Applied Machine Learning

Trees

Computer Science, Fall 2022

Instructor: Xuhong Zhang

Trees: Function Approximation

- A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning.
- Both for classification and regression.
- Supervised or unsupervised?
- Given a set of training tuples, learn model to predict one value from the others
  - Learned value typically a class

Function Approximation

- Suppose we have a set of possible instances $\mathcal{X}$ and set of possible labels $\mathcal{Y}$
- Unknown target function $f: \mathcal{X} \rightarrow \mathcal{Y}$
- Set of function hypotheses $H = \{h | h: \mathcal{X} \rightarrow \mathcal{Y}\}$

  - **Input**: Training examples of unknown target function f
  - **Output**: Hypothesis $h \in H$ that best approximates f

Based on slide by Tom Michell

Sample Dataset

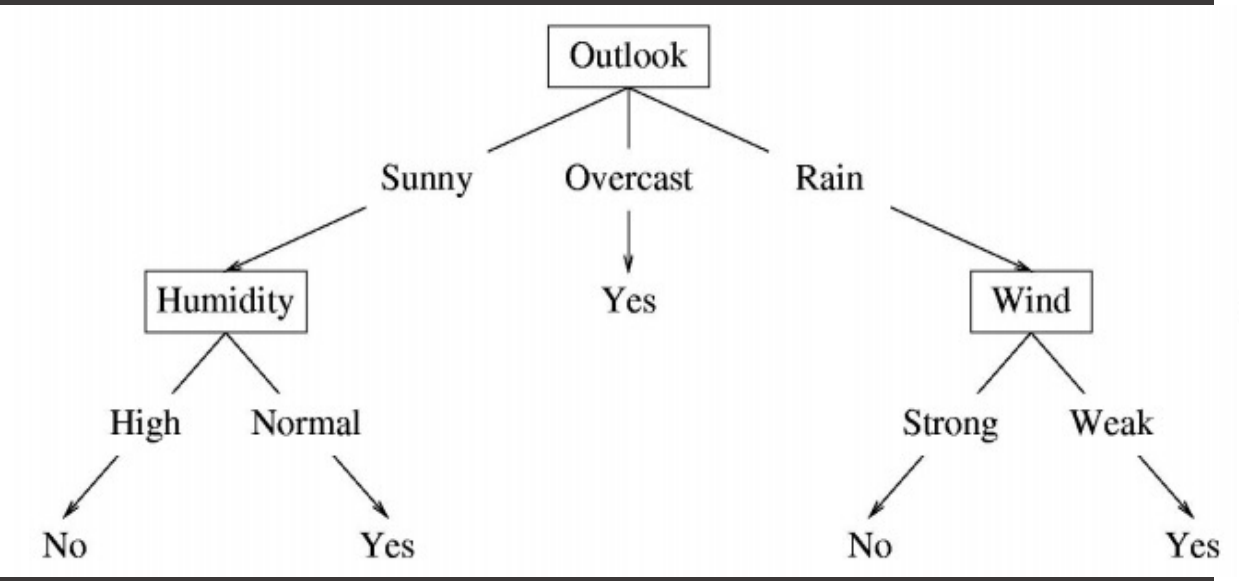
- Columns denote features X_i
- Rows denote labeled instances $\langle x_i, y_i \rangle$
- Class label denotes whether a tennis game was played

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

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

Decision Tree

- A possible decision tree for the data



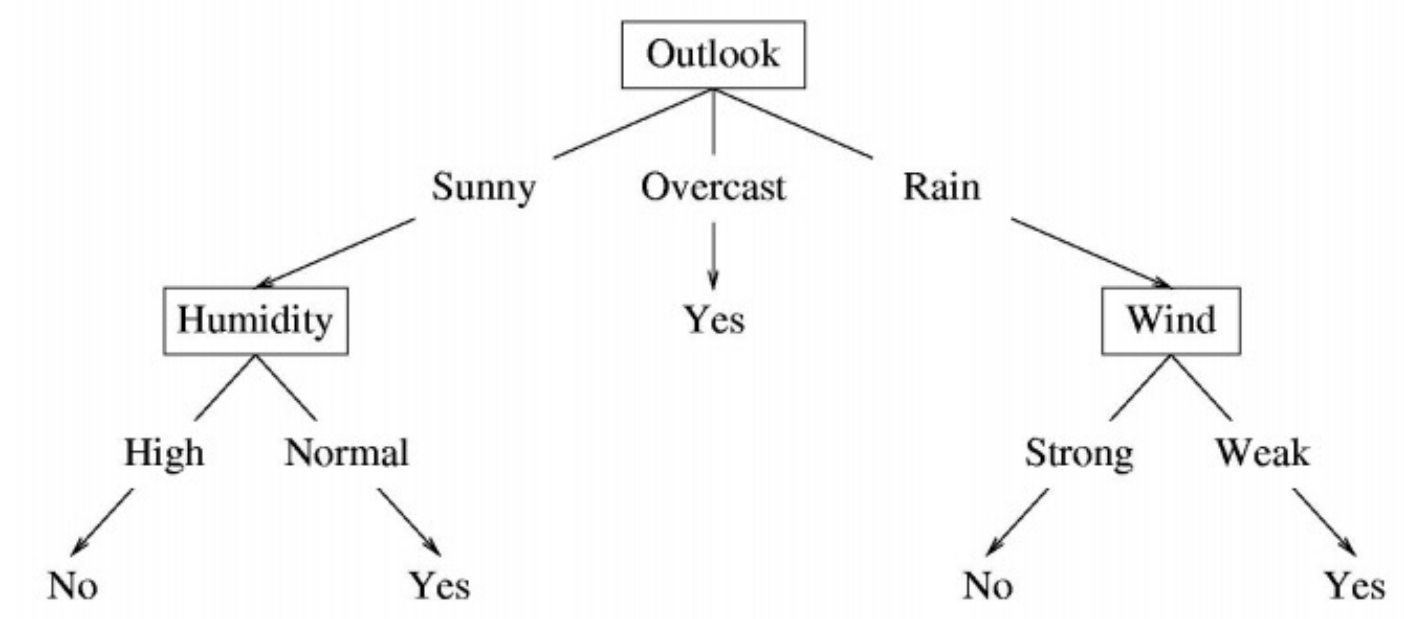
$\langle \boldsymbol{x}_i, y_i \rangle$

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

Based on slide by Tom Mitchell

Decision Tree

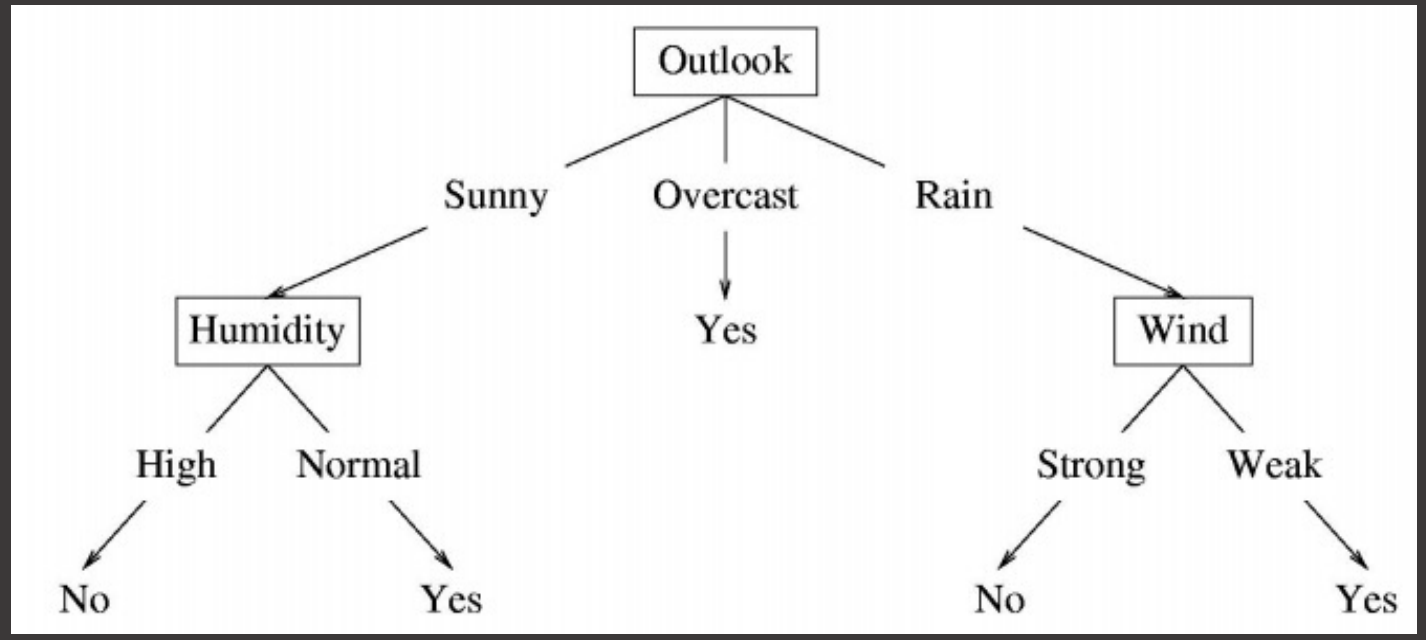
- A possible decision tree for the data



- Each internal node
  - test one attribute Xi
- Each branch from a node
  - selects one value for X
- Each leaf node
  - Predict Y (or $p(Y|x \in leaf)$)

Decision Tree

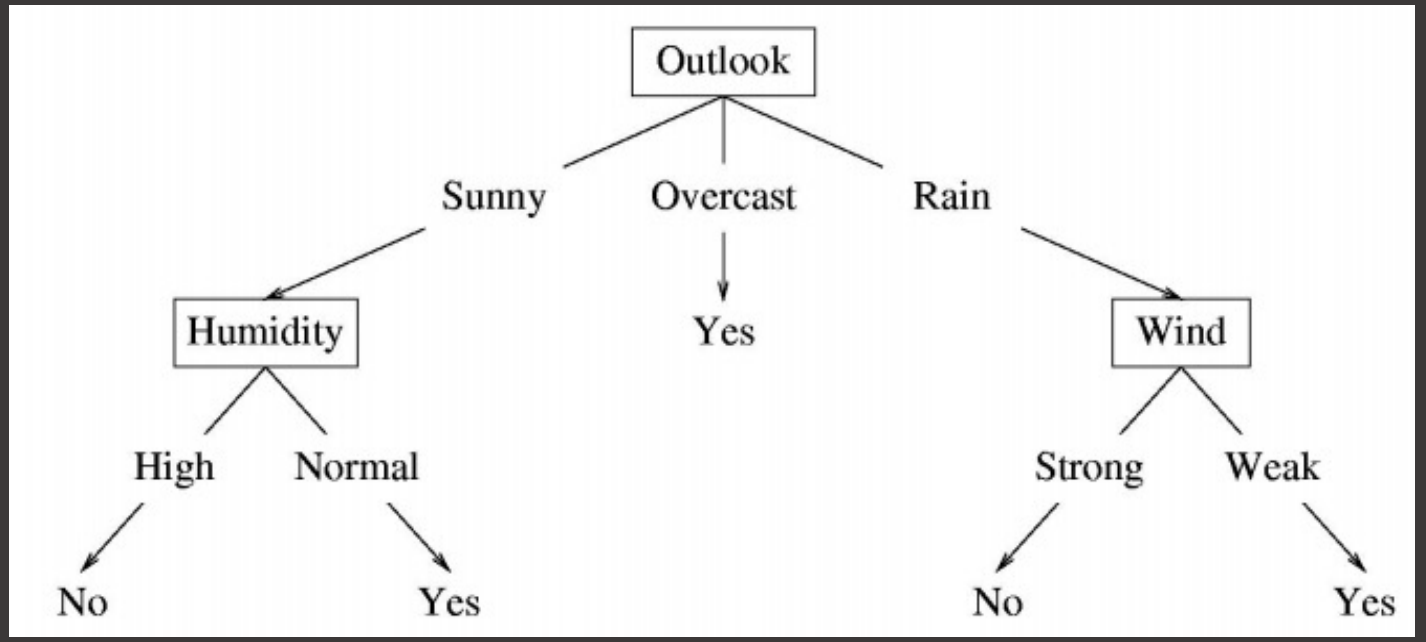
- A possible decision tree for the data



- Question:
  - What prediction would we make for (outlook = sunny, Temperature = hot, Humidity = high, Wind = weak)?

Decision Tree

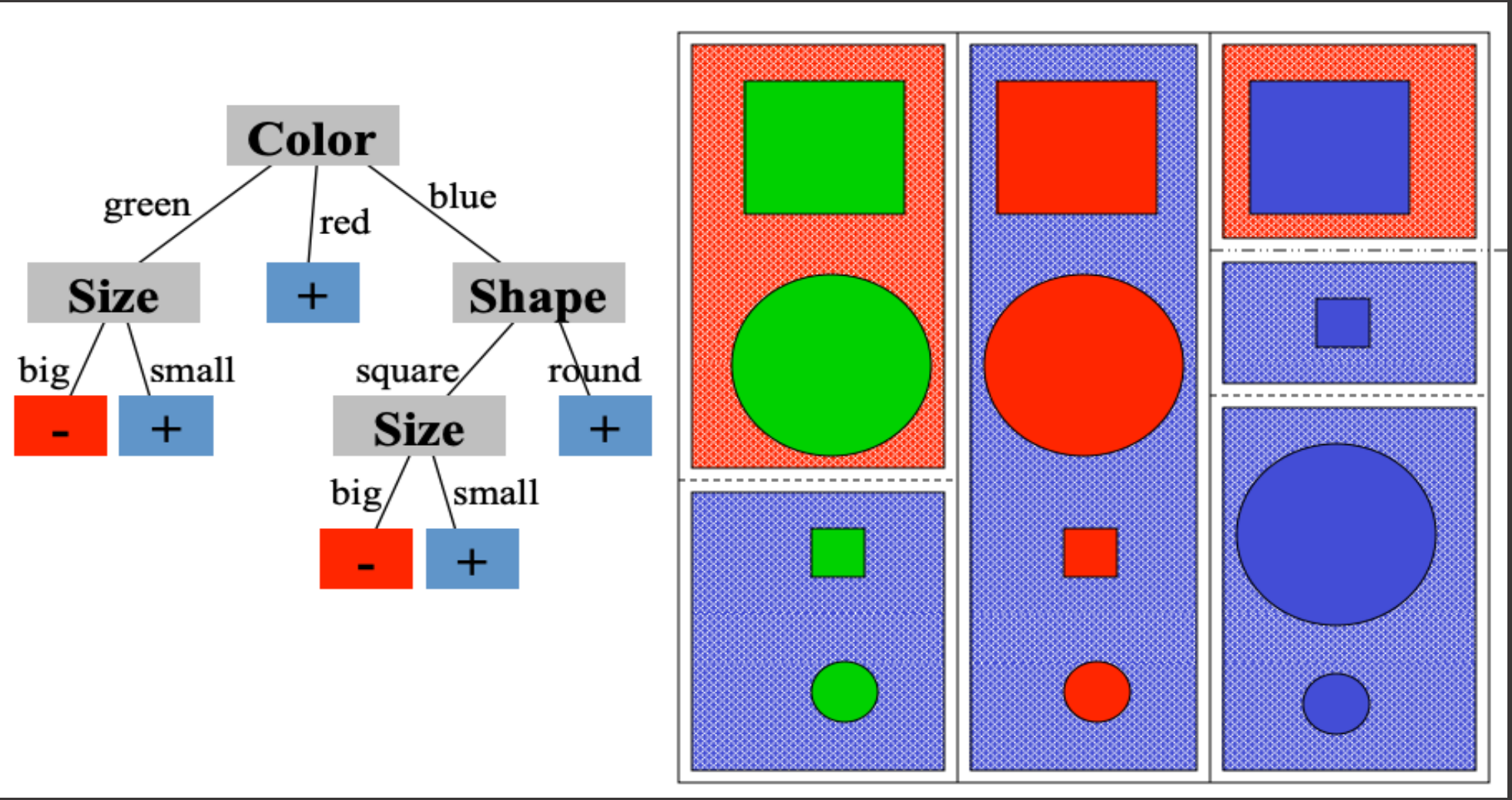
- If features are continuous, internal nodes can test the value of a feature against a threshold



Summary

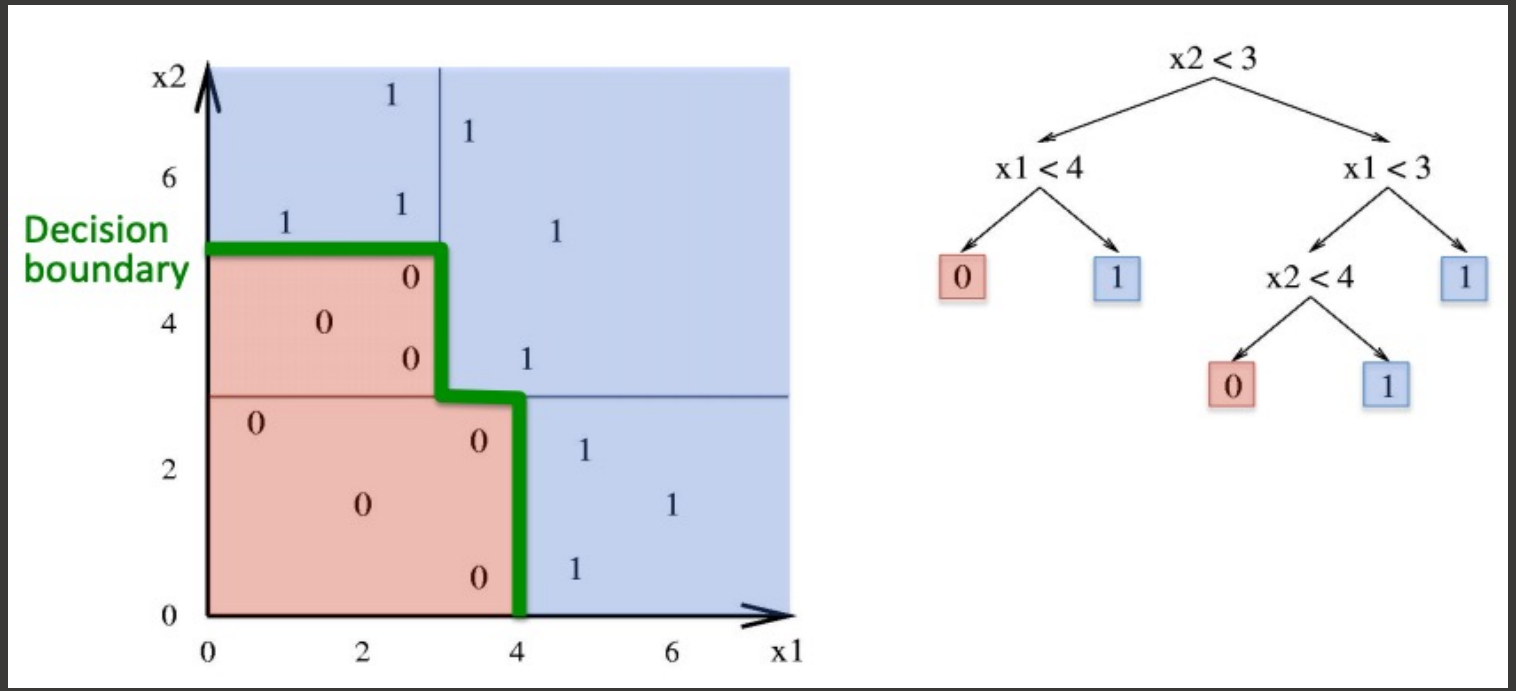
- Problem Setting
  - Set of possible instances $\mathcal{X}$
    - each instance x in $\mathcal{X}$ is a feature vector
    - e.g., <Humidity=low, Wind=weak, Outlook=rain, Temp=hot>
- Unknown target function $f: \mathcal{X} \rightarrow \mathcal{Y}$
  - Y is discrete valued
- Set of function hypotheses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

Decision Tree Induced Partition



Decision Tree Boundary

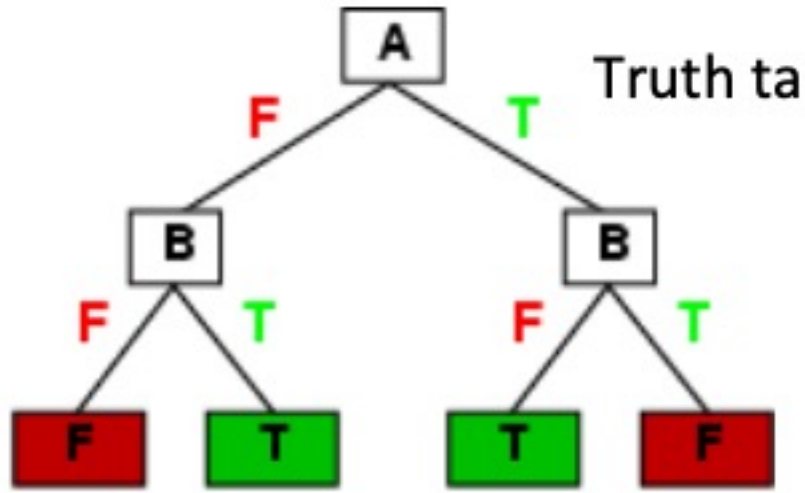
- Decision trees divide the feature space into axis-parallel (hyper-) rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels



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

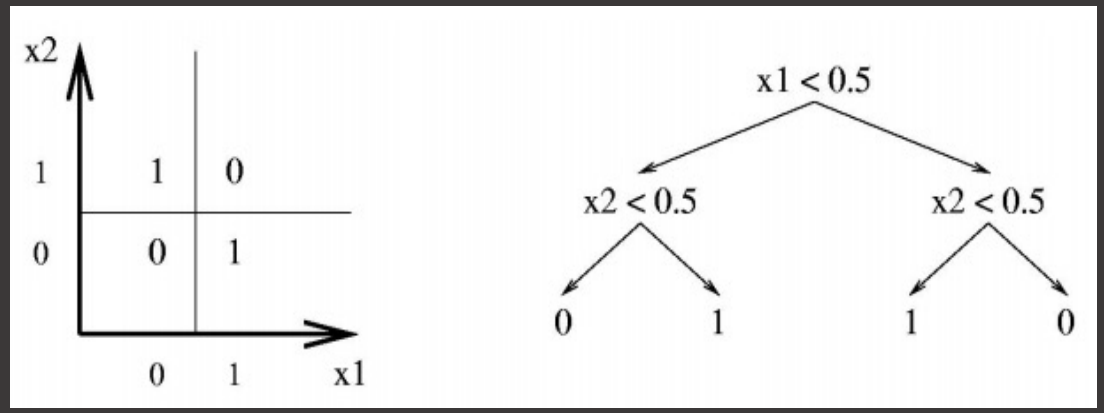


Truth table row → path to leaf

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

Expressiveness

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



Based on slide by Pedro Domingos

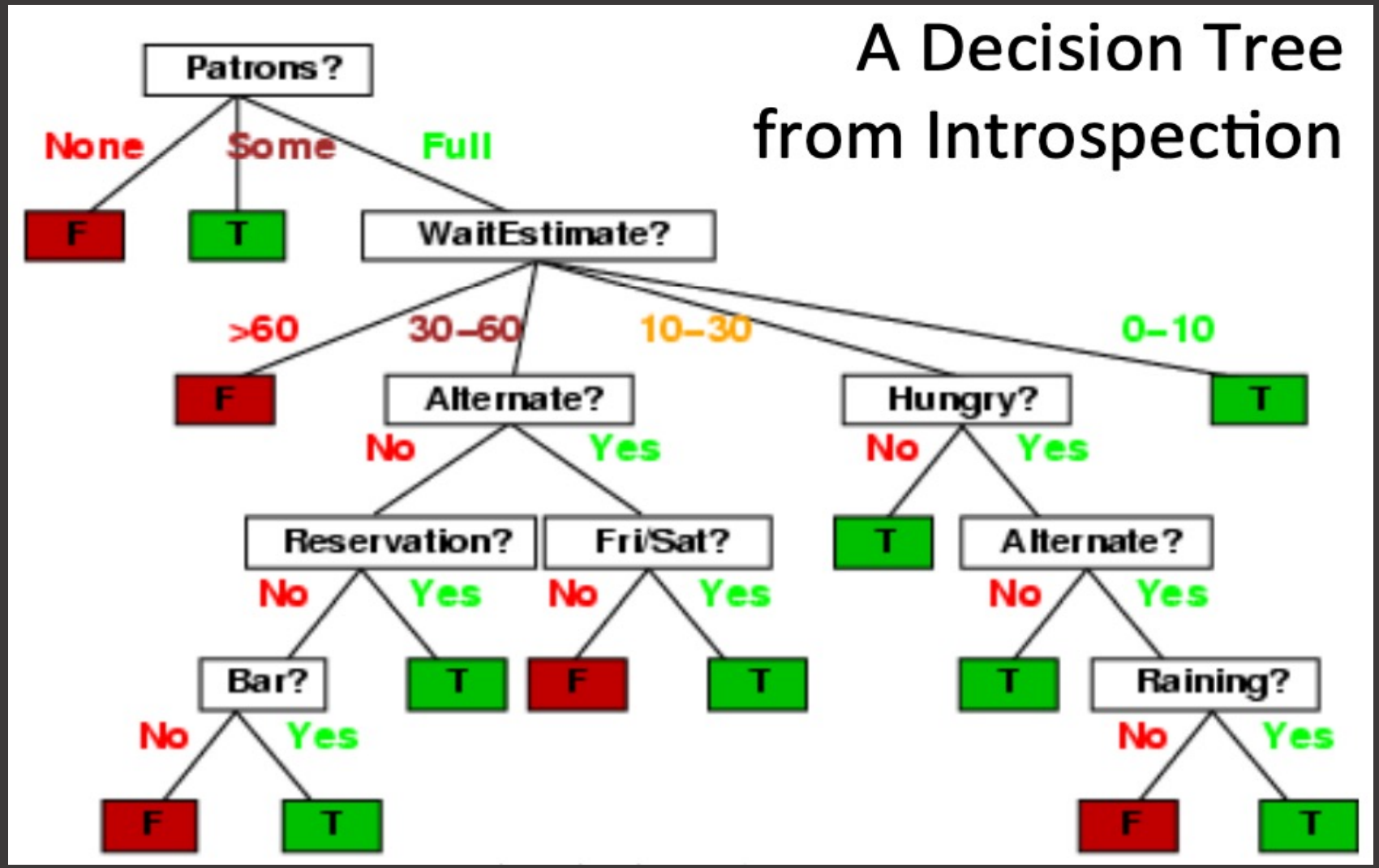
Another Example: Restaurant Domain (Russell & Norvig)

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

Another Example: Restaurant Domain (Russell & Norvig)



Question: Is this the best tree?

Preference bias: Ockham's Razor

- Principle stated by William Ockham (1285-1347)
  - "non sunt multiplicanda entia praeter necessitate"
  - 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

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. If training examples are perfectly classified, stop;

Else, recurse over new leaf nodes.

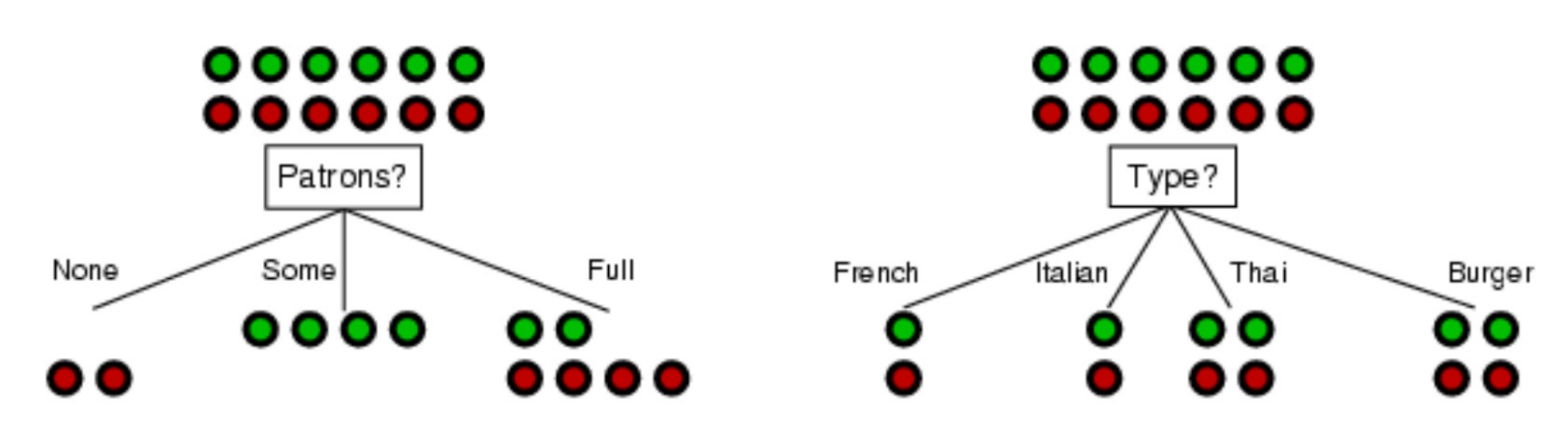
Key question: How do we choose which attribute is best?

Choosing the Best Attribute

- **Key problem**: choosing which attribute to split a given set of examples
- Some options:
  - **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
    - E.g., attribute that results in smallest expected size of subtrees rooted at its children

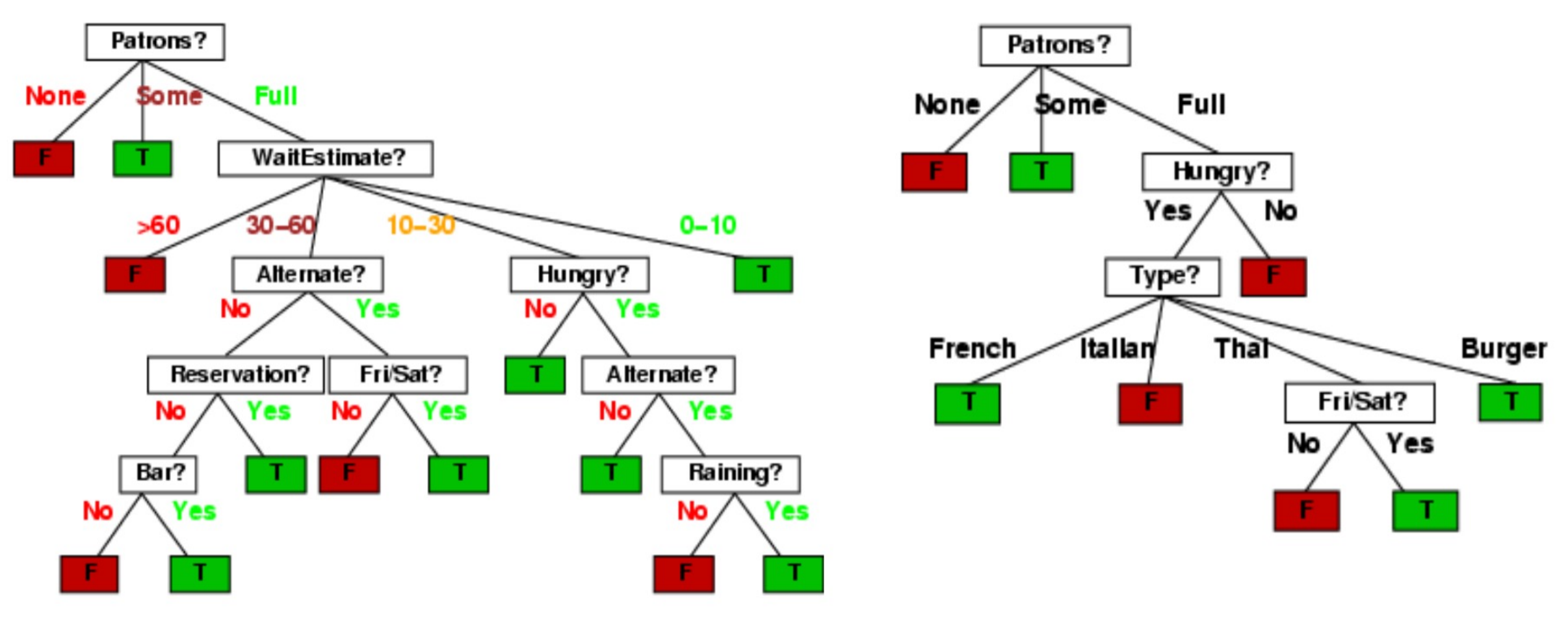
Choosing an Attribute

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



Question: Which split is more informative: Patrons? Or Type?

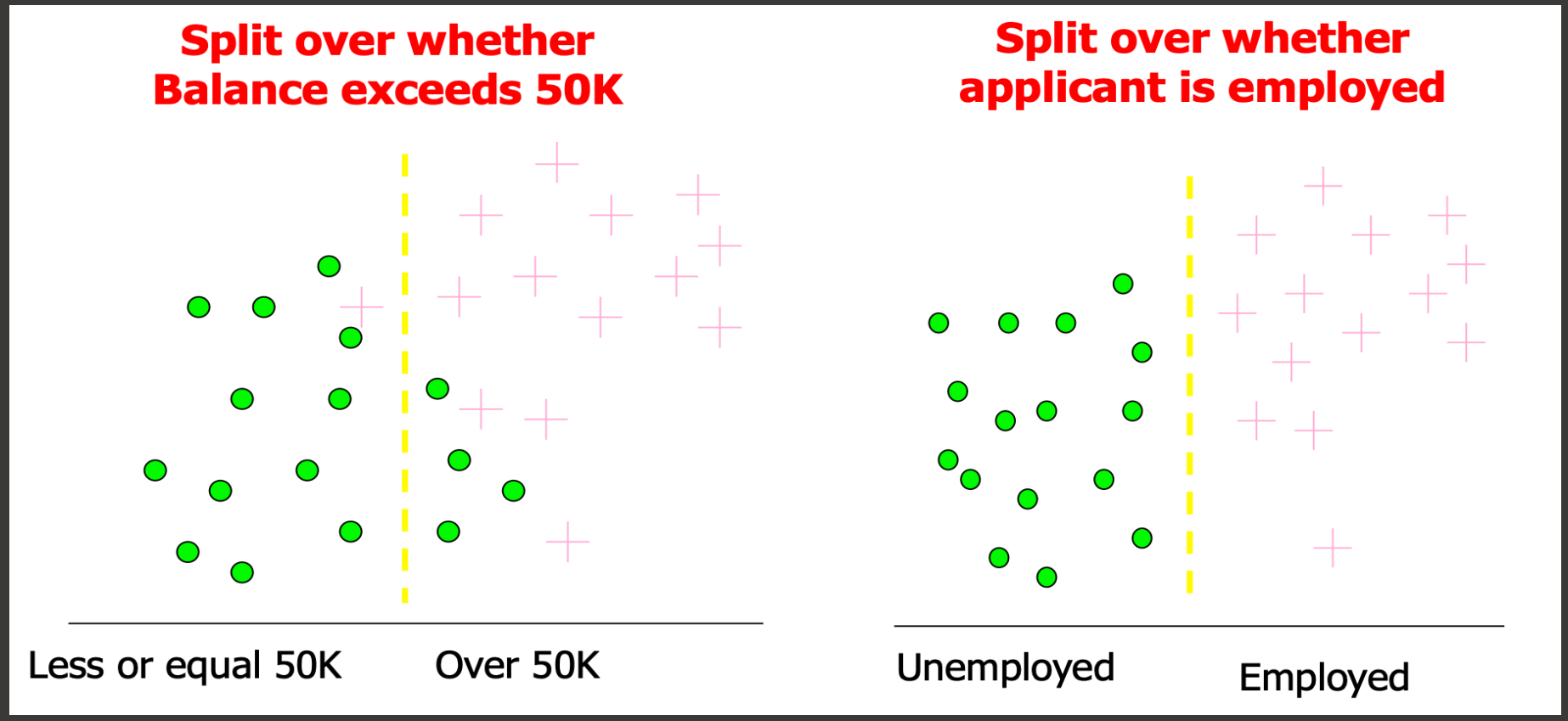
Compare the Two Decision Trees



Based on slide from M. DesJardians & T. Finin

Information Gain

Which test is more informative?

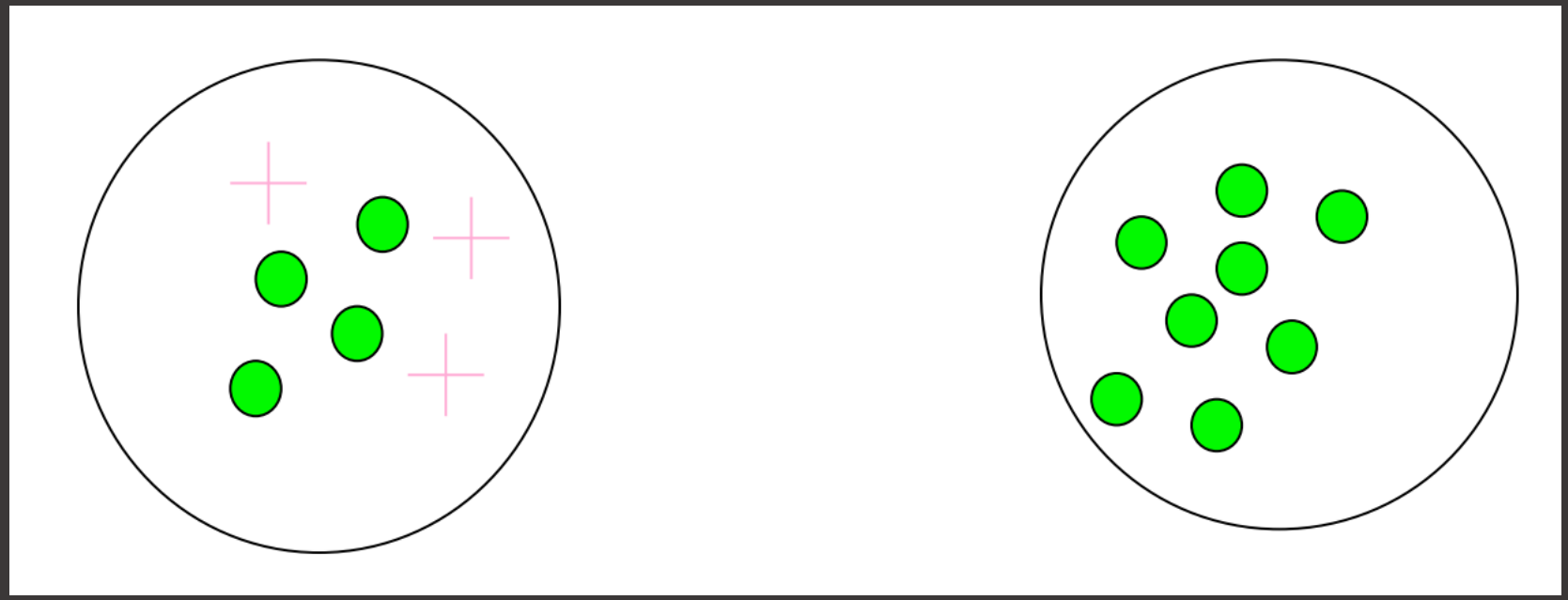


Based on slide by Pedro Domingos

Information Gain

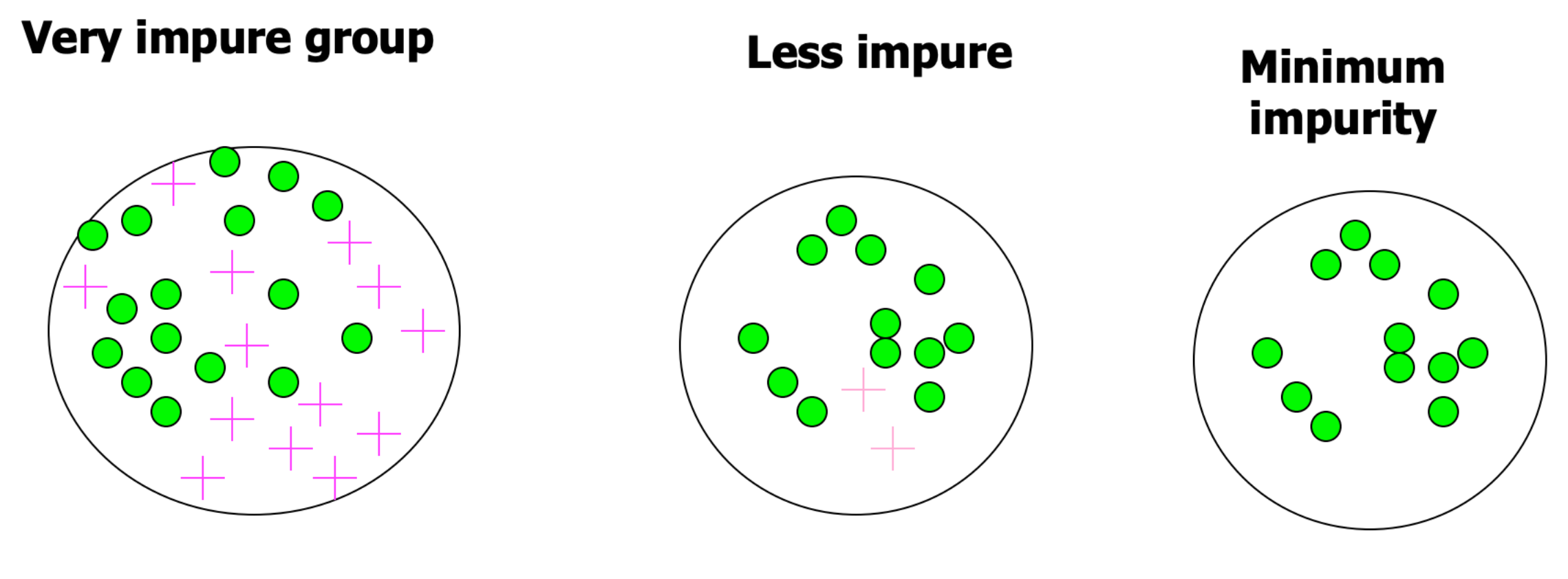
Impurity/Entropy (Informal)

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



Based on slide by Pedro Domingos

Impurity



Based on slide by Pedro Domingos

Entropy

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

Entropy: a common way to measure impurity

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

# of possible values for 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)

Slide credit: Tom Mitchell

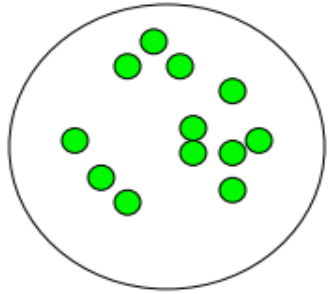
2-Class Cases

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

- 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

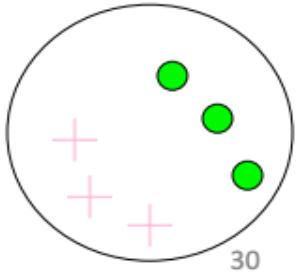
Minimum impurity



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

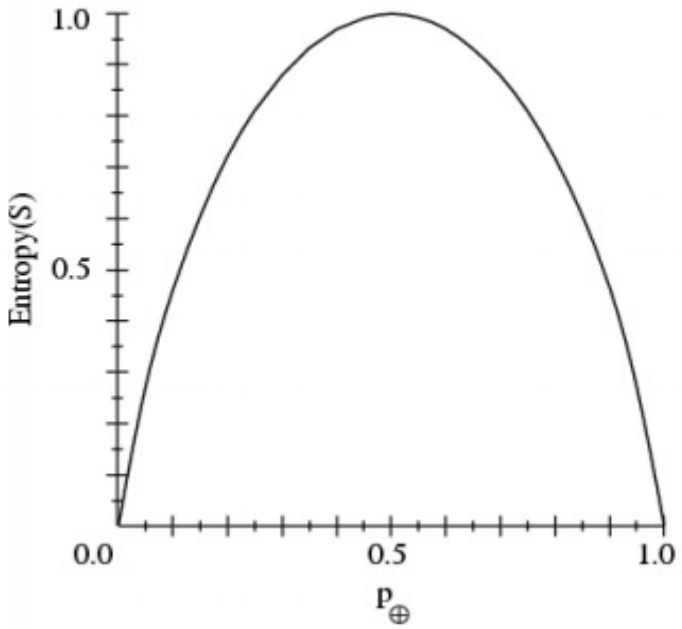
good training set for learning

Maximum impurity



Based on slide by Pedro Domingos

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

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

Slide credit: Tom Mitchell

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 vector is.

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

Slide credit: Pedro Domingos

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:

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

Slide credit: Tom Mitchell

From Entropy to Information Gain

Mutual information (aka Information Gain) of X and Y:

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

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

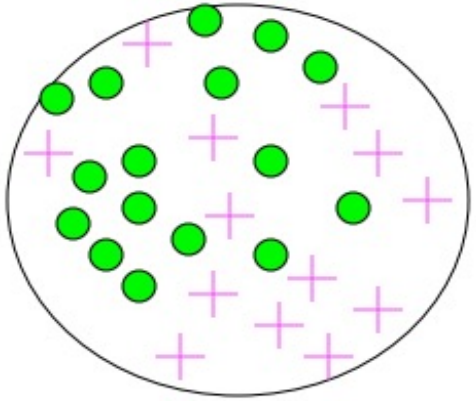
Slide credit: Tom Mitchell

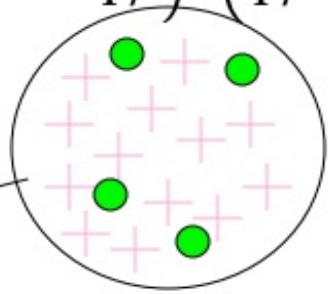
Calculating Information Gain

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

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

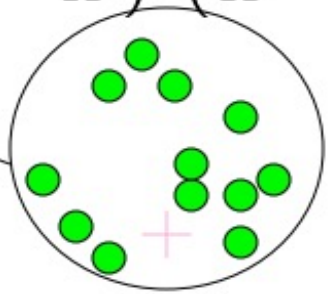
Entire population (30 instances)





17 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

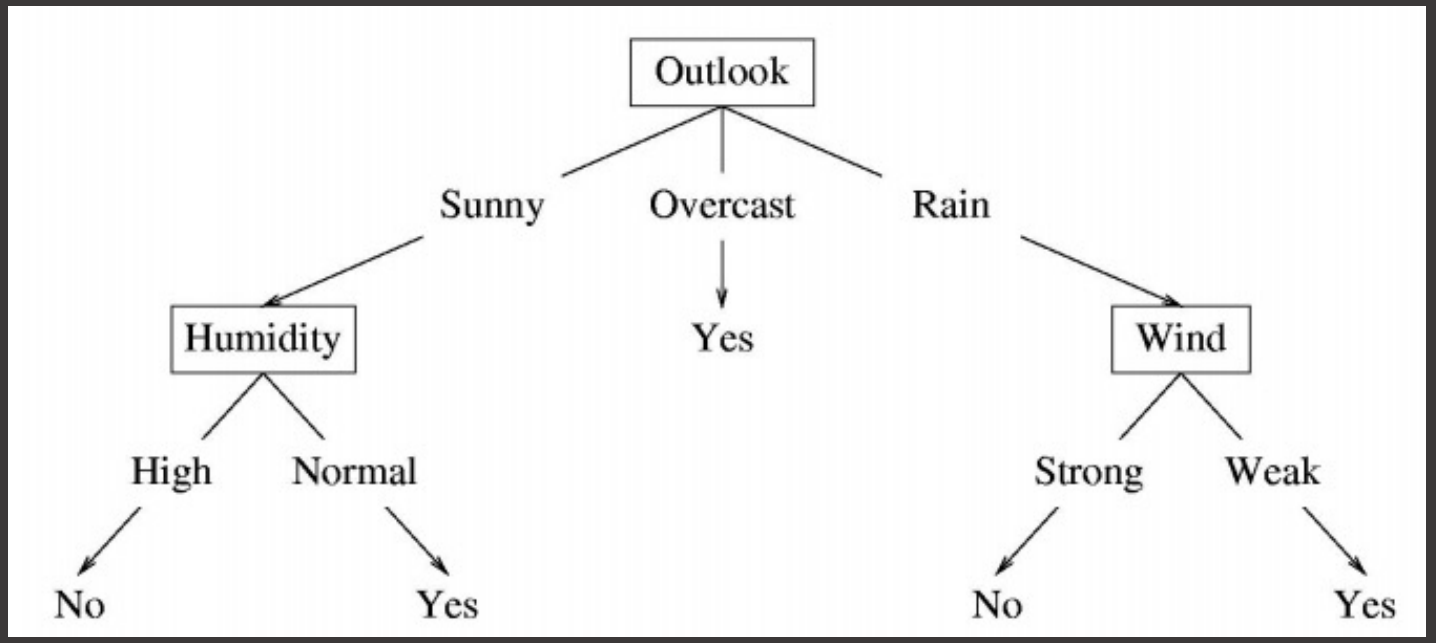
Based on slide by Pedro Domingos

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

Slide credit: M. desJardins & T. Finin

Summary of Decision Trees (so far)



- Decision tree induction → choose the best attribute
  - Choose a split via information gain
  - Build a tree greedily, recursing on children of split
  - Stop when we achieve homogeny
    - i.e., when all instance in a child have the same class

Summary of Decision Trees (so far)

Information Gain: Mutual information of attribute A and the class variable of data set X

$$InfoGain(X, A) = H(X) - H(X \mid A)$$

$$= H(X) - \sum_{v \in values(A)} \frac{|\{x \in X \mid x_A = v\}|}{|X|} \times H(\{x \in X \mid x_A = v\})$$

fraction of instances with value v in attribute A

entropy of those instances

Entropy:

$$H(X) = - \sum_{c \in Classes} \frac{|\{x \in X \mid class(x) = c\}|}{|X|} \log_2 \frac{|\{x \in X \mid class(x) = c\}|}{|X|}$$

fraction of instances of class c