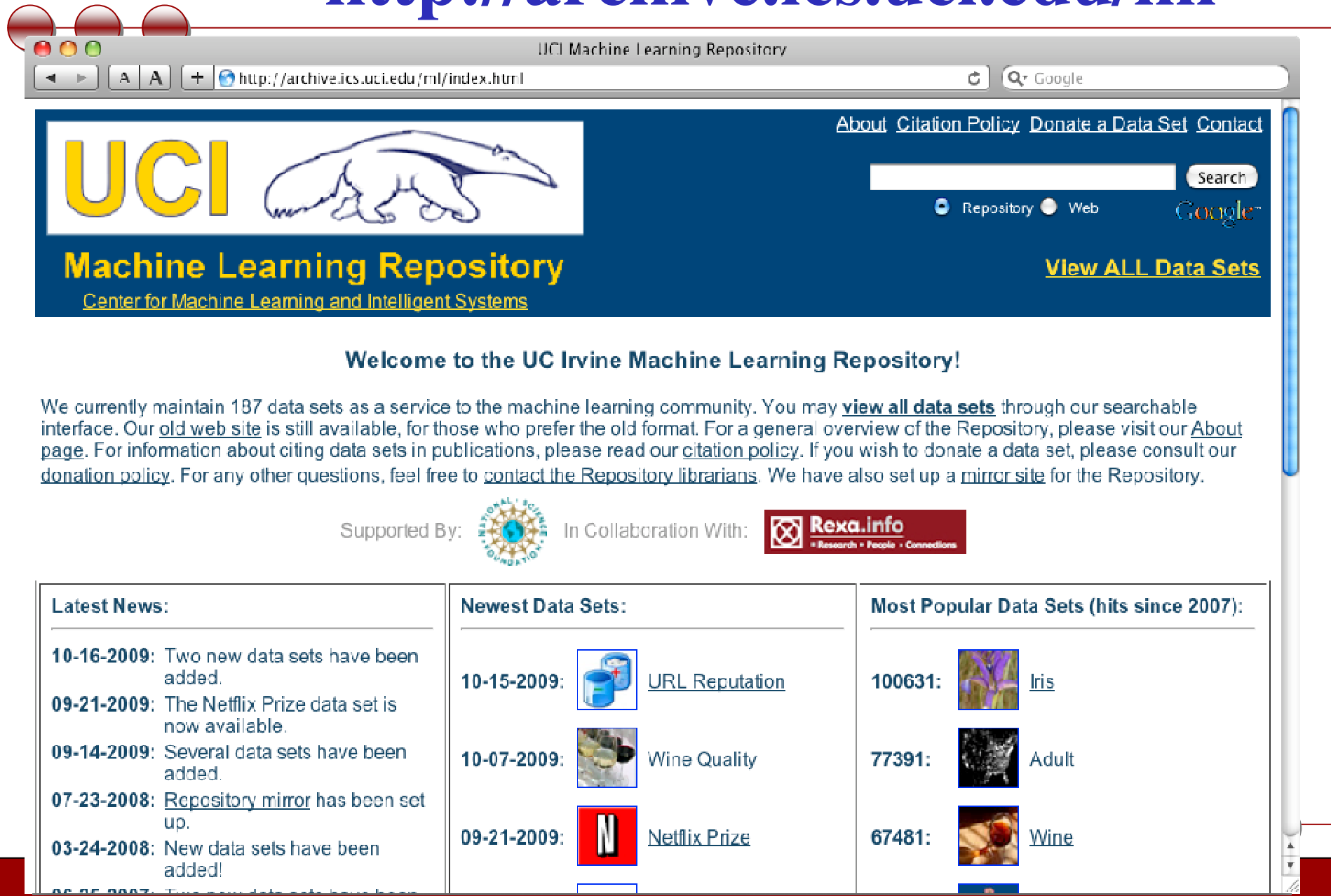
- Fix by removing irrelevant features
  - E.g., remove ‘year observed’, ‘month observed’, ‘day observed’, ‘observer name’ from feature vector
- Fix by getting more training data
- Fix by pruning lower nodes in the decision tree
  - E.g., if gain of the best attribute at a node is below a threshold, stop and make this node a leaf rather than generating children nodes



# Converting decision trees to rules

- It is easy to derive rules from a decision tree: write a rule for each path from the root to a leaf
- In that rule the left-hand side is built from the label of the nodes and the labels of the arcs
- The resulting rules set can be simplified:
  - Let LHS be the left hand side of a rule
  - LHS' obtained from LHS by eliminating some conditions
  - Replace LHS by LHS' in this rule if the subsets of the training set satisfying LHS and LHS' are equal
  - A rule may be eliminated by using meta-conditions such as “if no other rule applies”



# http://archive.ics.uci.edu/ml









The image shows a screenshot of a web browser displaying the UCI Machine Learning Repository website. The browser's address bar shows the URL <http://archive.ics.uci.edu/ml/index.html>. The website header features the UCI logo (a stylized bear) and the text "Machine Learning Repository" and "Center for Machine Learning and Intelligent Systems". Navigation links include "About", "Citation Policy", "Donate a Data Set", and "Contact". A search bar is present with a "Search" button and radio buttons for "Repository" and "Web". A "View ALL Data Sets" link is also visible.

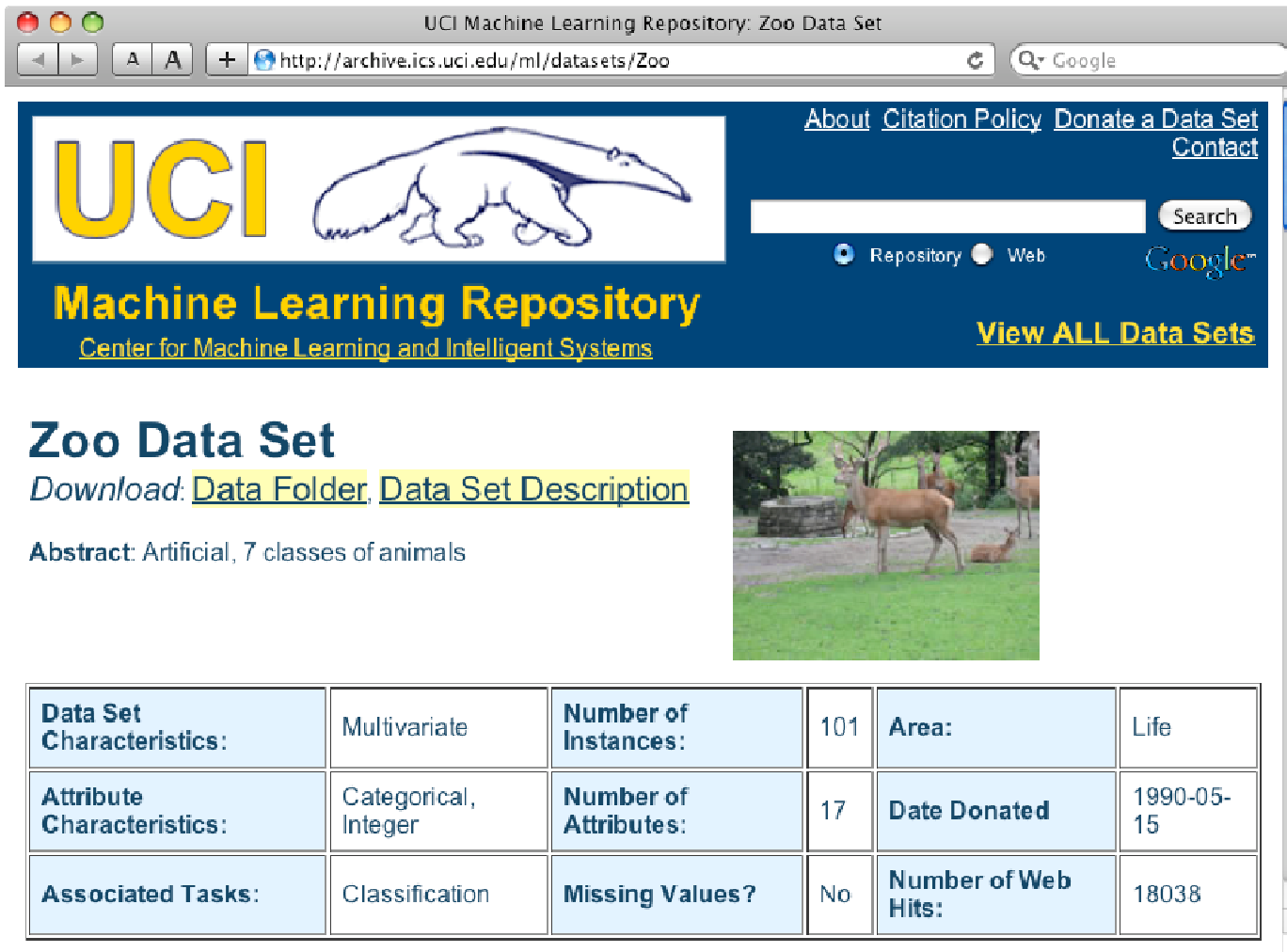
**Welcome to the UC Irvine Machine Learning Repository!**

We currently maintain 187 data sets as a service to the machine learning community. You may [view all data sets](#) through our searchable interface. Our [old web site](#) is still available, for those who prefer the old format. For a general overview of the Repository, please visit our [About page](#). For information about citing data sets in publications, please read our [citation policy](#). If you wish to donate a data set, please consult our [donation policy](#). For any other questions, feel free to [contact the Repository librarians](#). We have also set up a [mirror site](#) for the Repository.

Supported By:  In Collaboration With: 

Latest News:	Newest Data Sets:	Most Popular Data Sets (hits since 2007):
<b>10-16-2009:</b> Two new data sets have been added.	<b>10-15-2009:</b>  <a href="#">URL Reputation</a>	<b>100631:</b>  <a href="#">Iris</a>
<b>09-21-2009:</b> The Netflix Prize data set is now available.	<b>10-07-2009:</b>  <a href="#">Wine Quality</a>	<b>77391:</b>  <a href="#">Adult</a>
<b>09-14-2009:</b> Several data sets have been added.	<b>09-21-2009:</b>  <a href="#">Netflix Prize</a>	<b>67481:</b>  <a href="#">Wine</a>
<b>07-23-2008:</b> <a href="#">Repository mirror</a> has been set up.		
<b>03-24-2008:</b> New data sets have been added!		
<b>05-25-2007:</b> Two new data sets have been added.		

# Example: zoo data




The screenshot shows a web browser window with the URL <http://archive.ics.uci.edu/ml/datasets/Zoo>. The page features the UCI Machine Learning Repository logo, which includes a drawing of a sloth. Navigation links include "About", "Citation Policy", "Donate a Data Set", and "Contact". A search bar is present with a "Search" button and radio buttons for "Repository" and "Web". A "View ALL Data Sets" link is also visible.

## Zoo Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Artificial, 7 classes of animals



<b>Data Set Characteristics:</b>	Multivariate	<b>Number of Instances:</b>	101	<b>Area:</b>	Life
<b>Attribute Characteristics:</b>	Categorical, Integer	<b>Number of Attributes:</b>	17	<b>Date Donated</b>	1990-05-15
<b>Associated Tasks:</b>	Classification	<b>Missing Values?</b>	No	<b>Number of Web Hits:</b>	18038

<http://archive.ics.uci.edu/ml/datasets/Zoo>

# Zoo data

animal name: string  
hair: Boolean  
feathers: Boolean  
eggs: Boolean  
milk: Boolean  
airborne: Boolean  
aquatic: Boolean  
predator: Boolean  
toothed: Boolean  
backbone: Boolean  
breathes: Boolean  
venomous: Boolean  
fins: Boolean  
legs: {0,2,4,5,6,8}  
tail: Boolean  
domestic: Boolean  
catsize: Boolean  
type: {mammal, fish,  
bird, shellfish, insect,  
reptile, amphibian}

## 101 examples

aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal  
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal  
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish  
bear,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal  
boar,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal  
buffalo,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal  
calf,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1,mammal  
carp,0,0,1,0,0,1,0,1,1,0,0,1,0,1,1,0,fish  
catfish,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish  
cavy,1,0,0,1,0,0,0,1,1,1,0,0,4,0,1,0,mammal  
cheetah,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal  
chicken,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0,bird  
chub,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish  
clam,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,shellfish  
crab,0,0,1,0,0,1,1,0,0,0,0,0,4,0,0,0,shellfish  
...



# Zoo example

```
aima-python> python
```

```
>>> from learning import *
```

```
>>> zoo
```

```
<DataSet(zoo): 101 examples, 18 attributes>
```

```
>>> dt = DecisionTreeLearner()
```

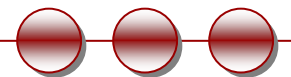
```
>>> dt.train(zoo)
```

```
>>> dt.predict(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0])
```

```
'fish'
```

```
>>> dt.predict(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0])
```

```
'mammal'
```



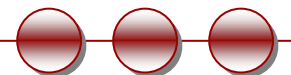


# Zoo example



>> dt.dt

```
DecisionTree(13, 'legs', {0: DecisionTree(12, 'fins', {0:
DecisionTree(8, 'toothed', {0: 'shellfish', 1: 'reptile'}), 1:
DecisionTree(3, 'eggs', {0: 'mammal', 1: 'fish'})}), 2:
DecisionTree(1, 'hair', {0: 'bird', 1: 'mammal'}), 4:
DecisionTree(1, 'hair', {0: DecisionTree(6, 'aquatic', {0:
'reptile', 1: DecisionTree(8, 'toothed', {0: 'shellfish', 1:
'amphibian'})}), 1: 'mammal'}), 5: 'shellfish', 6:
DecisionTree(6, 'aquatic', {0: 'insect', 1: 'shellfish'}), 8:
'shellfish'})
```



# Zoo example

```
>>> dt.dt.display()
```

```
Test legs
```

```
legs = 0 ==> Test fins
```

```
  fins = 0 ==> Test toothed
```

```
    toothed = 0 ==> RESULT = shellfish
```

```
    toothed = 1 ==> RESULT = reptile
```

```
  fins = 1 ==> Test eggs
```

```
    eggs = 0 ==> RESULT = mammal
```

```
    eggs = 1 ==> RESULT = fish
```

```
legs = 2 ==> Test hair
```

```
  hair = 0 ==> RESULT = bird
```

```
  hair = 1 ==> RESULT = mammal
```

```
legs = 4 ==> Test hair
```

```
  hair = 0 ==> Test aquatic
```

```
    aquatic = 0 ==> RESULT = reptile
```

```
    aquatic = 1 ==> Test toothed
```

```
      toothed = 0 ==> RESULT = shellfish
```

```
      toothed = 1 ==> RESULT = amphibian
```

```
  hair = 1 ==> RESULT = mammal
```

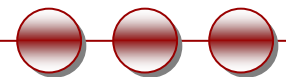
```
legs = 5 ==> RESULT = shellfish
```

```
legs = 6 ==> Test aquatic
```

```
  aquatic = 0 ==> RESULT = insect
```

```
  aquatic = 1 ==> RESULT = shellfish
```

```
legs = 8 ==> RESULT = shellfish
```



# Zoo example

```
>>> dt.dt.display()
```

```
Test legs
```

```
legs = 0 ==> Test fins
```

```
fins = 0 ==> Test toothed
```

```
toothed = 0 ==> RESULT = shellfish
```

```
toothed = 1 ==> RESULT = reptile
```

```
fins = 1 ==> Test milk
```

```
milk = 0 ==> RESULT = fish
```

```
milk = 1 ==> RESULT = mammal
```

```
legs = 2 ==> Test hair
```

```
hair = 0 ==> RESULT = bird
```

```
hair = 1 ==> RESULT = mammal
```

```
legs = 4 ==> Test hair
```

```
hair = 0 ==> Test aquatic
```

```
aquatic = 0 ==> RESULT = reptile
```

```
aquatic = 1 ==> Test toothed
```

```
toothed = 0 ==> RESULT = shellfish
```

```
toothed = 1 ==> RESULT = amphibian
```

```
hair = 1 ==> RESULT = mammal
```

```
legs = 5 ==> RESULT = shellfish
```

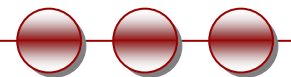
```
legs = 6 ==> Test aquatic
```

```
aquatic = 0 ==> RESULT = insect
```

```
aquatic = 1 ==> RESULT = shellfish
```

```
legs = 8 ==> RESULT = shellfish
```

Add the shark example  
to the training set and  
retrain



# Evaluation methodology

- Standard methodology:
  1. Collect large set of examples with correct classifications
  2. Randomly divide collection into two disjoint sets: *training* and *test*
  3. Apply learning algorithm to training set giving hypothesis  $H$
  4. Measure performance of  $H$  w.r.t. test set
- Important: keep the training and test sets disjoint!
- Study efficiency and robustness of algorithm: repeat steps 2-4 for different training sets and sizes of training sets
- On modifying algorithm, restart with step 1 to avoid evolving algorithm to work well on just this collection

# Measuring model quality

- How good is a model?
  - Predictive accuracy
  - False positives / false negatives for a given cutoff threshold
    - Loss function (accounts for cost of different types of errors)
  - Area under the (ROC) curve
  - Minimizing loss can lead to problems with overfitting
- Training error
  - Train on all data; measure error on all data
  - Subject to overfitting (of course we'll make good predictions on the data on which we trained!)
- Regularization
  - Attempt to avoid overfitting
  - Explicitly minimize the complexity of the function while minimizing loss. Trade off is modeled with a *regularization parameter*

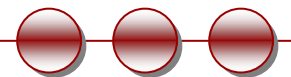
# K-fold Cross Validation

- Problem: getting “ground truth” data can be expensive
- Problem: ideally need different test data each time we test
- Problem: experimentation is needed to find right “feature space” and parameters for ML algorithm
- Goal: minimize amount of training+test data needed
- Idea: split training data into  $K$  subsets, use  $K-1$  for *training*, and one for *development testing*
- Common  $K$  values are 5 and 10

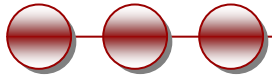
# K-fold Cross-validation, cont.



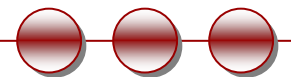
- k-fold cross-validation:
  - Divide data into  $k$  folds
  - Train on  $k-1$  folds, use the  $k$ th fold to measure error
  - Repeat  $k$  times; use average error to measure generalization accuracy
  - Statistically valid and gives good accuracy estimates



# Zoo evaluation



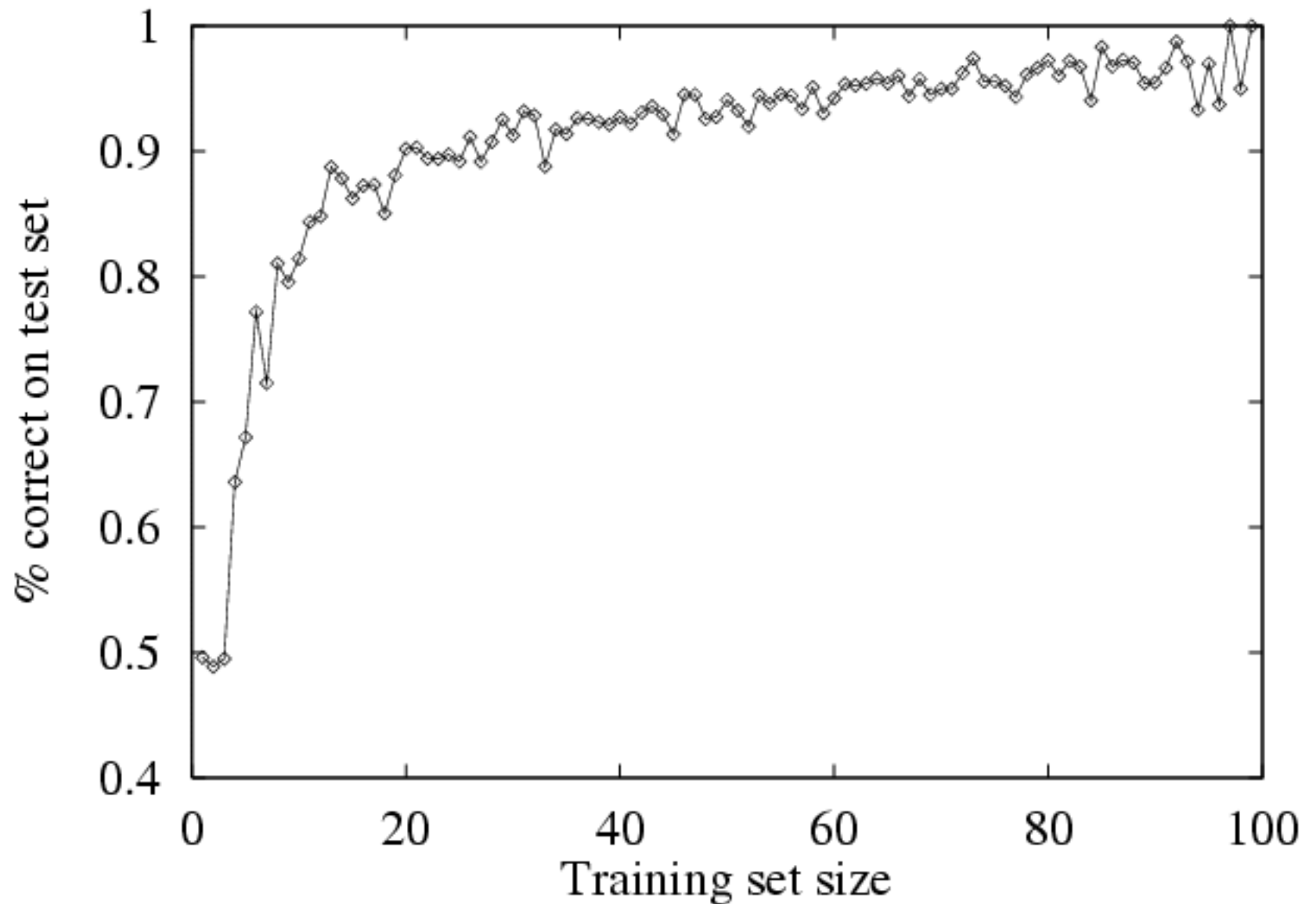
```
>>> train_and_test(DecisionTreeLearner(), zoo, 0, 10)
1.0
>>> train_and_test(DecisionTreeLearner(), zoo, 90, 100)
0.800000000000000004
>>> train_and_test(DecisionTreeLearner(), zoo, 90, 101)
0.81818181818181823
>>> train_and_test(DecisionTreeLearner(), zoo, 80, 90)
0.900000000000000002
>>> cross_validation(DecisionTreeLearner(), zoo, 10, 20)
0.955000000000000007
>>> leave1out(DecisionTreeLearner(), zoo)
0.97029702970297027
```





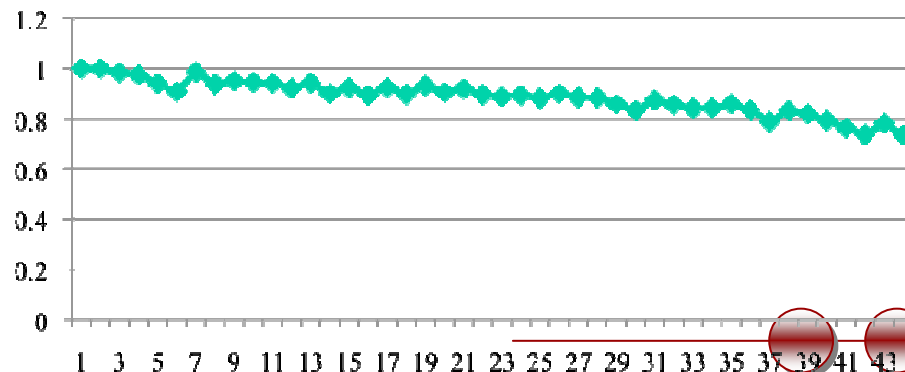
# Learning curve

Learning curve = % correct on test set as a function of training set size



# Zoo

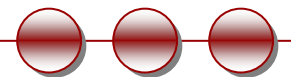
```
>>> learningcurve(DecisionTreeLearner(), zoo)
[(2, 1.0), (4, 1.0), (6, 0.983333333333333333339), (8,
0.974999999999999999998), (10, 0.940000000000000000006), (12,
0.908333333333333333321), (14, 0.98571428571428577), (16,
0.9375), (18, 0.949999999999999999996), (20,
0.944999999999999999995), ... (86, 0.78255813953488373), (88,
0.7363636363636363644), (90, 0.7077777777777777795)]
```



# Summary: Decision tree learning

---

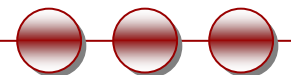
- Inducing decision trees is one of the most widely used learning methods in practice
- Can out-perform human experts in many problems
- Method for approximating discrete-valued functions
- Capable of learning disjunctive expressions (rules)
- Search over a completely expressive hypothesis space
- Inductive bias: a preference for small trees over large trees



# Summary: Decision tree learning

---

- Strengths include
  - Fast
  - Simple to implement
  - Can convert result to a set of easily interpretable rules
  - Empirically valid in many commercial products
  - Handles noisy data
- Weaknesses include:
  - Univariate splits/partitioning using only one attribute at a time so limits types of possible trees
  - Large decision trees may be hard to understand
  - Requires fixed-length feature vectors
  - Non-incremental (i.e., batch method)



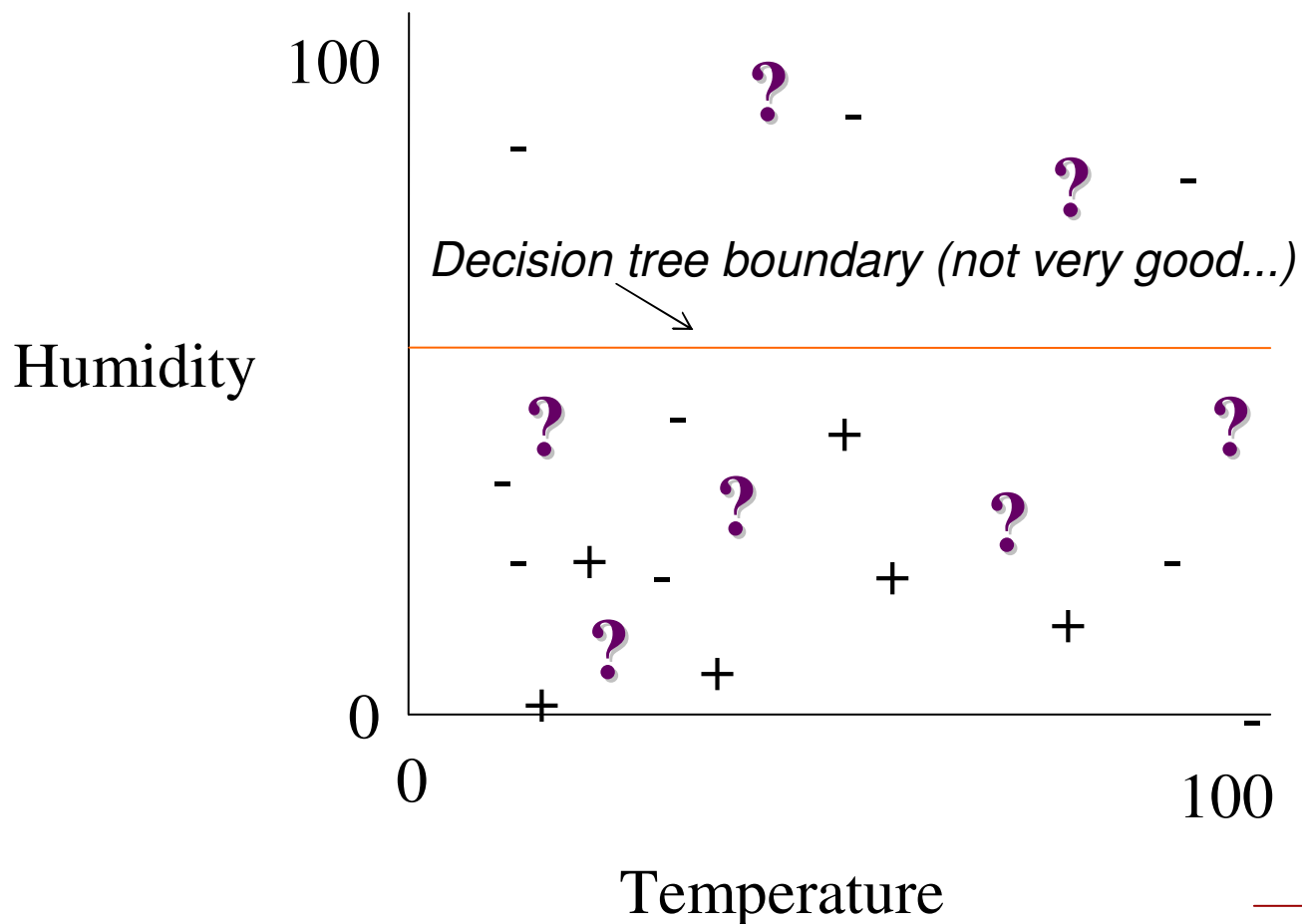
# Instance Based Learning

- Decision trees are a kind of *model*-based learning
  - We take the training instances and use them to build a *model* of the mapping from inputs to outputs
  - This model (e.g., a decision tree) can be used to make predictions on new (test) instances
- Another option is to do *instance*-based learning
  - Save all (or some subset) of the instances
  - Given a test instance, use some of the stored instances in some way to make a prediction
- Instance-based methods:
  - Nearest neighbor and its variants
  - Support vector machines

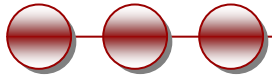
# Nearest Neighbor

- Vanilla “Nearest Neighbor”:
  - Save all training instances  $X_i = (C_i, F_i)$  in  $T$
  - Given a new test instance  $Y$ , find the instance  $X_j$  that is closest to  $Y$
  - Predict class  $C_i$
- What does “closest” mean?
  - Usually: Euclidean distance in feature space
  - Alternatively: Manhattan distance, or any other distance metric
- What if the data is noisy?
  - Generalize to k-nearest neighbor
  - Find the  $k$  closest training instances to  $Y$
  - Use majority voting to predict the class label of  $Y$
  - Better yet: use weighted (by distance) voting to predict the class label

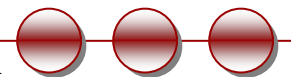
# Nearest Neighbor Example: Run Outside (+) or Inside (-)



- Noisy data
- Not linearly separable

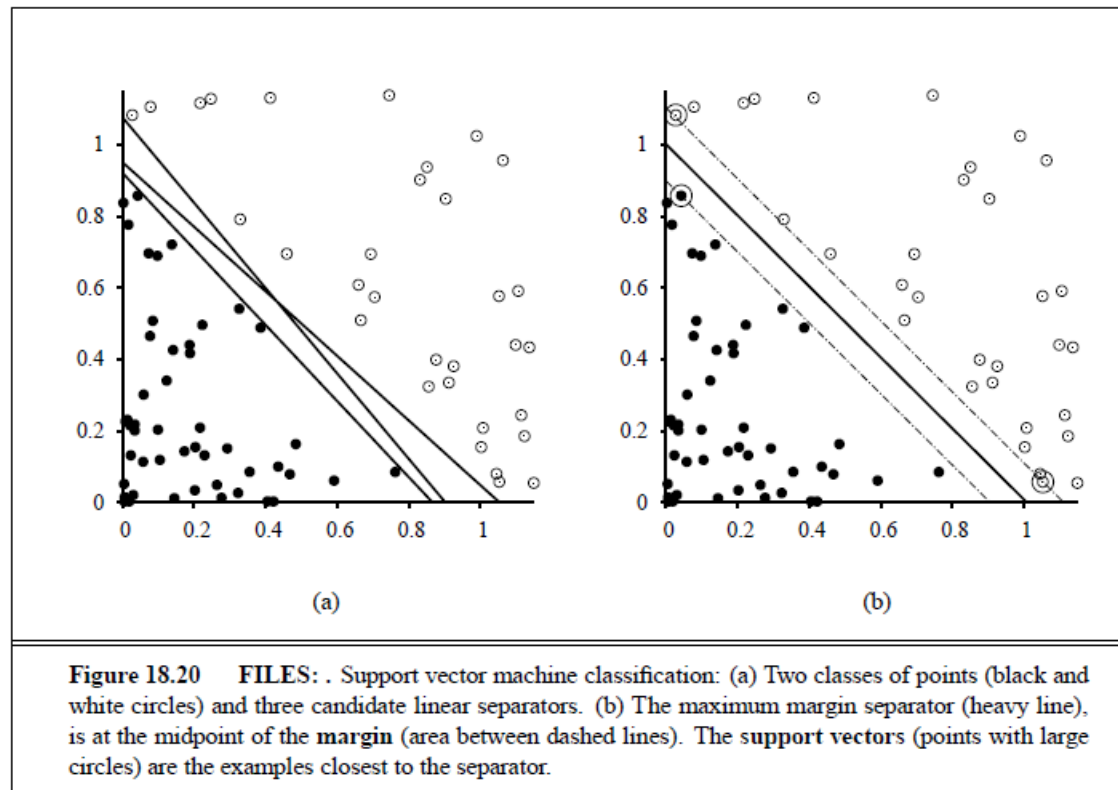


- A support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression or other tasks.
- Intuitively, a good separation is achieved by the hyperplane that has the **largest distance** to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.
  - Maximum margin separator
  - Linear separating hyperplane: data that are not linearly separable in the original input space, are easily separable in the higher dimensional space
- SVMs are currently the best-known classifier on a well-studied hand-written-character recognition benchmark

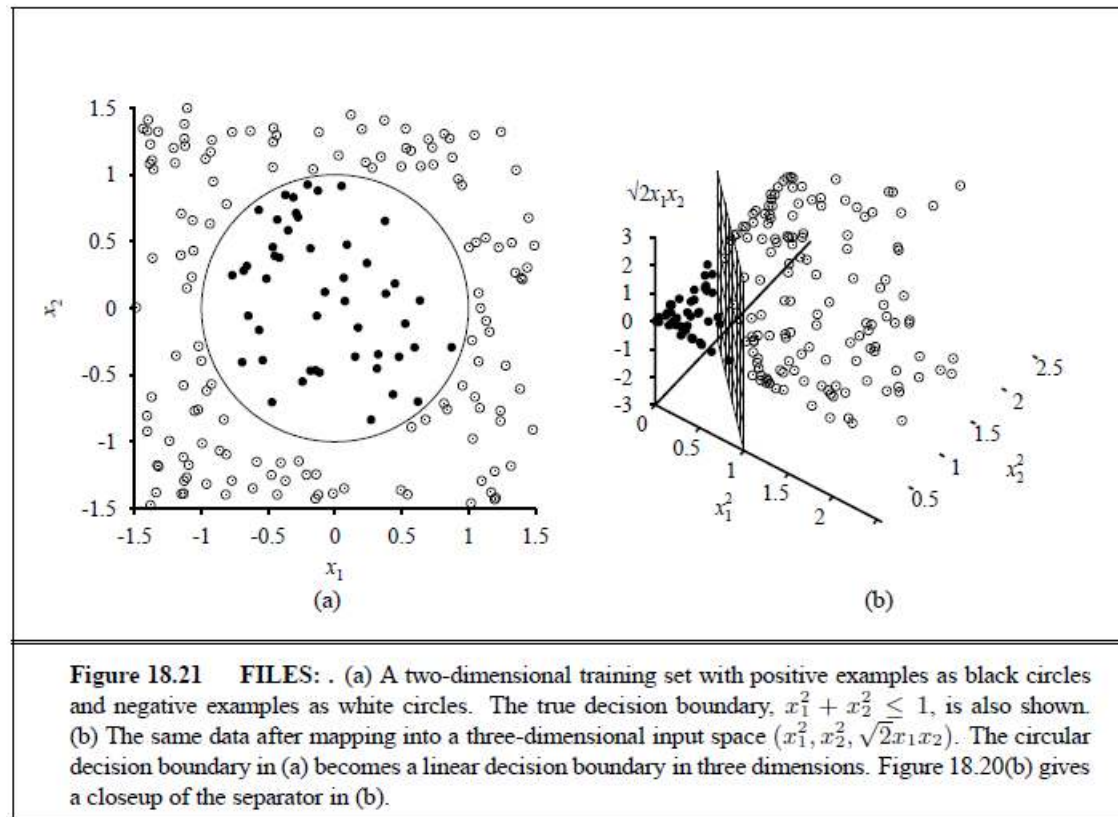




# SVM Classification



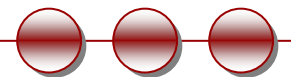
# SVM Mapping



- SVMs map data into a sufficiently high dimension where it is (almost) linearly separable

# Decision Trees Exercise

- Suppose we want ID3 to decide whether the weather is amenable to playing tennis. Over the course of 2 weeks, data is collected to help ID3 build a decision tree.
- The target (binary) classification is
  - "should we play **PlayTennis**?" which can be Yes or No (T or F; + or -).
- The weather attributes are outlook, temperature, humidity, and wind. They can have the following values:
  - Outlook = { sunny, overcast, rain }
  - Temperature = { hot, mild, cool }
  - Humidity = { high, normal }
  - Wind = { weak, strong }



# Training examples for the target concept PlayTennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

# Remember ...

- Entropy of a given collection S:

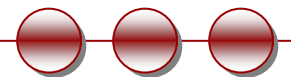
$$I(S) = -(p_+ \cdot \log_2(p_+) + p_- \cdot \log_2(p_-))$$

$$I([6+, 6-]) = 1 \rightarrow \text{entropy of the restaurant dataset}$$

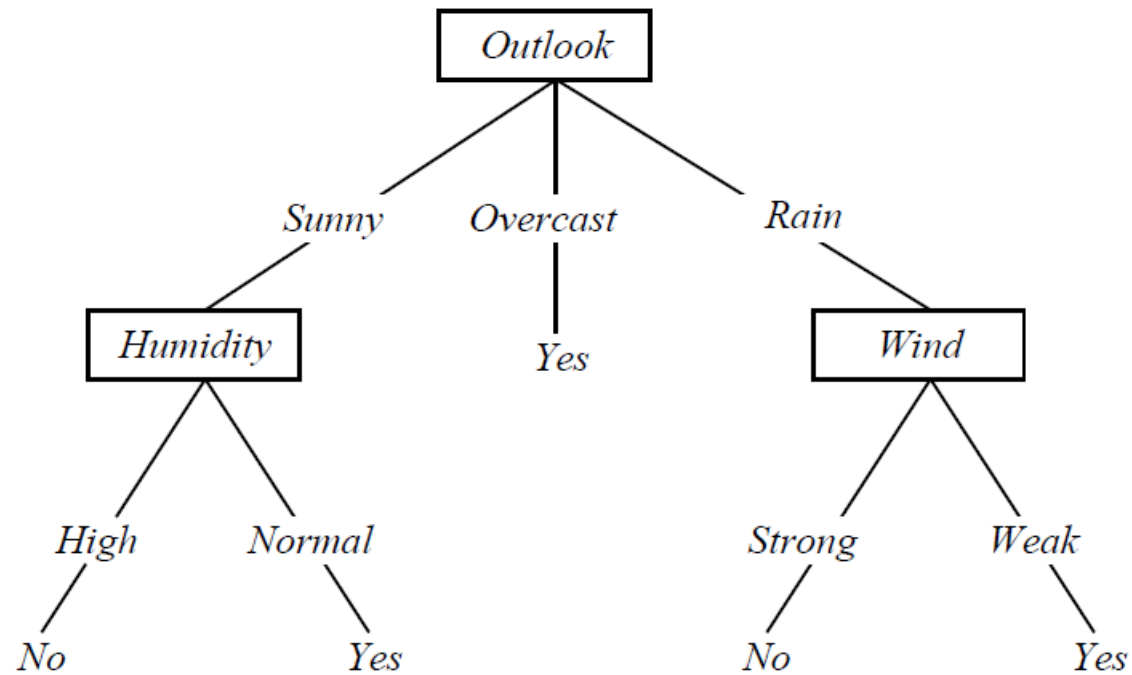
$$I([9+, 5-]) = 0.940$$

- Information gain of attribute A (on collection S):

$$IG(S, A) = I(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \times I(S_v)$$



# Induced decision tree



# Summary Learning from Examples

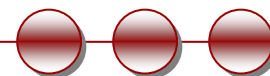


## ■ Supervised learning

- Learn an unknown function  $f(X) = Y$ , where  $X$  is an input example and  $Y$  is the desired output.
- **Supervised learning** implies we are given a **training set** of  $(X, Y)$  pairs by a “teacher”
- **Classification**: Learning a discrete-valued function
- **Regression**: Learning a continuous function

## ■ Inductive learning

- Generalizing from examples (finding a hypothesis that agrees with the examples)
- **Ockham’s razor**: choose the simplest consistent hypothesis



# Summary Learning from Examples

- **Model-based learning:** use training instances to build a *model* (of the mapping from inputs to outputs).
  - Example: decision trees
- **Parametric learning:** use training instances to summarize data with a set of parameters.
  - Example: Neural Networks
- **Non parametric learning:** no parameters or model are learned.
  - Instance-based learning: use all the instances every time in some way to make a prediction.
  - Examples: Nearest neighbor and Support vector machines
- **Evaluation**
  - Randomly divide collection into two disjoint sets: *training* and *test*; apply learning algorithm to training set; measure performance on test set
- **Applications**
  - SVMs are currently the best-known classifier on a well-studied hand-written-character recognition benchmark
  - Many case studies have shown that decision trees are at least as accurate as human experts, e.g., diagnosing breast cancer

