COMP3411/9414: Artificial Intelligence

Module 3

Learning and Decision Trees

Russell & Norvig, Chapter 18.1, 18.2, 18.3

Outline

- Learning agents
- Inductive learning
- Decision tree learning

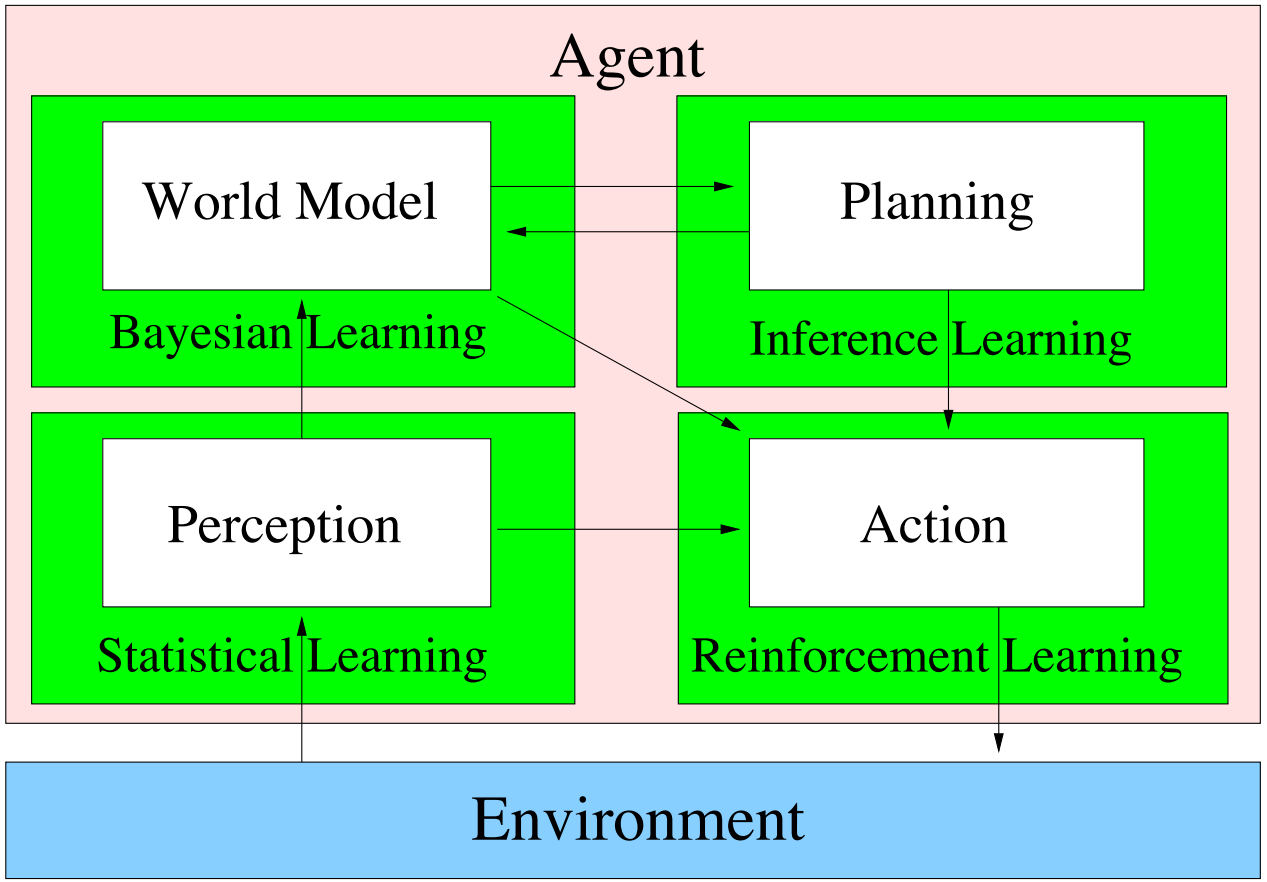
Motivation

- Reactive and Model-Based Agents choose their actions based only on what they currently perceive, or have perceived in the past.
- A Planning Agent can use Search techniques to plan several steps ahead in order to achieve its goal(s).
- Two classes of search strategies:
  - Uninformed search strategies can only distinguish goal states from non-goal states
  - Informed search strategies use heuristics to try to get "closer" to the goal

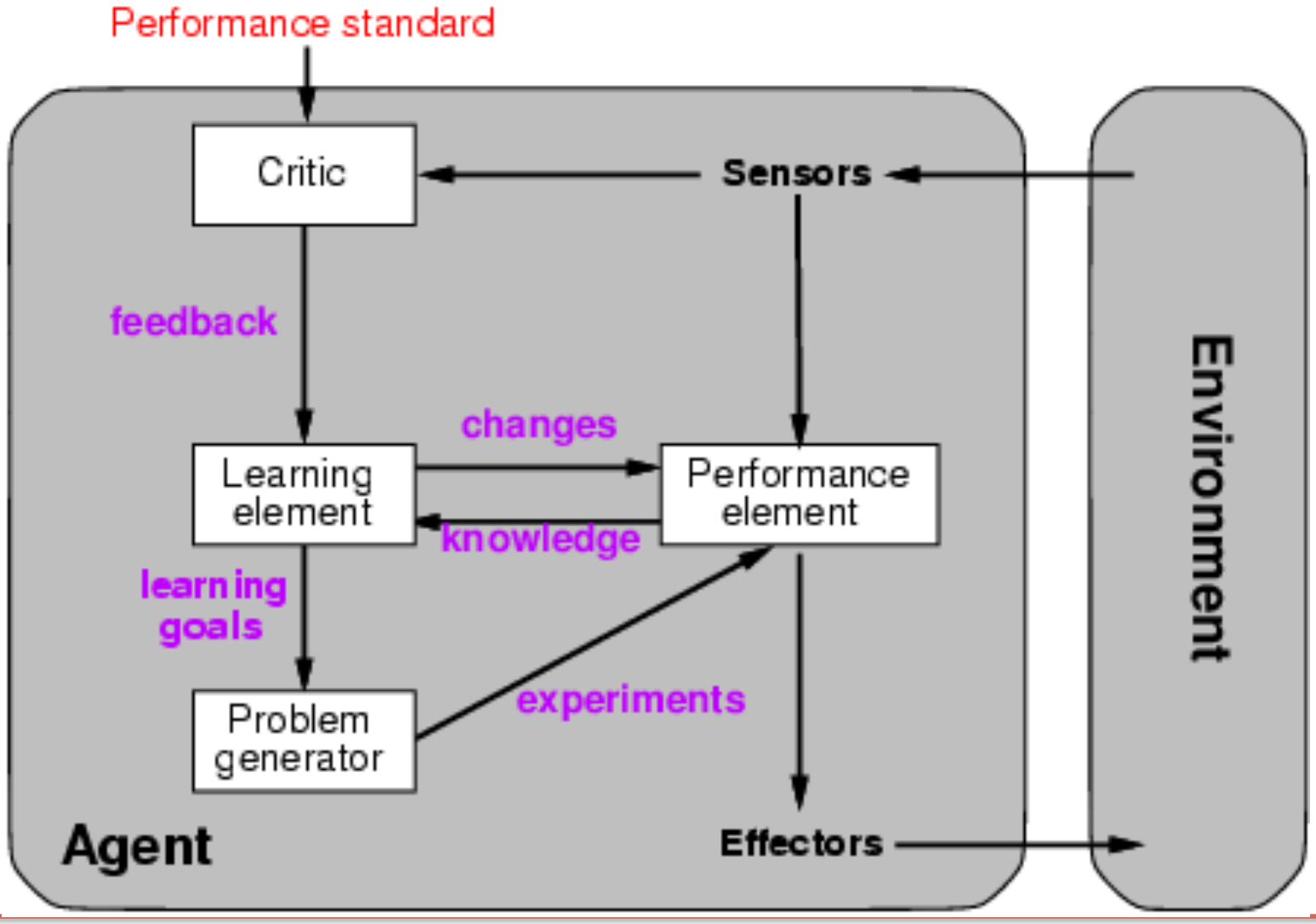
Learning

- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Learning Agent



Learning agents



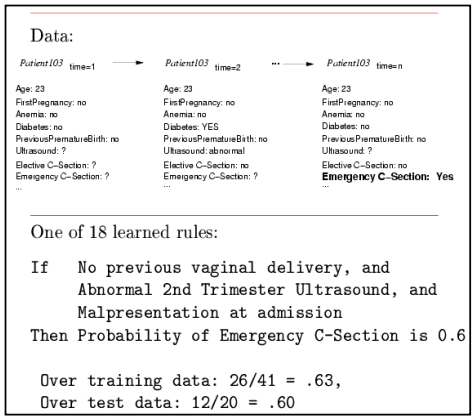
Learning agents

The Aim is to improve its performance on future tasks after making observations about the world.

Learning element

- Design of a learning element is affected by
  - Which components of the performance element are to be learned
  - What feedback is available to learn these components
  - What representation is used for the components

Machine Learning – Practice


Data:

Patient103 time=1 → Patient103 time=2 ··· → Patient103 time=n

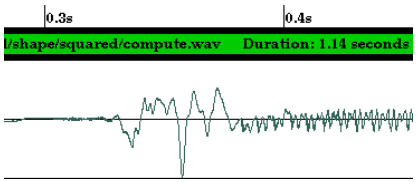
Age: 23	Age: 23	Age: 23
FirstPregnancy: no	FirstPregnancy: no	FirstPregnancy: no
Anemia: no	Anemia: no	Anemia: no
Diabetes: no	Diabetes: YES	Diabetes: no
PreviousPrematureBirth: no	PreviousPrematureBirth: no	PreviousPrematureBirth: no
Ultrasound: ?	Ultrasound: abnormal	Ultrasound: ?
Elective C–Section: ?	Elective C–Section: no	Elective C–Section: no
Emergency C–Section: ?	Emergency C–Section: ?	**Emergency C–Section: Yes**

One of 18 learned rules:

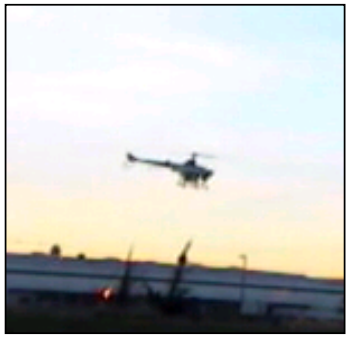
```
If   No previous vaginal delivery, and
     Abnormal 2nd Trimester Ultrasound, and
     Malpresentation at admission
Then Probability of Emergency C-Section is 0.6

 Over training data: 26/41 = .63,
 Over test data: 12/20 = .60
```

Mining Databases



Speech Recognition



Control learning

Object recognition

Text analysis

Peter H. van Oppen , **Chairman of the Board & Chief Executive Officer**
Mr. van Oppen has served as chairman of the board and chief executive officer of ADIC since its acquisition by Interpoint in 1994 and a director of ADIC since 1986. Until its acquisition by Crane Co. in October 1996, Mr. van Oppen served as chairman of the board of directors, president and chief executive officer of Interpoint . Prior to 1985, Mr. van Oppen worked as a consulting manager at Price Waterhouse LLP and at Bain & Company in Boston and London. He has additional experience in medical electronics and venture capital. Mr. van Oppen also serves as a director of Seattle FilmWorks Inc. and Spacelabs Medical, Inc.. He holds a B.A. from Whitman College and an M.B.A. from Harvard Business School, where he was a Baker Scholar.

- Supervised learning
- Bayesian networks
- Hidden Markov models
- Unsupervised clustering
- Reinforcement learning

Types of Learning

- Supervised Learning
  - agent is presented with examples of inputs and their target outputs
- Reinforcement Learning
  - agent is not presented with target outputs, but is given a reward
  - signal, which it aims to maximize
- Unsupervised Learning
  - agent is only presented with the inputs themselves, and aims to find structure in these inputs

Supervised Learning

- We have a training set and a test set, each consisting of a set of items; for each item, a number of input attributes and a target value are specified.
- The aim is to predict the target value, based on the input attributes.
- Agent Is presented with the input and target output for each item in the training set; it must then predict the output for each item in the test set
- Various learning paradigms are available:
  - Decision Tree
  - Neural Network
  - Support Vector Machine, etc.

Supervised Learning – Issues

- framework (decision tree, neural network, SVM, etc.)
- representation (of inputs and outputs)
- pre-processing / post-processing
- training method (perceptron learning, backpropagation, etc.)
- generalization (avoid over-fitting)
- evaluation (separate training and testing sets)

Inductive learning

Simplest form: learn a function from examples

f is the target function
An example is a pair $(x, f(x))$

Problem: find a hypothesis h
such that $h \approx f$
given a training set of examples

Inductive learning

Simplest form: learn a function from examples

f is the target function
An example is a pair $(x, f(x))$

Problem: find a hypothesis *h*
such that $h \approx f$
given a training set of examples

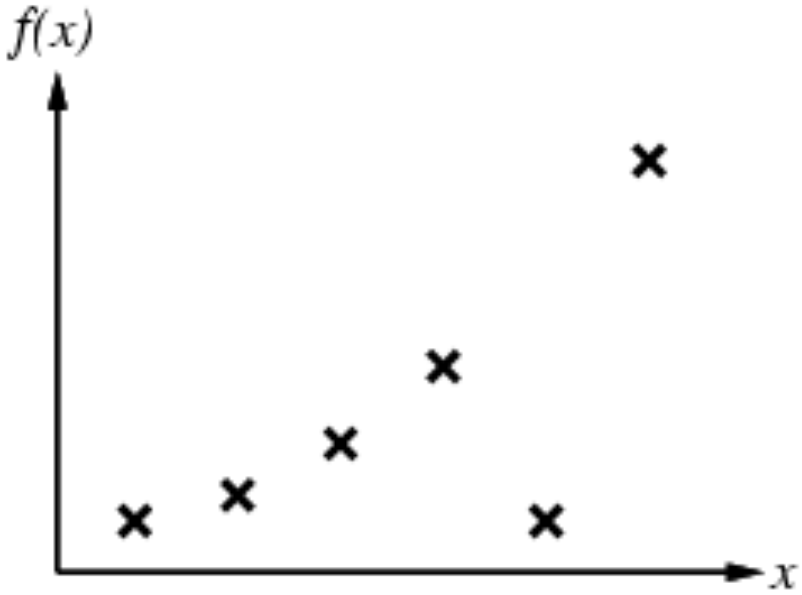
(This is a highly simplified model of real learning:
- Ignores prior knowledge
- Assumes examples are given)

Inductive learning method

Construct/adjust *h* to agree with *f* on training set

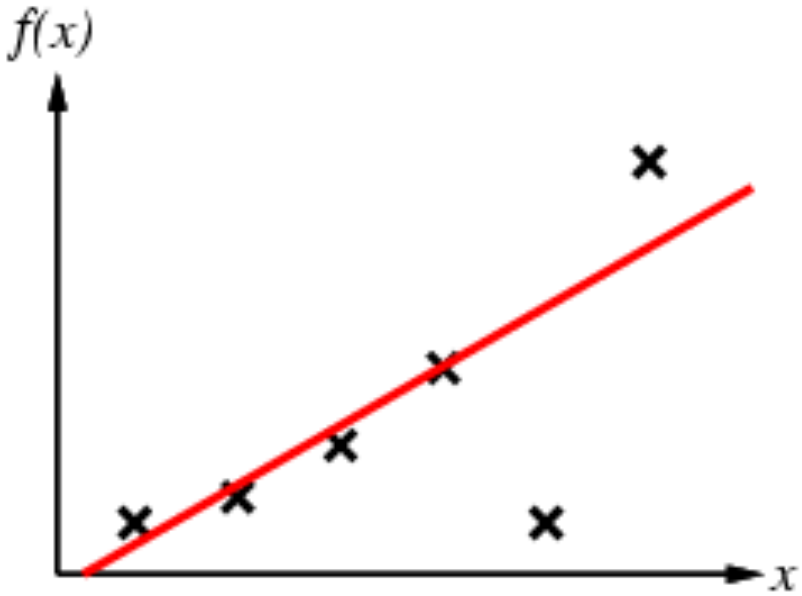
(*h* is consistent if it agrees with *f* on all examples)

- E.g., curve fitting:



Inductive learning method – Curve Fitting

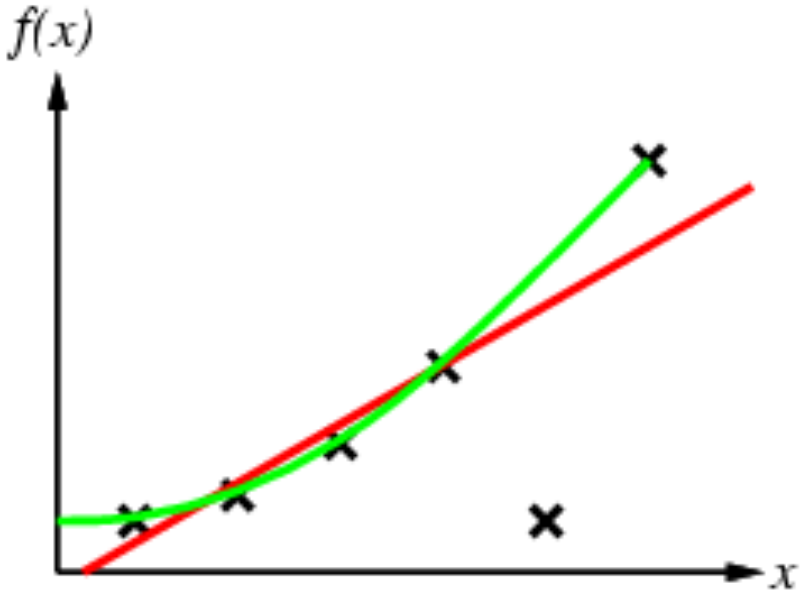
Which curve gives the "best fit" to these data?



straight line?

Curve Fitting

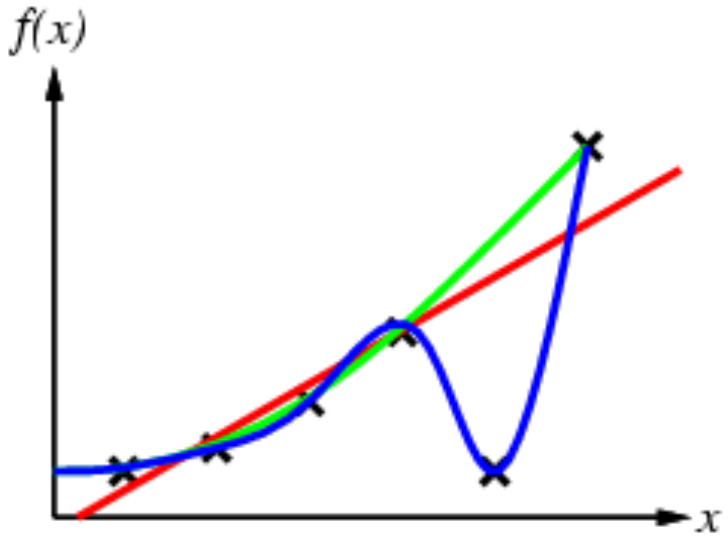
Which curve gives the "best fit" to these data?



parabola?

Inductive learning method

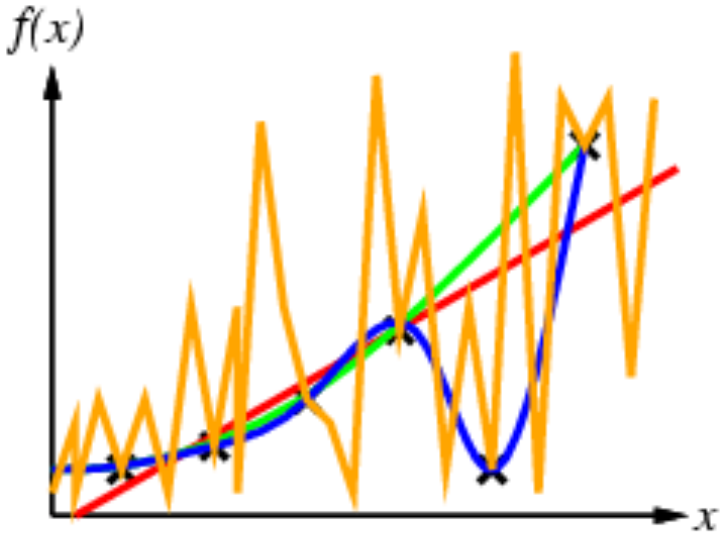
Which curve gives the "best fit" to these data?



4th order polynomial?

Inductive learning method

Which curve gives the "best fit" to these data?



Something else?

Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."

Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."

- Occam's razor (sometimes spelled Ockham's razor) is a principle attributed to the 14th- century English logician and Franciscan friar William of Ockham.

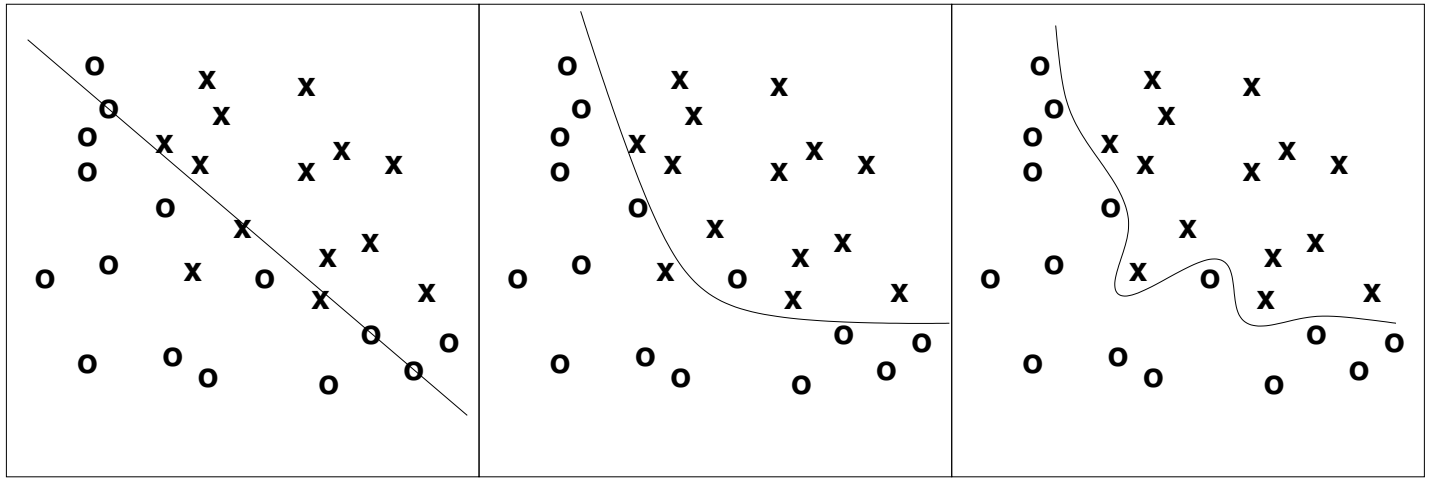
The principle states that the explanation of any phenomenon should make as few assumptions as possible, eliminating those that make no difference in the observable predictions of the explanatory hypothesis or theory.

This is often paraphrased as "All other things being equal, the simplest solution is the best."

Prefer the simplest hypothesis that fits the data

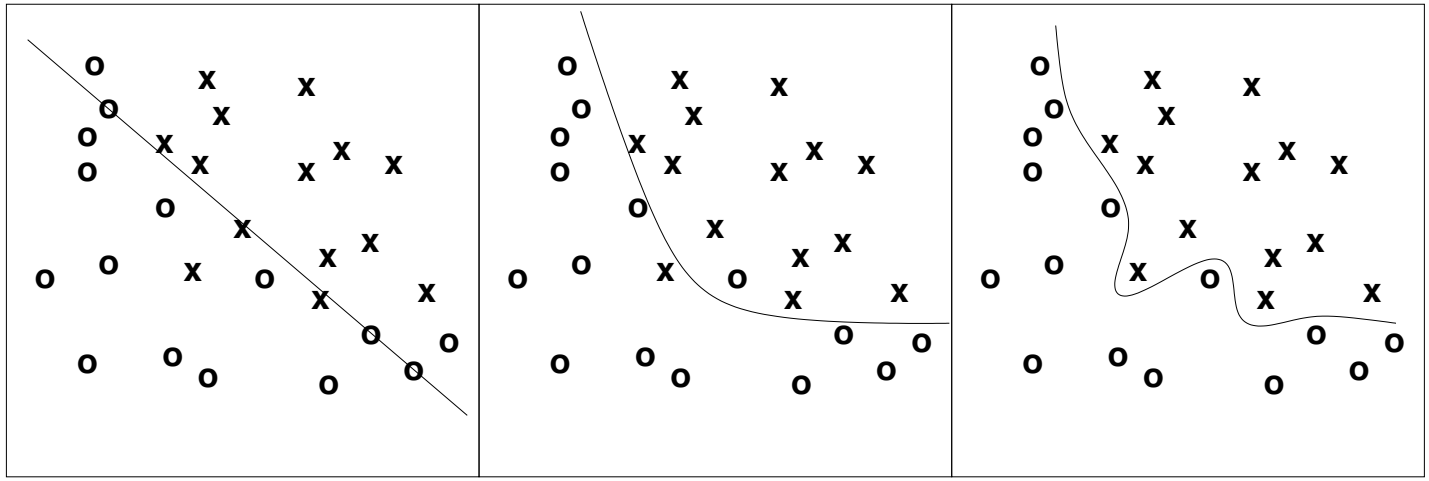
Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."



Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."

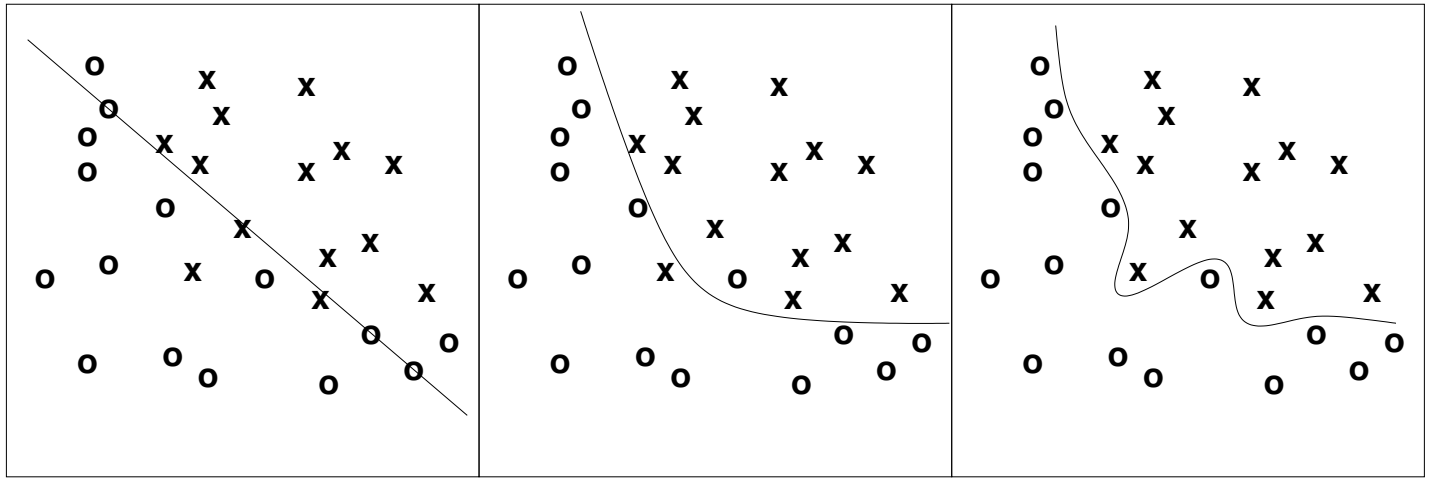


inadequate

Since there can be noise in the measurements, in practice need to make a tradeoff between simplicity of the hypothesis and how well it fits the data.

Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."

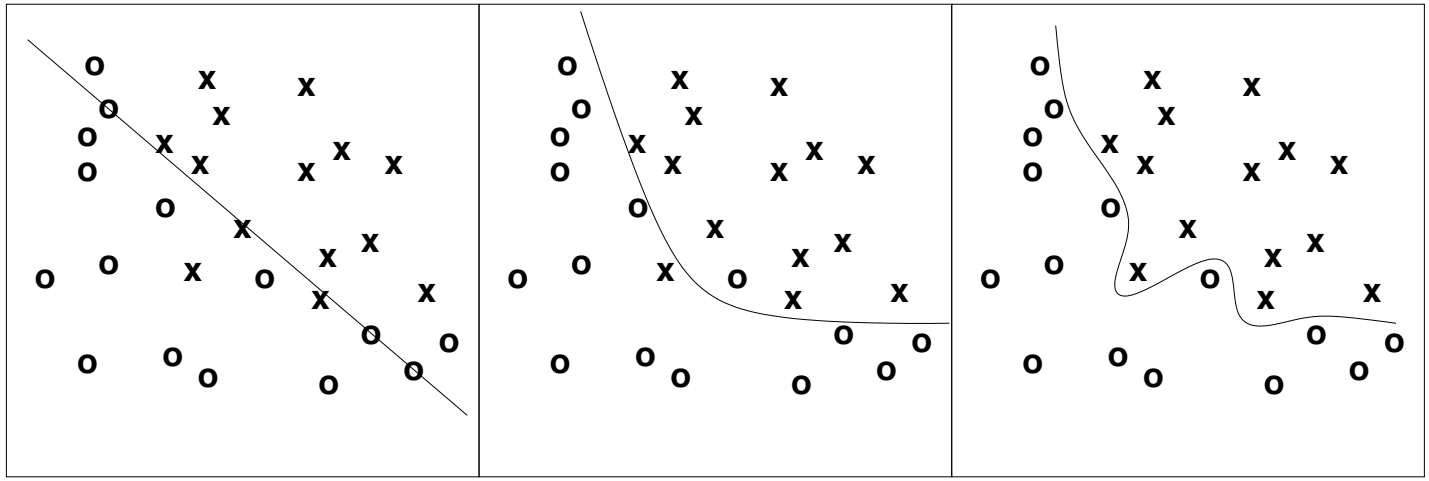


inadequate good compromise

Since there can be noise in the measurements, in practice need to make a tradeoff between simplicity of the hypothesis and how well it fits the data.

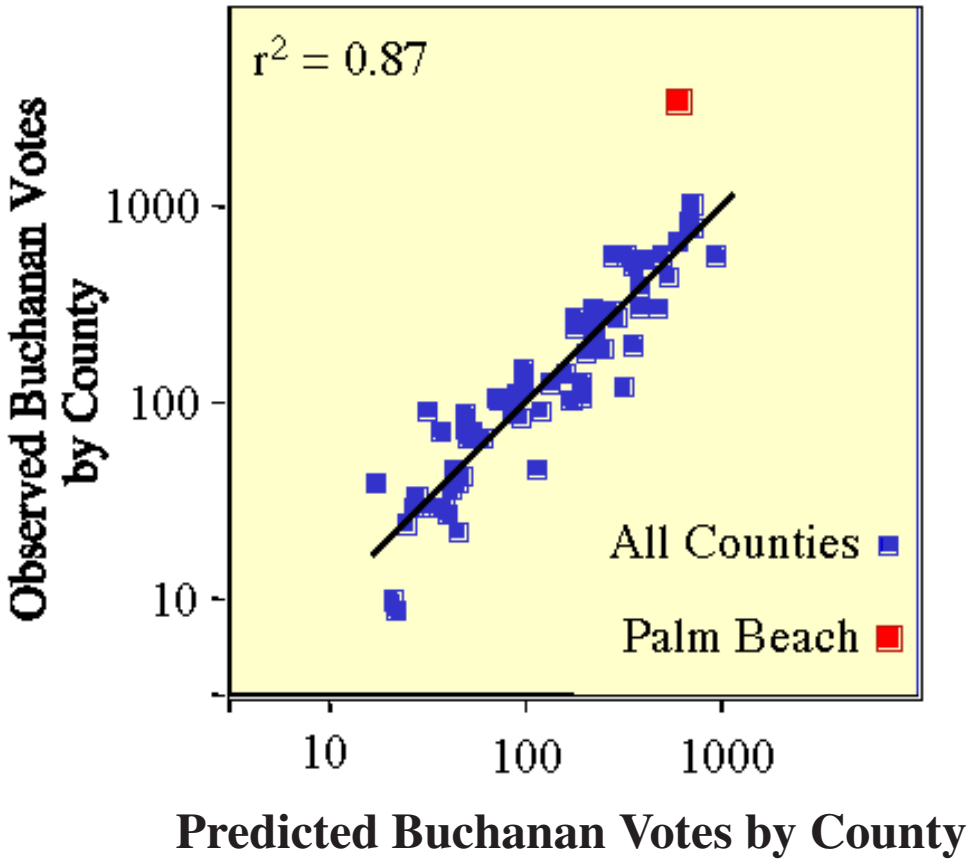
Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."



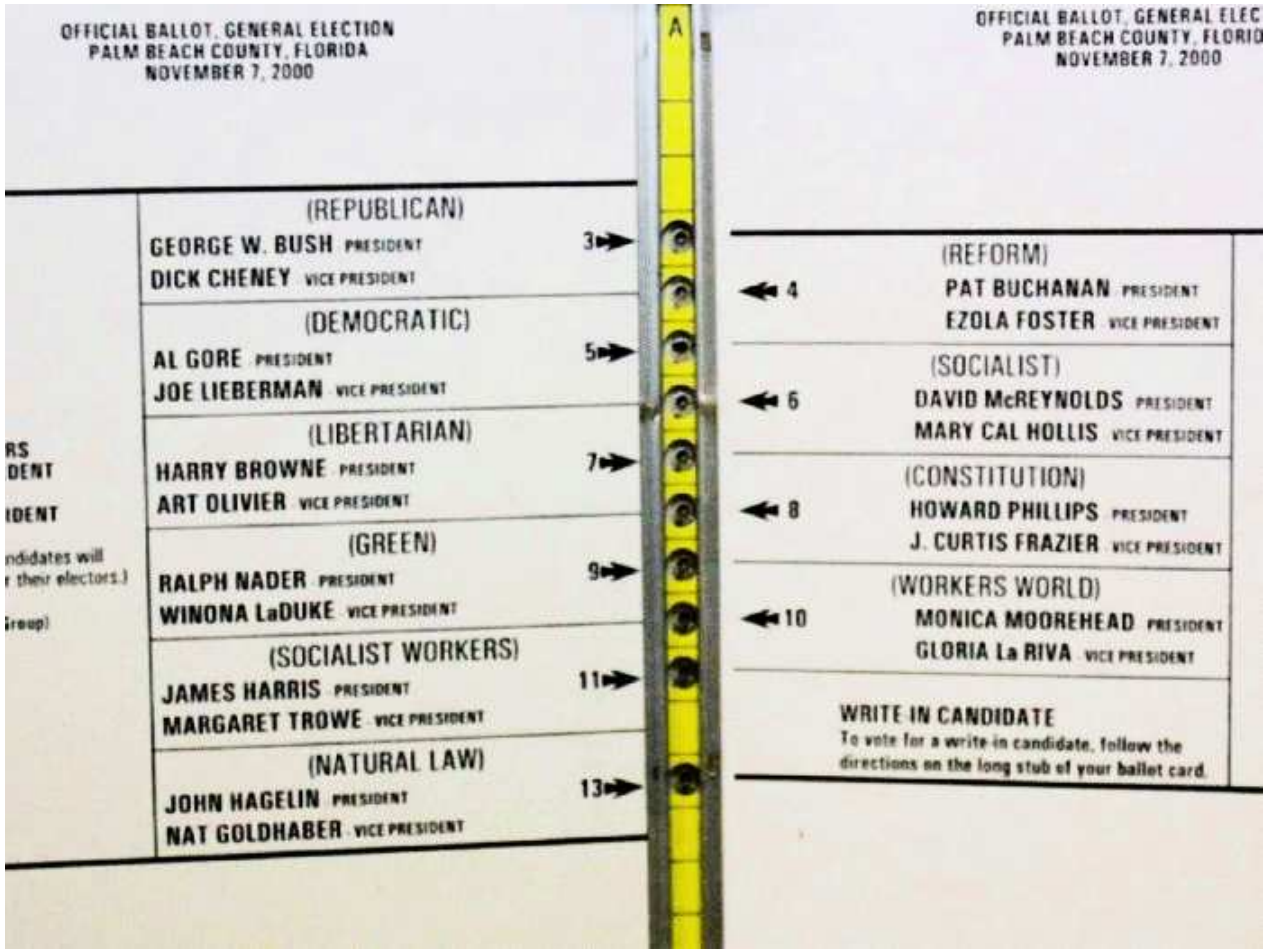
Since there can be noise in the measurements, in practice need to make a tradeoff between simplicity of the hypothesis and how well it fits the data.

Outliers



[faculty.washington.edu/mtbrett]

Butterfly Ballot



[faculty.washington.edu/mtbrett]

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Restaurant Training Data

	Alt	Bar	F/S	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

Attribute-based representations

- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

	Alt	Bar	F/S	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

Attribute-based representations

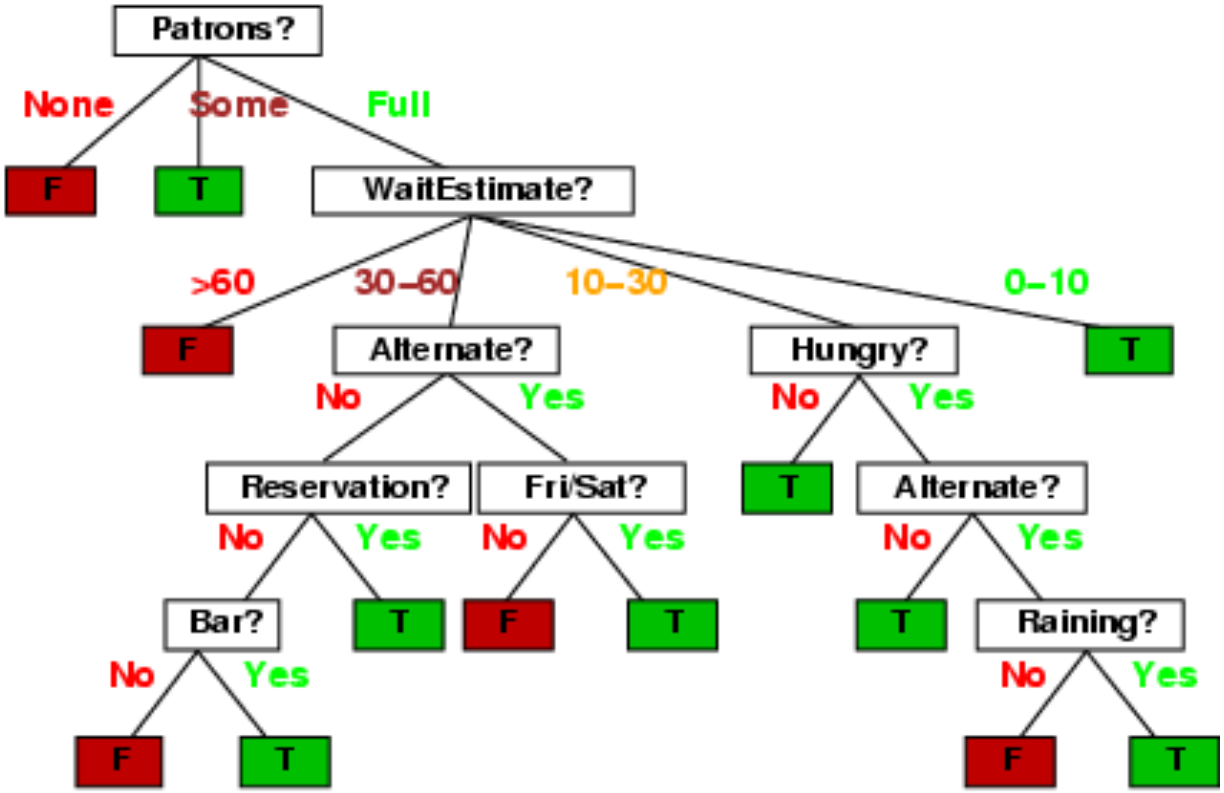
- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

	Alt	Bar	F/S	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

- Classification of examples is positive (T) or negative (F)

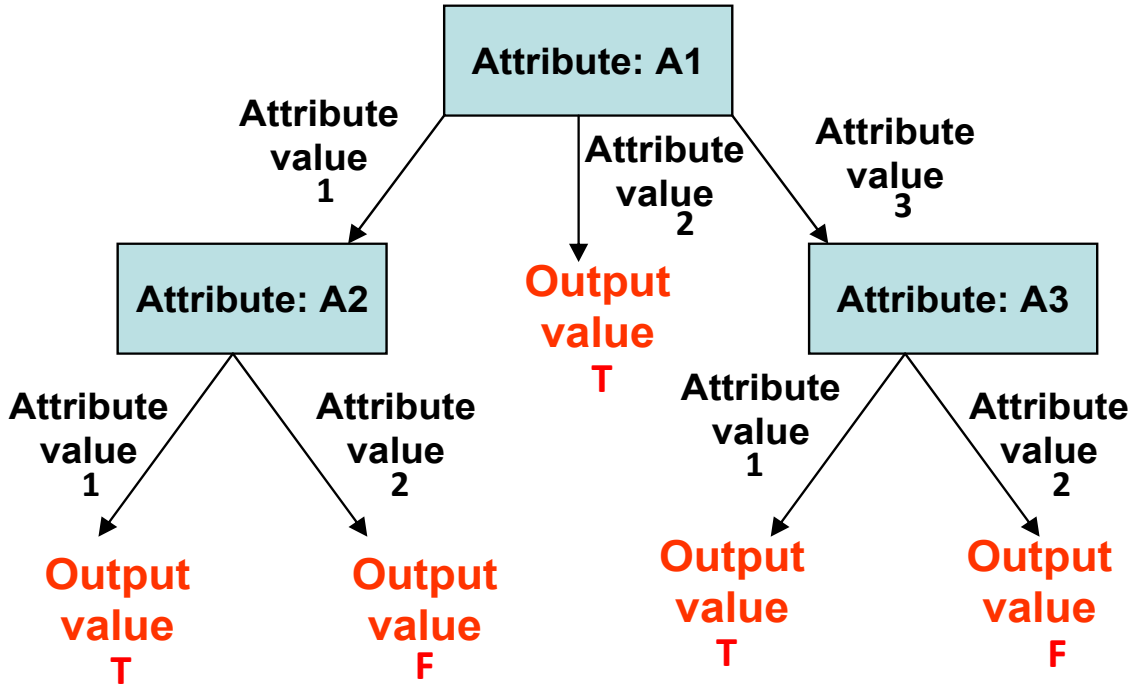
Decision trees

- One possible representation for hypotheses
  - E.g., here is the "true" tree for deciding whether to wait:



Decision trees - general

- One possible representation for hypotheses



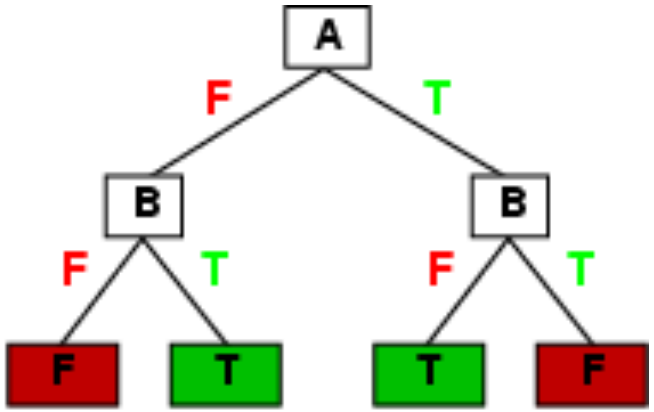
Generalization

- Provided the training data are not inconsistent, we can split the attributes in any order and still produce a tree that correctly classifies all examples in the training set.
- However, we really want a tree which is likely to generalize to correctly classify the (unseen) examples in the test set.
- In view of Ochkam's Razor, we prefer a simpler hypothesis, i.e. a smaller tree.
- But how can we choose attributes in order to produce a small tree?

Expressiveness

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

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



- 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

- Prefer to find more compact decision trees

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., *Hungry* $\wedge$ $\neg$*Rain*)?

- Each attribute can be in (positive), in (negative), or out

  $\Rightarrow 3^n$ distinct conjunctive hypotheses
- More expressive hypothesis space
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent with training set

    $\Rightarrow$ may get worse predictions

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

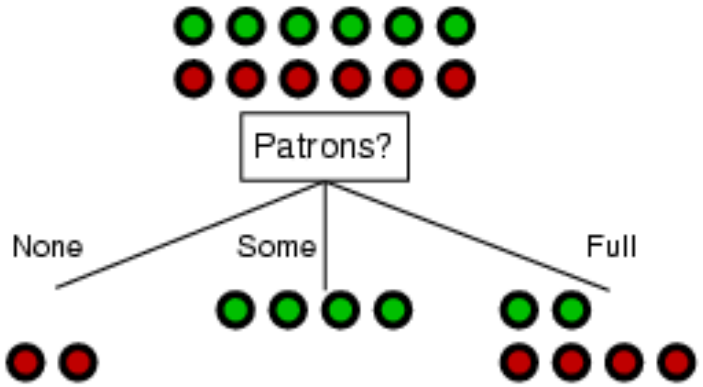
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

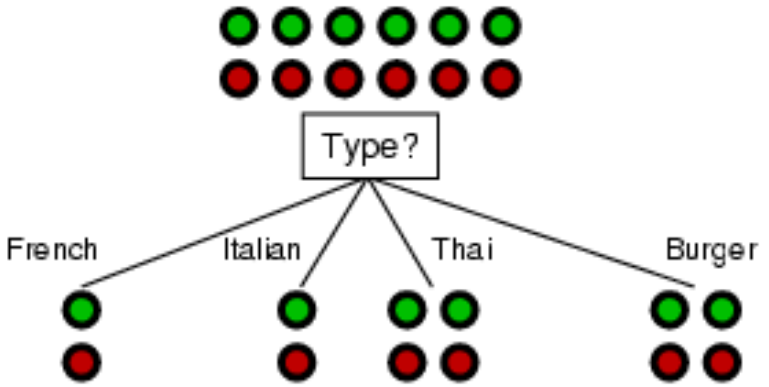
Choosing an attribute

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

Choosing an attribute

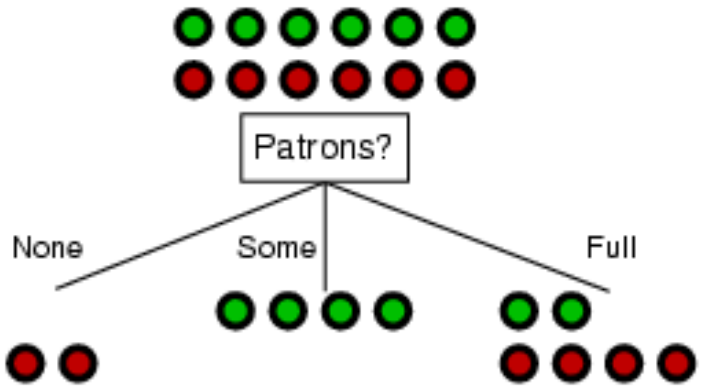
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

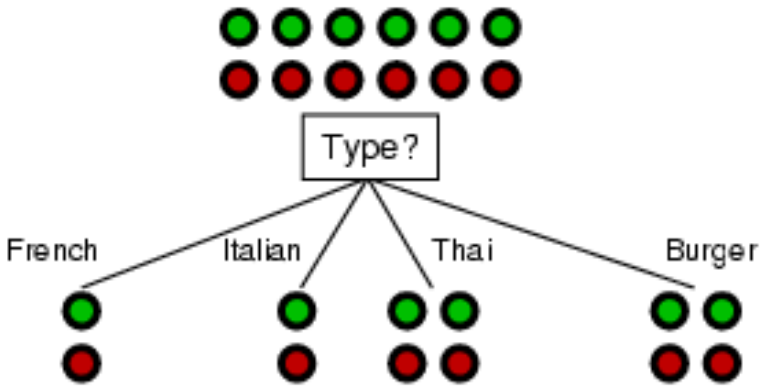




Choosing an attribute

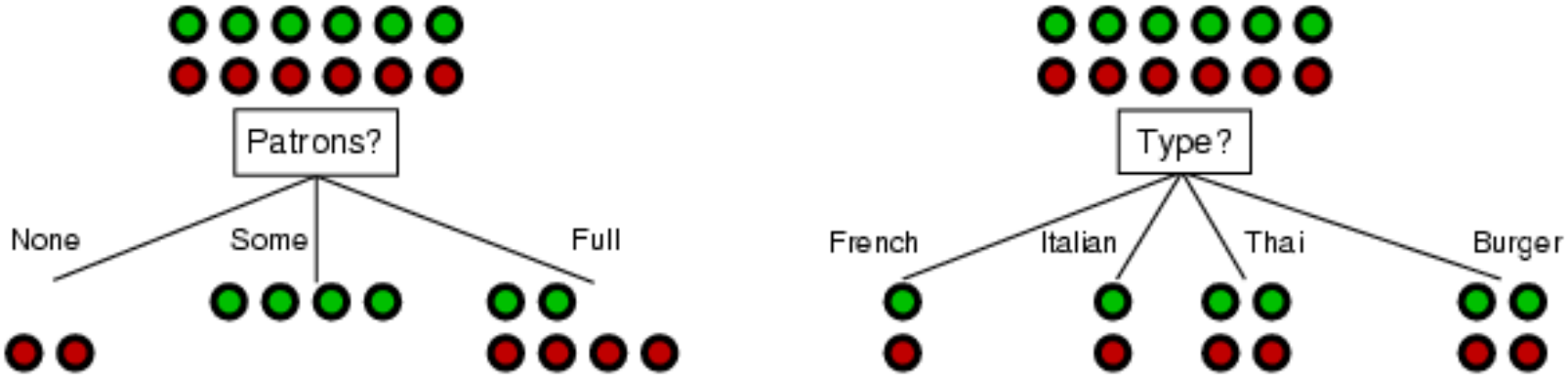
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





- *Patrons?* is a better choice

Choosing an attribute



- *Patrons?* is a better choice
- Patrons is a "more informative" attribute than Type, because it splits the examples more nearly into sets that are "all positive" or "all negative".
- This notion of "informativeness" can be quantified using the mathematical concept of "entropy".
- A parsimonious tree can be built by minimizing the entropy at each step

Entropy

Entropy is a measure of how much information we gain when the target attribute is revealed to us. In other words, it is not a measure of how much we know, but of how much we don't know.

If the prior probabilities of the n target attribute values are $p_1, \ldots, p_n$ then the entropy is

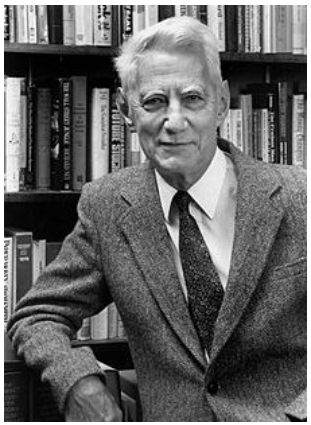
$$H(\langle p_1, \ldots, p_n \rangle) = \sum_{i=1} -p_i \log_2 p_i$$

Information Gain

Information gain is based on information theory concept called ***Entropy***

In information theory, the **Shannon entropy or information entropy** is a measure of the **uncertainty** associated with a random variable.

- It quantifies the information contained in a message, usually in bits or bits/symbol.
- It is the minimum message length necessary to communicate information.



Claude Elwood Shannon (April 30, 1916 – February 24, 2001), an American electrical engineer and mathematician, has been called "the father of information theory"

Entropy and Huffmann Coding

Entropy is the number of bits per symbol achieved by a (block) Huffman Coding scheme.

Example 1: $H(\langle 0.5, 0.5 \rangle) = 1$ bit.

Suppose we want to encode, in zeroes and ones, a long message composed of the two letters A and B, which occur with equal frequency. This can be done efficiently by assigning A=0, B=1. In other words, one bit is needed to encode each letter.

Entropy and Huffmann Coding

Example 2: $H(\langle 0.5, 0.25, 0.25 \rangle) = 1.5$ bits.

Suppose we need to encode a message consisting of the letters A, B and C, and that B and C occur equally often but A occurs twice as often as the other two letters. In this case, the most efficient code would be A=0, B=10, C=11. The average number of bits needed to encode each letter is 1.5 .

If the letters occur in some other proportion, we would need to "block" them together in order to encode them efficiently. But, the average number of bits required by the most efficient coding scheme is given by

$$H(\langle p_1, \ldots, p_n \rangle) = \sum_{i=1}^{n} -p_i \log_2 p_i$$

Entropy

- Suppose we have p positive and n negative examples at a node.
  $\rightarrow H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example.
  - e.g. for 12 restaurant examples, $p = n = 6$ so we need 1 bit.

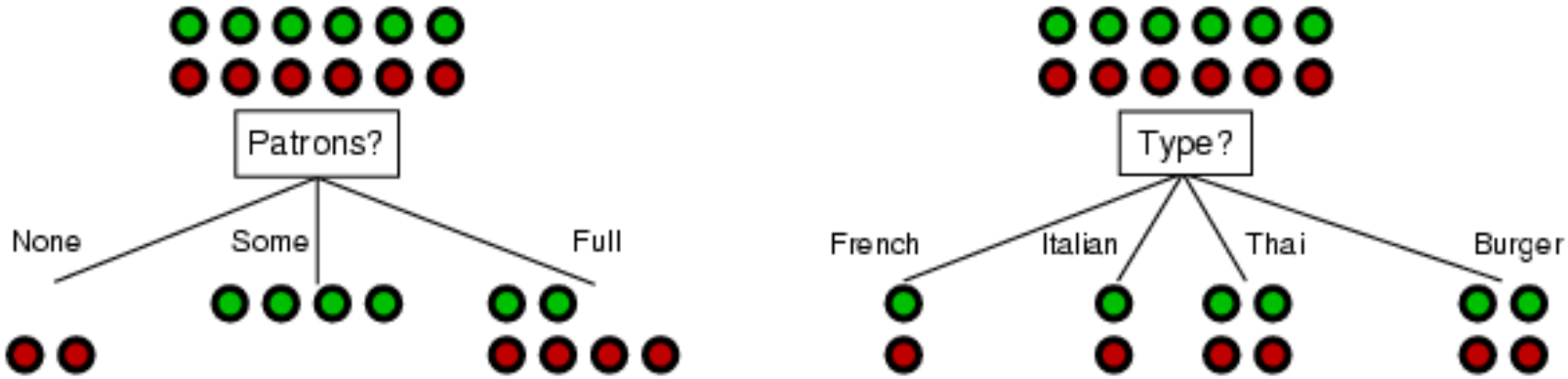
- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification.

  Let E_i have p_i positive and n_i negative examples
  $\rightarrow H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
  $\rightarrow$ expected number of bits per example over all branches is

$$\sum_i \frac{p_i+n_i}{p+n} H(\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\rangle)$$

For Patrons, this is 0.459 bits, for Type this is (still) 1 bit.

Choosing and Attribute

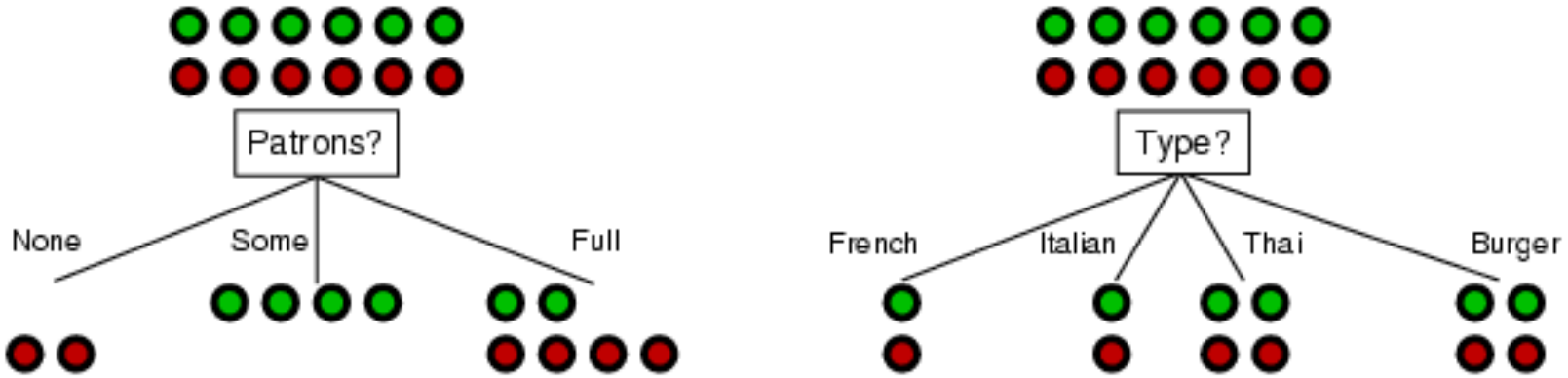


$$\text{For Patrons, Entropy} = \frac{1}{6}(0)+\frac{1}{3}(0)+\frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3})-\frac{2}{3}\log(\frac{2}{3})\right]$$

$$= 0+0+\frac{1}{2}\left[\frac{1}{3}(1.585)+\frac{2}{3}(0.585)\right] = 0.459$$

$$\text{For Type, Entropy} = \frac{1}{6}(1)+\frac{1}{6}(1)+\frac{1}{3}(1)+\frac{1}{3}(1) = 1$$

Choosing and Attribute

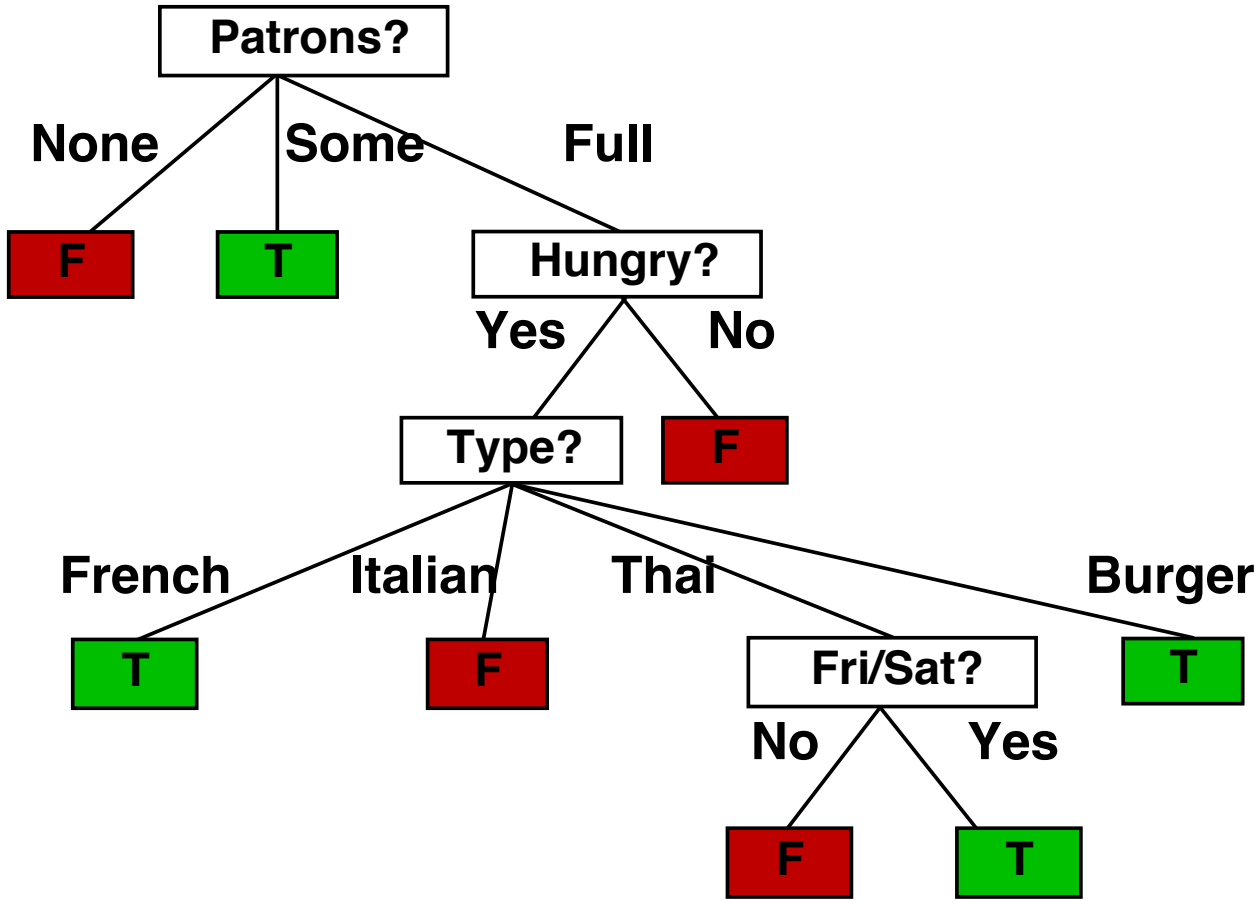


$$\text{For Patrons, Entropy} = \frac{1}{6}(0)+\frac{1}{3}(0)+\frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3})-\frac{2}{3}\log(\frac{2}{3})\right]$$

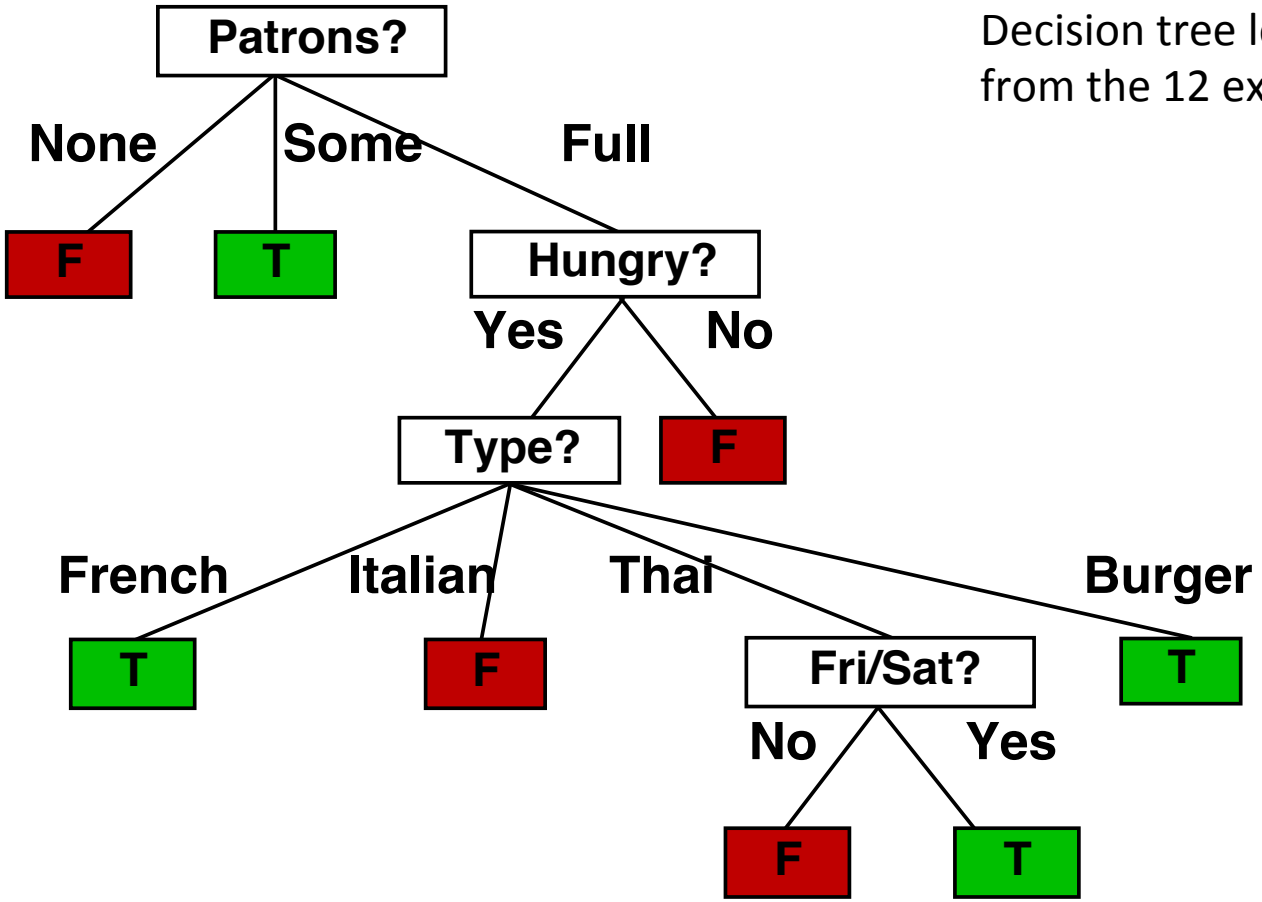
$$= 0+0+\frac{1}{2}\left[\frac{1}{3}(1.585)+\frac{2}{3}(0.585)\right] = 0.459$$

$$\text{For Type, Entropy} = \frac{1}{6}(1)+\frac{1}{6}(1)+\frac{1}{3}(1)+\frac{1}{3}(1) = 1$$

Induced Tree

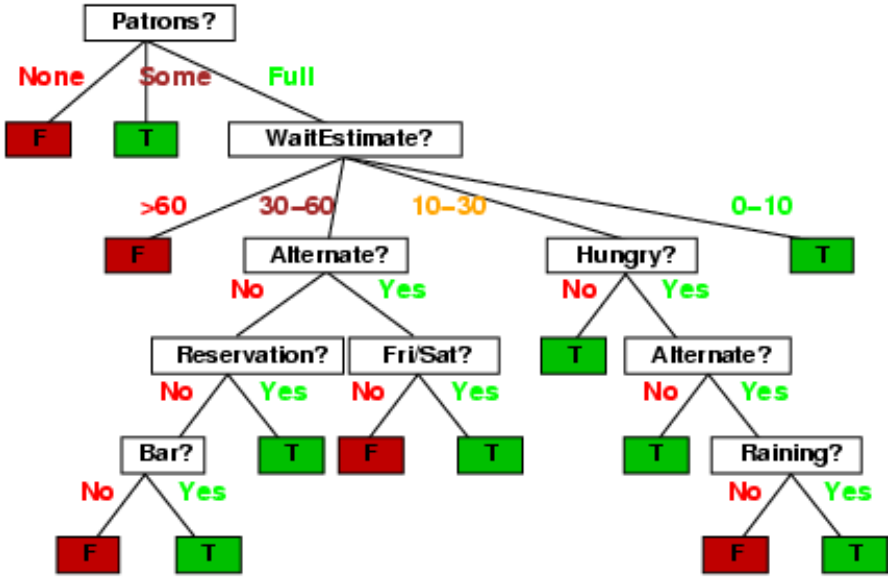


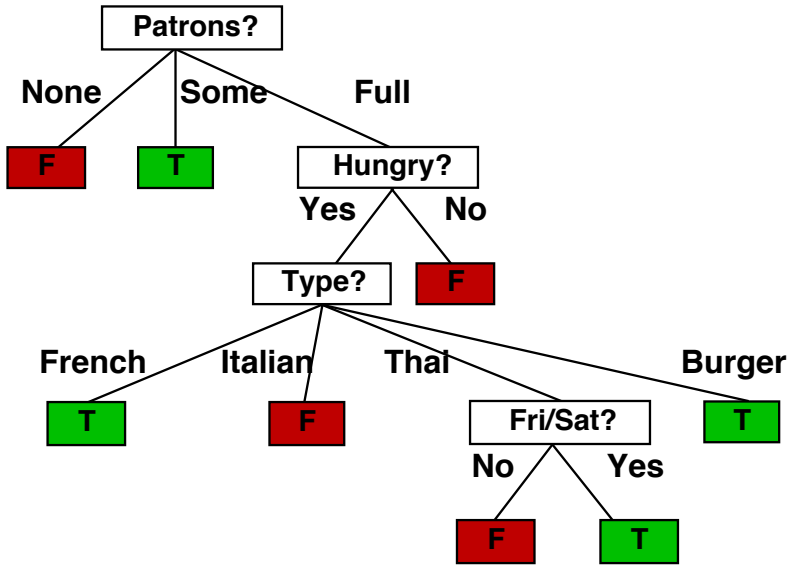
Induced Tree



Decision tree learned from the 12 examples.

Induced Tree





Decision Trees

Decision tree learning is a method for approximating discrete value target functions, in which the learned function is represented by a decision tree.

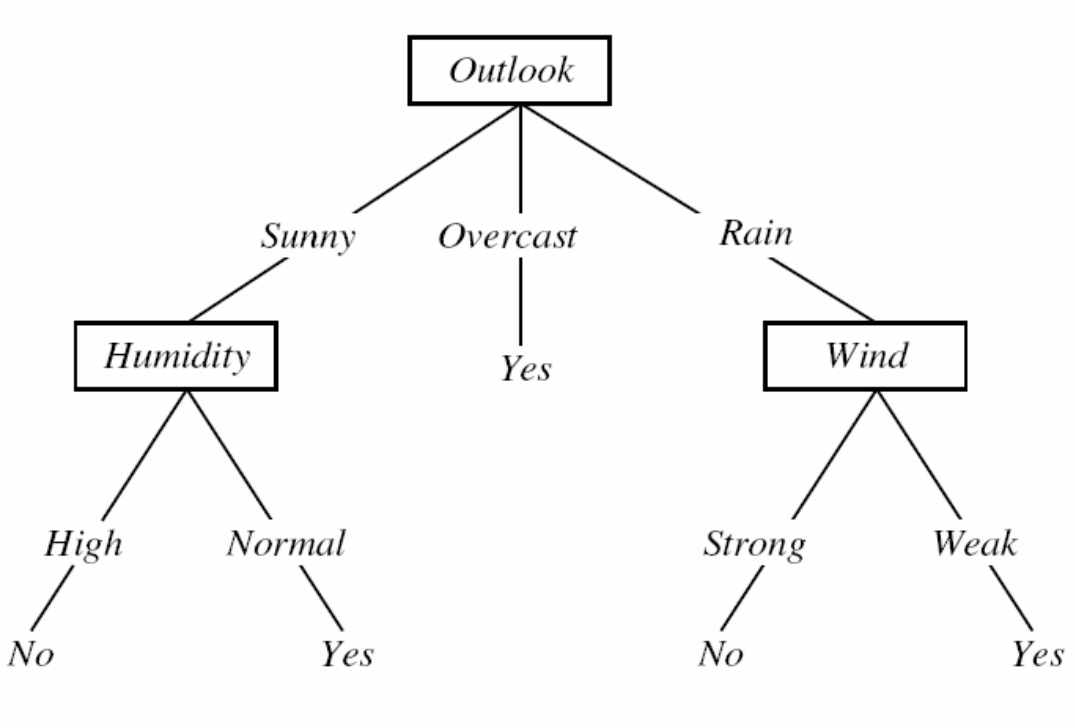
Decision trees can also be represented by if-then-else rule.

Decision tree learning is one of the most widely used approach for inductive inference .

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenis
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

Decision tree



Using the tree
An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node.

Basic Decision Tree Learning Algorithm

- ID3 Algorithm (Quinlan 1986) and it's successors C4.5 and C5.0
- ***Employs a top-down induction***

[ID3, C4.5, Quinlan]

node = Root

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 perfectly classified, Then STOP, Else iterate over new leaf nodes

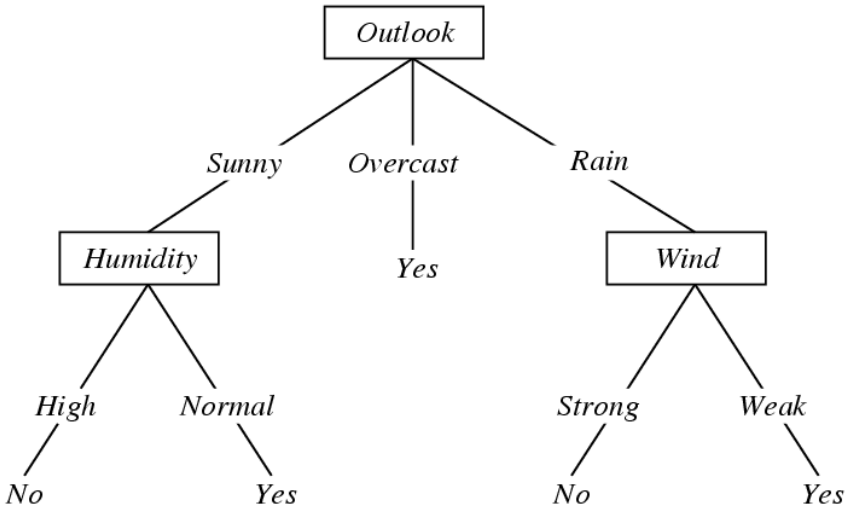
- Greedy search the space of possible decision trees.
- The algorithm never backtracks to reconsider earlier choices.

http://www.rulequest.com/Personal/

Example – Pay Tenis?

A Decision tree for

f: <Outlook, Humidity, Wind, Temp> → PlayTennis?



More generally, f: <X_1, … X_n> → Y

Each internal node: discrete test on one attribute, X_i

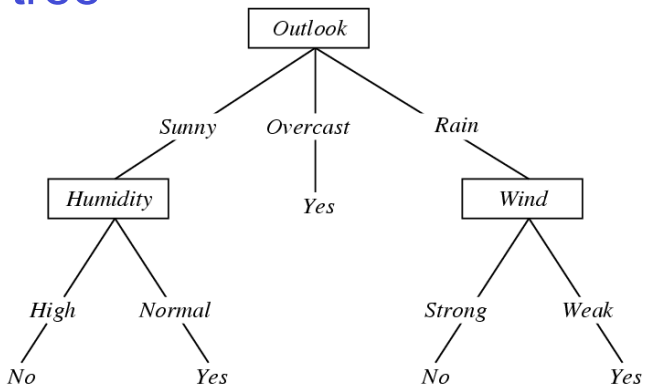
Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or P(Y|X ∈ leaf))

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=1 if we play tennis on this day, else 0
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree



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

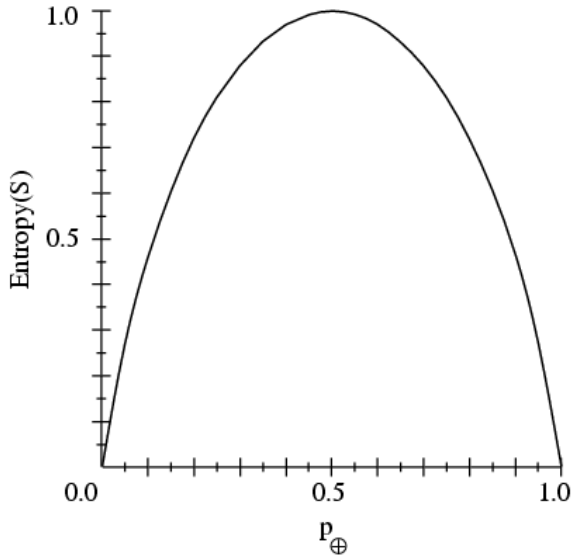
Input:

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

Output:

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

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

- Information gain is based on information theory concept called ***Entropy***

Entropy

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)

Why? Information theory:

- Most efficient possible 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 and 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 :

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

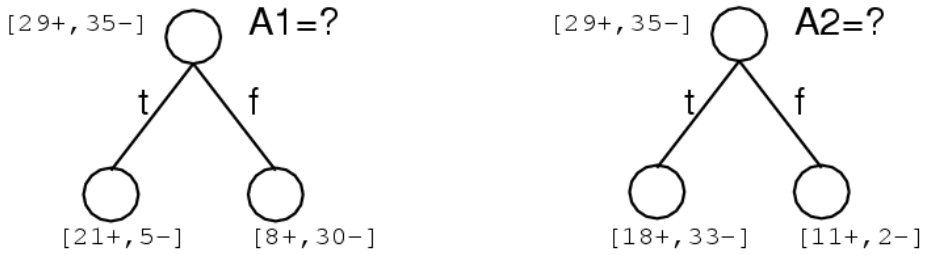
Mutual information (aka Information Gain) of X and Y :

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

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

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



The value of $Gain(S, A)$ is the number of bits saved when encoding the target value of an arbitrary member of S, by knowing the value of attribute *A*.

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenis
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

Which Attribute to Select ??

We would like to select the attribute that is most useful for classifying examples.

What is a good quantitative measure of the worth of an attribute?

ID3 uses this **information gain** measure to select among the candidate attributes at each step while growing the tree.

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenr
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

There are 14 examples. **9 positive** and **5 negative** examples [9+, 5-]**.**

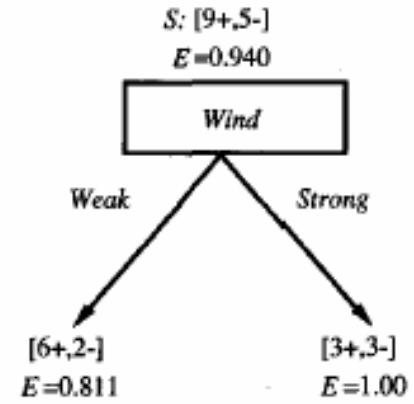
The entropy of S relative to this boolean (yes/no) classification is

$$Entropy([9+,5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14)$$
$$= 0.940$$

Gain (S, Attribute = Wind)

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

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$



Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

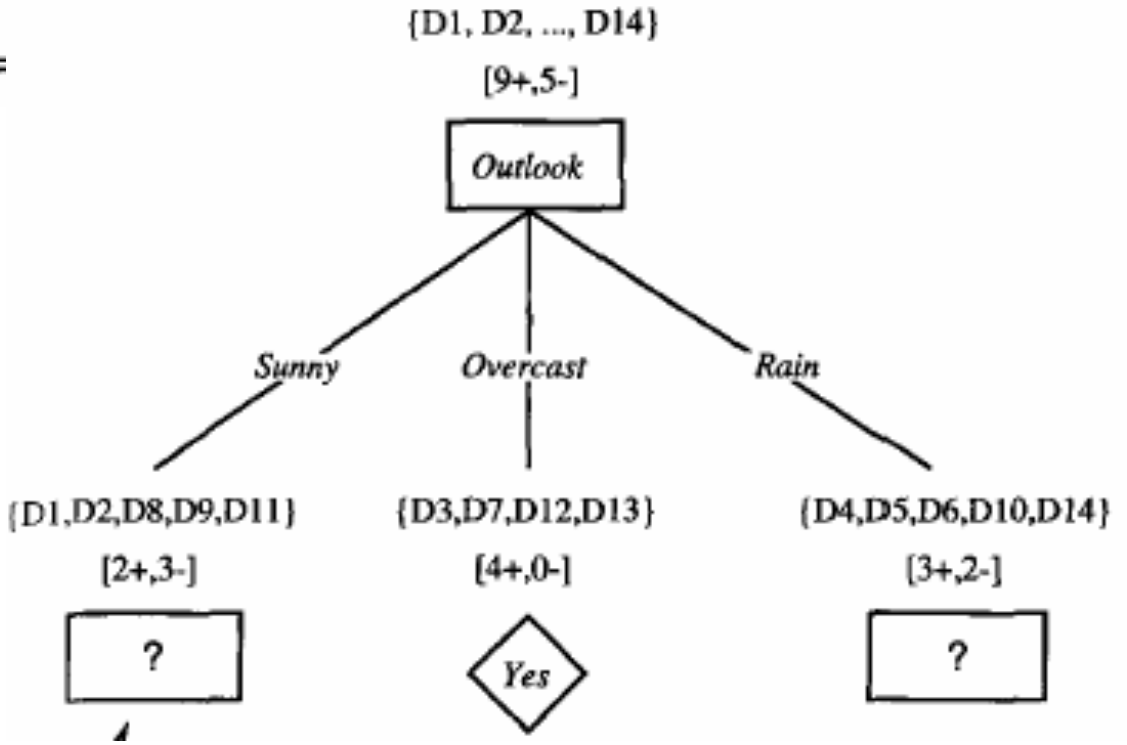
Gain (S, A)

$$Gain(S, Outlook) = 0.246$$

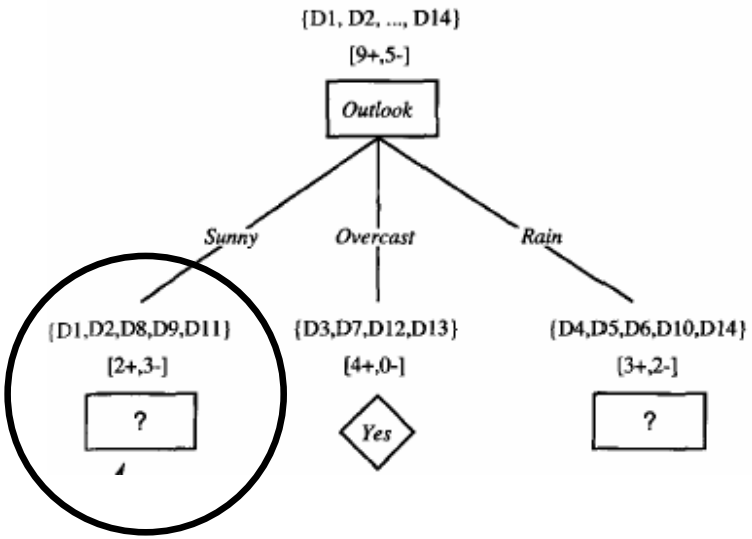
$$Gain(S, Humidity) = 0.151$$

$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) =$$



Gain (S_{Sunny}, A)



Day	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Temperature

(Hot) {0+, 2-)

(Mild) {1+, 1-}

(Cool) {1+, 0-}

Humidity

(High) {0+, 3-}

(Normal) {2+, 0-}

Wind

(Weak) {1+, 2-}

(Strong) {1+, 1-}

Gain (S_{Sunny}, A)

$$Entropy(S_{Sunny}) = - \{ 2/5 \log(2/5) + 3/5 \log(3/5)\} = 0.97095$$

Temperature
(Hot) {0+, 2-)
(Mild) {1+, 1-}
(Cool) {1+, 0-}

Entropy(Hot) = 0

Entropy(Mild) = 1

Entropy(Cool) = 0

Gain(S1, Temperature) = 0.97095 – 2/5*0 – 2/5*1 – 1/5*0 = 0.57095

Humidity
(High) {0+, 3-}
(Normal) {2+, 0-}

Entropy(High) = 0

Entropy(Normal) = 0

Gain(S1, Humidity) = 0.97095 – 3/5*0 – 2/5*0 = **0.97095**

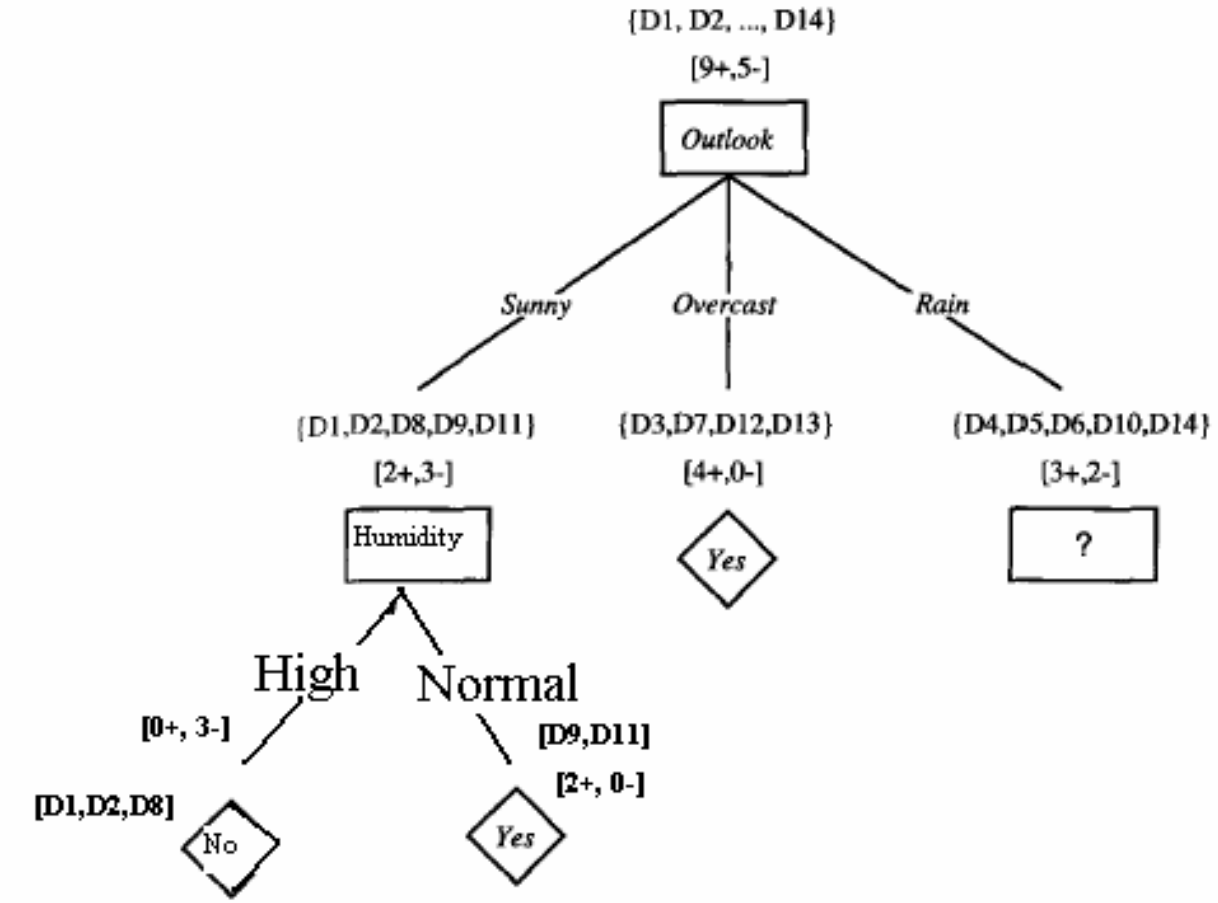
Wind
(Weak) {1+, 2-}
(Strong) {1+, 1-}

Entropy(Weak) = 0.9183

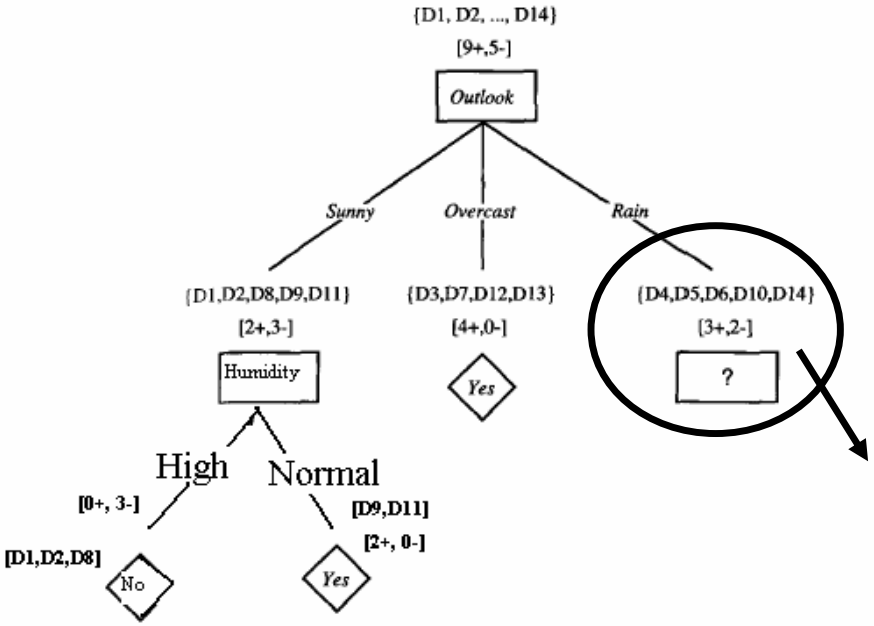
Entropy(Normal) = 1.0

Gain(S1, Wind) = 0.97095 – 3/5*0.9183 – 2/5*1 = 0.01997

Modified Decision Tree



Gain (S_{Rain}, A)



Day	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
~~D1~~	~~Hot~~	~~High~~	~~Weak~~	~~No~~
~~D2~~	~~Hot~~	~~High~~	~~Strong~~	~~No~~
~~D3~~	~~Hot~~	~~High~~	~~Weak~~	~~Yes~~
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
~~D7~~	~~Cool~~	~~Normal~~	~~Strong~~	~~Yes~~
~~D8~~	~~Mild~~	~~High~~	~~Weak~~	~~No~~
~~D9~~	~~Cool~~	~~Normal~~	~~Weak~~	~~Yes~~
D10	Mild	Normal	Weak	Yes
~~D11~~	~~Mild~~	~~Normal~~	~~Strong~~	~~Yes~~
~~D12~~	~~Mild~~	~~High~~	~~Strong~~	~~Yes~~
~~D13~~	~~Hot~~	~~Normal~~	~~Weak~~	~~Yes~~
D14	Mild	High	Strong	No

Temperature
(Hot) {0+, 0-)
(Mild) {2+, 1-}
(Cool) {1+, 1-}

Humidity
(High) {1+, 1-}
(Normal) {2+, 1-}

Wind
(Weak) {3+, 0-}
(Strong) {0+, 2-}

Gain (S_{Rain}, A)

$$\text{Entropy}(S_{Rain}) = -\{ 2/5 \log(2/5) + 3/5 \log(3/5)\} = 0.97095$$

Temperature
(Hot) {0+, 0-)
(Mild) {2+, 1-}
(Cool) {1+, 1-}

Entropy(Hot) = 0
Entropy(Mild) = 0.1383
Entropy(Cool) = 1.0
Gain(S1, Temperature) = 0.97095 – 0 – 2/3*0.1383 - 2/5*1 = 0.4922

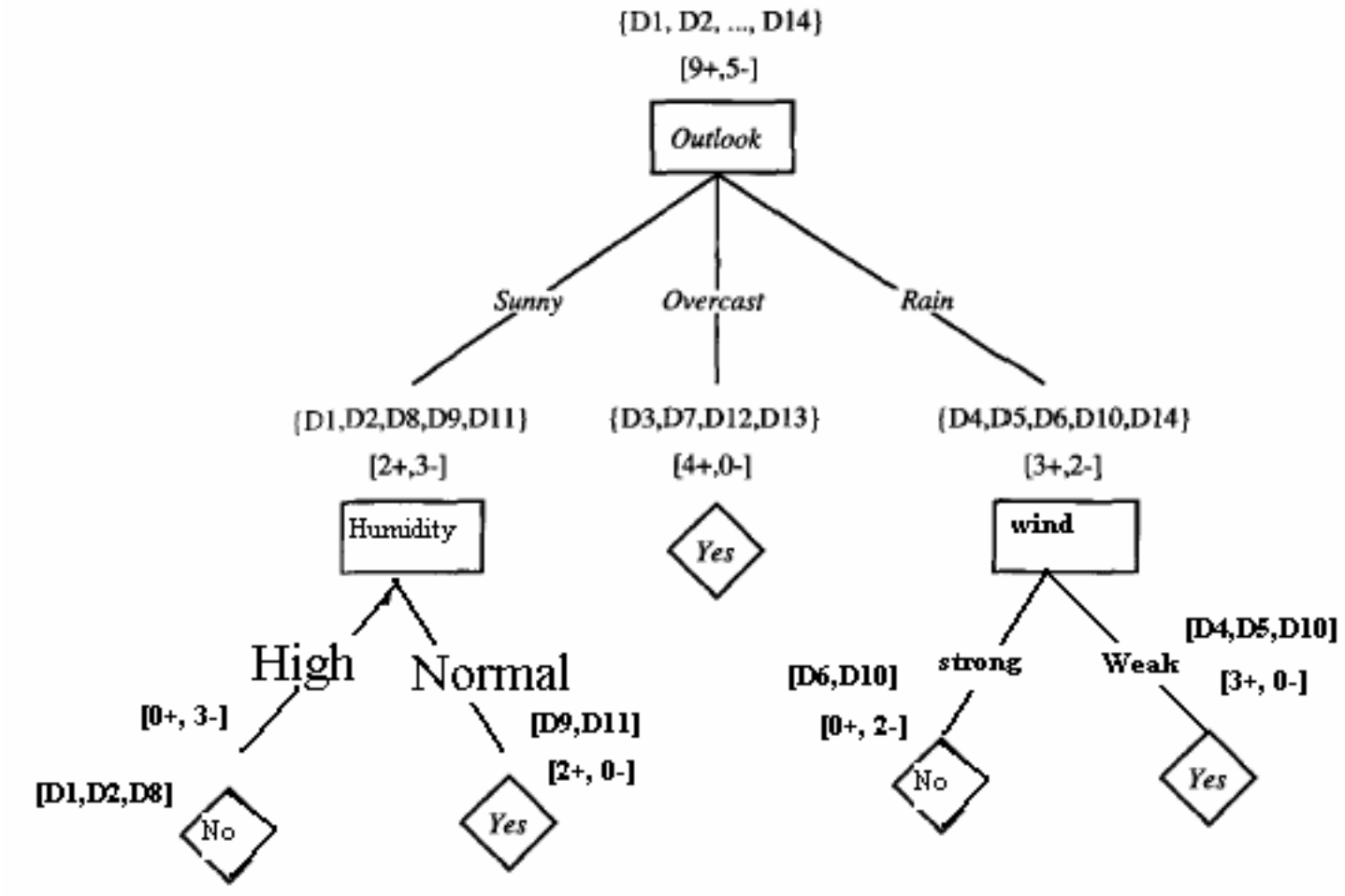
Humidity
(High) {1+, 1-}
(Normal) {2+, 1-}

Entropy(High) = 1.0
Entropy(Normal) = 0.1383
Gain(S1, Humidity) = 0.97095 – 2/5*1.0 – 3/5*0.1383 = 0.4922

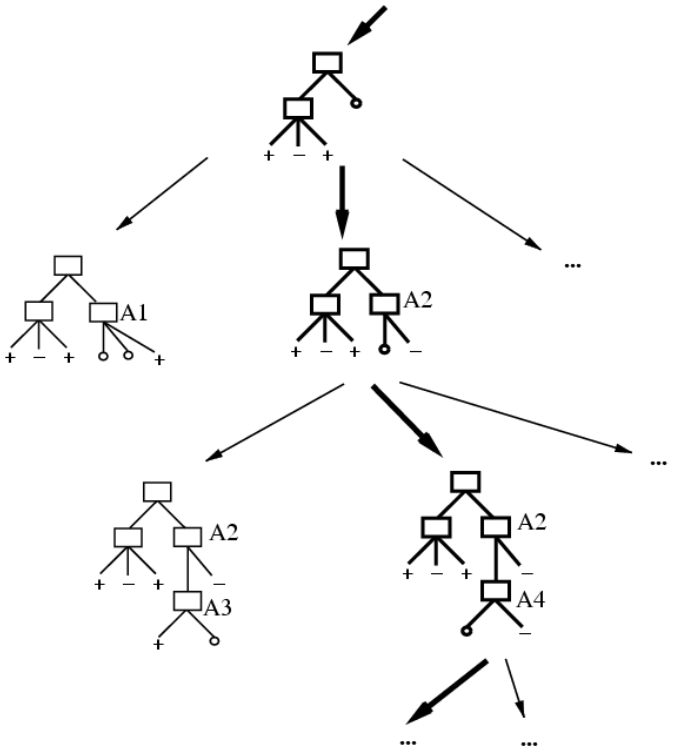
Wind
(Weak) {3+, 0-}
(Strong) {0+, 2-}

Entropy(Weak) = 0.0
Entropy(Normal) = 0.0
Gain(S1, Humidity) = 0.97095 - 3/5*0 – 2/5*0 = **0.97095**

Final Decision Tree

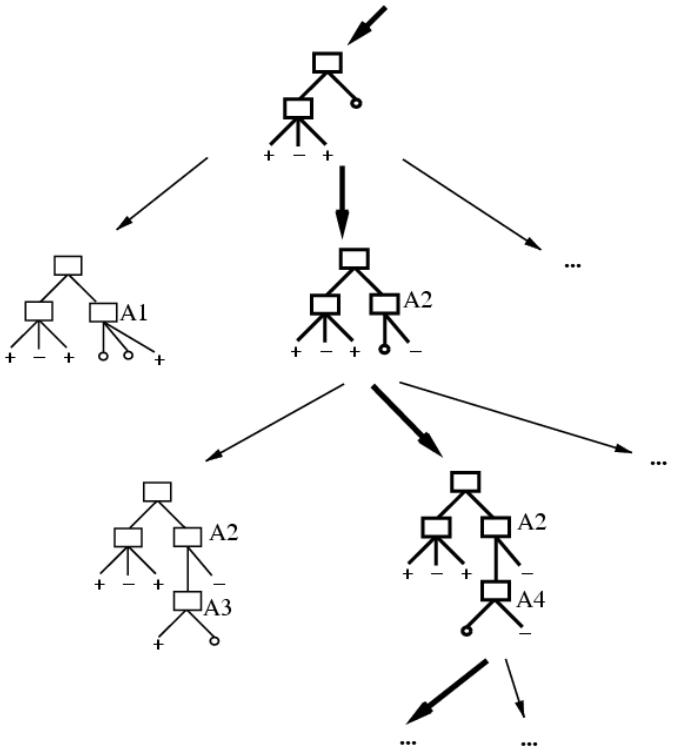


Which Tree Should We Output?



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

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

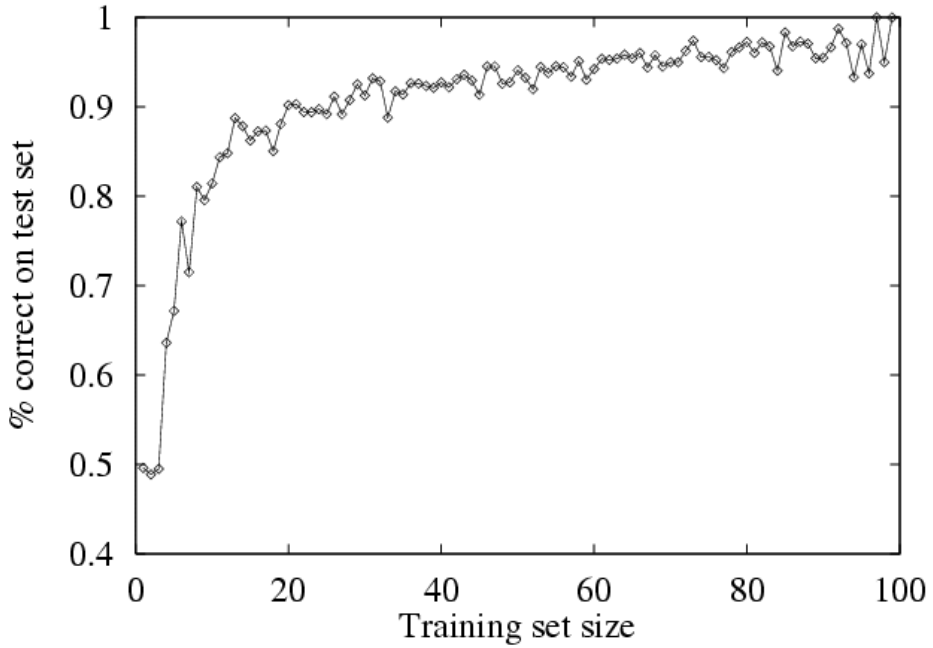
Performance measurement

How do we know that $h \approx f$?

1. Use theorems of computational/statistical learning theory
2. Try h on a new test set of examples

   (use same distribution over example space as training set)

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



Training and Testing

For classification problems, a classifier's performance is measured in terms of the *error rate*.

The classifier predicts the class of each instance: if it is correct, that is counted as a *success;* if not, it is an *error.*

The error rate is just the proportion of errors made over a whole set of instances, and it measures the overall performance of the classifier.

Training and Testing

- Self-consistency Test: When training and test dataset are same
- Hold out Strategy: Holdout method reserves a certain amount for testing and uses the remainder for training (and sets part of that aside for validation, if required).
  - In practical scenario we have limited number of example with us...
- K-fold Cross validation technique:
  - In the k-fold cross-validation, the dataset was partitioned randomly into k equal-sized sets. The training and testing of each classifier were carried out k times using one distinct set for testing and other k-1 sets for training.

K-Fold Cross Validation

Idea: train multiple times, leaving out a disjoint subset of data each time for test. Average the test set accuracies.

Partition data into K disjoint subsets
For k=1 to K

testData = kth subset

h←classifier trained on all data except for testData

accuracy(k) = accuracy of h on testData

end

FinalAccuracy = mean of the K recorded testset accuracies

Leave-One-Out Cross Validation

This is just k-fold cross validation leaving out one example each iteration. Average the test set accuracies.

Partition data into K disjoint subsets *each containing one example*

For k=1 to K

testData = kth subset

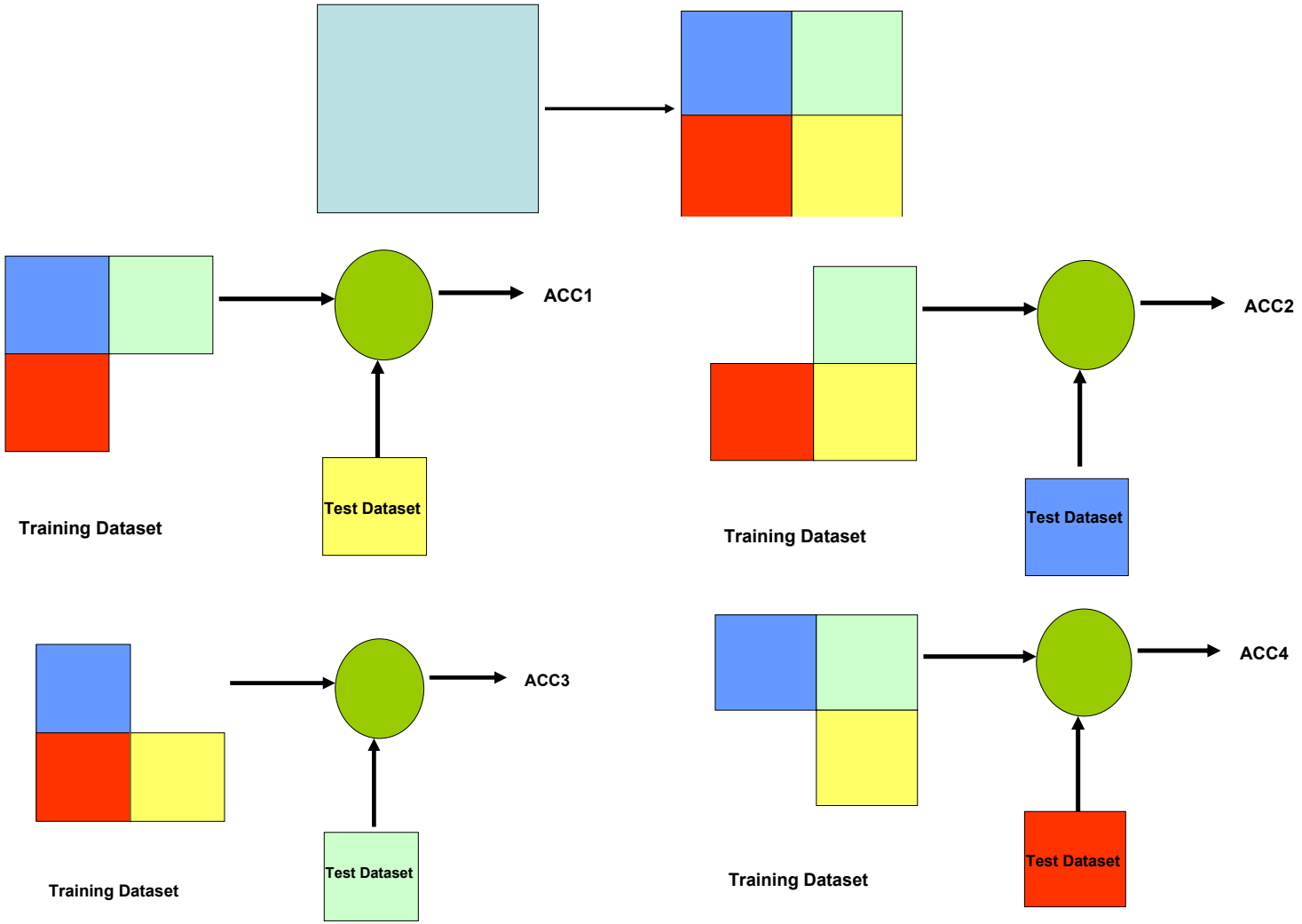
h←classifier trained on all data except for testData

accuracy(k) = accuracy of h on testData

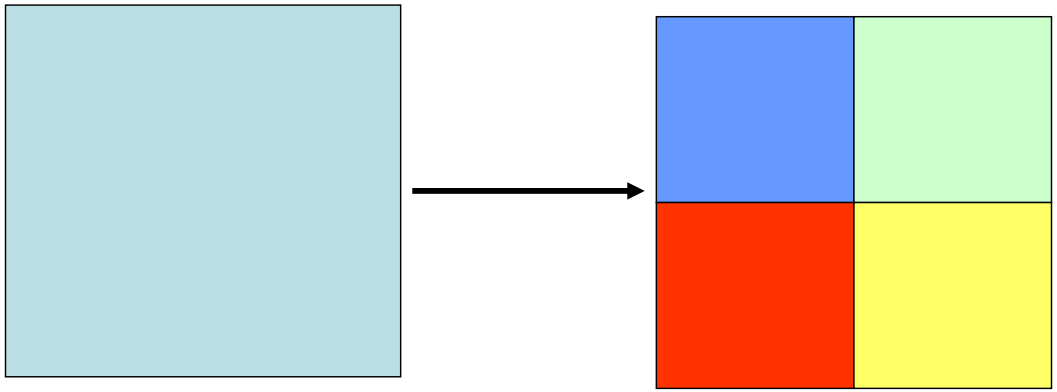
end

FinalAccuracy = mean of the K recorded testset accuracies

4-Fold Cross-validation



4-Fold Cross-validation



ACC = (ACC1 + ACC2 + ACC3 + ACC4) / 4

Learning From Noisy Data

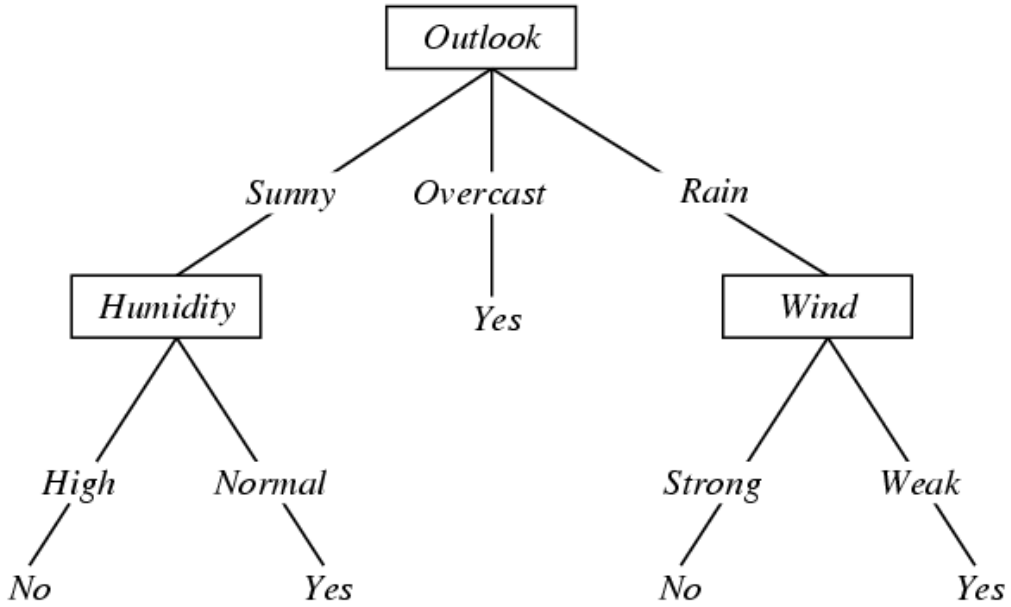
- Noise: errors in examples, incompleteness, clashes between class values
- Problems with noisy data:
  - Overfitting data, tracing noise
  - Low accuracy on new data
  - Large descriptions (trees), poor comprehensibility
- Main technique for handling noise in tree induction: tree pruning

Overfitting in Decision Trees

Consider adding noisy training example #15:

$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$

What effect on earlier tree?



Overfitting

Consider a hypothesis h and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

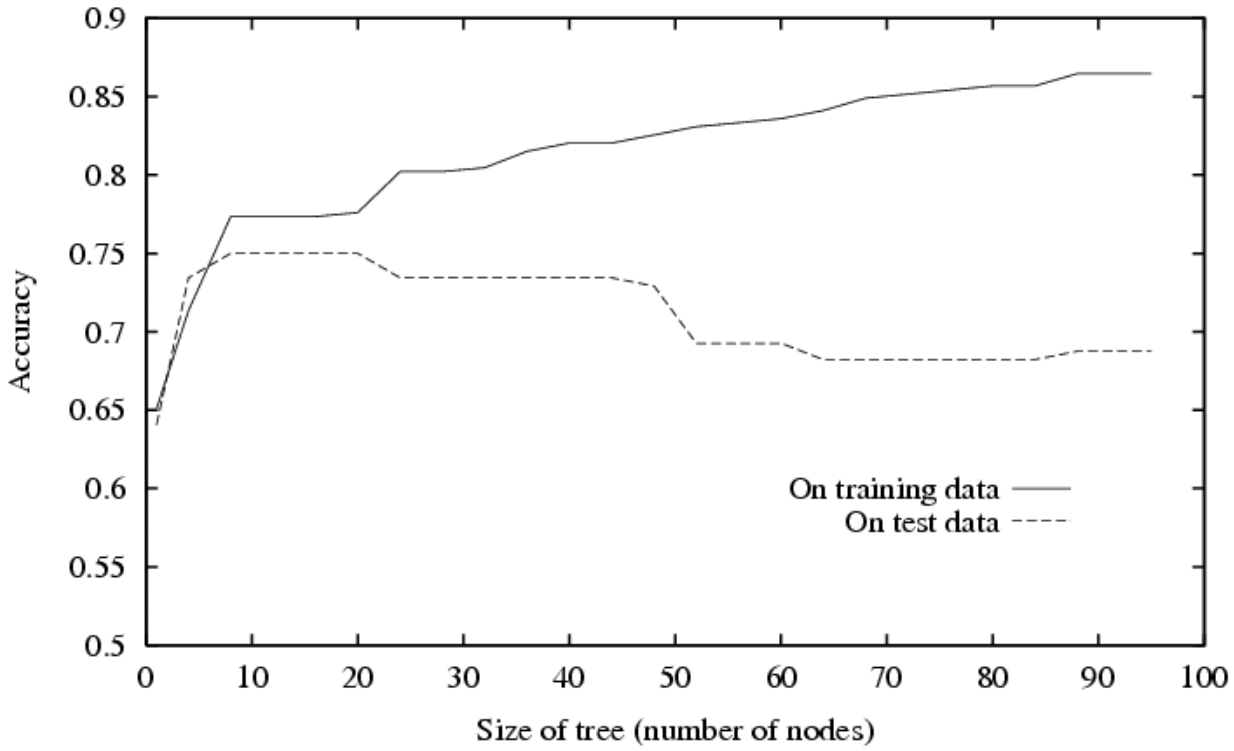
- We say h overfits the training data if

$$error_{true}(h) > error_{train}(h)$$

- Amount of overfitting =

$$error_{true}(h) - error_{train}(h)$$

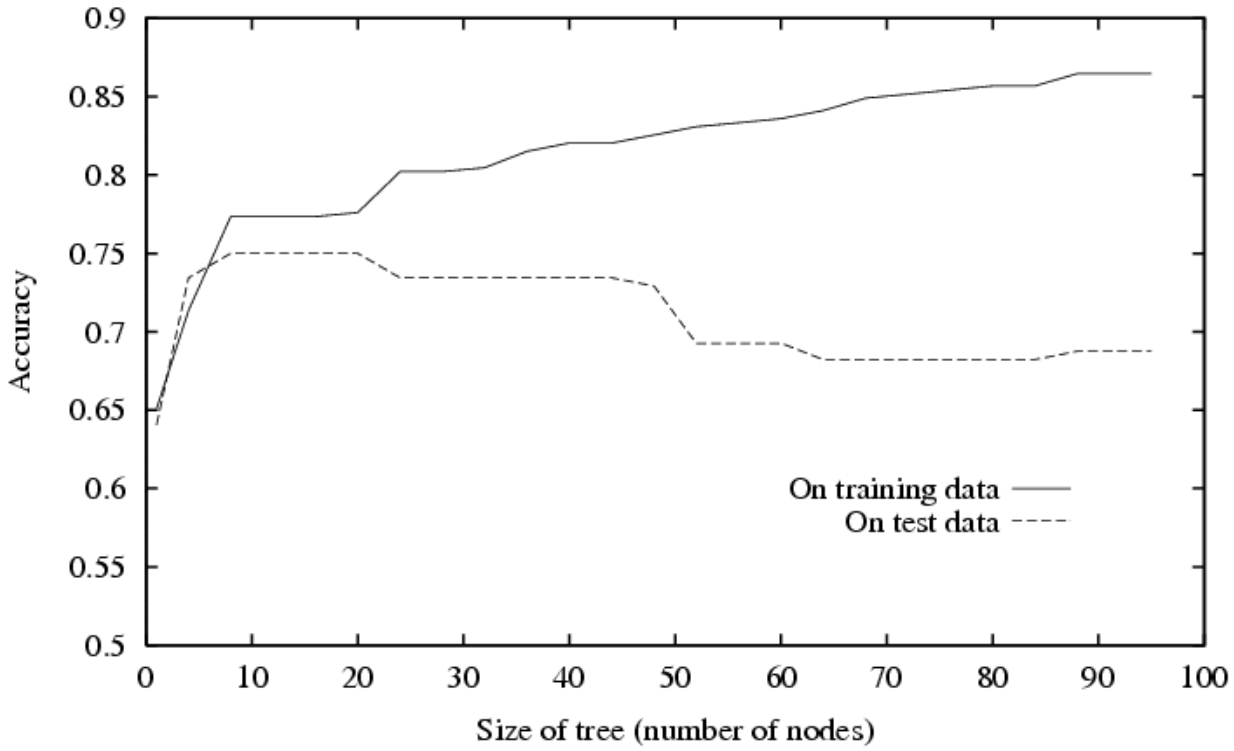
Overfitting in Decision Tree Learning



Typical relation between size and accuracy

How to prune optimally?

Overfitting in Decision Tree Learning



Typical relation between size and accuracy

How to prune optimally?

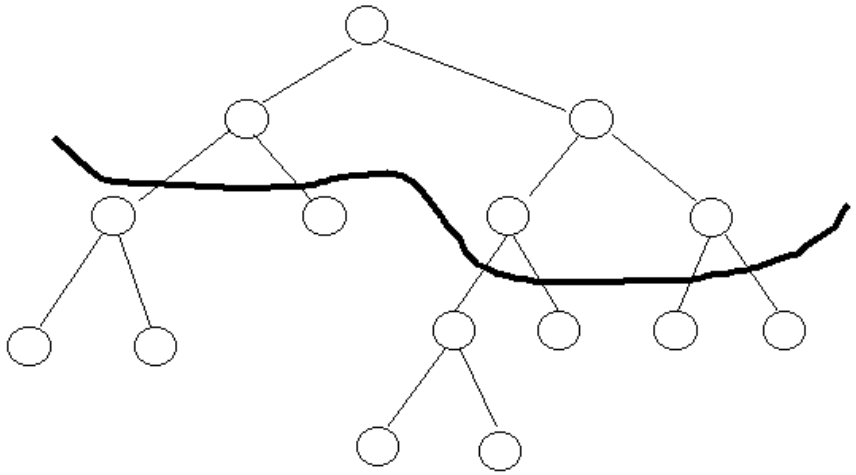
Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data spilt not statistically significant
- Grow full tree, then prost-prune

A key question is what criterion is to be used to determine the correct final tree size.

Pruning Of Trees



Prune below this

After pruning accuracy may increase!

Tree Pruning To Minimise Error

- Maximise expected accuracy, minimise expected probability of error
- For given tree, estimate its expected error
- Also estimate errors for variously pruned tree
- Choose tree with minimal estimated error

How to estimate error?

- One possibility: use "pruning set"
- Another possibility: estimate error probability from data in tree

Probability Estimates

- Laplace estimate:

- $p = (n + 1)/(N + k)$
- N = number of all trials
  n = number of successful outcomes
- k = number of possible outcomes

Laplace Error and Pruning

According to Ockham's Razor, we may wish to prune off branches that do not provide much benefit in classifying the items.

When a node becomes a leaf, all items will be assigned to the majority class at that node. We can estimate the error rate on the (unseen) test items using the Laplace error:

$$E = 1 - \frac{n+1}{N+k}$$

N = total number of (training) items at the node

n = number of (training) items in the majority class

k = number of classes

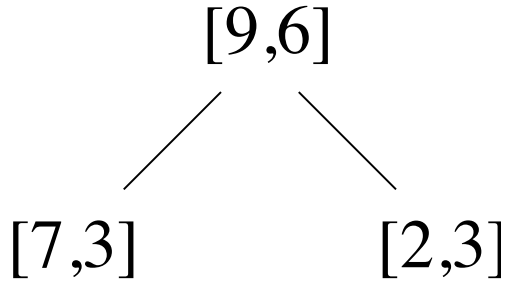
If the average Laplace error of the children exceeds that of the parent node, we prune off the children.

Minimal Error Pruning

Should the children of this node be pruned or not?

Left child has class frequencies [7,3]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{7+1}{10+2} = 0.333$$



Right child has $E = 0.429$

Parent node has $E = 0.412$

Average for Left and Right child is

$$E = \frac{10}{15}(0.333) + \frac{5}{15}(0.429) = 0.365$$

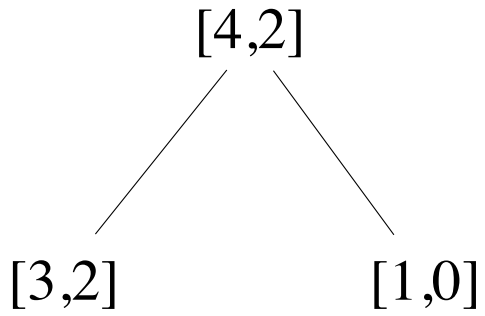
Since $0.365 < 0.412$, children should NOT be pruned.

Minimal Error Pruning

Should the children of this node be pruned or not?

Left child has class frequencies [3,2]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{3+1}{5+2} = 0.429$$



Right child has $E = 0.333$

Parent node has $E = 0.375$

Average for Left and Right child is

$$E = \frac{5}{6}(0.429) + \frac{1}{6}(0.333) = 0.413$$

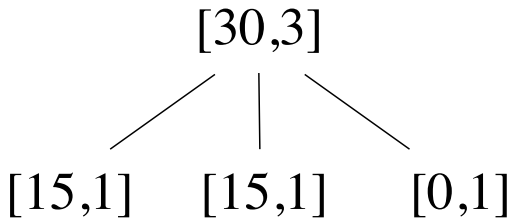
Since $0.413 > 0.375$, children should be pruned.

Minimal Error Pruning

Should the children of this node be pruned or not?

Left and Middle child have class frequencies [15,1]

$$E = 1 - \frac{n+1}{N+k} = 1 - \frac{15+1}{16+2} = 0.111$$



Right child has $E = 0.333$

Parent node has $E = \frac{4}{35} = 0.114$

Average for Left, Middle and Right child is

$$E = \frac{16}{33}(0.111) + \frac{16}{33}(0.111) + \frac{1}{33}(0.333) = 0.118$$

Since $0.118 > 0.114$, children should be pruned.

Problems Suitable for Decision Trees

Instances are represented by attribute-value pairs

Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot).

- The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold). However, extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing Temperature numerically).

The target function has discrete output values

The training data

- The training data may contain errors
- The training data may contain missing attribute values

Some TDIDT Systems

- ID3 (Quinlan 79)
- CART (Breiman et al. 84)
- Assistant (Cestnik et al. 87)
- C4.5 (Quinlan 93)
- C5 (Quinlan 97)
- Trees in WEKA (Witten, Frank 2000 - ...)

TDIDT - Top-Down Induction of Decision Trees

Inducing Regression Trees

Similar to induction of decision trees

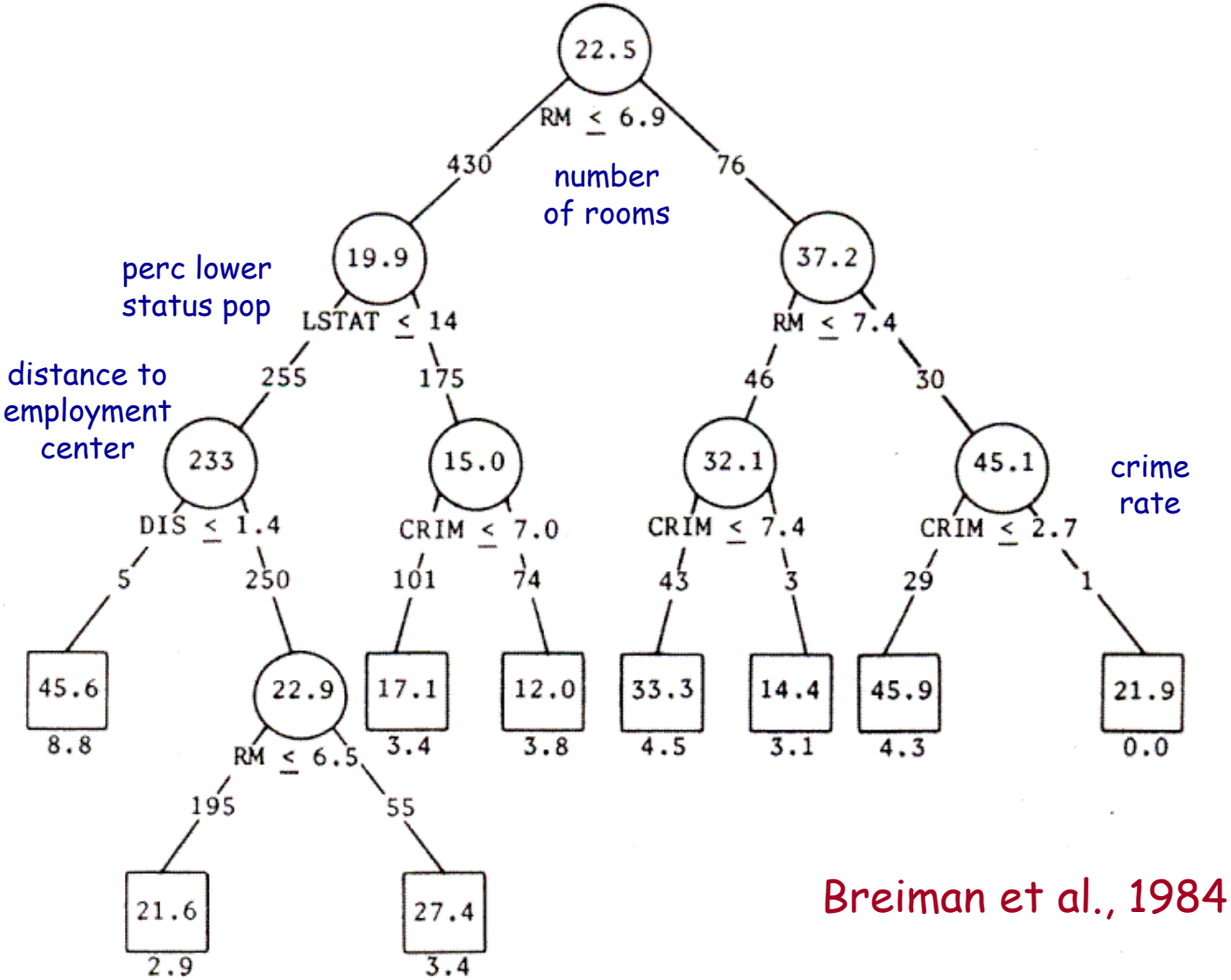
Regression trees useful in continuous domains,

- e.g. predict biomass of algae

Decision trees: discrete class

Regression trees: continuous, real-valued class

Example: Boston Housing Values



Breiman et al., 1984

Some Systems That Induce Regression Trees

- CART (Breiman et al. 1984)
- RETIS (Karalič 1992)
- M5 (Quinlan 1993)
- WEKA (Witten and Frank 2000-...)

Summary

- Supervised Learning
  - training set and test set
  - try to predict target value, based on input attributes
- Ockham's Razor
  - tradeoff between simplicity and accuracy
- Decision Trees
  - improve generalisation by building a smaller tree (using entropy)
  - prune nodes based on Laplace error

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set