Decision Trees

Function Approximation

Problem Se*ng

- Set of possible instances
- Set of possible labels
- Unknown target function $f : \mathcal{X} \to \mathcal{Y}$
- Set of function hypotheses $H = \{ h \mid h : \mathcal{X} \to \mathcal{Y} \}$

Input: Training examples of unknown target function f

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

Output: Hypothesis $h \in H$ that best approximates f

Sample Dataset

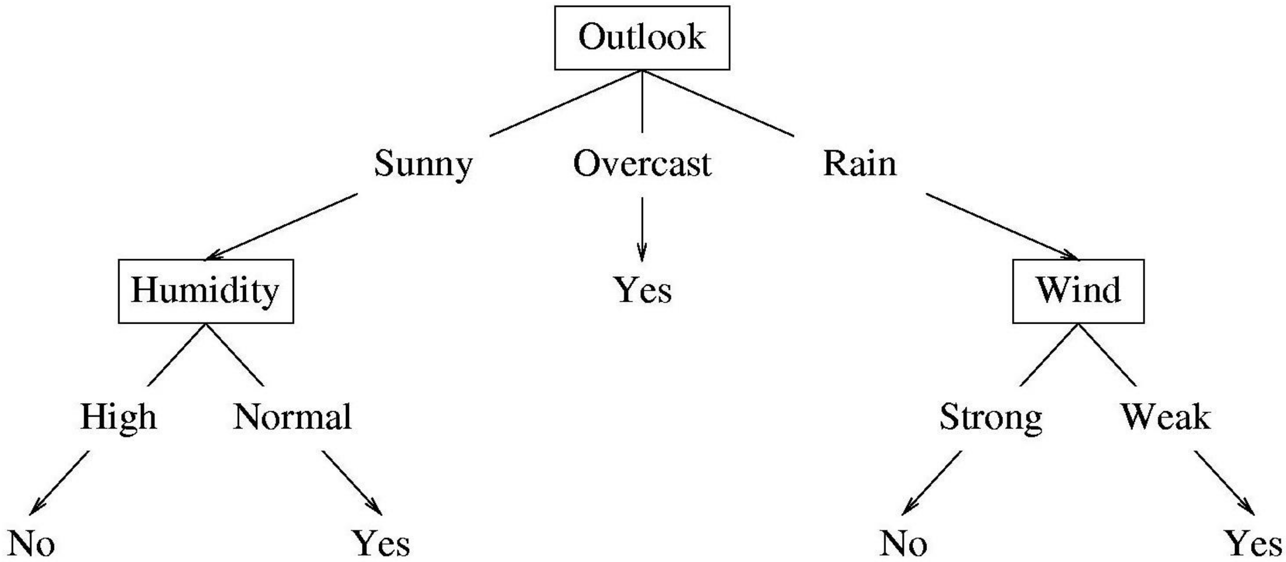
- Columns denote features X_i
- Rows denote labeled instances $\boldsymbol{x}_i, y_i$
- Class label denotes whether a tennis game was played

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

$\boldsymbol{x}_i, y_i$

Decision Tree

- A possible decision tree for the data:

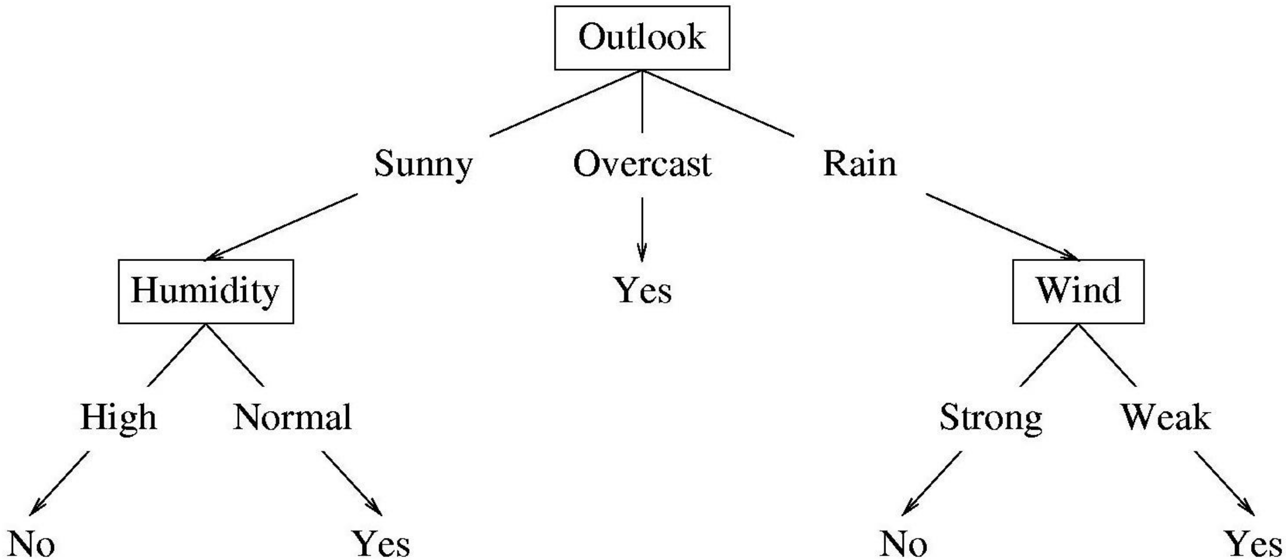


- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y \mid \boldsymbol{x} \; 2 \; \text{leaf})$)

Based on slide by Tom Mitchell

Decision Tree

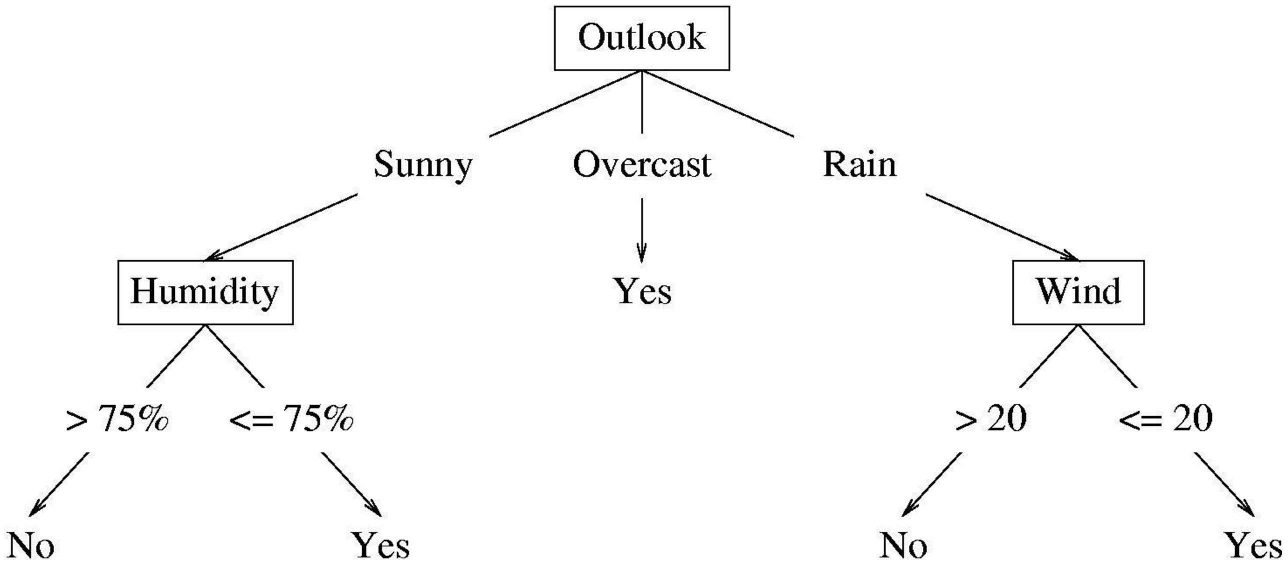
- A possible decision tree for the data:



- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Decision Tree

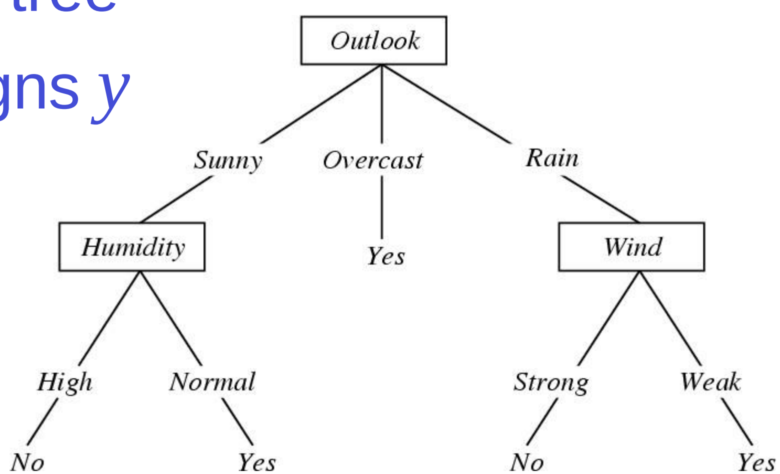
- If features are continuous, internal nodes can test the value of a feature against a threshold



Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses H={ h | $h : X \rightarrow Y$ }
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y

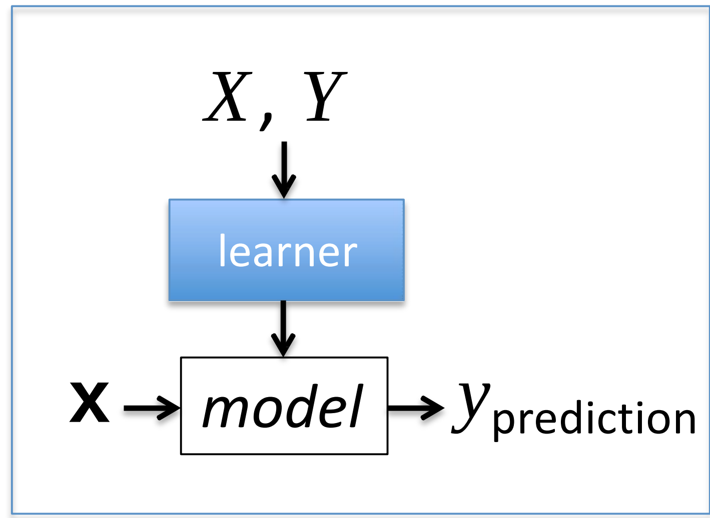


Slide by Tom Mitchell

Stages of (Batch) Machine Learning

Given: labeled training data $X, Y = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n}$

- Assumes each $\boldsymbol{x}_i \dashleftarrow \mathcal{D}(\mathcal{X})$ with $y_i = f_{target}(\boldsymbol{x}_i)$



Train the model:

model ← *classifier*.train(X, Y)

Apply the model to new data:

- Given: new unlabeled instance $\boldsymbol{x} \dashleftarrow \mathcal{D}(\mathcal{X})$

$y_{\text{prediction}}$ ← *model*.predict(**x**)

Example Application: A Tree to Predict Caesarean Section Risk

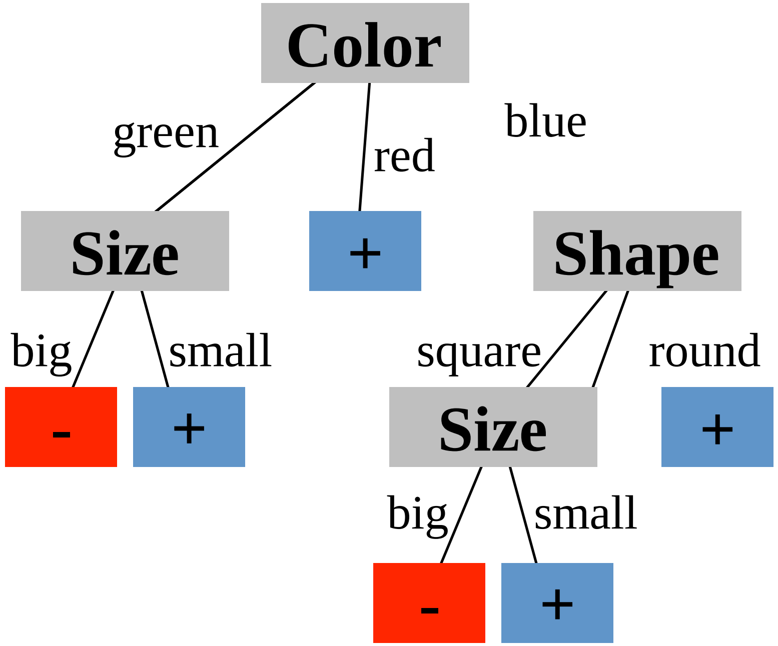
Learned from medical records of 1000 women

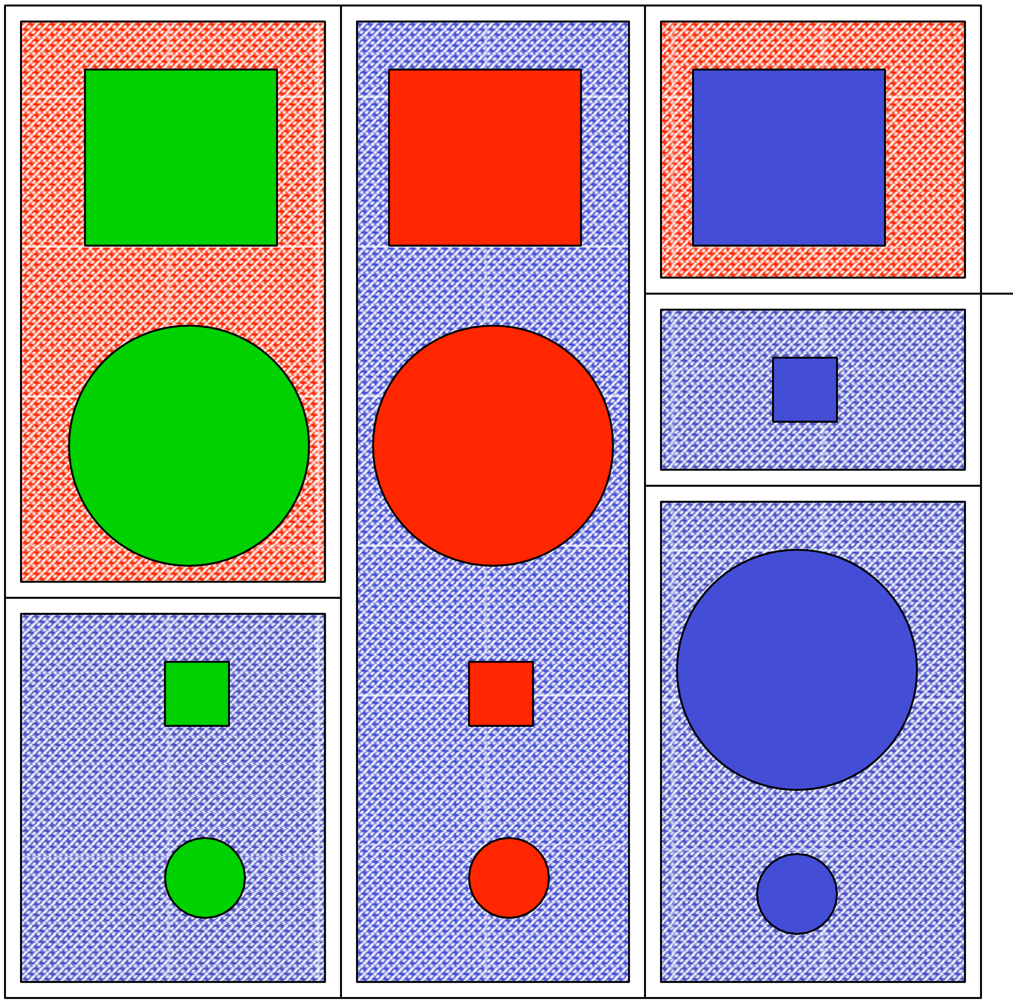
Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Based on Example by Tom Mitchell

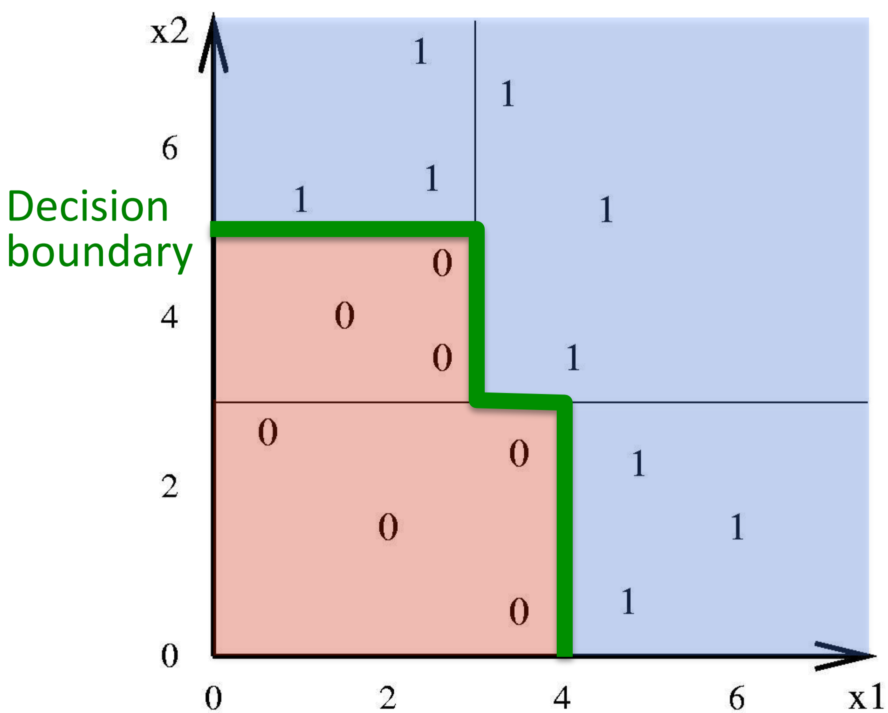
Decision Tree Induced Partition

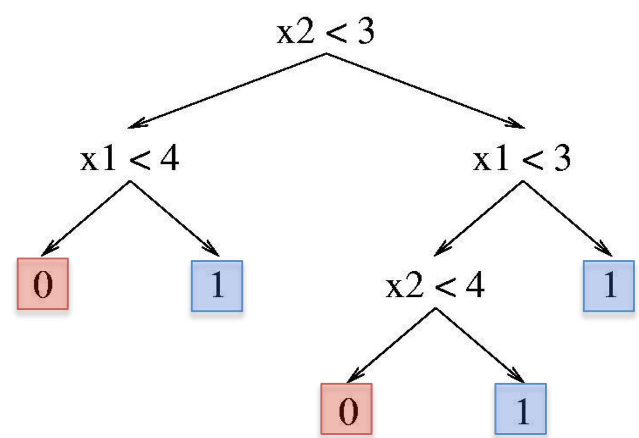




Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - – or a probability distribution over labels

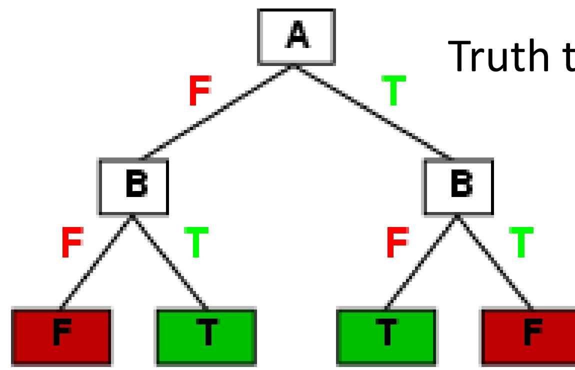




Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



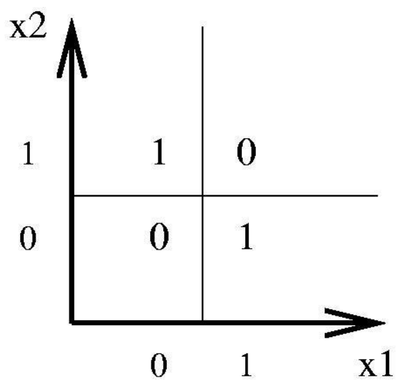
Truth table row → path to leaf

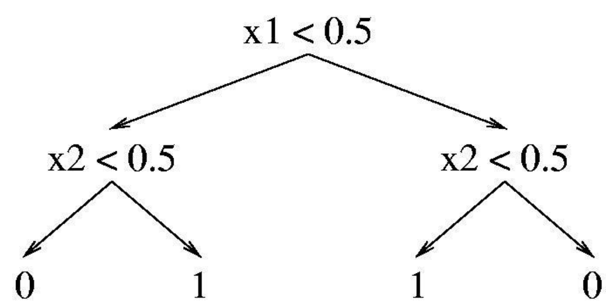
- In the worst case, the tree will require exponentially many nodes

Expressiveness

Decision trees have a variable-sized hypothesis space

- As the #nodes (or depth) increases, the hypothesis space grows
  - Depth 1 ("decision stump"): can represent any boolean function of one feature
  - Depth 2: any boolean fn of two features; some involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$)
  - etc.





Based on slide by Pedro Domingos

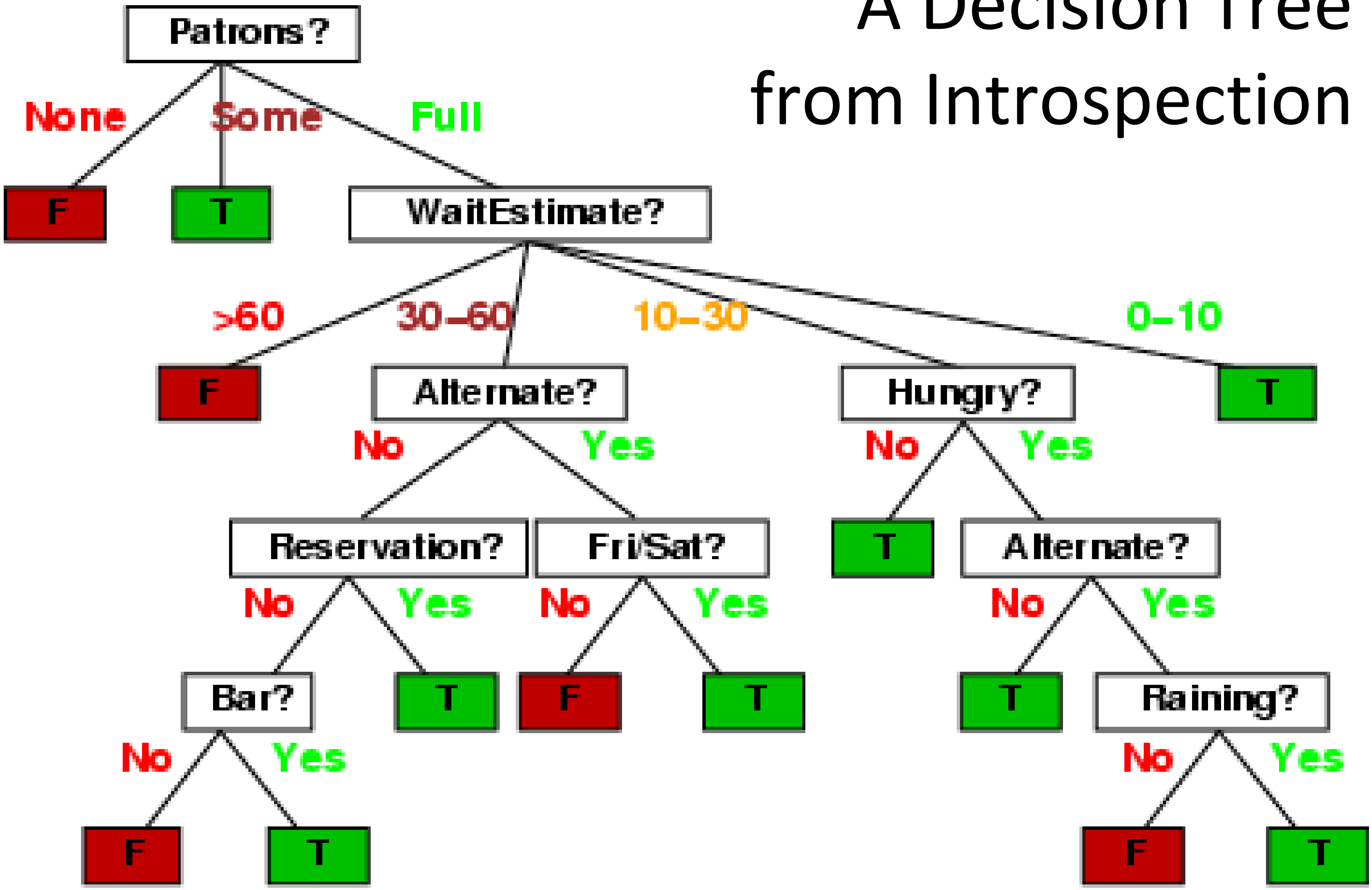
Another Example: Restaurant Domain (Russell & Norvig)

Model a patron's decision of whether to wait for a table at a 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

~7,000 possible cases

A Decision Tree from Introspection



Is this the best decision tree?

Preference bias: Ockham's Razor

- Principle stated by William of Ockham (1285-1347)
  - "*non sunt multiplicanda entia praeter necessitatem*"
  - entities are not to be multiplied beyond necessity
  - AKA Occam's Razor, Law of Economy, or Law of Parsimony

Idea: The simplest consistent explanation is the best

- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  - Finding the provably smallest decision tree is NP-hard
  - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

node = root of decision tree

Main loop:

1. *A* ← the "best" decision attribute for the next node.
2. Assign *A* as decision attribute for *node*.
3. For each value of *A*, create a new descendant of *node*.
4. Sort training examples to leaf nodes.
5. If training examples are perfectly classified, stop.
   Else, recurse over new leaf nodes.

How do we choose which attribute is best?

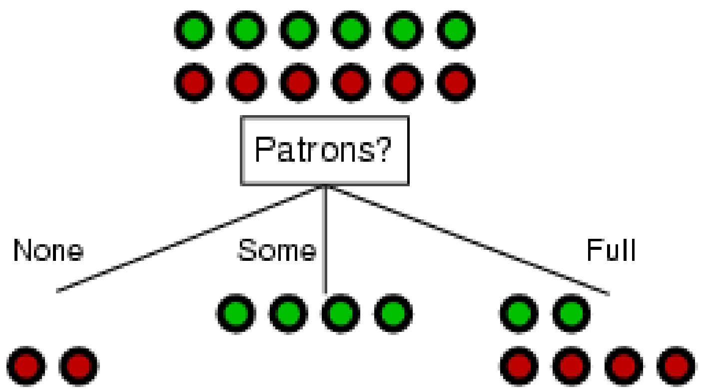
Choosing the Best Attribute

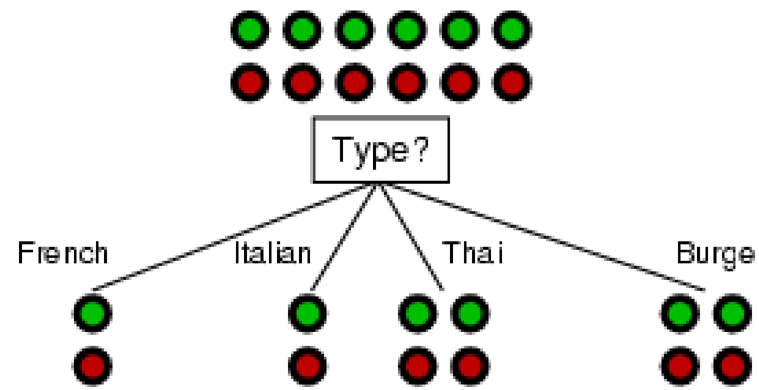
Key problem: choosing which attribute to split a given set of examples

- 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 algorithm uses the Max-Gain method of selecting the best attribute

Choosing an Attribute

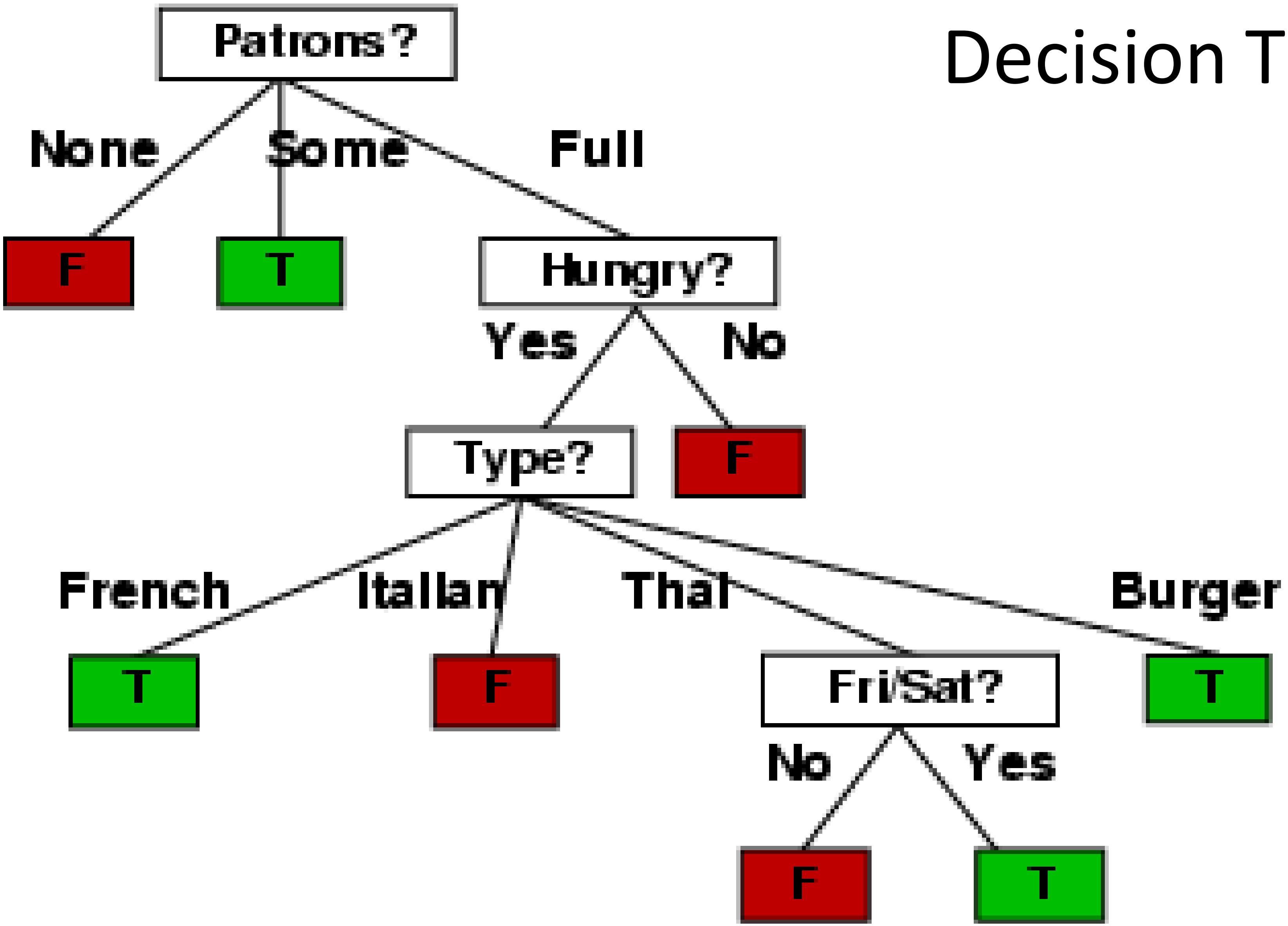
Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





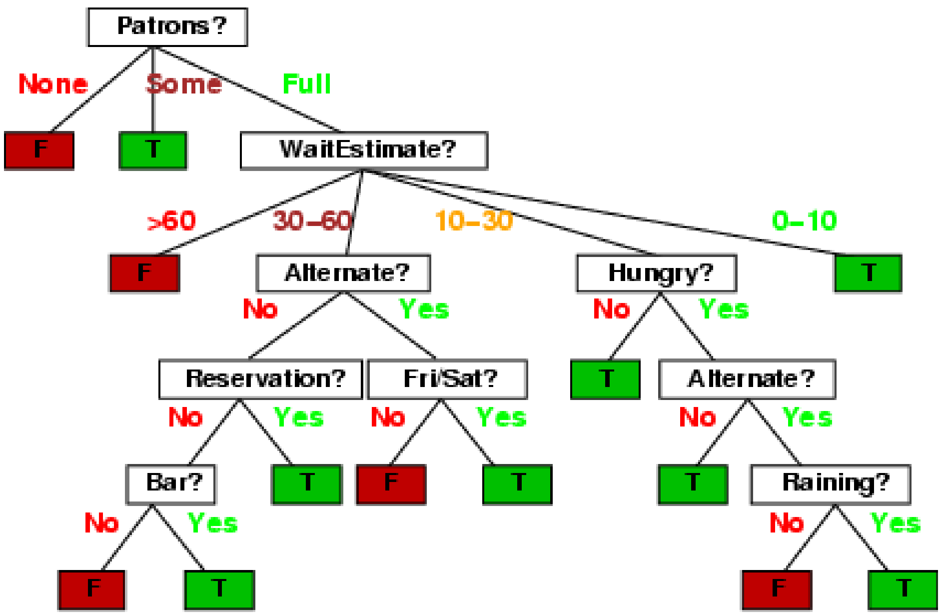
Which split is more informative: *Patrons?* or *Type?*

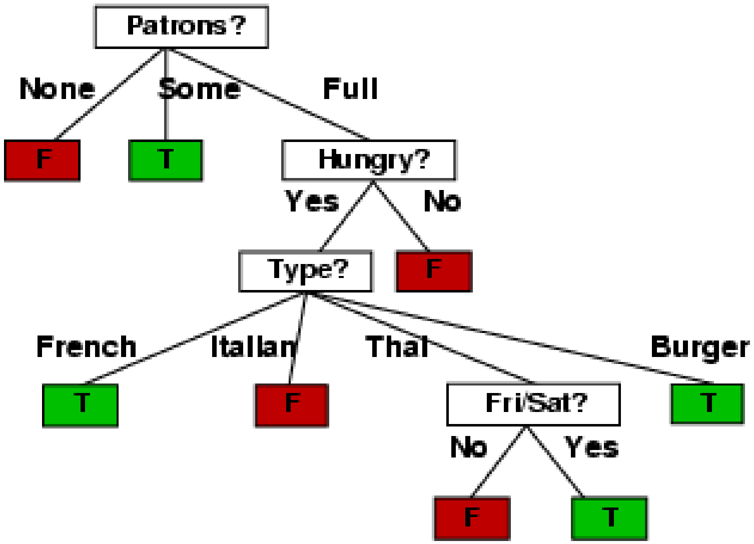
ID3-induced Decision Tree



Based on Slide from M. desJardins & T. Finin

Compare the Two Decision Trees



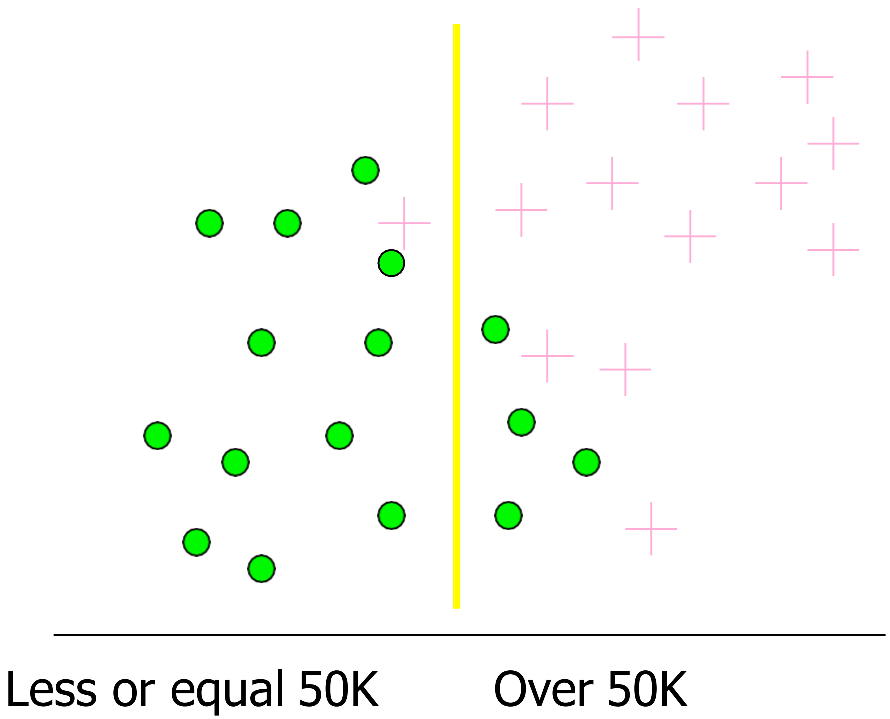


Based on Slide from M. desJardins & T. Finin

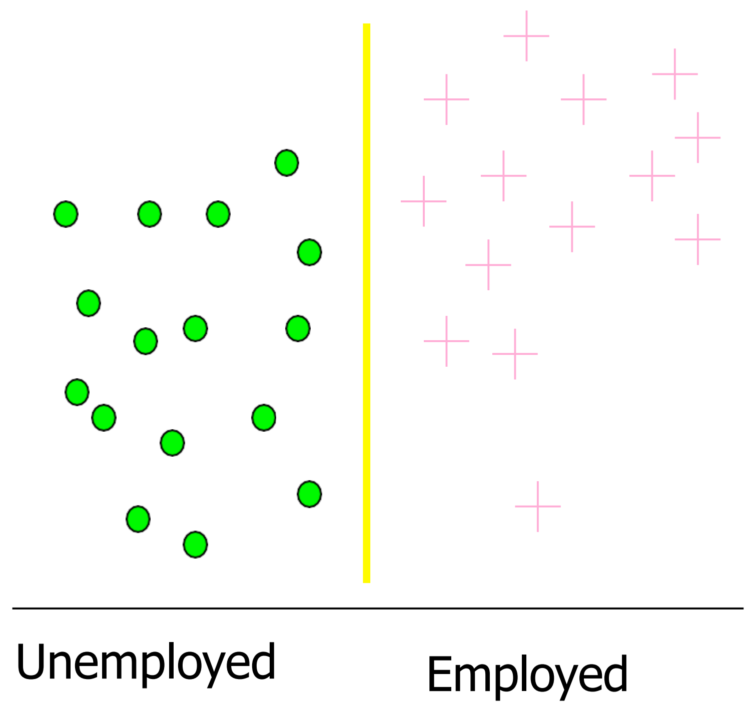
Information Gain

Which test is more informative?

Split over whether Balance exceeds 50K



Split over whether applicant is employed

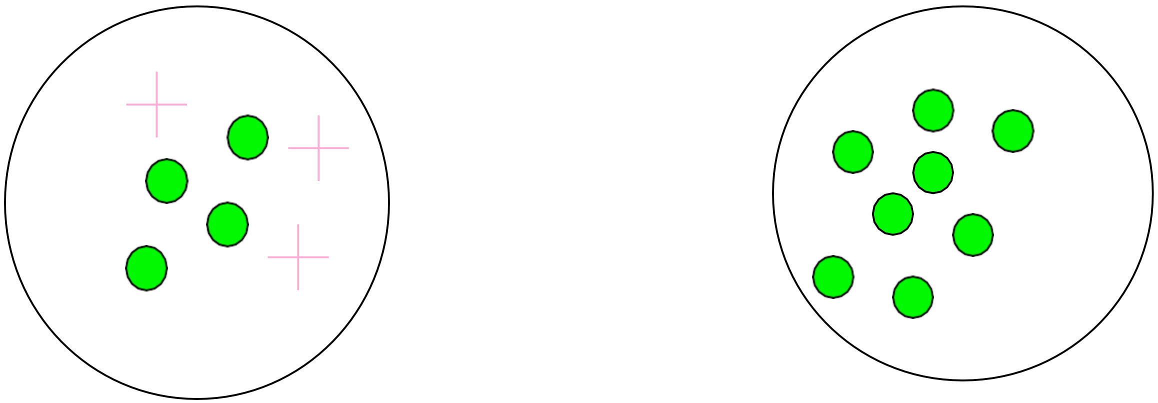


Based on slide by Pedro Domingos

Information Gain

Impurity/Entropy (informal)

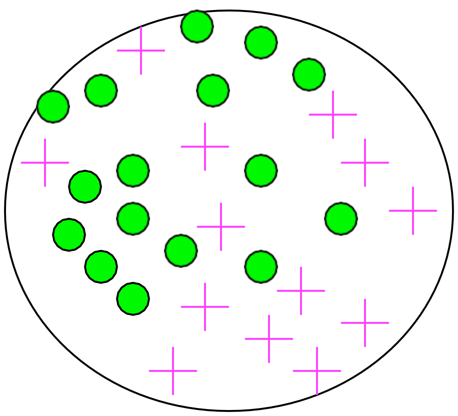
– Measures the level of **impurity** in a group of examples



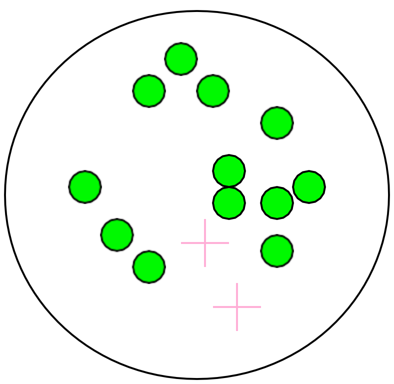
Based on slide by Pedro Domingos

Impurity

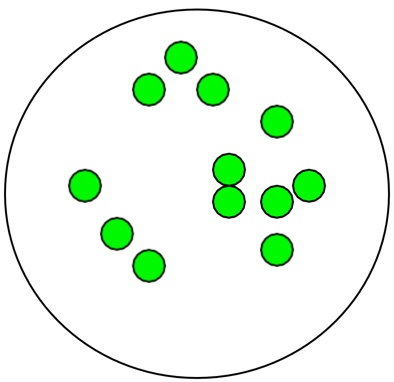
Very impure group



Less impure



Minimum impurity



Based on slide by Pedro Domingos

Entropy: a common way to measure impurity

of possible values for X

- Entropy *H(X)* of a random variable *X*

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

H(X) is the expected number of bits needed to encode a randomly drawn value of *X* (under most efficient code)

Entropy: a common way to measure impurity

of possible values for X

- Entropy $H(X)$ of a random variable X

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)}$$

$H(X)$ is the expected number of bits needed to encode a randomly drawn value of X (under most efficient code)

Why? Information theory:

- Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

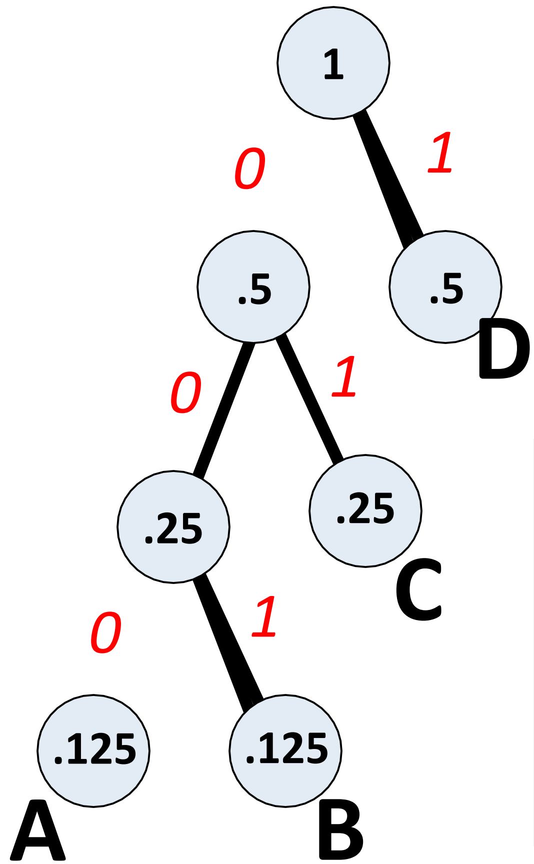
Slide by Tom Mitchell

Example: Huffman code

- In 1952 MIT student David Huffman devised, in the course of doing a homework assignment, an elegant coding scheme which is optimal in the case where all symbols' probabilities are integral powers of 1/2.
- A Huffman code can be built in the following manner:
  - Rank all symbols in order of probability of occurrence
  - Successively combine the two symbols of the lowest probability to form a new composite symbol; eventually we will build a binary tree where each node is the probability of all nodes beneath it
  - Trace a path to each leaf, noticing direction at each node

Huffman code example

M	P
A	.125
B	.125
C	.25
D	.5



M	code	length	prob	
A	000	3	0.125	0.375
B	001	3	0.125	0.375
C	01	2	0.250	0.500
D	1	1	0.500	0.500
average message leng th				1.750

If we use this code to many messages (A,B,C or D) with this probability distribution, then, over time, the average bits/message should approach 1.75

Based on Slide from M. desJardins & T. Finin

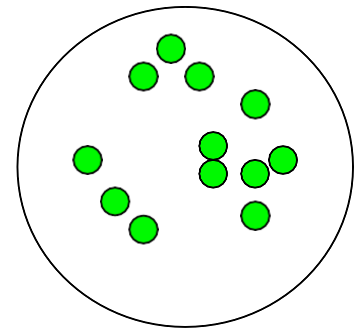
2-Class Cases:

$$\text{Entropy} \quad H(x) = -\sum_{i=1}^{n} P(x=i)\log_2 P(x=i)$$

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = - 1 $\log_2 1$ = 0

not a good training set for learning

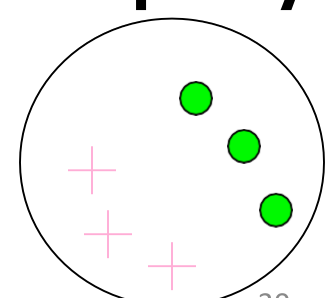
Minimum impurity



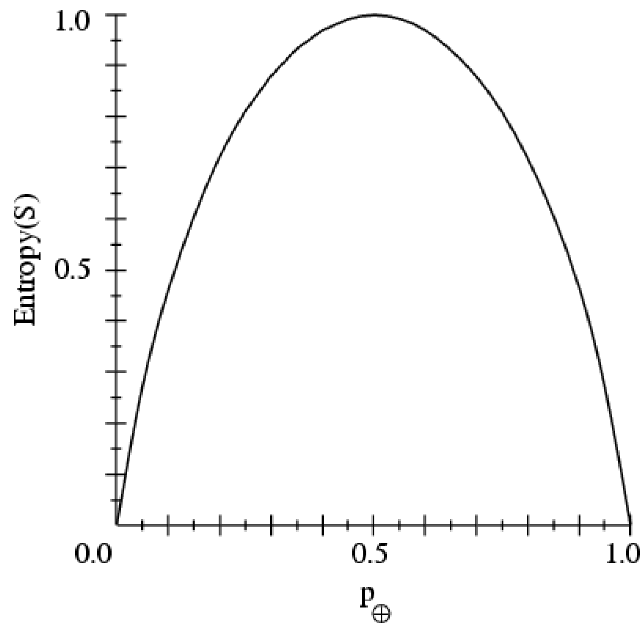
- What is the entropy of a group with 50% in either class?
  - entropy = -0.5 $\log_2 0.5$ – 0.5 $\log_2 0.5$ =1

good training set for learning

Maximum impurity



Sample Entropy



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$H(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Information Gain

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.

- Information gain tells us how important a given attribute of the feature vectors is.

- We will use it to decide the ordering of attributes in the nodes of a decision tree.

From Entropy to Information Gain

Entropy $H(X)$ of a random variable X

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)}$$

From Entropy to Information Gain

Entropy $H(X)$ of a random variable X

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)}$$

Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v) \log_2 P(X=i|Y=v)$$

From Entropy to Information Gain

Entropy *H(X)* of a random variable *X*

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)}$$

Specific conditional entropy *H(X|Y=v)* of *X* given *Y=v* :

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v)\log_2 P(X=i|Y=v)$$

Conditional entropy *H(X|Y)* of *X* given *Y* :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v) H(X|Y=v)$$

Slide by Tom Mitchell

From Entropy to Information Gain

Entropy *H(X)* of a random variable *X*

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)}$$

Specific conditional entropy *H(X|Y=v)* of *X* given *Y=v* :

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v) \log_2 P(X=i|Y=v)$$

Conditional entropy *H(X|Y)* of *X* given *Y* :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v) H(X|Y=v)$$

Mututal information (aka Information Gain) of *X* and *Y* :

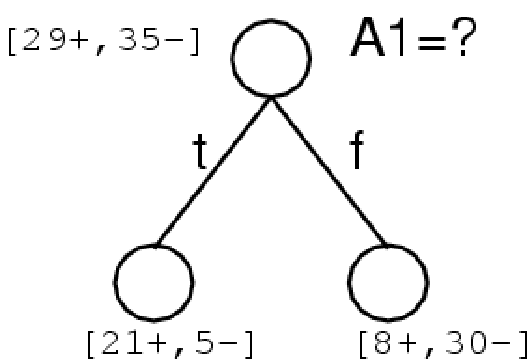
$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

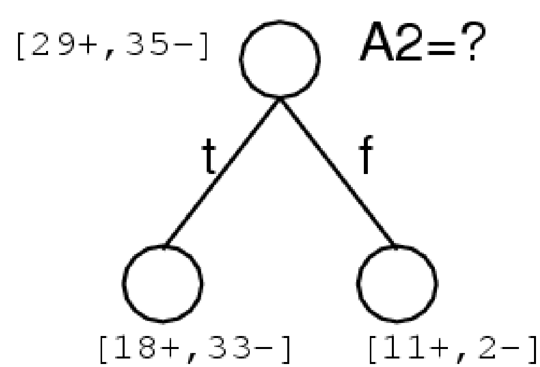
Information Gain

Information Gain is the mutual information between input attribute A and target variable Y

Information Gain is the expected reduction in entropy of target variable Y for data sample S, due to sorting on variable A

$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$



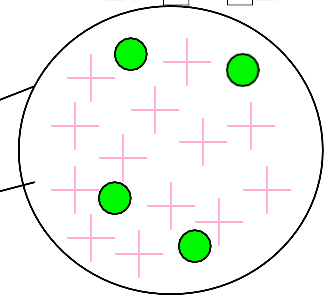


Slide by Tom Mitchell

Calculating Information Gain

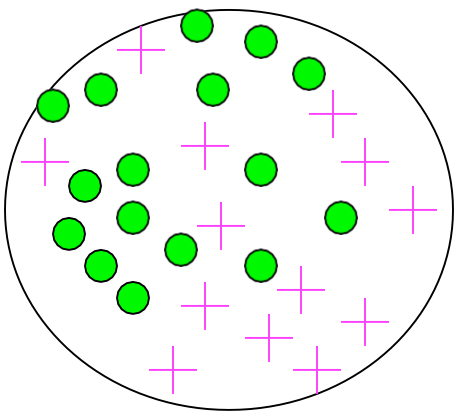
Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

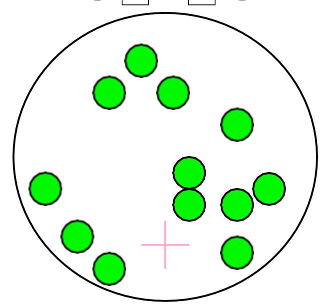


17 instances

Entire population (30 instances)



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$



13 instances

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

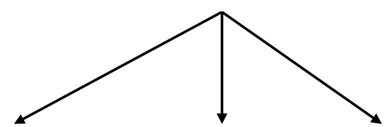
(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

Information Gain= 0.996 - 0.615 = 0.38

Entropy-Based Automatic Decision Tree Construction

Training Set X
x1=(f11,f12,…f1m)
x2=(f21,f22, f2m)
.
.
xn=(fn1,f22, f2m)

Node 1
What feature should be used?



What values?

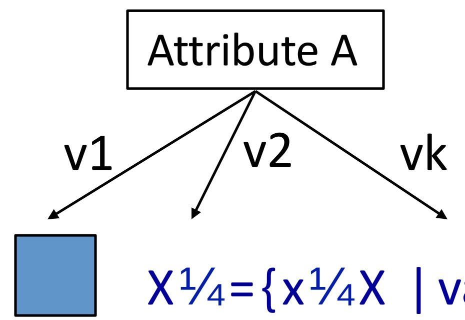
Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.

Based on slide by Pedro Domingos

Using Information Gain to Construct a Decision Tree

Full Training Set X

Choose the attribute A with highest information gain for the full training set at the root of the tree.



Construct child nodes for each value of A. Each has an associated subset of vectors in which A has a particular value.

Set X¼

X¼={x¼X | value(A)=v1}

repeat recursively till when?

Disadvantage of information gain:

- It prefers attributes with large number of values that split the data into small, pure subsets
- Quinlan's gain ratio uses normalization to improve this

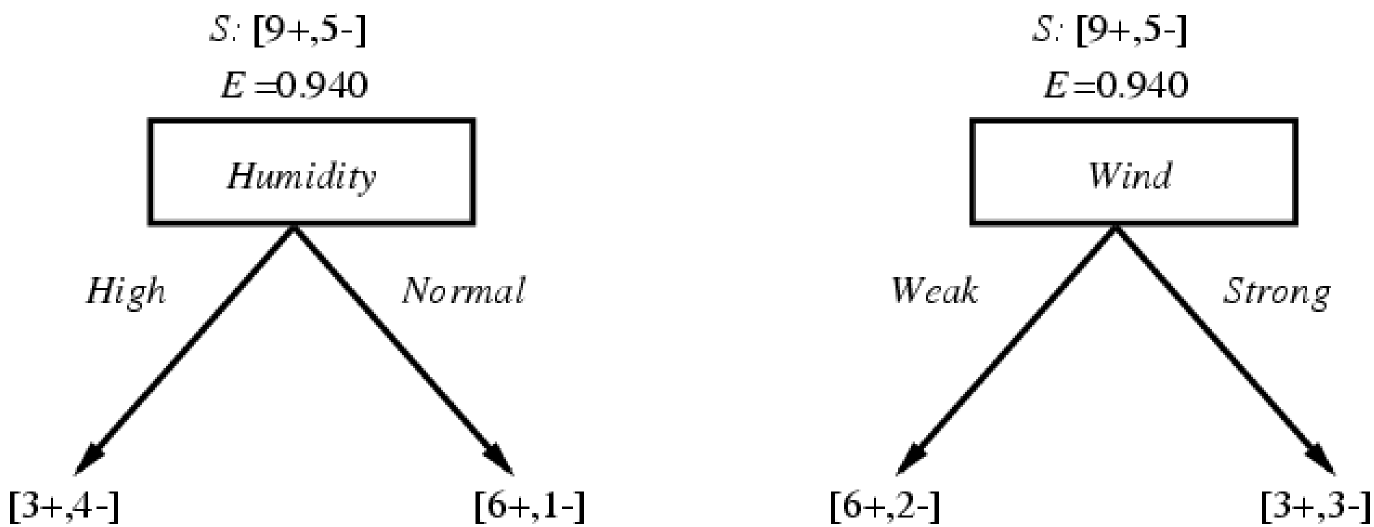
Based on slide by Pedro Domingos

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenı
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

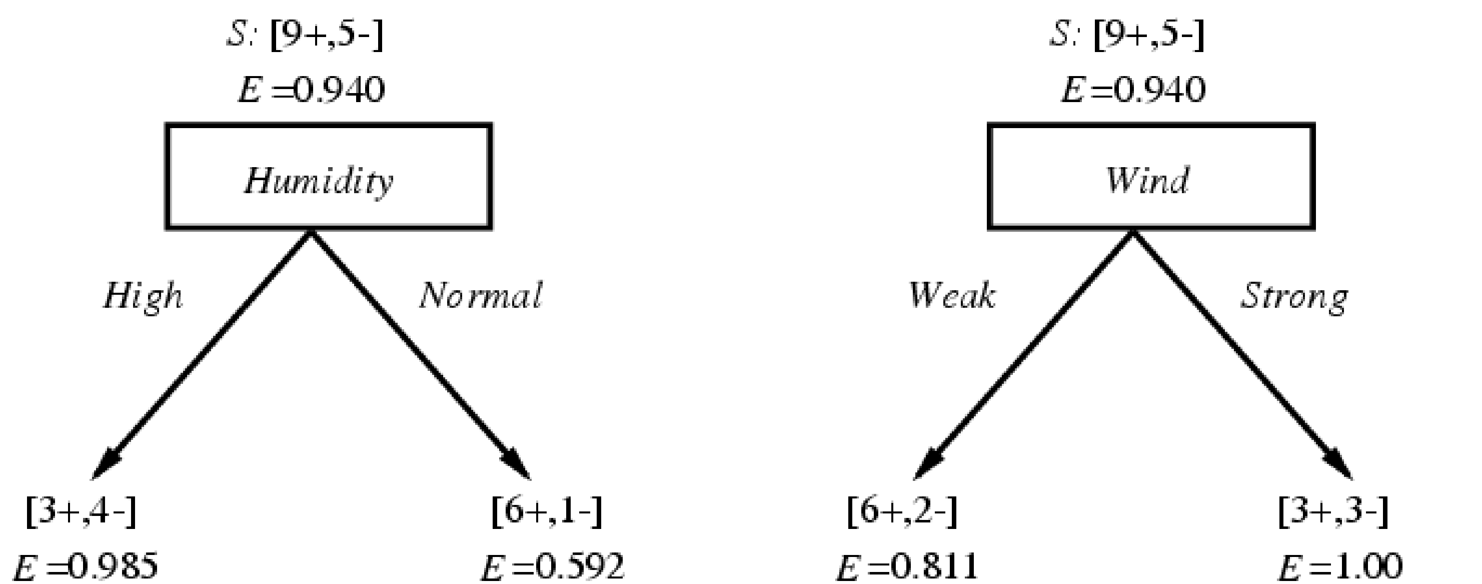
Selecting the Next Attribute

Which attribute is the best classifier?



Selecting the Next Attribute

Which attribute is the best classifier?

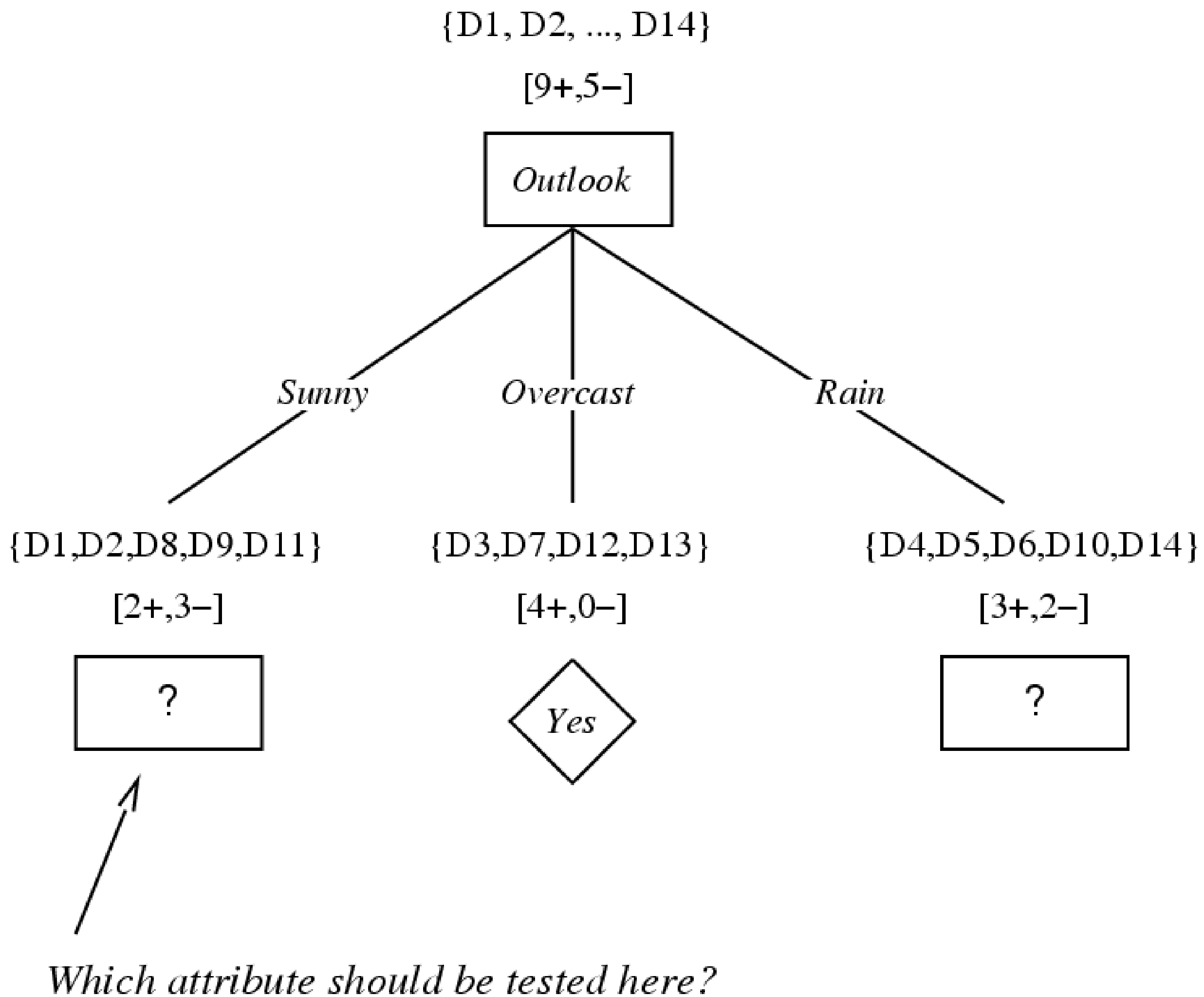


Gain (S, Humidity)

$= .940 - (7/14).985 - (7/14).592$
$= .151$

Gain (S, Wind)

$= .940 - (8/14).811 - (6/14)1.0$
$= .048$



S_{sunny} = {D1,D2,D8,D9,D11}

$Gain\,(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

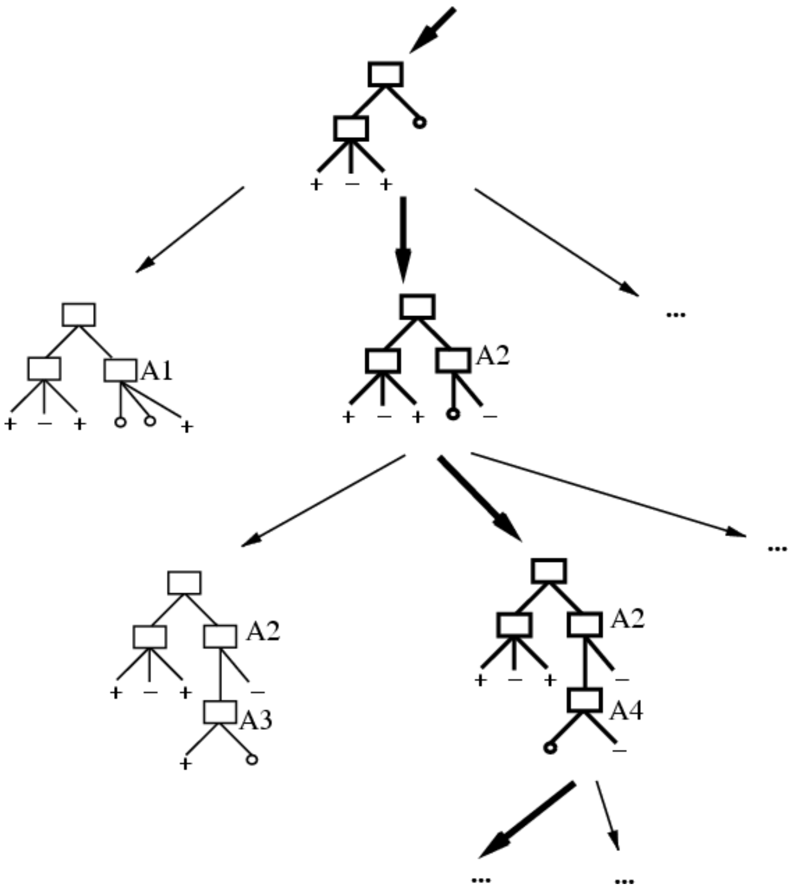
$Gain\,(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain\,(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Decision Tree Applet

http://webdocs.cs.ualberta.ca/~aixplore/learning/DecisionTrees/Applet/DecisionTreeApplet.html

Which Tree Should We Output?



- ID3 performs heuristic search through space of decision trees
- It stops at smallest acceptable tree. Why?

Occam's razor: prefer the simplest hypothesis that fits the data

The ID3 algorithm builds a decision tree, given a set of non-categorical attributes C1, C2, .., Cn, the class attribute C, and a training set T of records

```
function ID3(R:input attributes, C:class attribute,
S:training set) returns decision tree;
   If S is empty, return single node with value Failure;
   If every example in S has same value for C, return
   single node with that value;
   If R is empty, then return a single node with most
   frequent of the values of C found in examples S;
   # causes errors -- improperly classified record
   Let D be attribute with largest Gain(D,S) among R;
   Let {dj| j=1,2, .., m} be values of attribute D;
   Let {Sj| j=1,2, .., m} be subsets of S consisting of
             records with value dj for attribute D;
   Return tree with root labeled D and arcs labeled
     d1..dm going to the trees ID3(R-{D},C,S1). . .
     ID3(R-{D},C,Sm);
```

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