Decision Tree Learning

Lecture Slides for textbook

Machine Learning

T. Mitchell, Mc. Graw Hill

Outline

- Decision Tree Representation
- ID3 learning algorithm
- Entropy, Information Gain
- Overfitting

Machine Learning

Study of algorithms that:

- improve their **performance P**
- at some **task T**
- with **experience E**

Well-defined learning task: <P, T, E>

Function Approximation

Problem Setting:

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

superscript: ith training example

Input:

- Training examples $\{<x^{(i)},y^{(i)}>\}$ of unknown target function f

Output:

- Hypothesis $h \in H$ that best approximates target function f

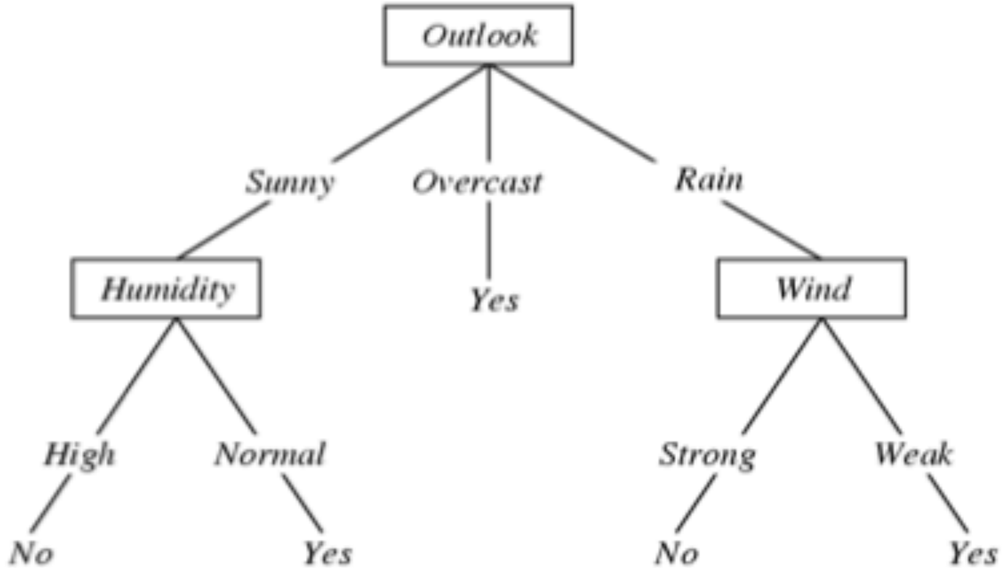
Sample Dataset

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

$\langle \boldsymbol{x}_i, y_i \rangle$

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

A Decision tree for
F: <Outlook, Humidity, Wind, Temp> → PlayTennis?



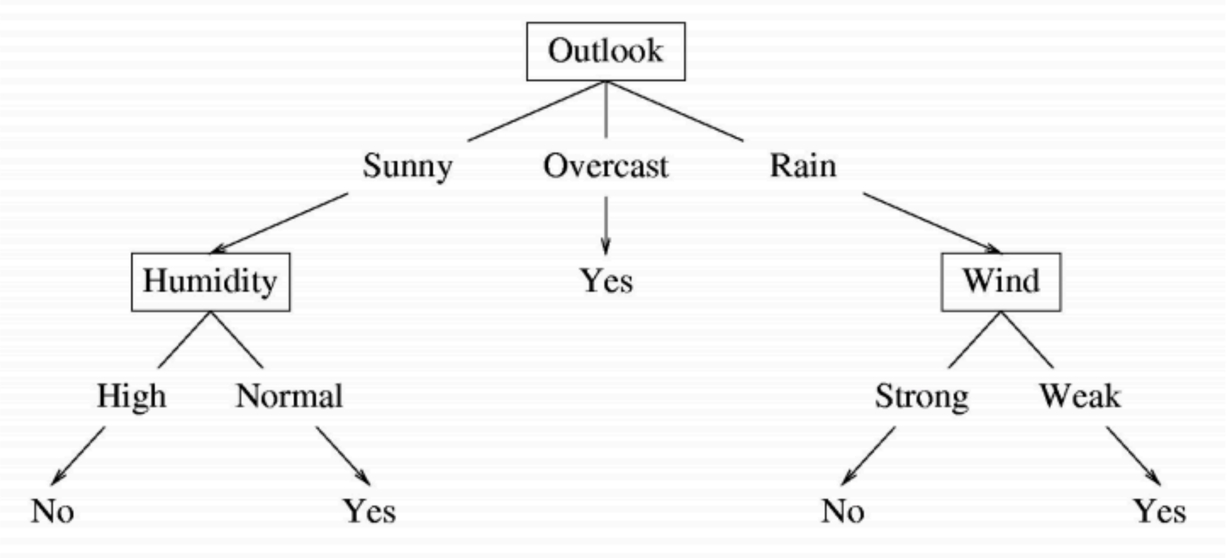
Each internal node: test one attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or $P(Y|X \in \text{leaf})$)

Decision Tree

- A possible decision tree for the data:

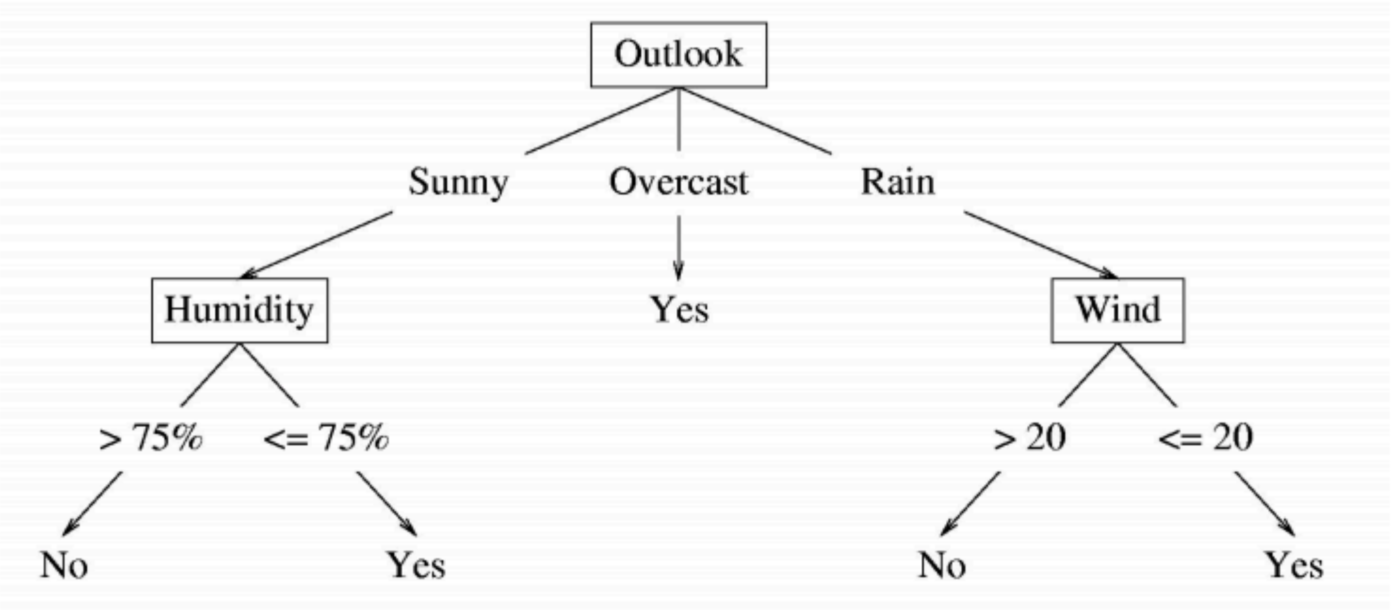


- What prediction would we make for

  <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 $H = \{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y

Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector $x = < x_1, x_2 \ldots x_n>$
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree

Input:

- Training examples $\{<x^{(i)},y^{(i)}>\}$ of unknown target function f

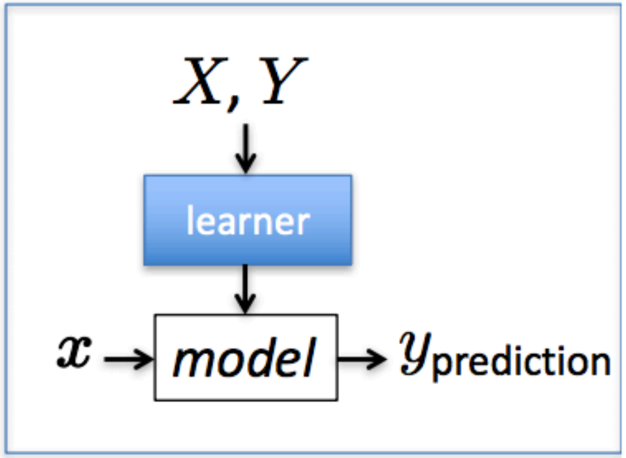
Output:

- Hypothesis $h \in H$ that best approximates target function f

Stages of (Batch) Machine Learning

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

- Assumes each $\boldsymbol{x}_i \sim \mathcal{D}(\mathcal{X})$ 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 $\boldsymbol{x} \sim \mathcal{D}(\mathcal{X})$

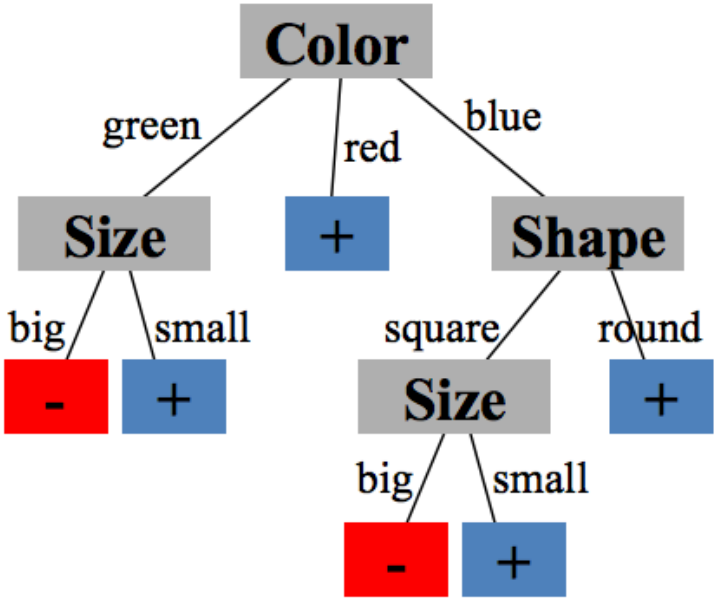
$y_{\text{prediction}}$ ← *model*.predict($\boldsymbol{x}$)

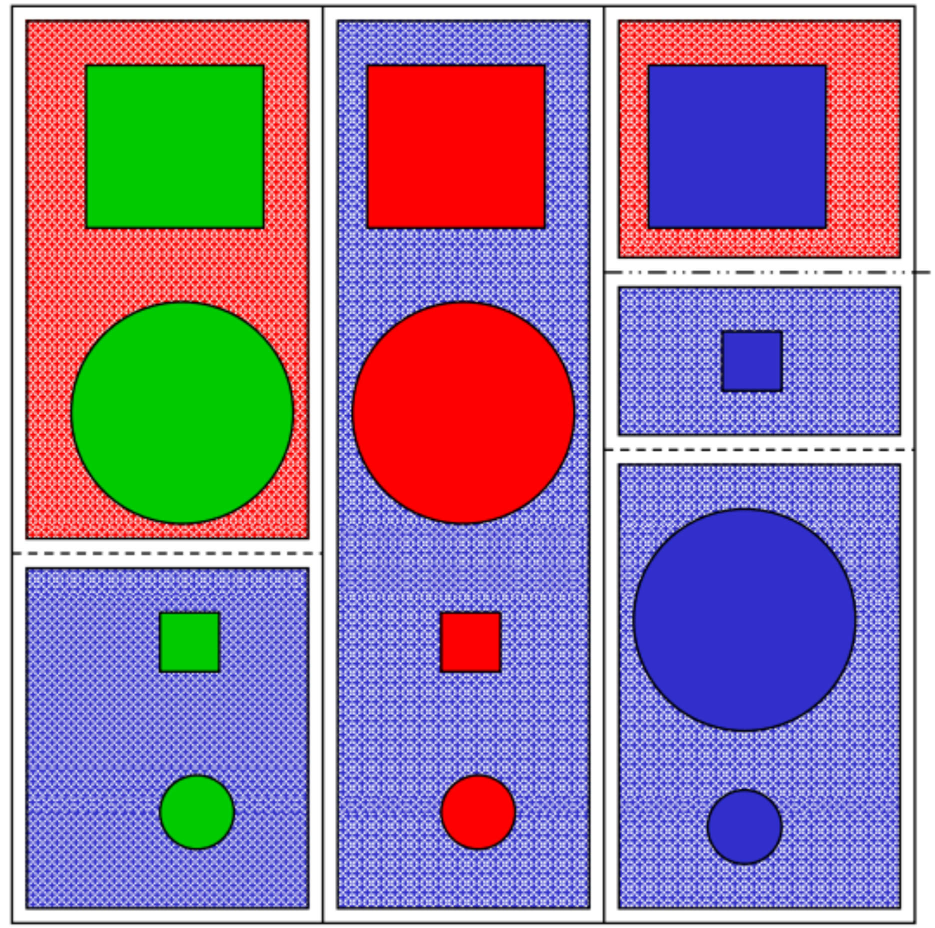
A Tree to Predict C-Section Risk

- 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+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | 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-
```

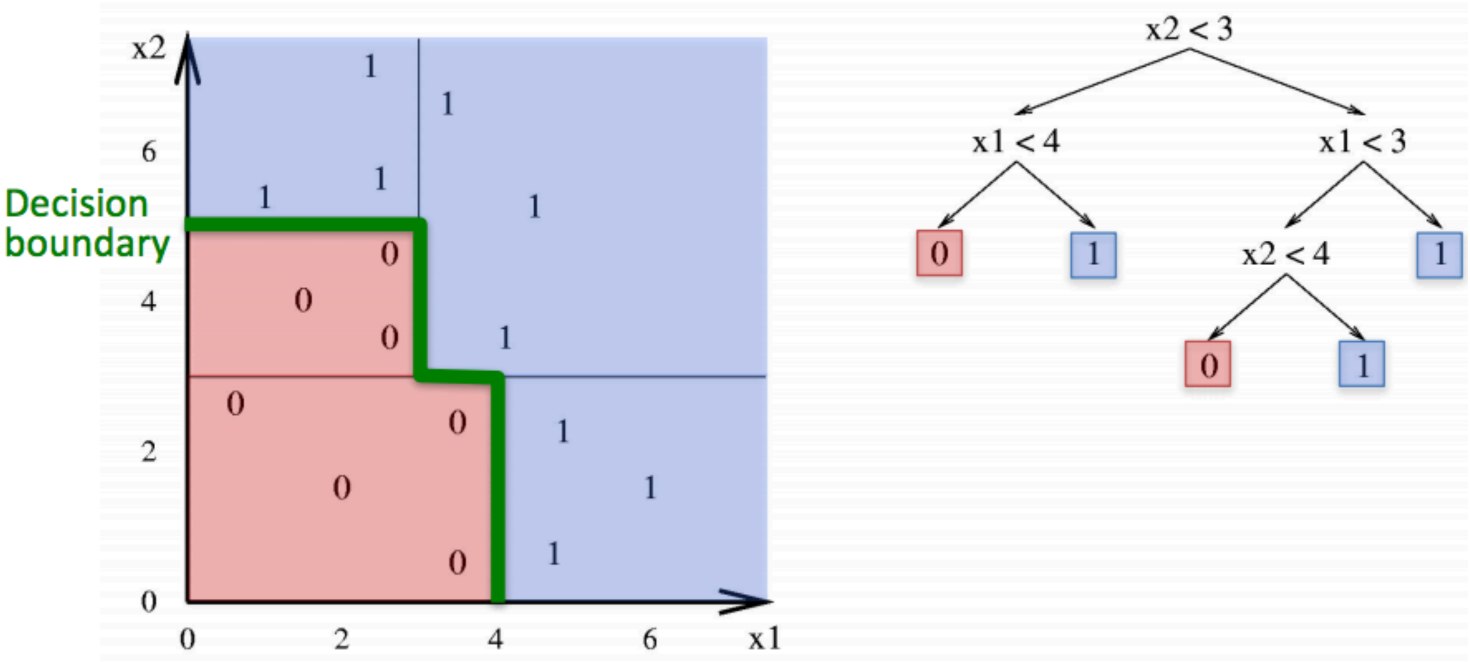
Decision Tree Induced Partition





Decision Tree Induced Partition

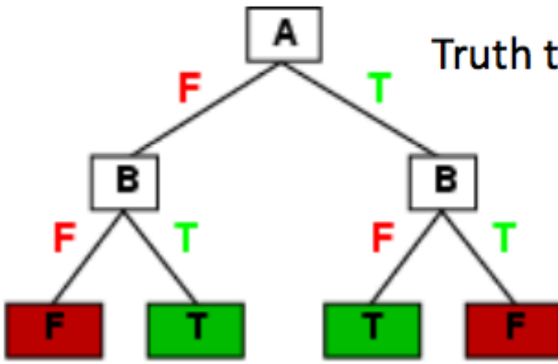
- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels



Decision Tree Induced Partition

- 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



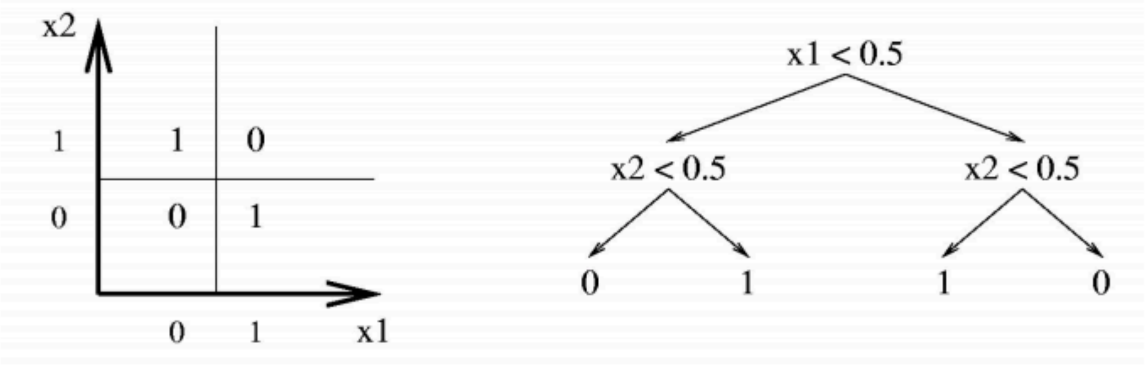
Truth table row → path to leaf

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

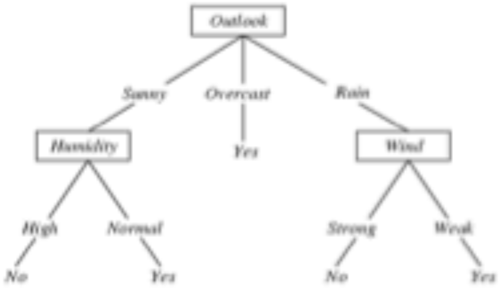
Expressiveness

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.



Decision Trees

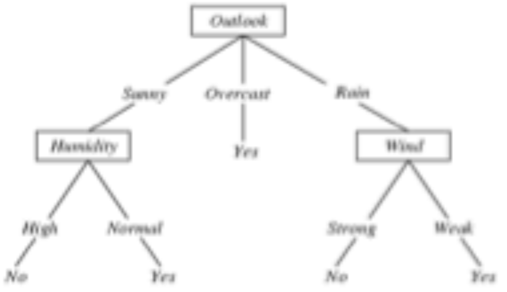


Suppose $X = <X_1, \ldots X_n>$
where X_i are boolean variables

How would you represent $Y = X_2 X_5$? $Y = X_2 \vee X_5$

How would you represent $X_2 X_5 \vee X_3 X_4 (\neg X_1)$

Decision Trees

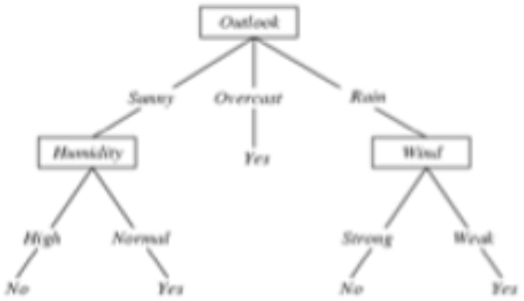


Suppose $X = \langle X_1, \ldots X_n \rangle$
where X_i are boolean variables

How would you represent $Y = X_2 X_5$? $Y = X_2 \vee X_5$

How would you represent $X_2 X_5 \vee X_3 X_4 (\neg X_1)$

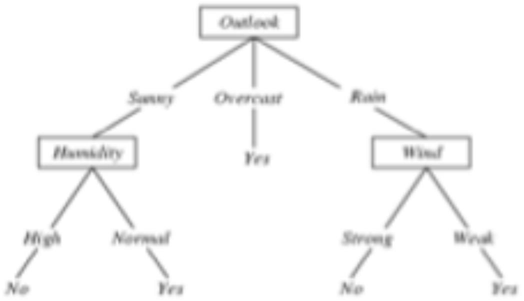
Decision Trees



Suppose $X = \langle X_1, \ldots X_n \rangle$
where X_i are boolean variables

How would you represent $Y = X_2 X_5$? $Y = X_2 \vee X_5$

Decision Trees



Suppose $X = \langle X_1, \ldots X_n \rangle$
where X_i are boolean variables

How would you represent $X_2 X_5 \vee X_3 X_4(\neg X_1)$

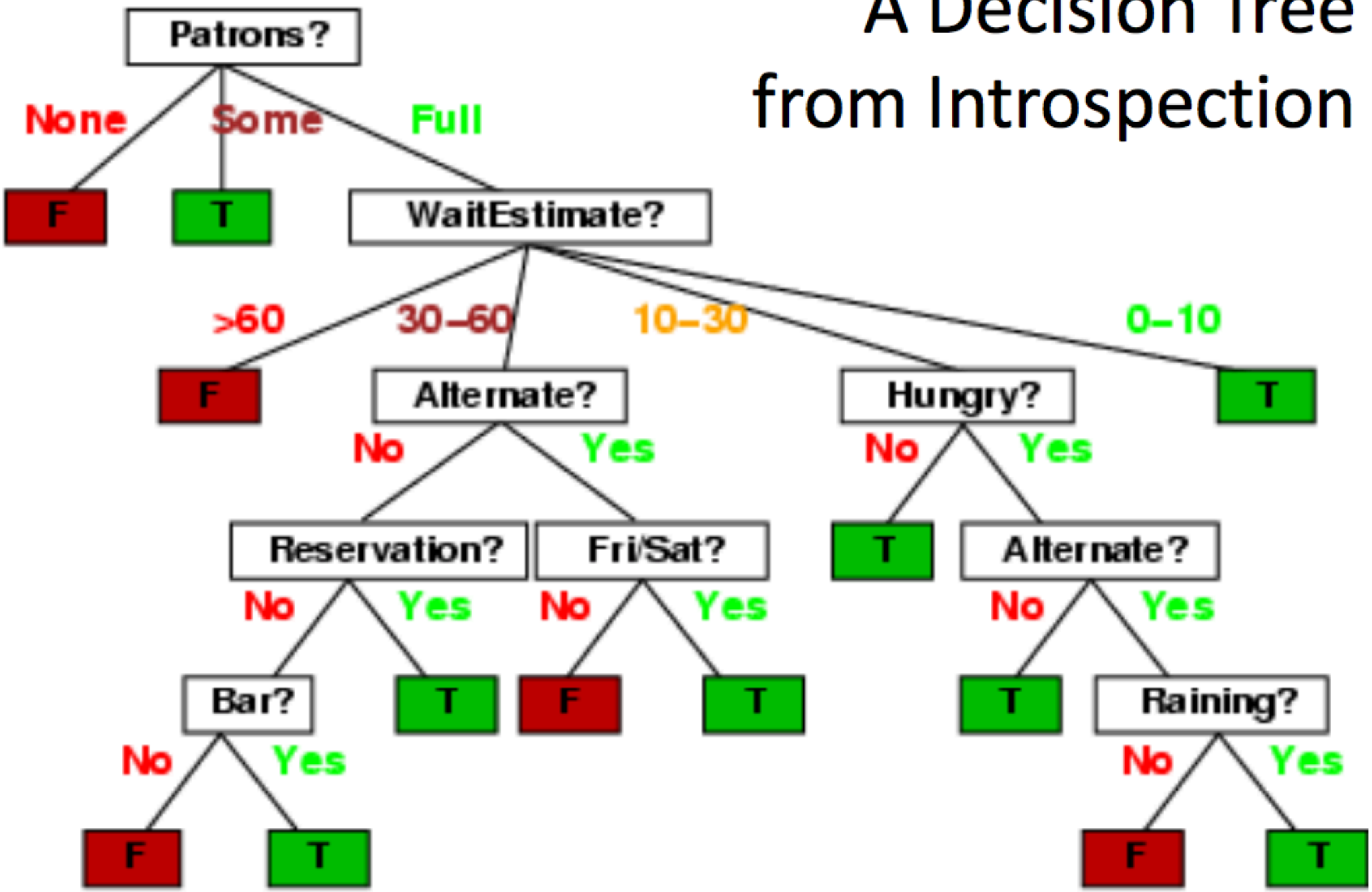
Another Example: Restaurant Domain (Russell & Norvig)

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?

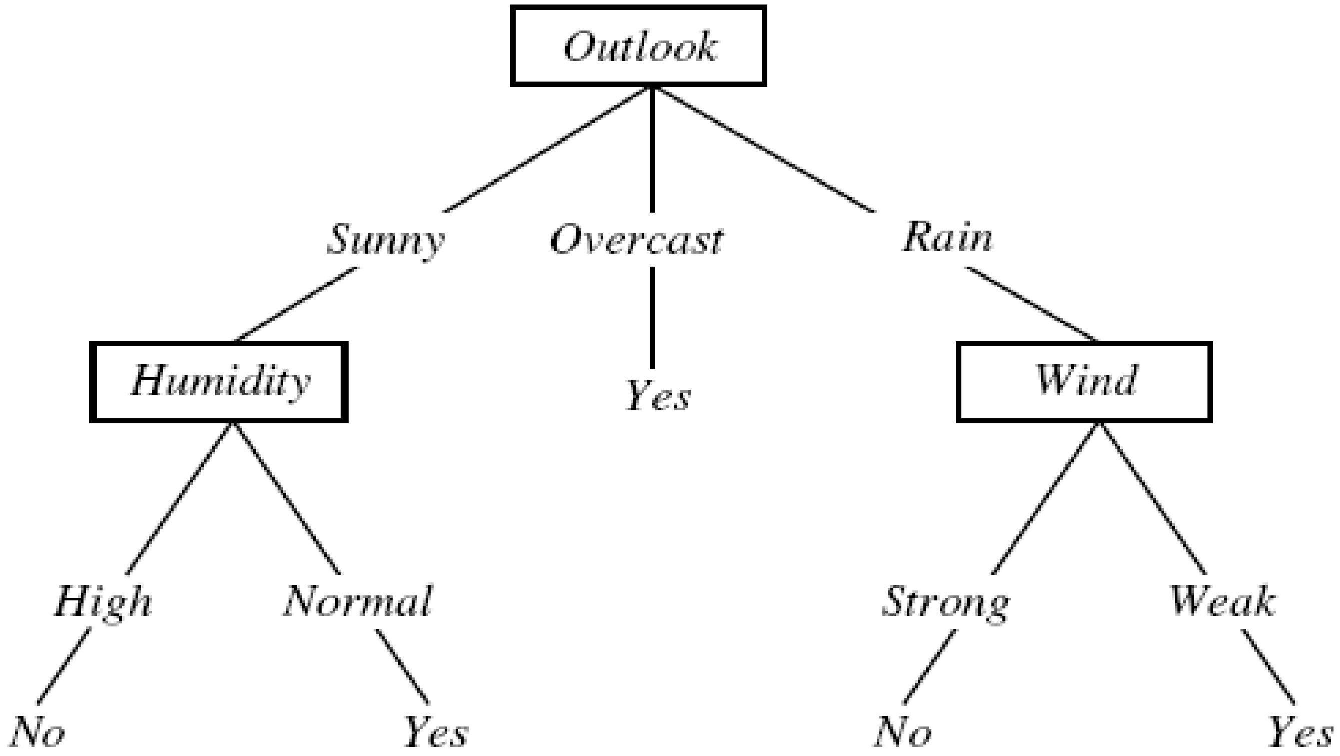
Preference bias: Ockham's Razor

- 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

Decision Tree for *PlayTennis*



When to Consider Decision Trees

- Instances can be described by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data, or data with missing values

Example

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Top-Down Induction of Decision Trees

Main Loop:

1. A $\leftarrow$ the "best" decision attribute for next node
2. Assign A as decision attribute for node
3. **For** each value of A, create new descendant of node
4. Sort training examples to leaf nodes
5. **If** training examples are perfectly classified, **Then** STOP, **Else** iterate over new leaf nodes

Which attribute is best?

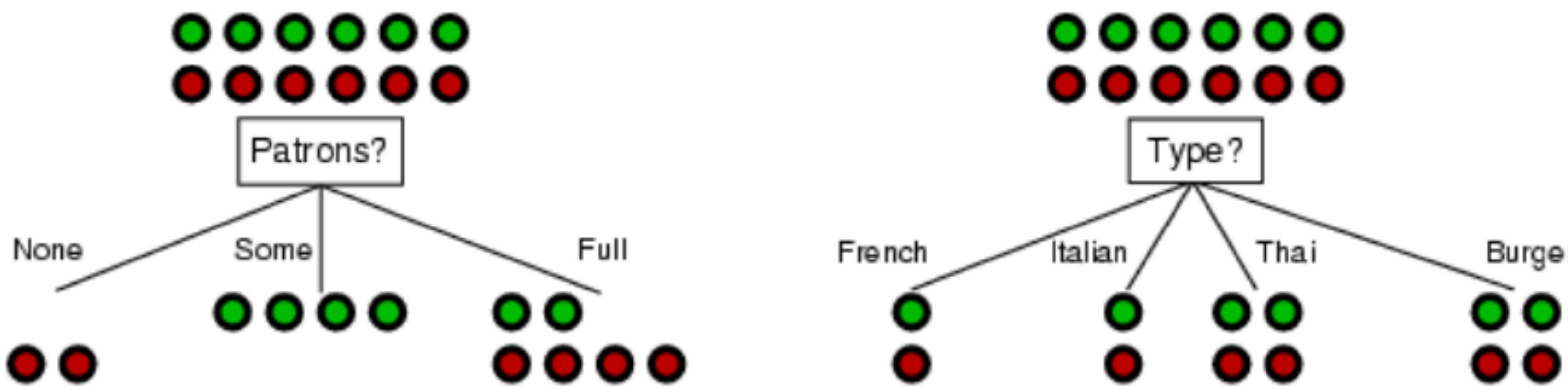
Choosing the Best Attribute

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

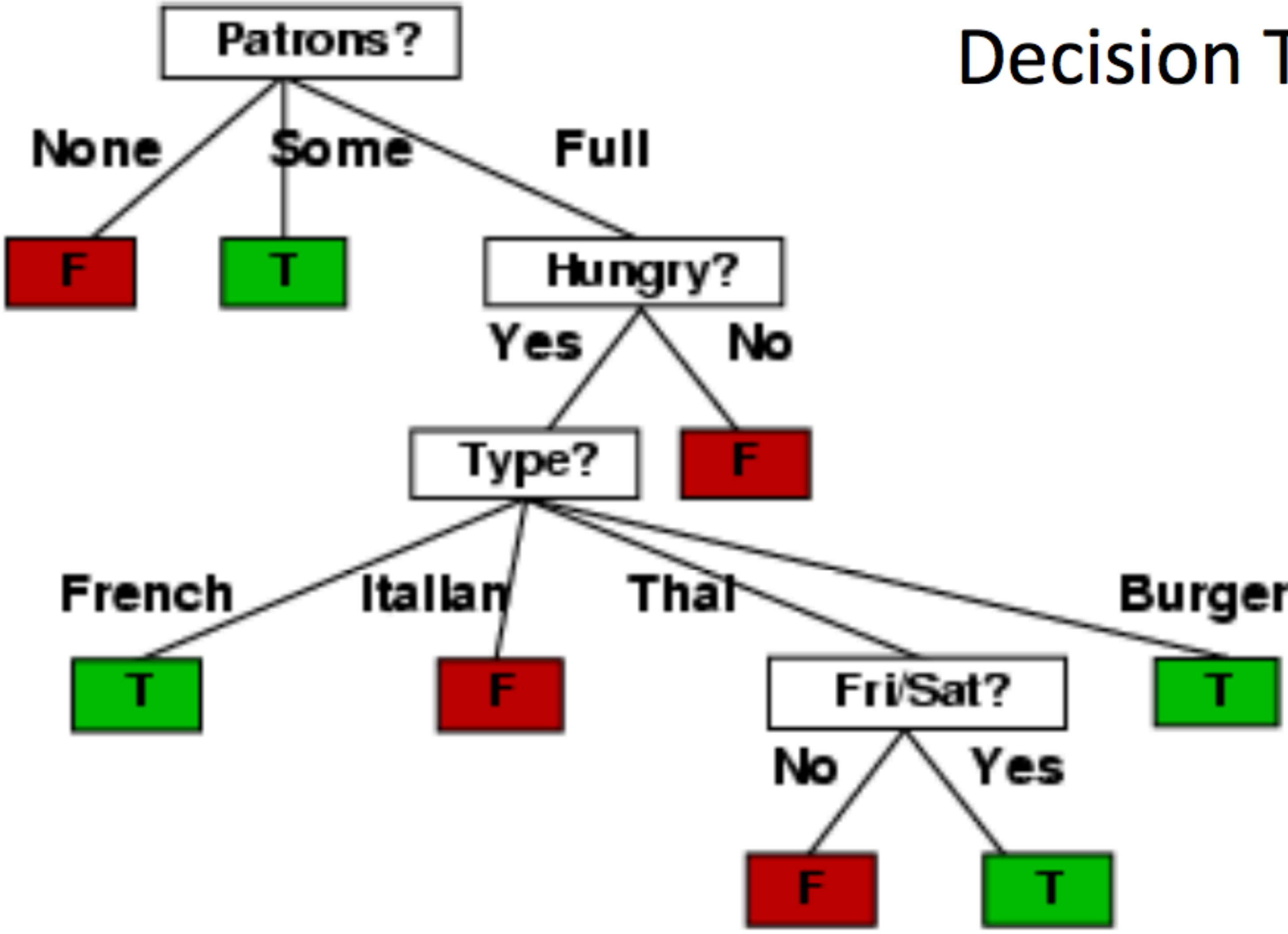
Choosing an Attribute

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

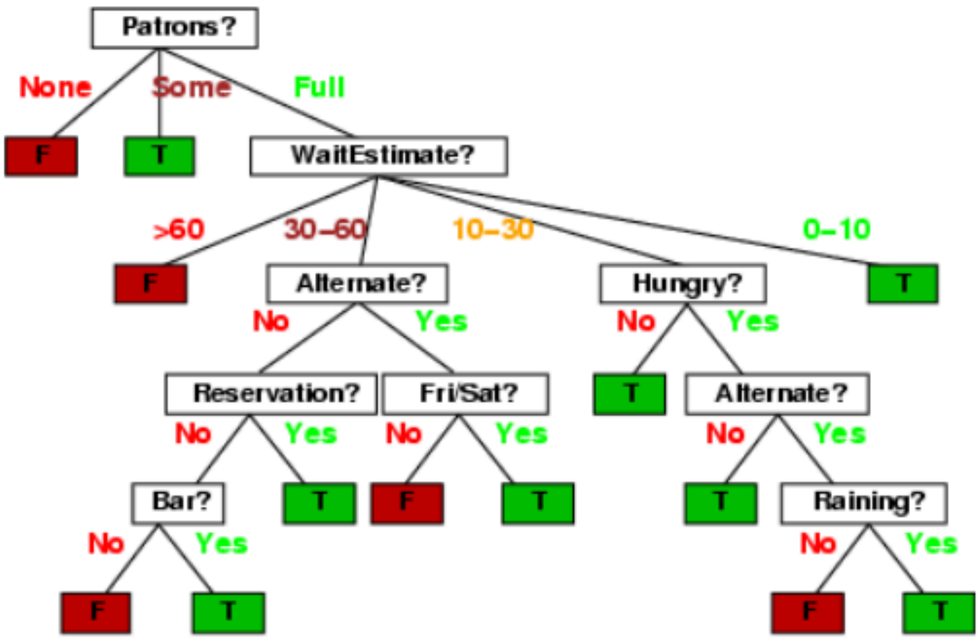


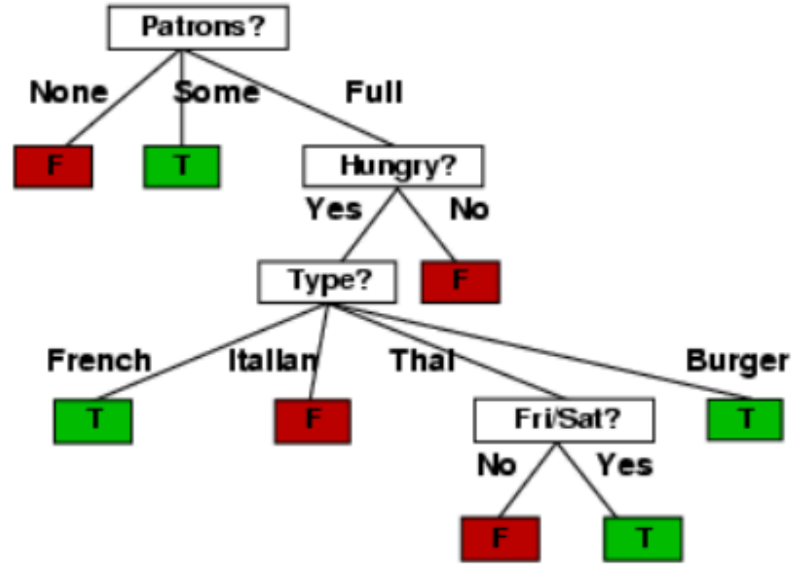
Which split is more informative: *Patrons?* or *Type?*

ID3-induced Decision Tree



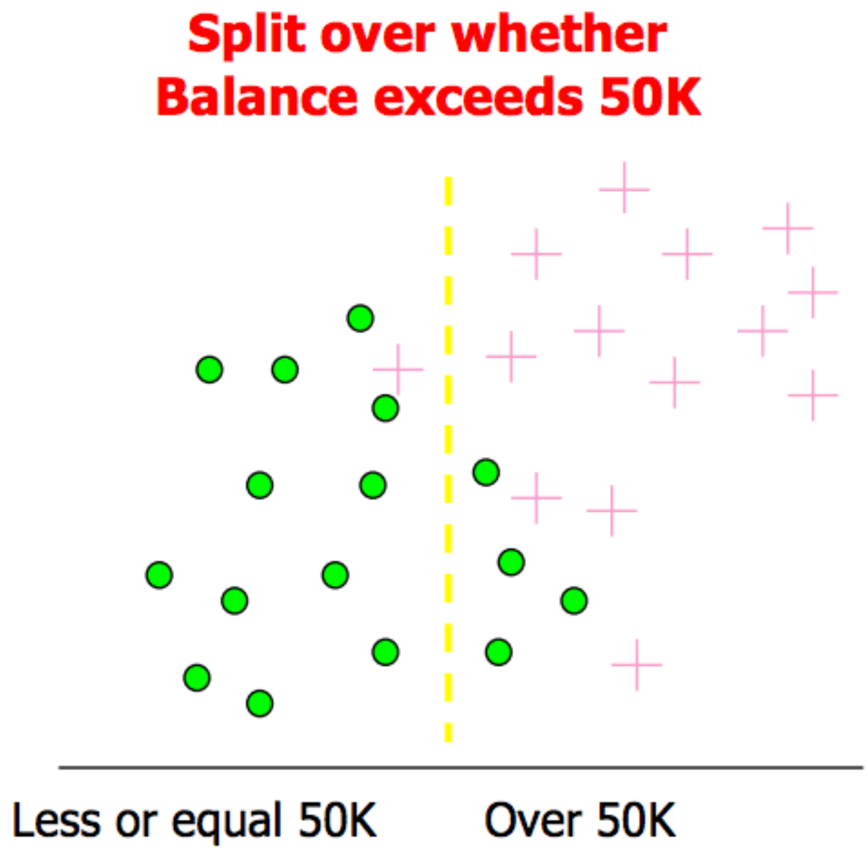
Compare the Two Decision Trees

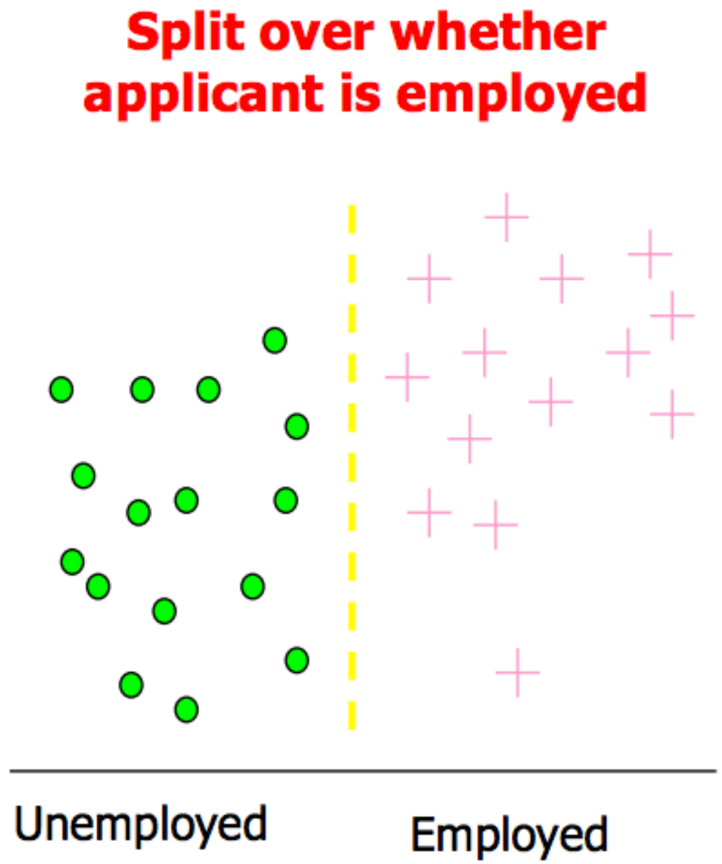




Information Gain

Which test is more informative?

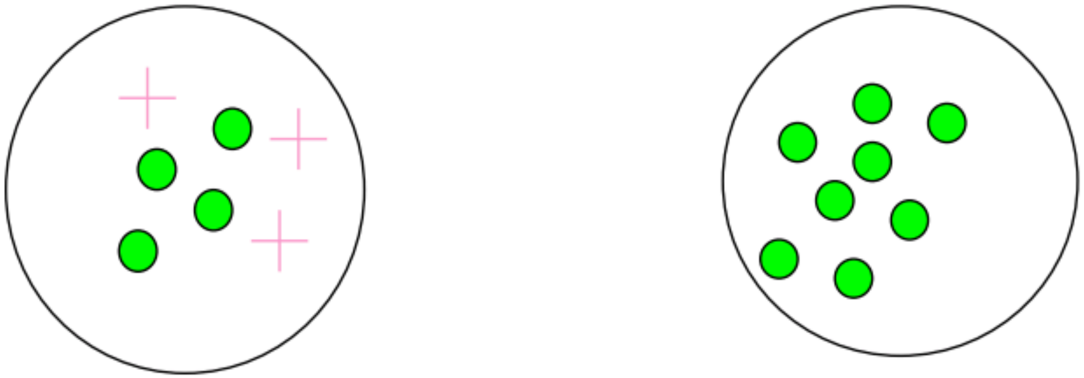




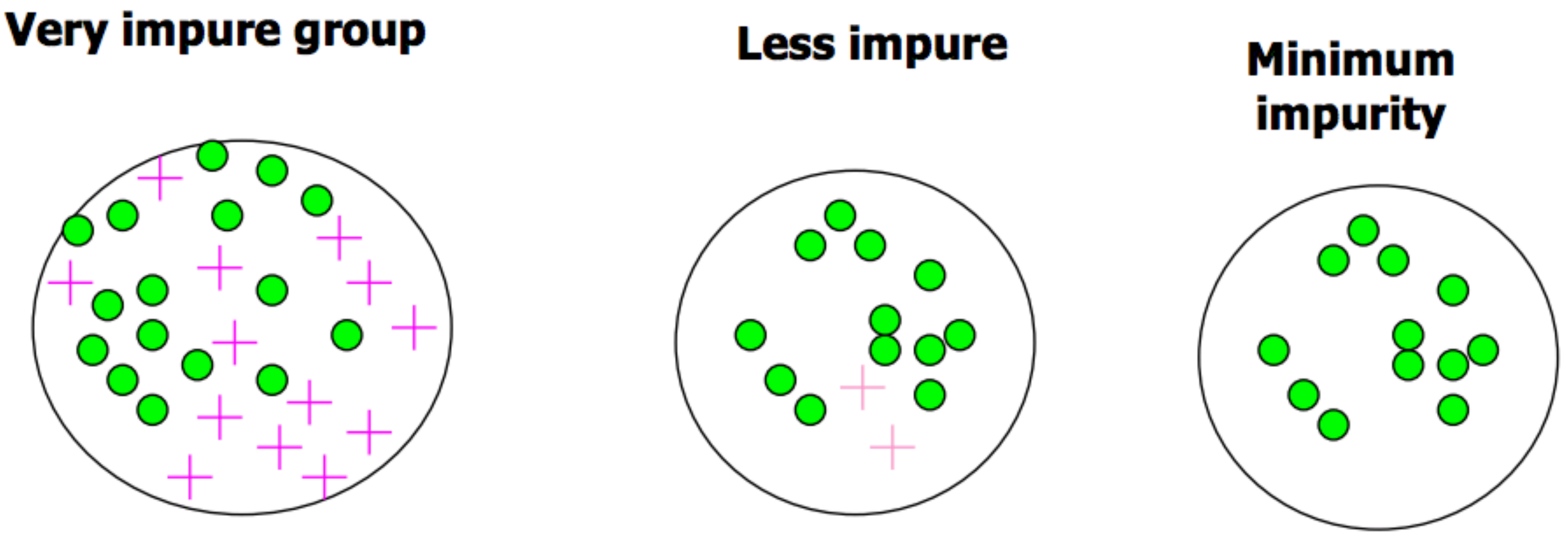
Information Gain

Impurity/Entropy (informal)

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

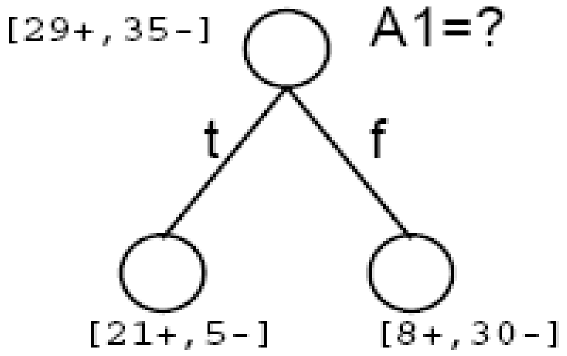


Impurity



Top-Down Induction of Decision Trees

Which attribute is best?



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

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

$H(X)$ 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:

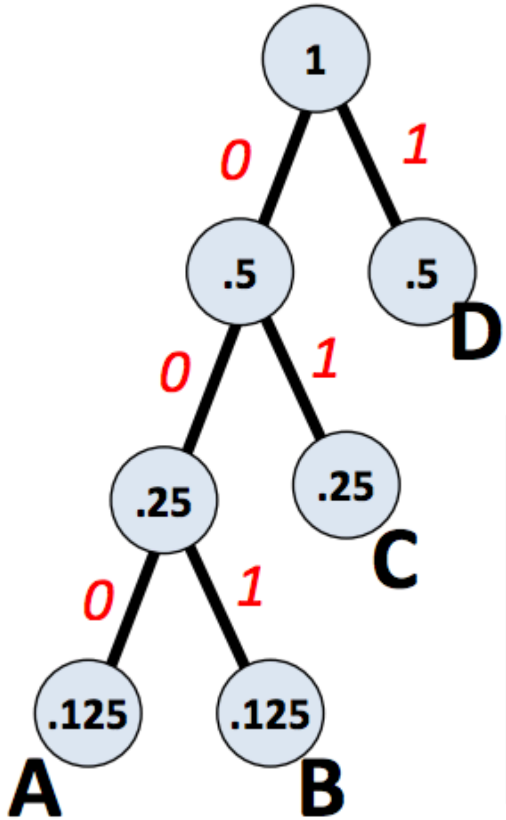
$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

Example: Huffman Code

- 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 length				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

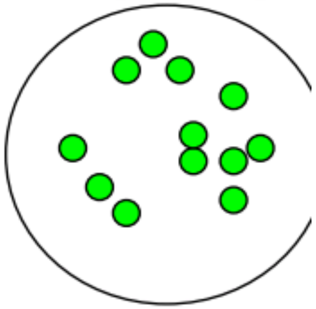
2-Class Cases:

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

- 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

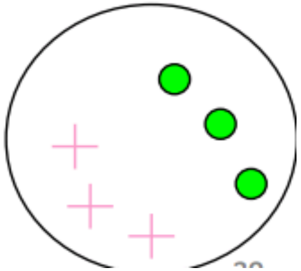
Minimum impurity



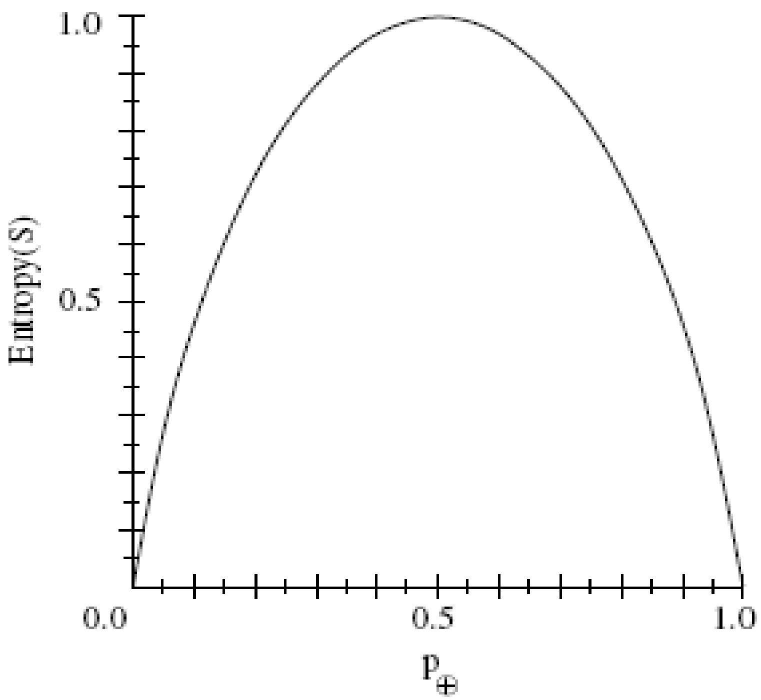
- What is the entropy of a group with 50% in either class?
  - entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

  good training set for learning

Maximum impurity



Entropy



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.

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

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

From Entropy to Information Gain

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

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

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*

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

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* :

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

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* :

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

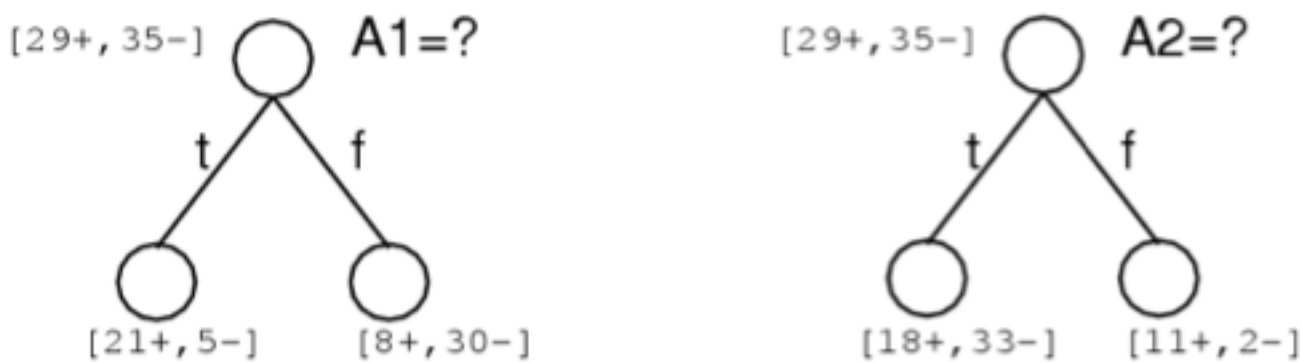
Mututal information (aka Information Gain) of *X* and *Y* :

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

Information Gain

Information Gain is the mutual information between attribute A and Target Variable Y.

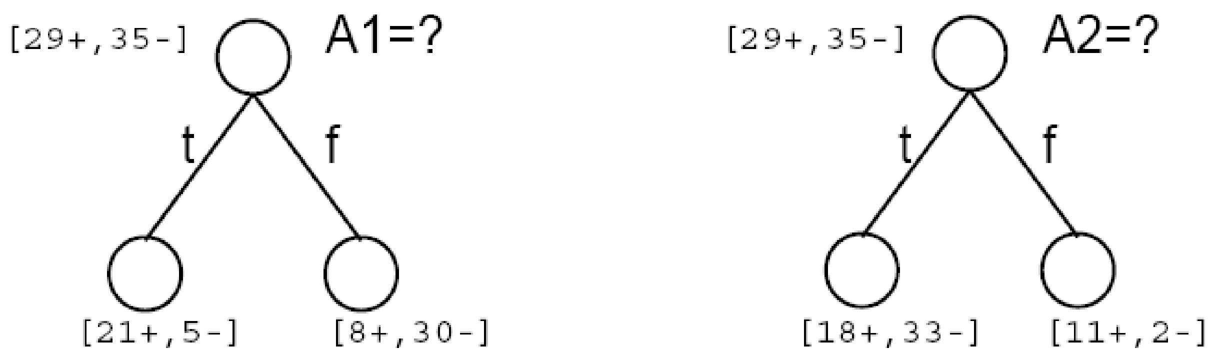
$$Gain(S, A) = I_S(A, Y) = H_S(Y) - H_S(Y|A)$$



Information Gain

Gain (S, A) = expected reduction in entropy of target variable Y for data sample S due to sorting on attribute A.

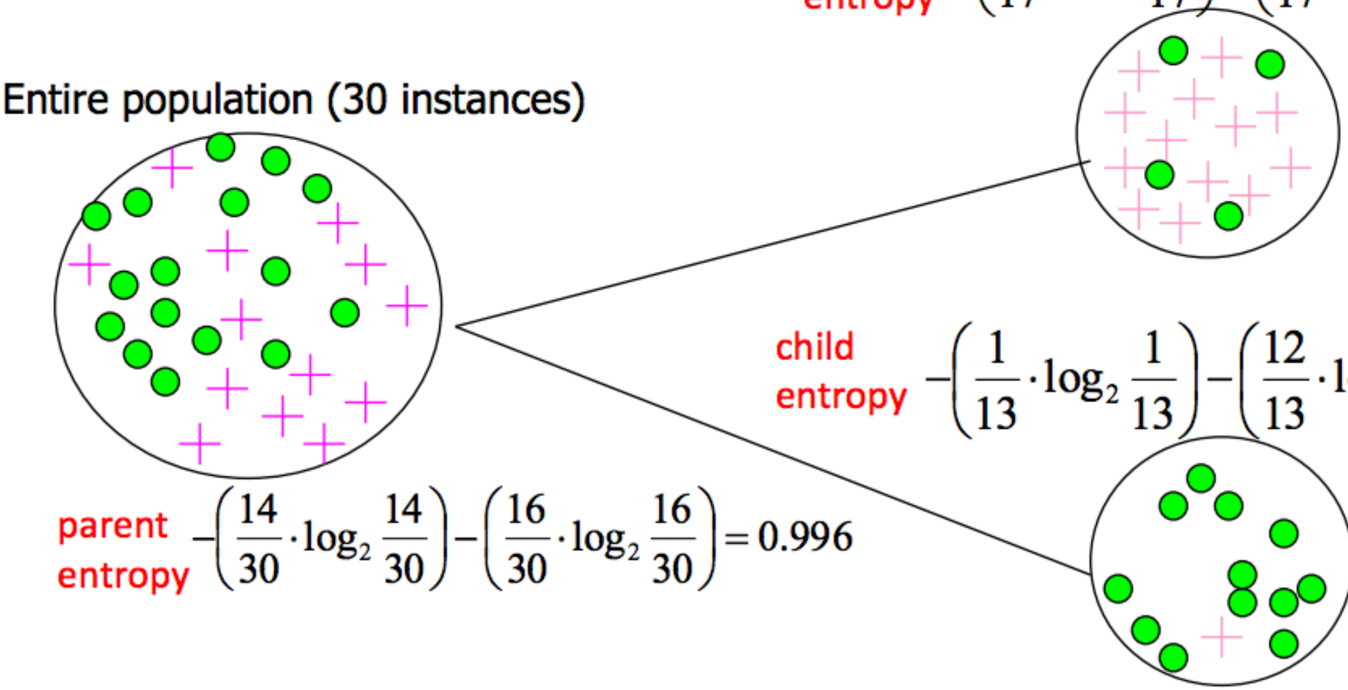
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



Calculating Information Gain

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

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



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

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?

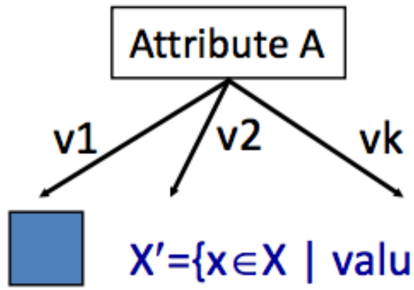
Quinlan suggested information gain in his ID3 system and later the gain ratio, both based on entropy.

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 X ′

$X' = \{x \in X \mid \text{value}(A) = v1\}$

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

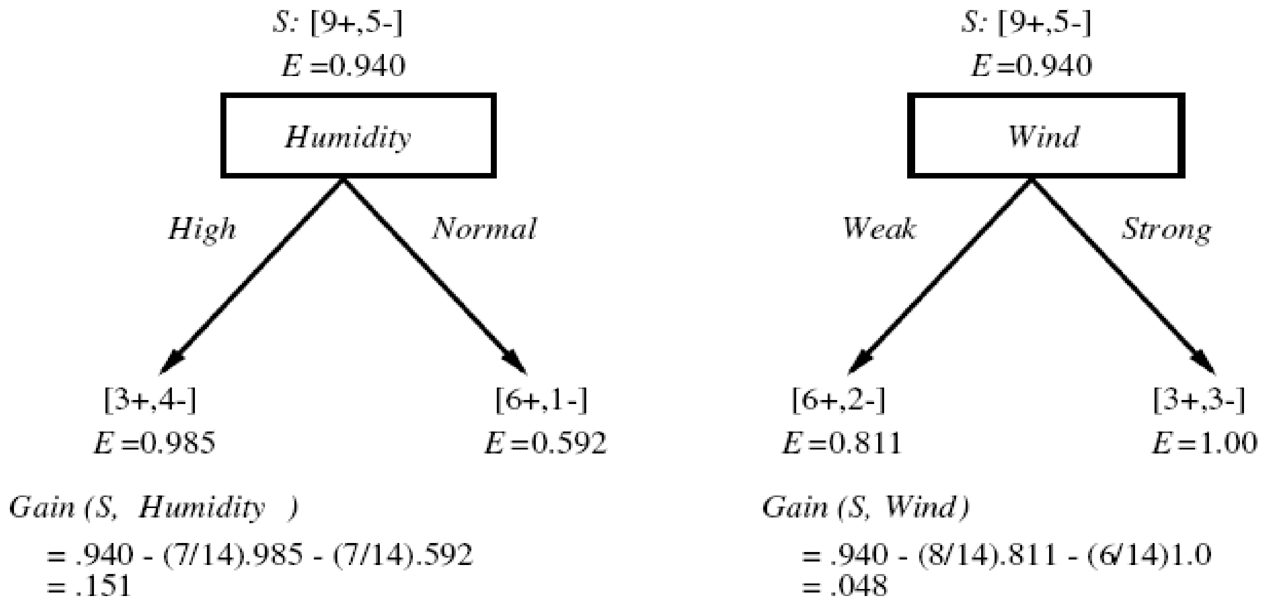
Based on slide by Pedro Domingos

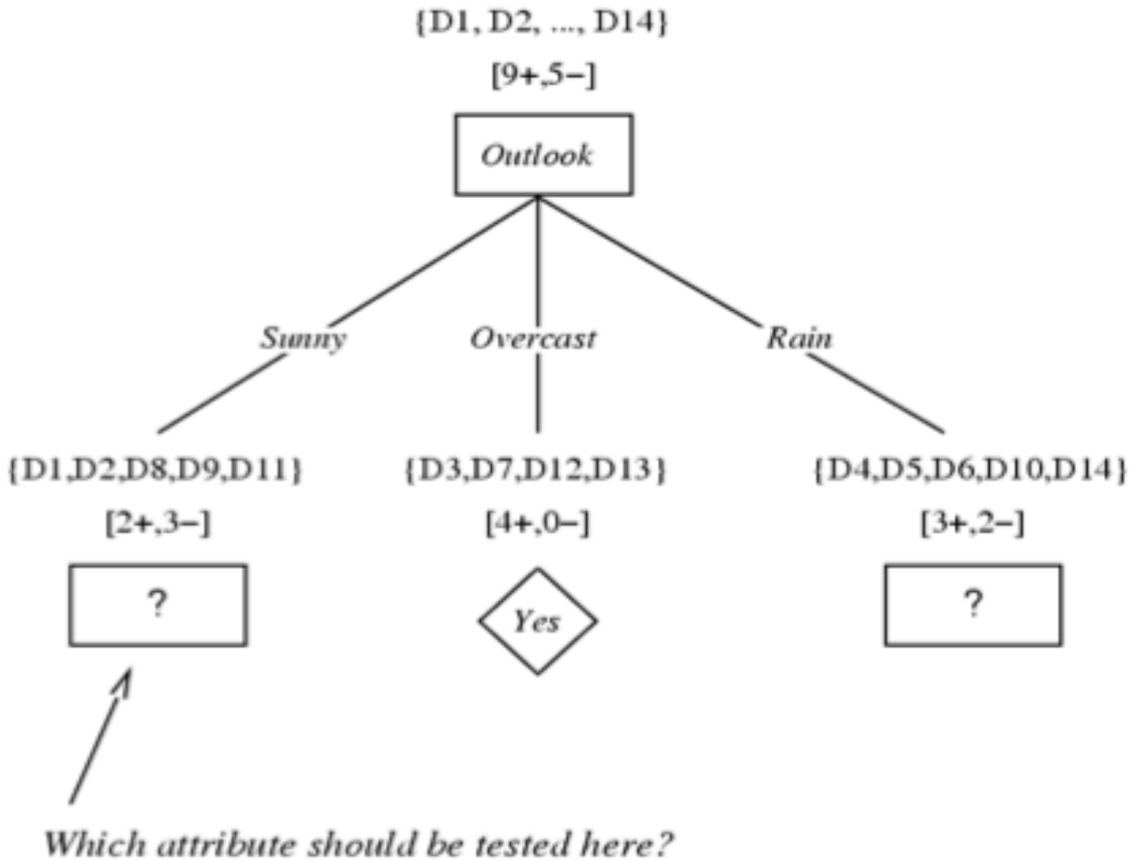
Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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?





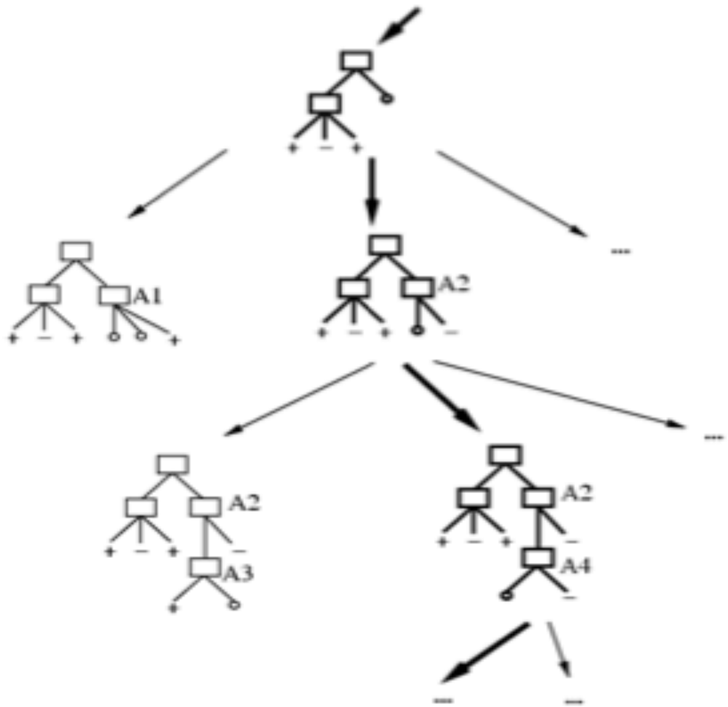
S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

Function Approximation as Search for the Best Hyphotheses



- 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

Hypothesis Space Search by ID3

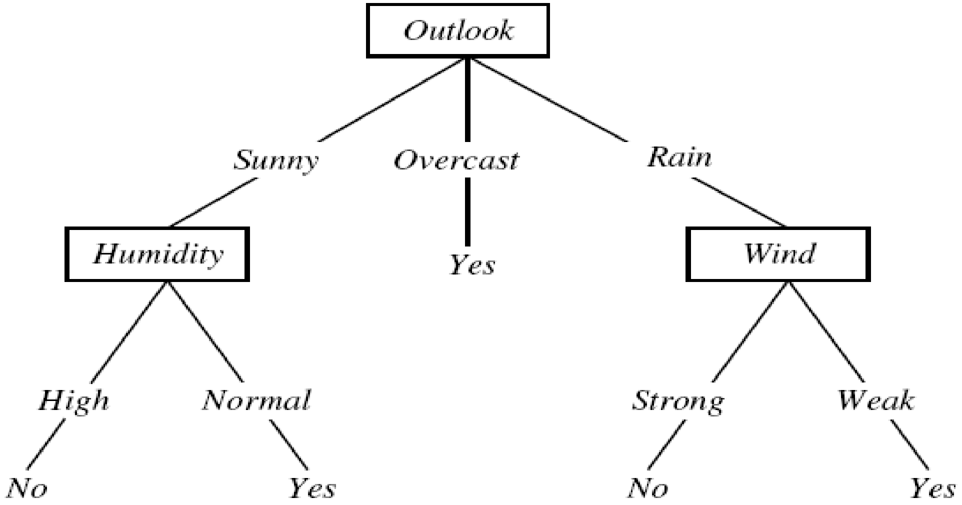
- Hypothesis space is complete?
  - Target function surely in there
- Outputs a single hypothesis (which one?)
- No back tracking
  - Local minima …
- Statistically-based search choices
  - Robust to noisy data …
- Inductive bias: approx "prefer shortest tree"

Overfitting in Decision Trees

Consider adding noisy training example #15

Sunny, Hot, Normal, Strong, PlayTennis = NO

What effect on earlier tree?



Overfitting

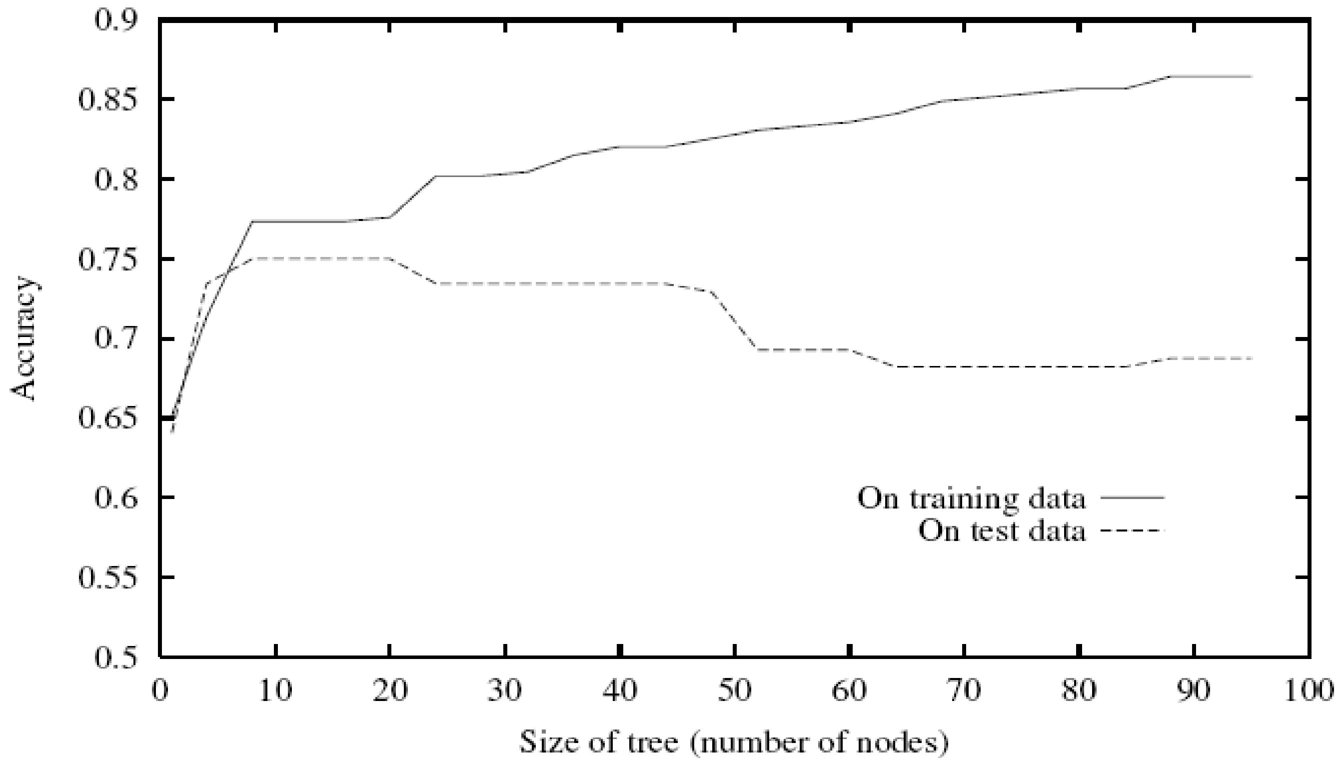
- Consider *error* of hypothesis *h* over
  - Training data: $error_{train}(h)$
  - Entire distribution *D* of data: $error_D(h)$
- Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

- How can we avoid overfitting?
  - Stop growing when data split not statistically significant
  - Grow full tree, then post-prune

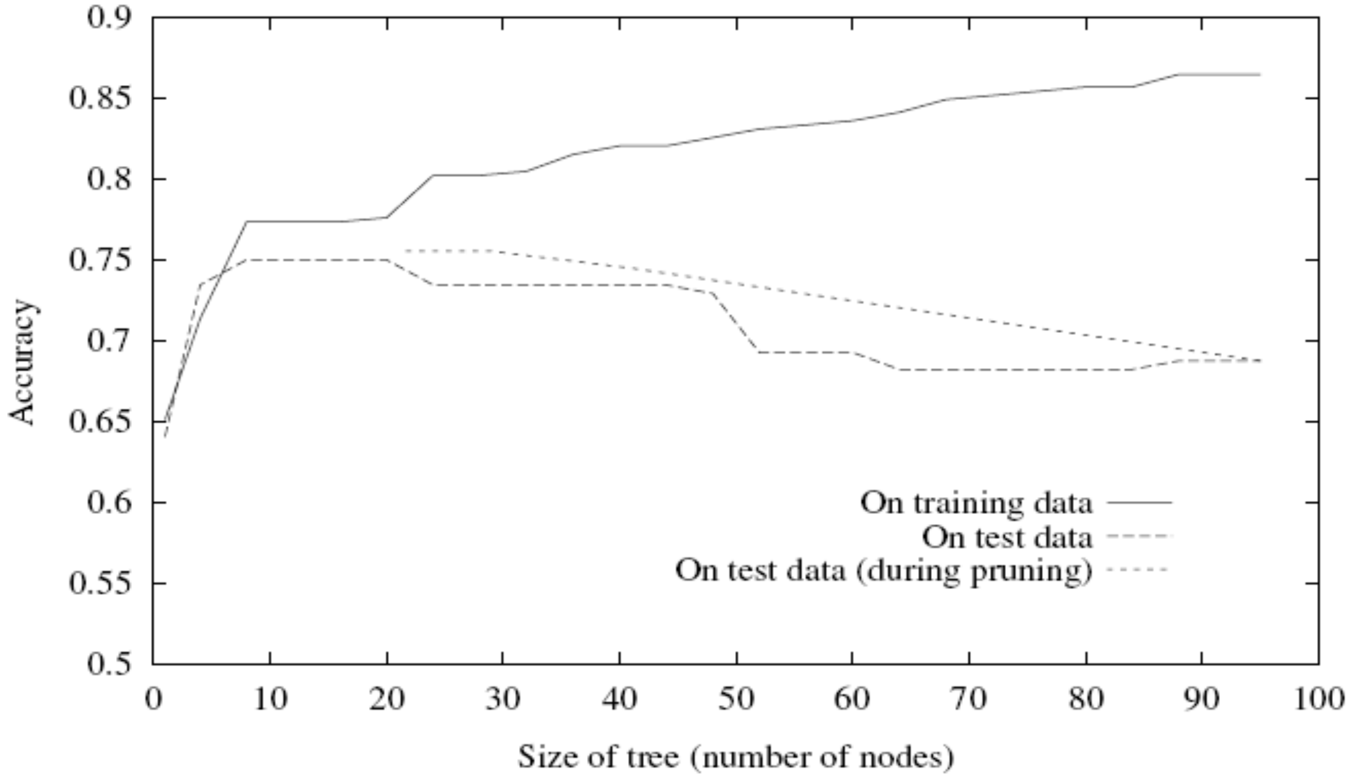
- How to select "best" tree?
  - Measure performance over training data
  - Measure performance over separate validation data set
  - MDL: minimize

    size(tree) + size (misclassification(tree))

Reduced-Error Pruning

- Split data into training and validation set
- Do Until further pruning is harmful:
    1. Evaluate impact on validation set of pruning each possible node (plus those below it)
    2. Greedily remove the one that most improves validation set accuracy
- Produces smallest version of most accurate subtree
- What if data is limited?

Effect of Reduced-Error Pruning

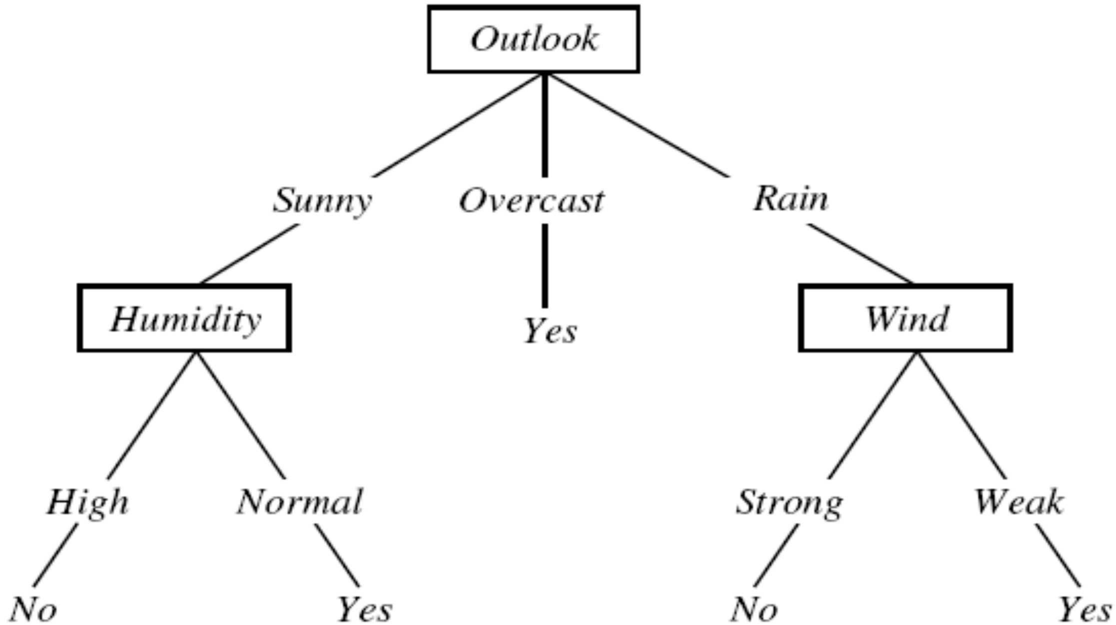


Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

Converting A Tree to Rules



IF (*Outlook* = *Sunny*) AND (*Humidity* = *High*)
THEN *PlayTennis* = *No*

IF (*Outlook* = *Sunny*) AND (*Humidity* = *Normal*)
THEN *PlayTennis* = *Yes*

Continuous Valued Attributes

- Create a discrete attribute to test continuous
  - *Temperature* = 82.5
  - (*Temperature* > 72.3) = t, f

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

Attributes with Many Values

- Problem:
  - If attribute has many values, Gain with select it
  - Imagine using Date = June_3_1996 as attribute

- One approach: use Gain Ratio instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Attributes with Costs

- Consider
  - Medical diagnosis, BloodTest has cost \$150
  - Robotics, Width_from_1ft has cost 23 sec.
- How to learn a consistent tree with low expected cost? One approach, replace gain by:

- Tan and Schlimmer (1990)

$$\frac{Gain^2(S, A)}{Cost(A)}.$$

- Nunez (1988)

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

where $w \in [0, 1]$ determines importance of cost

Unknown Attribute Values

- What if some examples missing values of A? Use training example anyway, sort through tree
  - If node n tests A, assign most common value of A among other examples sorted to node n
  - Assign most common value of A among other examples with same target value
  - Assign probability p_i to each possible value v_i of A and assign fraction p_i of example to each descendant in tree
- Classify new examples in same fashion