Learning and Decision Trees

LECTURE 3 - part I

Learning agents • Inductive learning
Decision tree learning

UNSW
SYDNEY

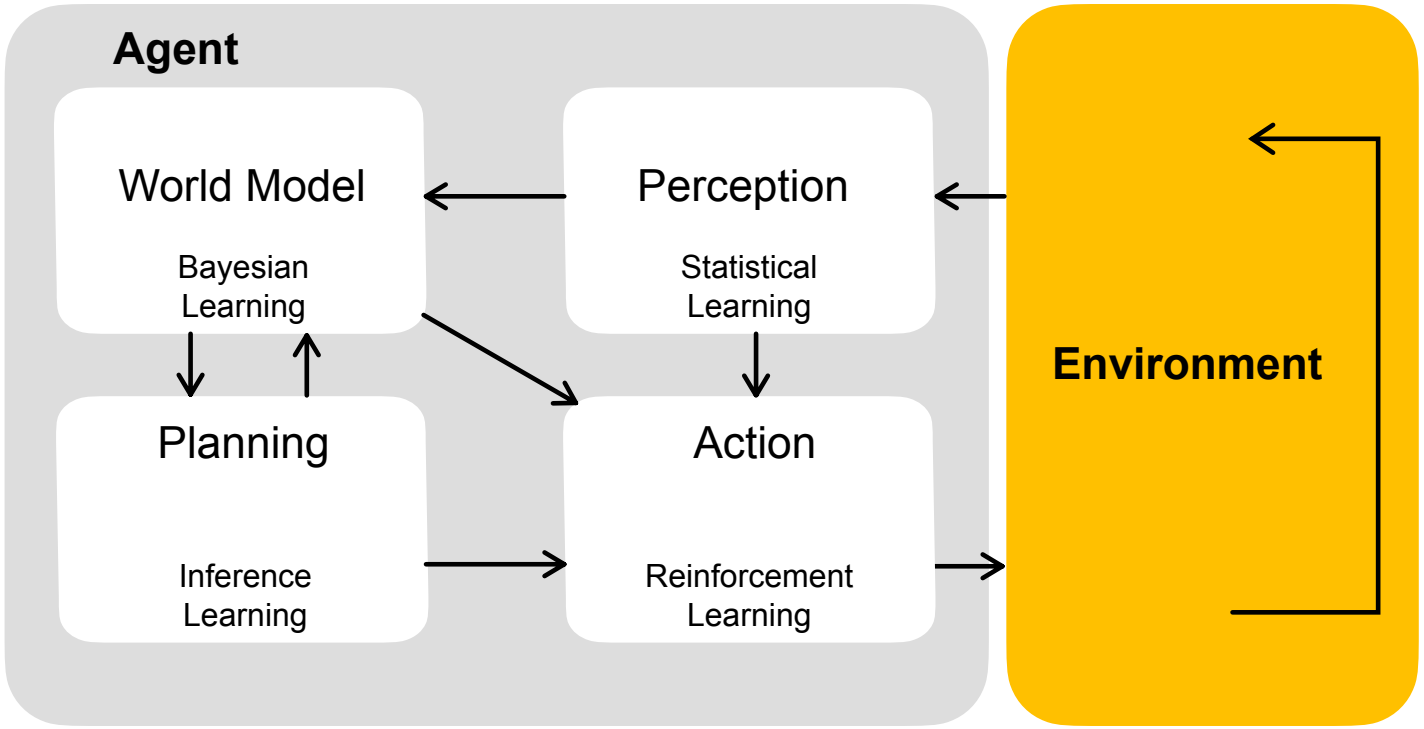
Motivation

- 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



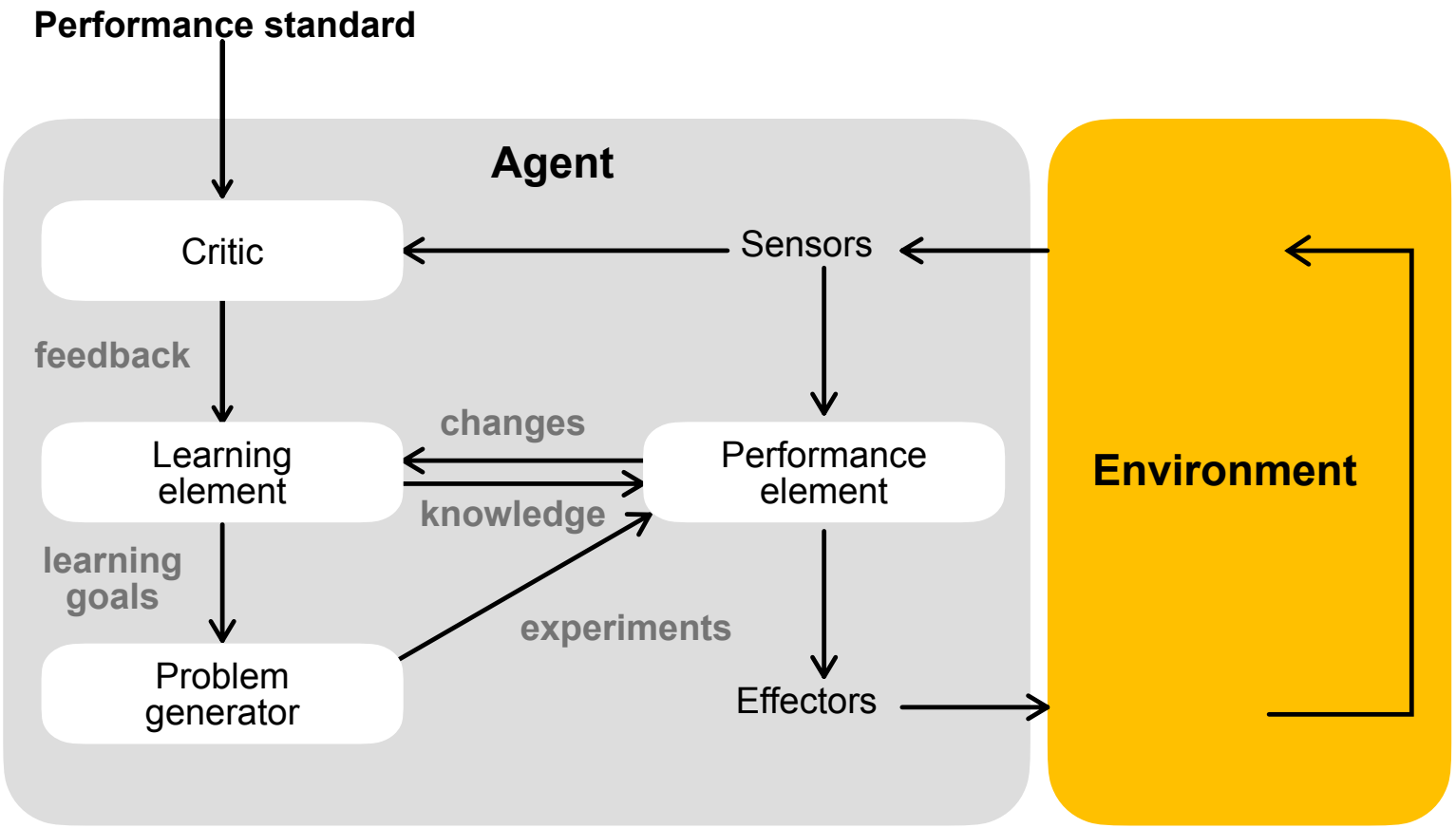
Learning agents

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

Learning element

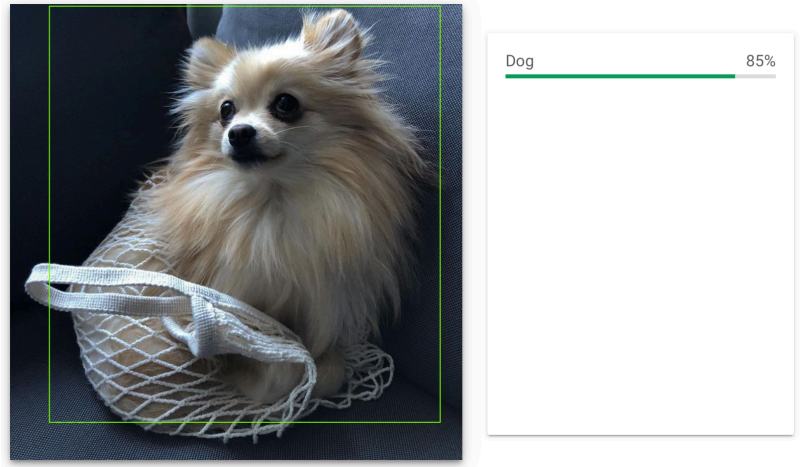
- Design of a learning element is affected by
  - Which components of the performance element are to be learned
  - What feedback is available to learn these components
  - What representation is used for the components

Learning agents

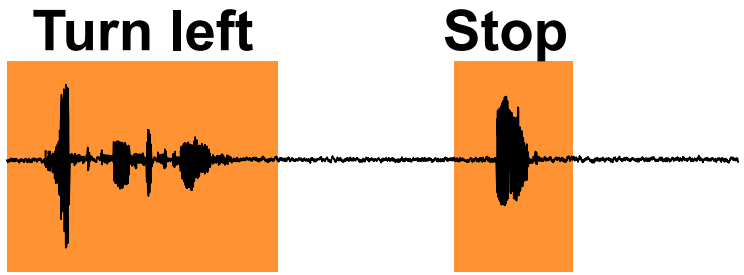


Machine Learning – Practice

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



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Libero volutpat sed cras ornare arcu dui vivamus arcu felis. Libero id faucibus nisl tincidunt eget nullam non. Libero justo laoreet sit amet cursus sit amet. Risus viverra adipiscing at in tellus integer feugiat. Odio eu feugiat pretium nibh ipsum consequat. Erat imperdiet sed euismod nisi porta lorem mollis aliquam. Odio ut sem nulla pharetra. Eleifend mi in nulla posuere sollicitudin aliquam. Consectetur adipiscing elit pellentesque habitant morbi tristique senectus. Ut ornare lectus sit amet est placerat. Ut tellus elementum sagittis vitae et leo. Lacinia at quis risus sed vulputate odio ut enim blandit. Laoreet suspendisse interdum consectetur libero id faucibus nisl tincidunt.
Sit amet mattis vulputate enim. Massa tincidunt nunc pulvinar sapien. Congue quisque egestas diam in. Sed faucibus turpis in eu. Velit scelerisque in dictum non consectetur a erat. Semper quis lectus nulla at volutpat. Sagittis orci a scelerisque purus semper eget. Mus mauris vitae ultricies leo integer. Ultricies tristique nulla aliquet enim tortor at auctor urna. Mi sit amet mauris commodo. Dignissim diam quis enim lobortis. Vitae



Types of Learning

- Supervised Learning
  - Agent is presented with examples of inputs and their target outputs, and must learn a function from inputs to outputs that agrees with the training examples and generalises to new examples
- Reinforcement Learning
  - Agent is not presented with target outputs for each input, but is periodically given a reward, and must learn to maximize (expected) rewards over time
- Unsupervised Learning
  - Agent is only presented with a series of inputs, and must find 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

- Given a training set and a test set, each consisting of a set of items for each item in the training set, a set of features and a target output
- Learner must learn **a model** that can **predict** the target output for any given item (characterised by its set of features)
- Learner is given the input features and target output for each item in the training set
  - Items may be presented all at once (batch) or in sequence (online)
  - Items may be presented at random or in time order (stream)
  - Learner **cannot** use the test set **at all** in defining the model
- Model is evaluated by its performance on predicting the output for each item in the **test set**

Methods vs Models

- Various learning **methods** can be used to generate models
  - Decision Trees
  - Support Vector Machines
  - Neural Networks/Deep Learning
- Evaluate methods by evaluating models on a variety of datasets
  - Problem with availability of standard benchmark datasets
  - Models depend on problem formulation and on parameters
  - End users may only care about a model, not a general method
  - Most machine learning research evaluates methods, not models

Supervised Learning – Methodology

- Feature “engineering” – select relevant features
- Choose representation of input features and outputs
- Pre-processing method to extract features from raw data
- Choose learning method(s) to evaluate
- Choose training regime (including parameters)
- Evaluation
  - Choose **realistic** baseline for comparison
  - Choose type of internal **validation**, e.g. cross-validation
  - Sanity check results with human expertise, other benchmark

Supervised Learning – Issues

- Framework (decision tree, neural network, SVM, etc.)
  - representation (of inputs and outputs)
  - pre-processing / post-processing
  - training method (perceptron learning, back-propagation, etc.)
  - generalisation (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

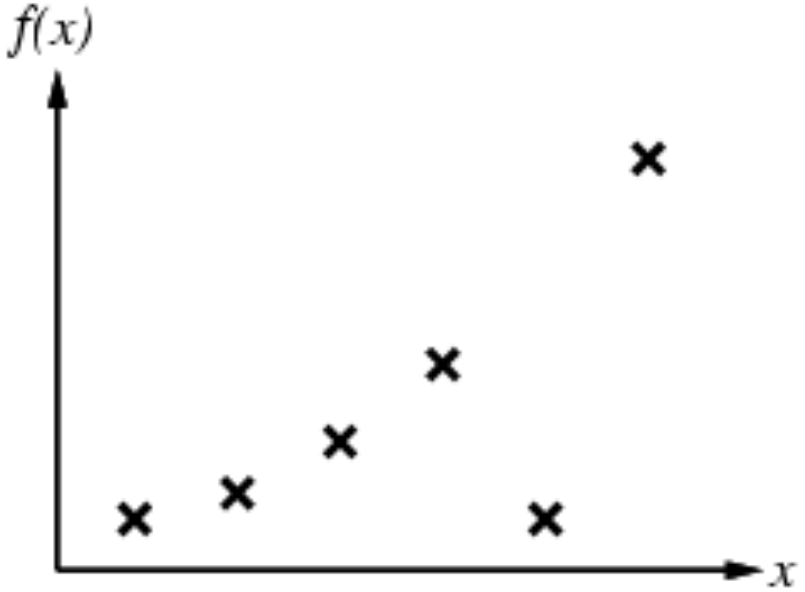
- (This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes examples are given)

Inductive learning method

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

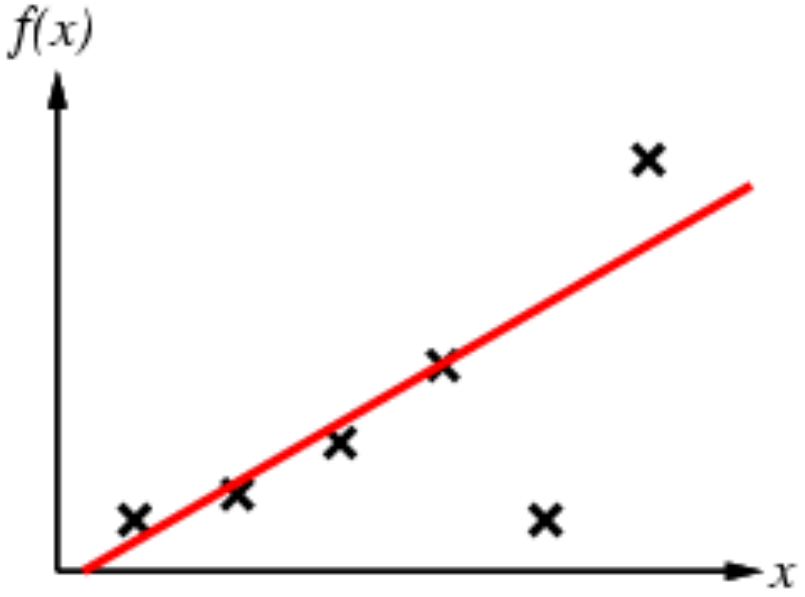
(*h* is **consistent** if it agrees with *f* on all examples)

- E.g., curve fitting:



Inductive learning method – Curve Fitting

Which curve gives the “best fit” to these data?

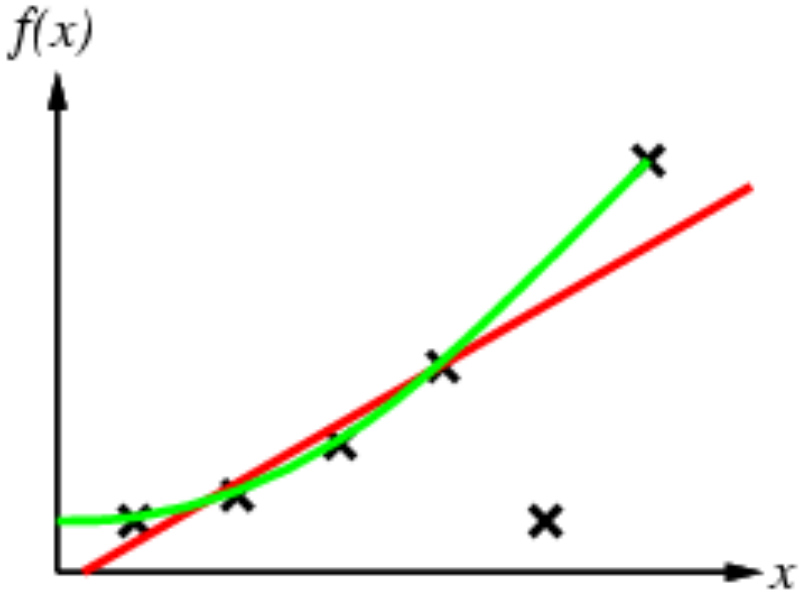


straight line?

Curve Fitting

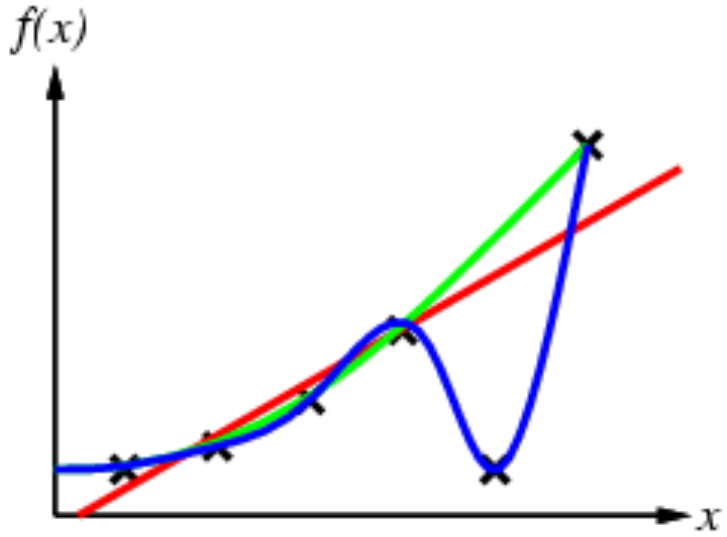
Which curve gives the “best fit” to these data?

parabola?



Inductive learning method

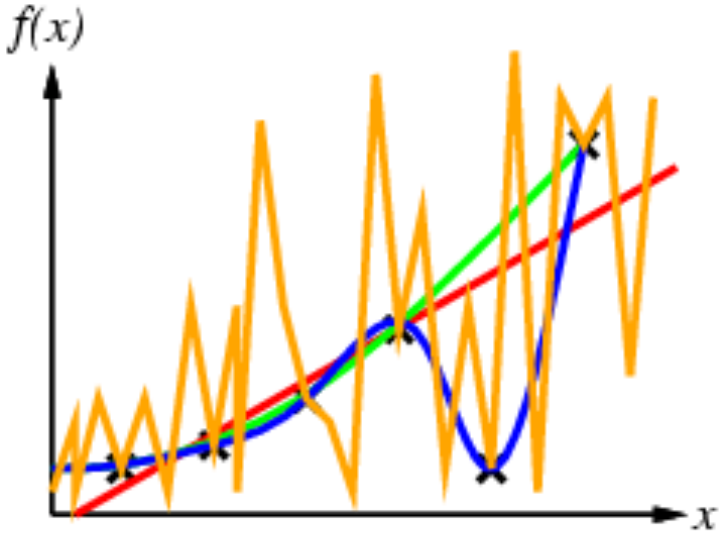
Which curve gives the “best fit” to these data?



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.

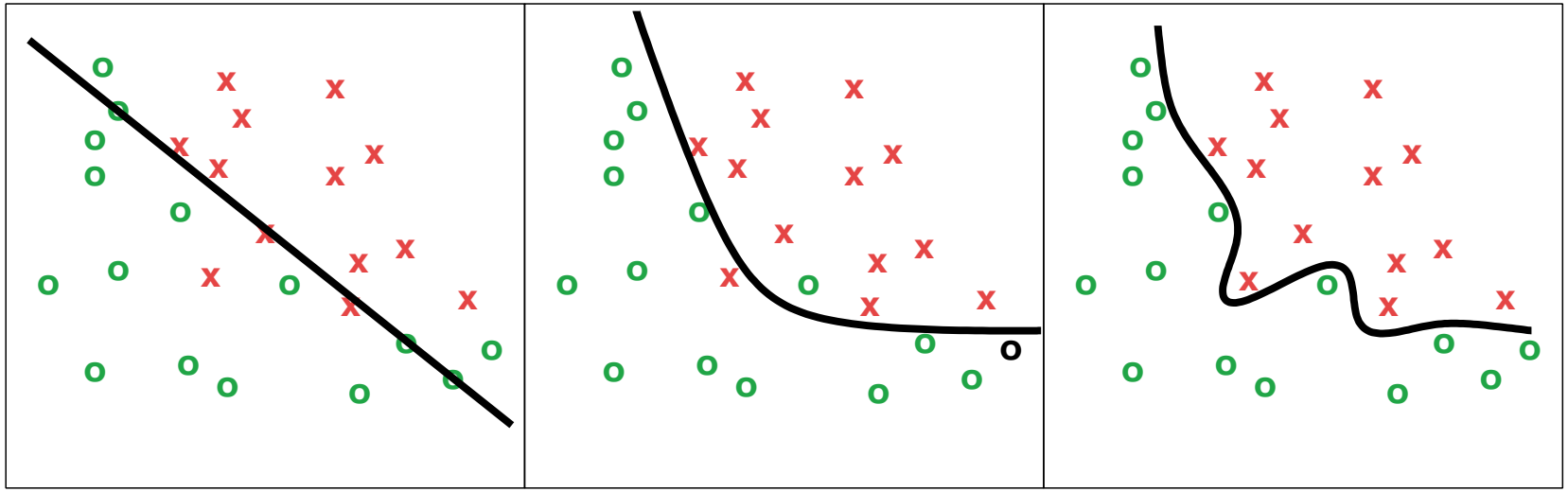
The principle states that the explanation of any phenomenon should make as few assumptions as possible, eliminating those that make no difference in the observable predictions of the explanatory hypothesis or theory.

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

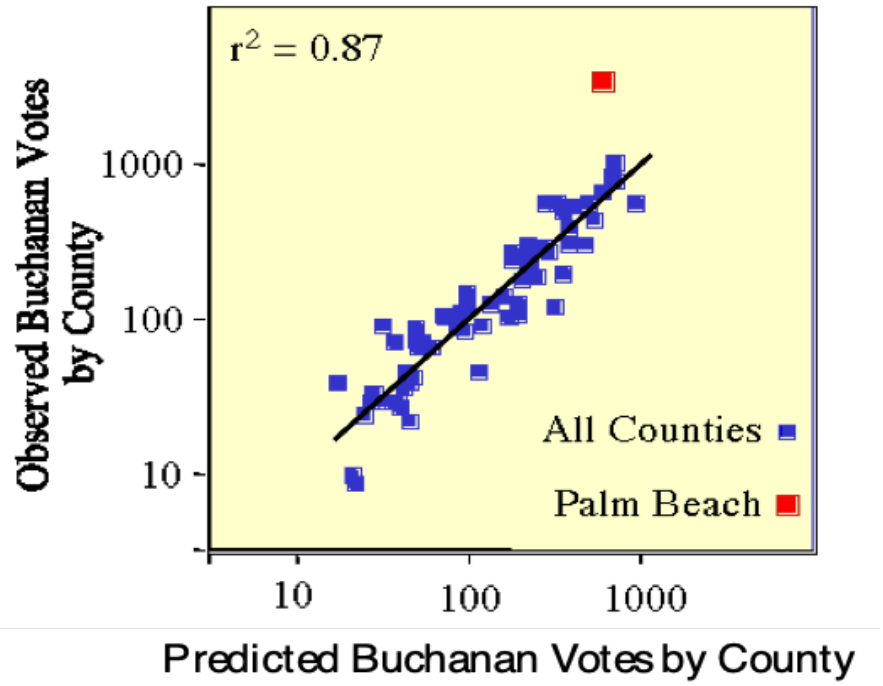
"The most likely hypothesis is the **simplest** one consistent with the data."



inadequate | good compromise | over-fitting

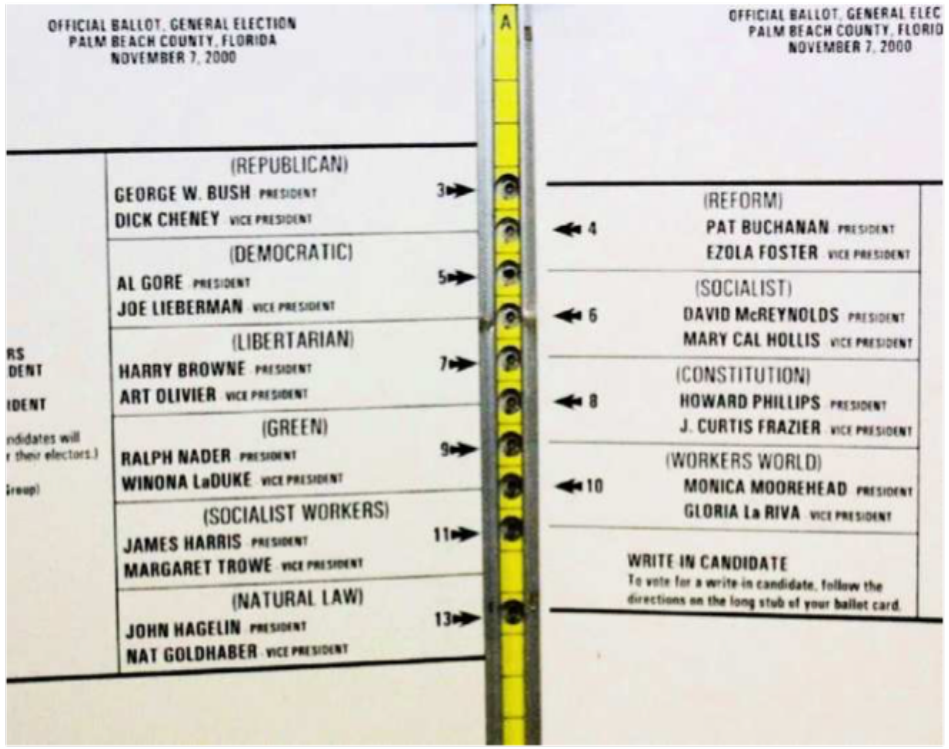
Since there can be **noise** in the measurements, in practice need to make a **tradeoff** between simplicity of the hypothesis and how well it fits the data.

Outliers



[faculty.washington.edu/mtbrett]

Butterfly Ballot



[faculty.washington.edu/mtbrett]

Components of a learning problem

The following components are part of any learning problem:

- **task:** the behaviour or task that's being improved.
    - For example: classification, acting in an environment
- **data:** the experiences that are being used to improve performance in the task.
- **measure of improvement:** how can the improvement be measured? For example: increasing accuracy in prediction, new skills that were not present initially, improved speed.

Supervised Learning

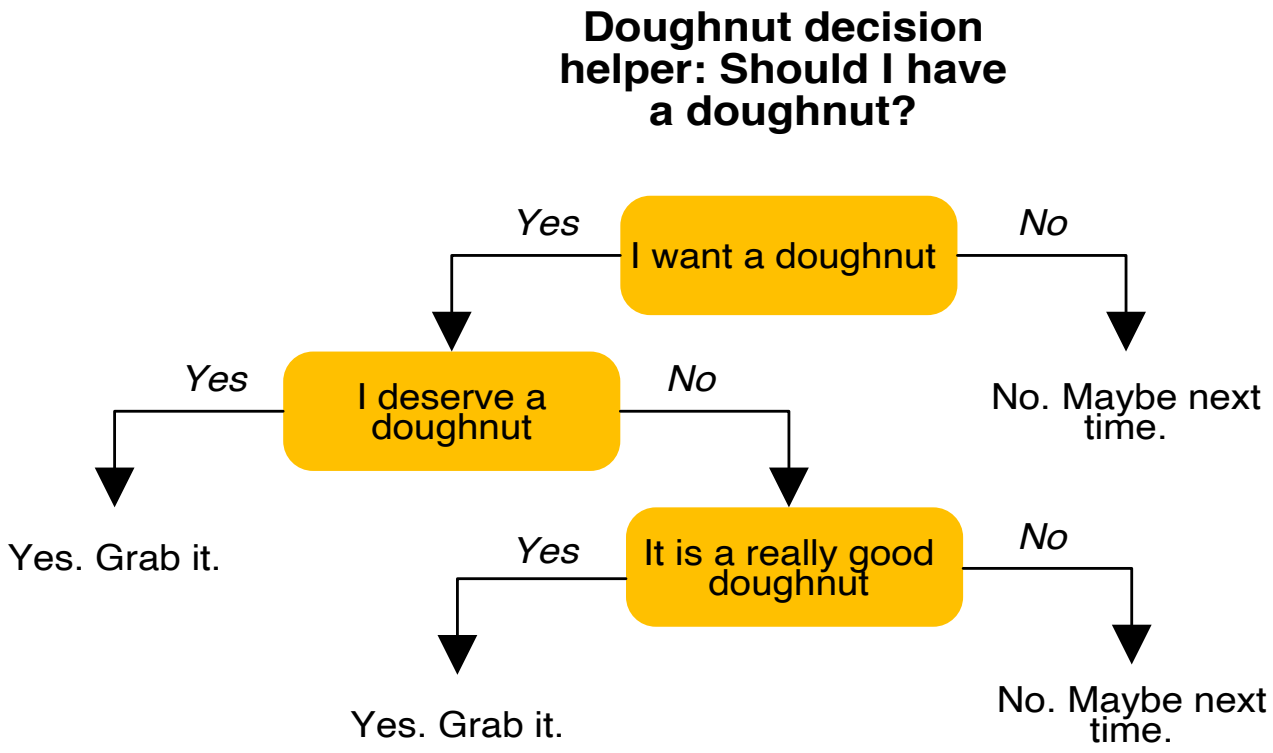
Given:

- a set of inputs features X1,...,Xn a
- set of target features Y1,...,Yk
- a set of training examples where the values for the input features and the target features are given for each example

- a new example, where only the values for the input features are given

- predict the values for the target features for the new example.
  - classification when the Yi are discrete
  - regression when the Yi are continuous

Learning decision trees



Learning decision trees

#	Did they want a doughnut?	Did they deserve a doughnut?	Was it a good doughnut?	Did they eat the doughnut?
1	yes	yes	yes	yes
2	yes	no	no	no
3	yes	no	yes	yes
4	no	yes	yes	no
5	no	no	no	yes
		. . .		

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

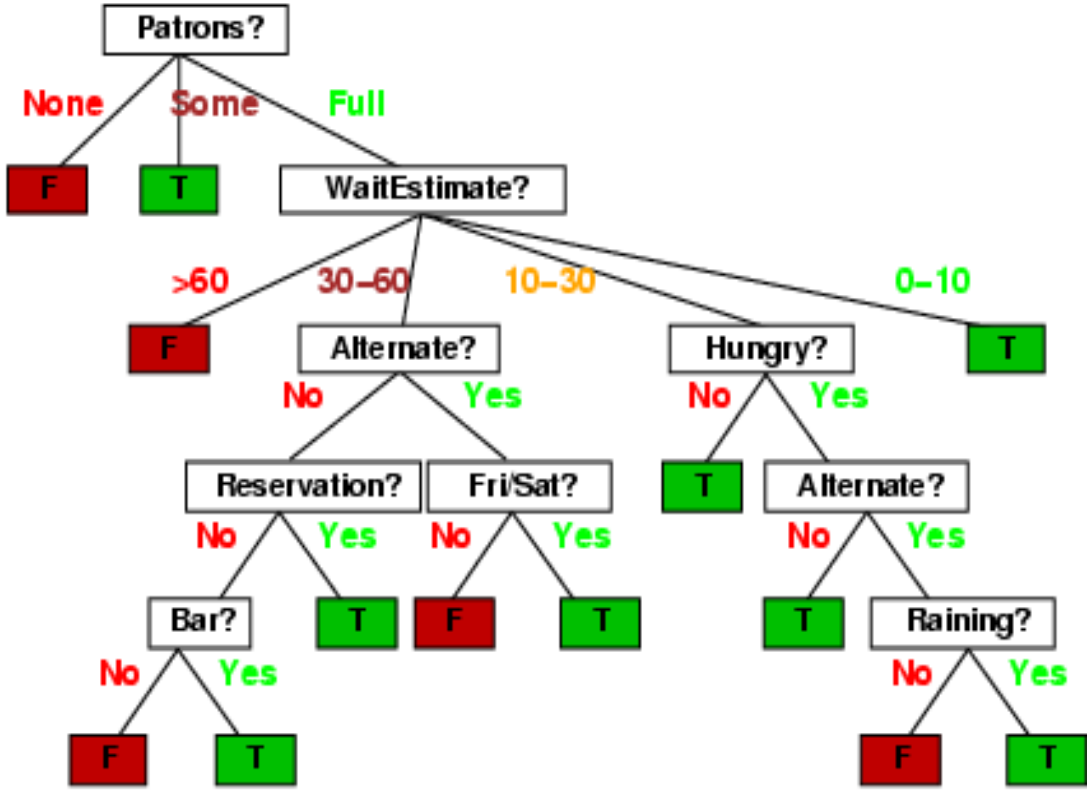
- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

	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

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

Decision trees

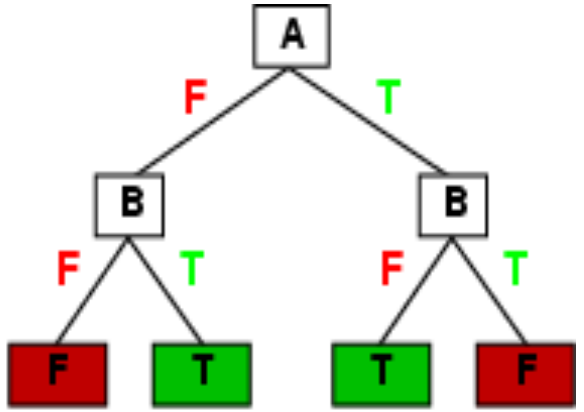
- One possible representation for hypotheses
  - E.g., here is the “true” tree for deciding whether to wait:



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 generalise to new examples

- Prefer to find more compact decision trees

Generalisation

- 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 generalise 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?

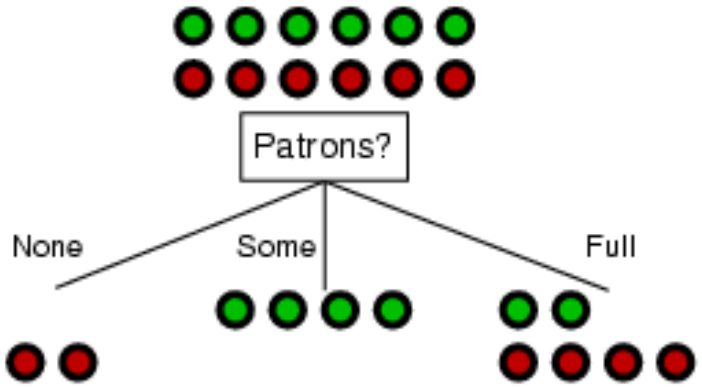
Decision tree learning algorithm

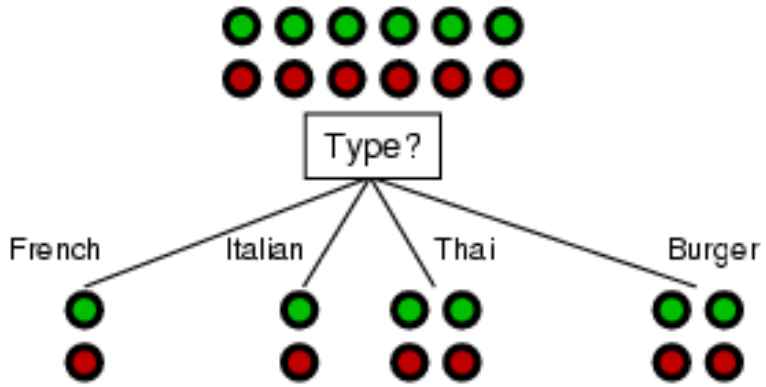
Aim: Find the smallest tree consistent with the training samples.
Idea: Recursively choose “most significant” attribute as root of (sub)tree

1. If the current split is inconsistent - mixed true & false samples, then choose the most informative attribute to split them.
2. If there are no samples left for a split then no such sample has been observed. Assign the label of the majority of the parent node.
3. If the split is consistent then we have reached a leaf node. Assign the corresponding label - true or false.
4. If split is inconsistent but there are no attributes left. Assign the label of the majority. There could be several reasons for that:
   - Inconsistent data
   - Lack of attributes
   - Non-deterministic data

Decision tree learning algorithm

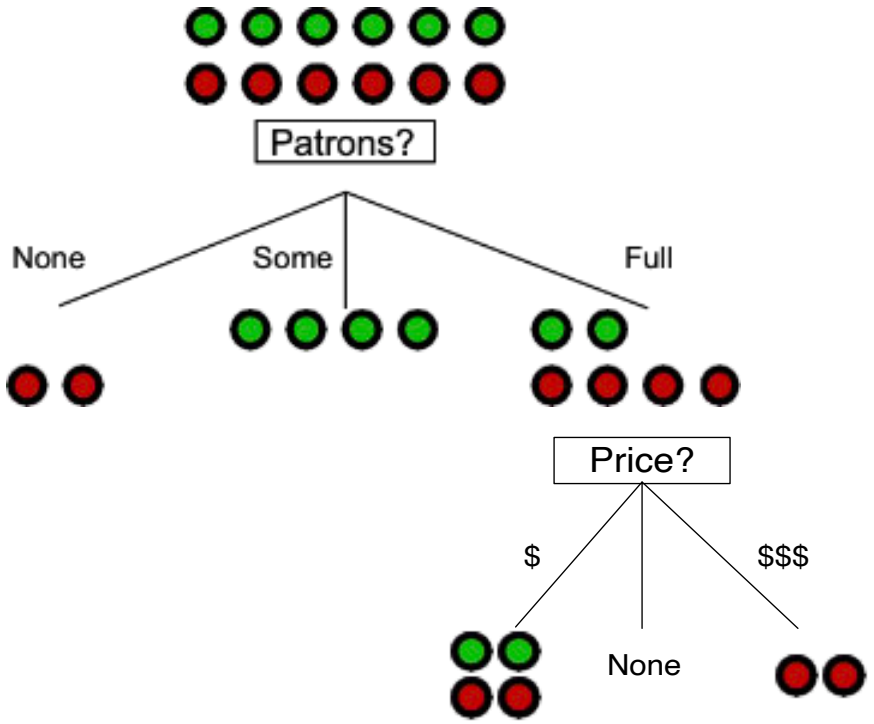
1. If the current split is inconsistent - mixed true & false samples, then choose the most informative attribute to split them.





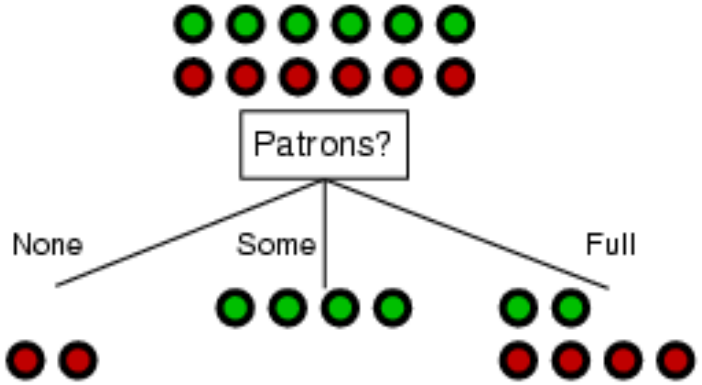
Decision tree learning algorithm

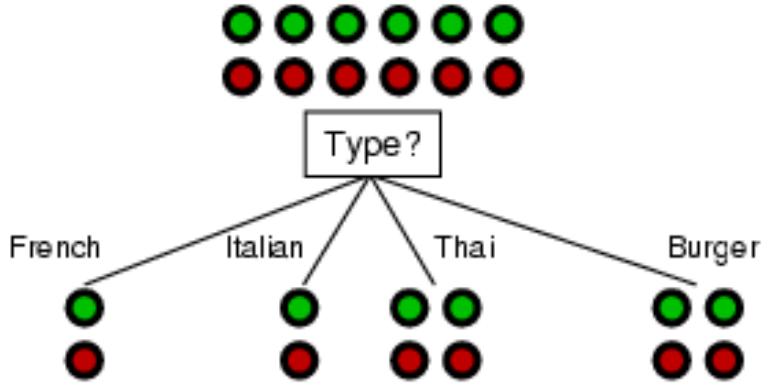
2. If there are no samples left for a split then no such sample has been observed. Assign the label of the majority of the parent node.



Decision tree learning algorithm

3. If the split is consistent then we have reached a leaf node. Assign the corresponding label - true or false.



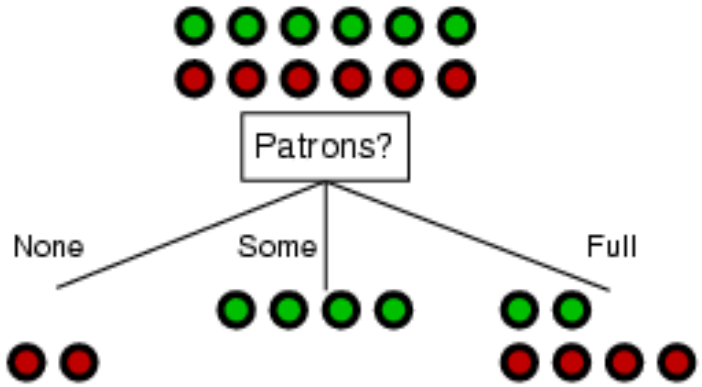


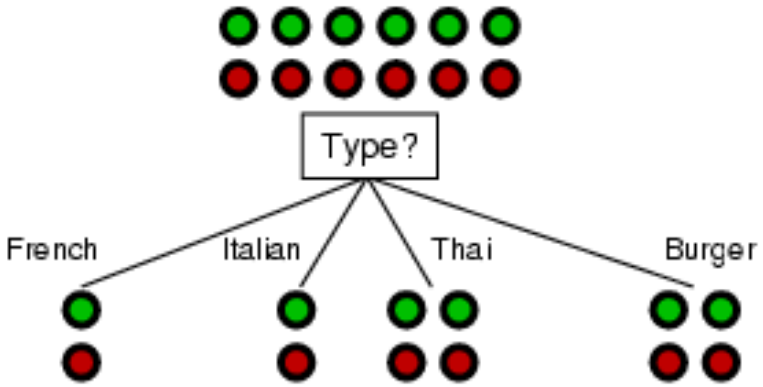
Choosing an attribute

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

Choosing an attribute

- Patrons is a “more informative” attribute than Type, because it splits the examples more nearly into sets that are “all positive” or “all negative”.
- An important (or good) attribute splits samples into groups that are (ideally) all positive or negative.

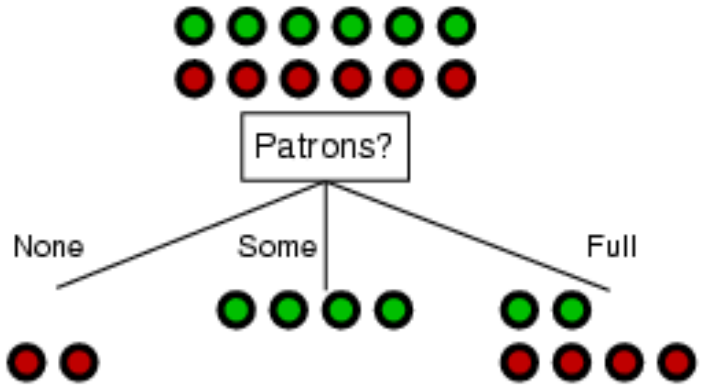


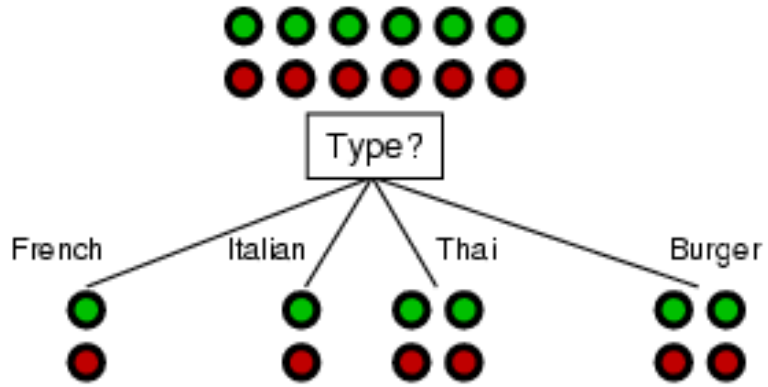


Choosing an attribute

Therefore $Patrons$ is more important than $Type$.

Testing good attributes first allows us to minimise the tree depth.





This notion of "informativeness" can be quantified using the mathematical concept of **"entropy".**

A parsimonious tree can be built by minimising 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 $p_1, .., p_n$ then the entropy is

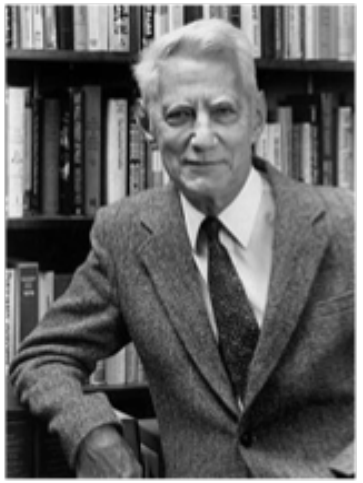
$$H(<p_1, .., p_n>) = \sum - p_i \log_2 p_i$$

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:

Suppose we want to encode, in 0s and 1s, 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** (binary digit) is needed to encode each letter.

$$H(\langle 0.5, 0.5 \rangle) = 1 \text{ bit}$$

Entropy and Huffmann Coding

Example 2:

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.

$$H(\langle 0.5, 0.5 \rangle) = 1 \text{ bit}$$

Entropy and Huffmann Coding

Example 3:

Consider a code to distinguish elements of {a, b, c, d} with

$$P(a) = \frac{1}{2}, P(b) = \frac{1}{4}, P(c) = \frac{1}{8}, P(d) = \frac{1}{8}$$

Consider the code: a 0 b 10 c 110 d 111

This code uses 1 to 3 bits. On average, it uses

$$\begin{aligned} & P(a) \times 1 + P(b) \times 2 + P(c) \times 3 + P(d) \times 3 \\ = \ & \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = 1\frac{3}{4} \text{ bits.} \end{aligned}$$

Entropy and Huffmann Coding

Information theory

- A **bit** is a binary digit.
- 1 bit can distinguish 2 items
- k bits can distinguish 2k items
- n items can be distinguished using $\log_2 n$ bits

Entropy and Huffmann Coding

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

$$H(<p_1, .., p_n>) = \sum - p_i \log_2 p_i$$

Entropy

Suppose we have p positive and n negative examples at a node.

$\rightarrow H(<p/(p+n), .., n/(p+n)>)$ 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

$\rightarrow H(<p_i/(p_i+n_i), n_i/(p_i+n_i)>)$ bits needed to classify a new example

expected number of bits per example over all branches is

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

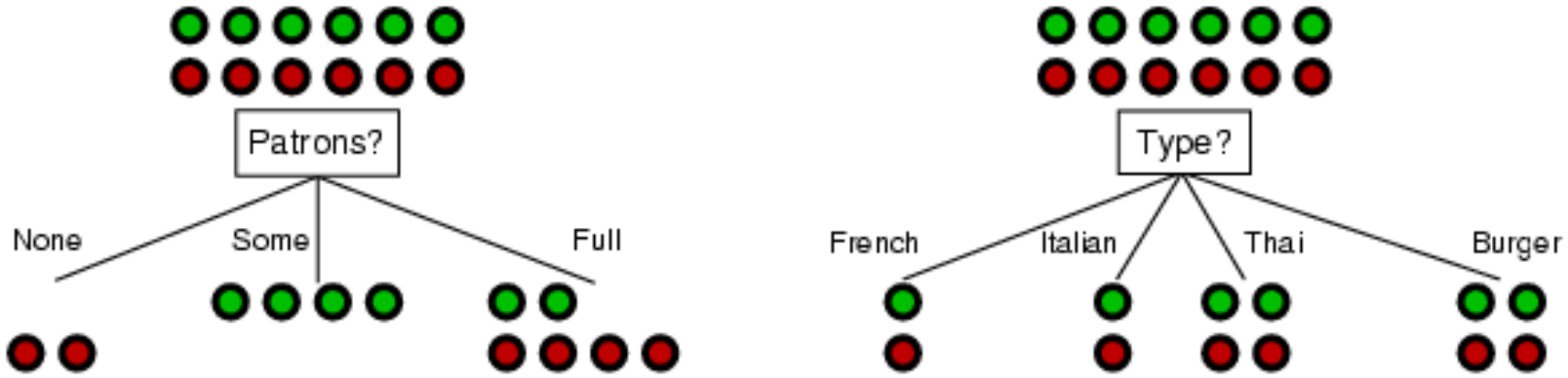
Entropy

- Entropy is a measure of “randomness” (lack of uniformity)
  - Related to prior distribution of some random variable
  - Higher entropy means more randomness
  - “Information” (about distribution) reduces entropy
- Idea: Split based on information gain
  - Loss of entropy based on “communicating” value of attribute
  - Related to Shannon’s information theory
  - Measure information gain in bits

Definition. If the prior probabilities of n attribute values are $p_1, .., p_n$, then the entropy of the distribution is

$$H(<p_1, .., p_n>) = \sum - p_i \log_2 p_i$$

Choosing and Attribute

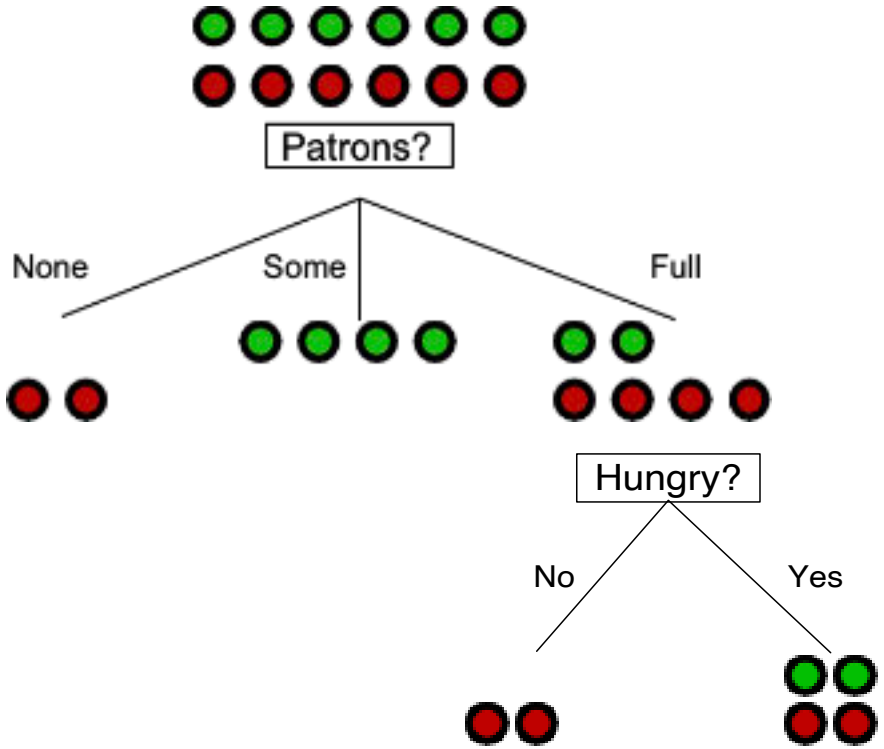


$$\text{For Patrons, Entropy} = \frac{1}{6}(0) + \frac{1}{3}(0) + \frac{1}{2}\left[-\frac{1}{3}\log\left(\frac{1}{3}\right) - \frac{2}{3}\log\left(\frac{2}{3}\right)\right]$$

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

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

Choosing Next Attribute

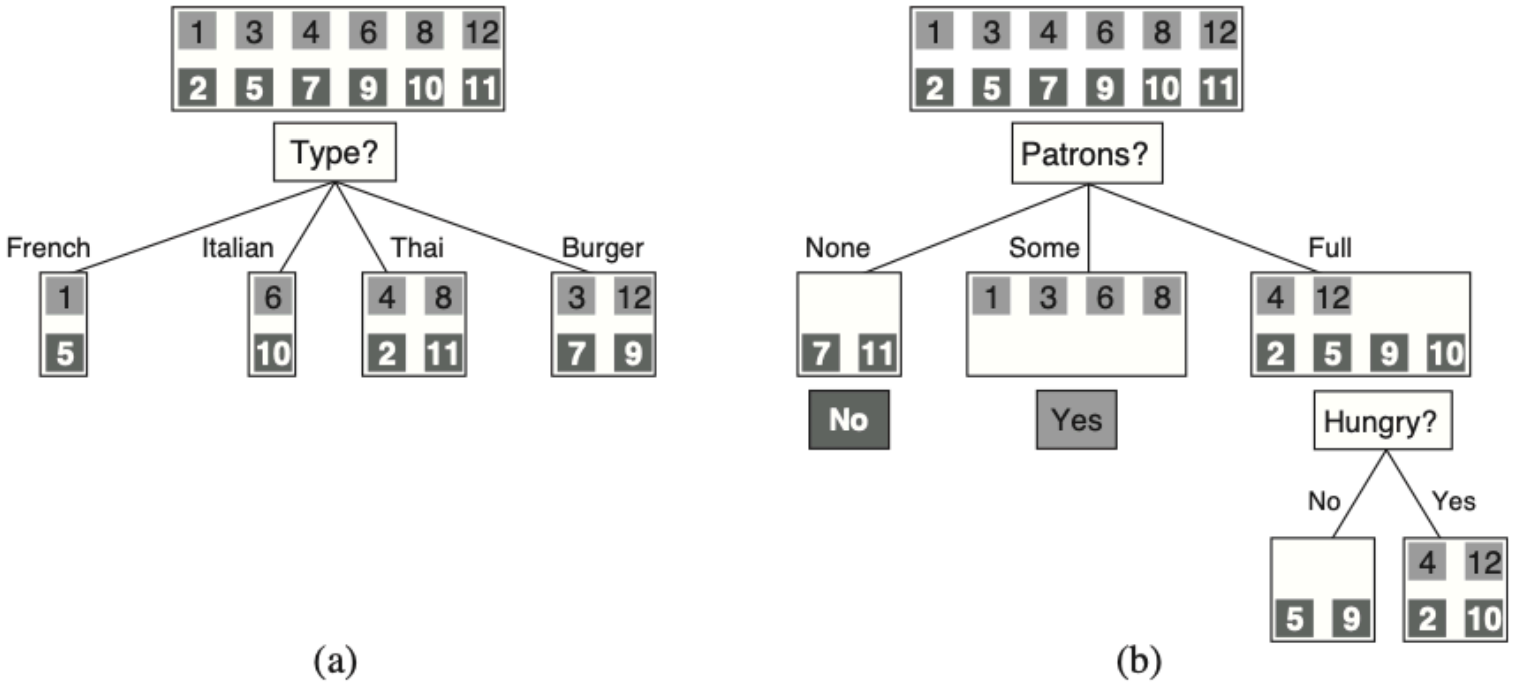


$$\sum_i \frac{p_i + n_i}{p+n} H(\langle \frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i} \rangle)$$

After splitting on Patrons, split on Hungry ?

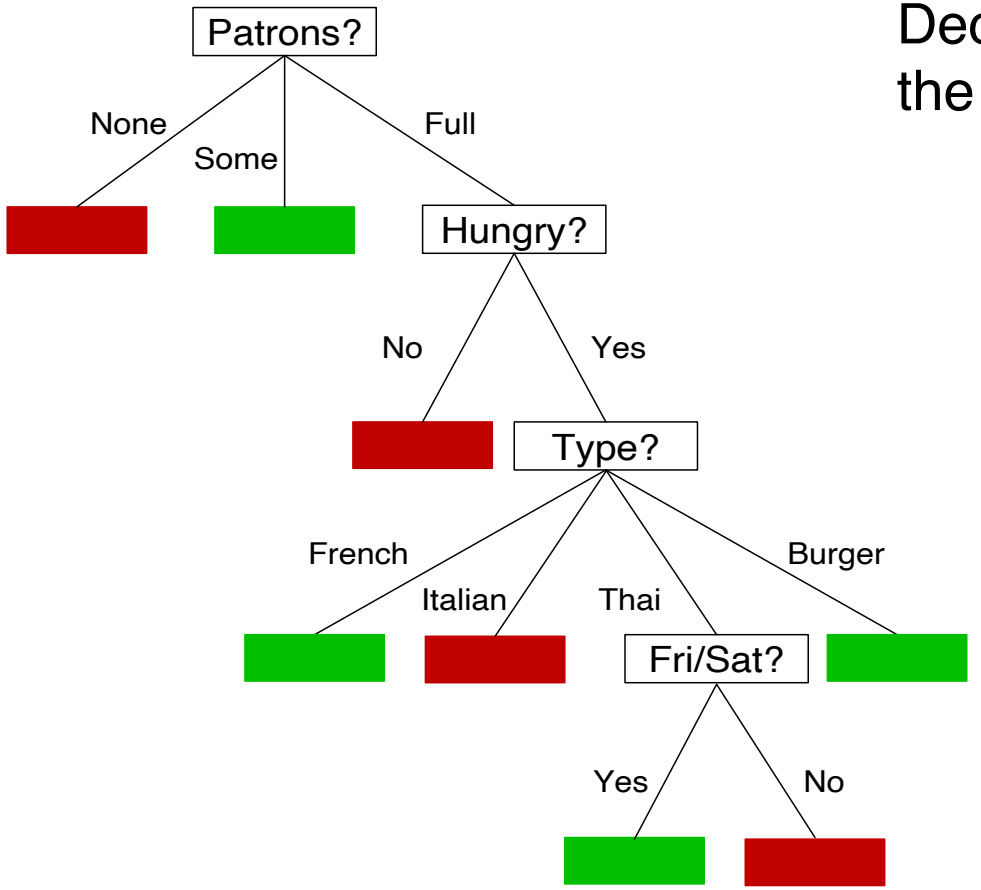
Choosing Next Attribute

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



After splitting on Patrons, split on Hungry

Induced Tree



Decision tree learned from the 12 examples.

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 .

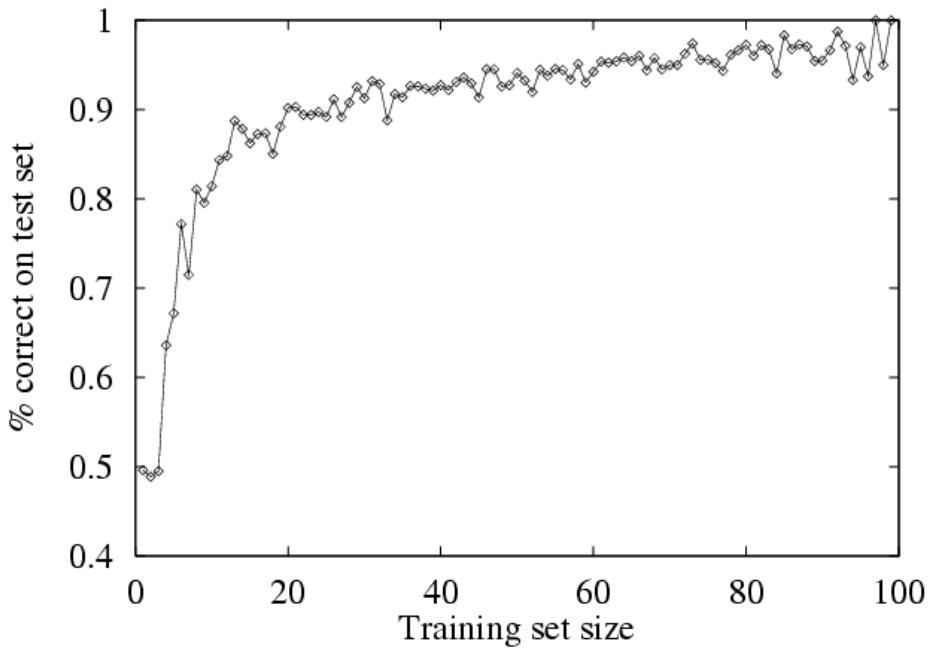
Performance measurement

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

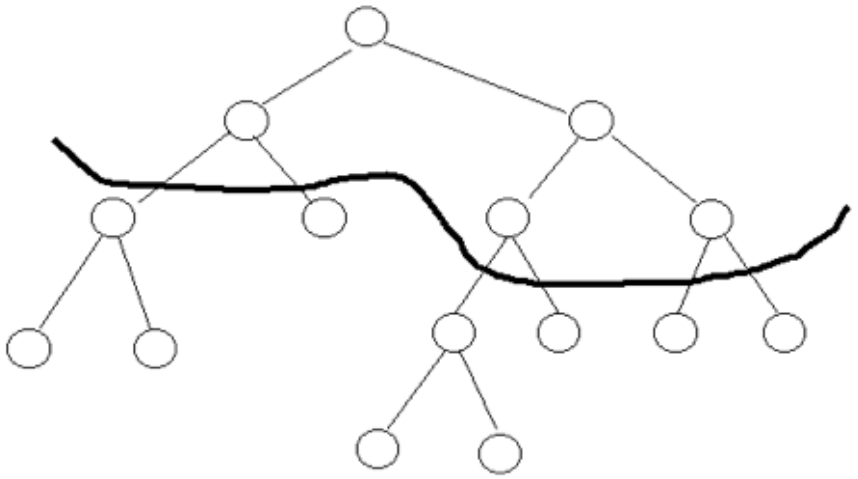
1. Use theorems of computational/statistical learning theory
2. Try h on a new test set of examples

   (use same distribution over example space as training set)

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



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

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

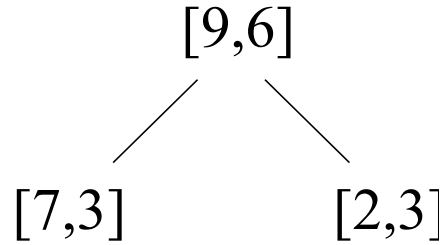
If the average Laplace error of the children exceeds that of the parent node, we prune off the children.

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

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

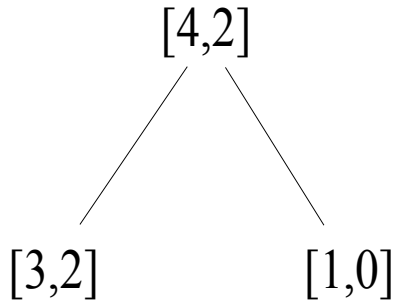
Since $0.365 < 0.412$, children should NOT be pruned.

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

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

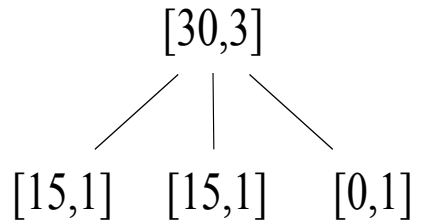
Since $0.413 > 0.375$, children should be pruned.

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.

Basic Decision Tree Learning Algorithm

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

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

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

Information Gain

- Information Gain is the mutual (相互的) 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

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

Information Gain

$Gain(S, A)$ = expected reduction in entropy due to sorting on 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*.

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

回归树

Similar to induction of decision trees

Regression trees useful in continuous domains,

- e.g. predict biomass of algae

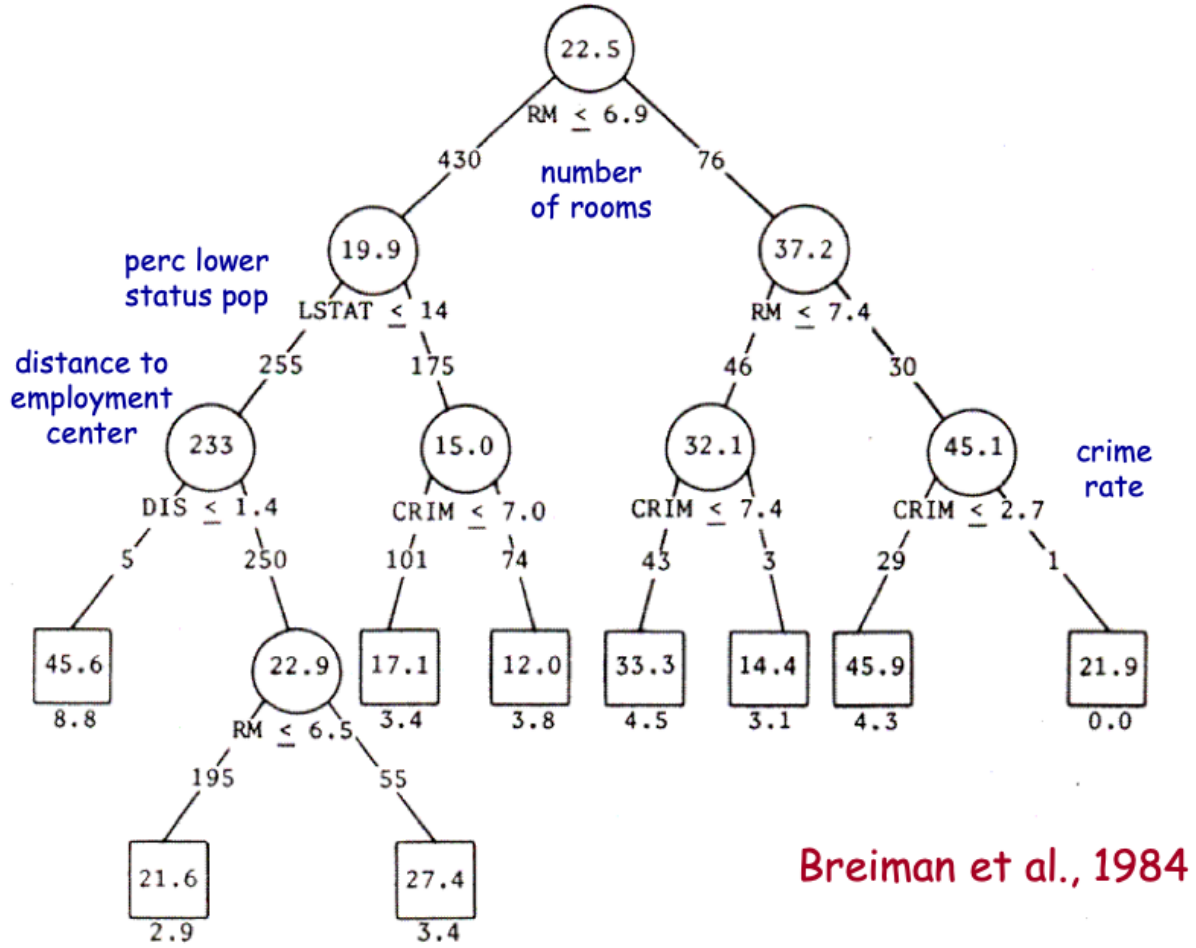
Decision trees: discrete class

Regression trees: continuous, real-valued class

Predicting Boston Housing Prices

- 'RM' is the average number of rooms among homes in the neighborhood.
- 'LSTAT' is the percentage of homeowners in the neighborhood considered "lower class" (working poor).
- 'PTRATIO' is the ratio of students to teachers in primary and secondary schools in the neighborhood.

Example: Boston Housing Values



Some Systems That Induce Regression Trees

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

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.

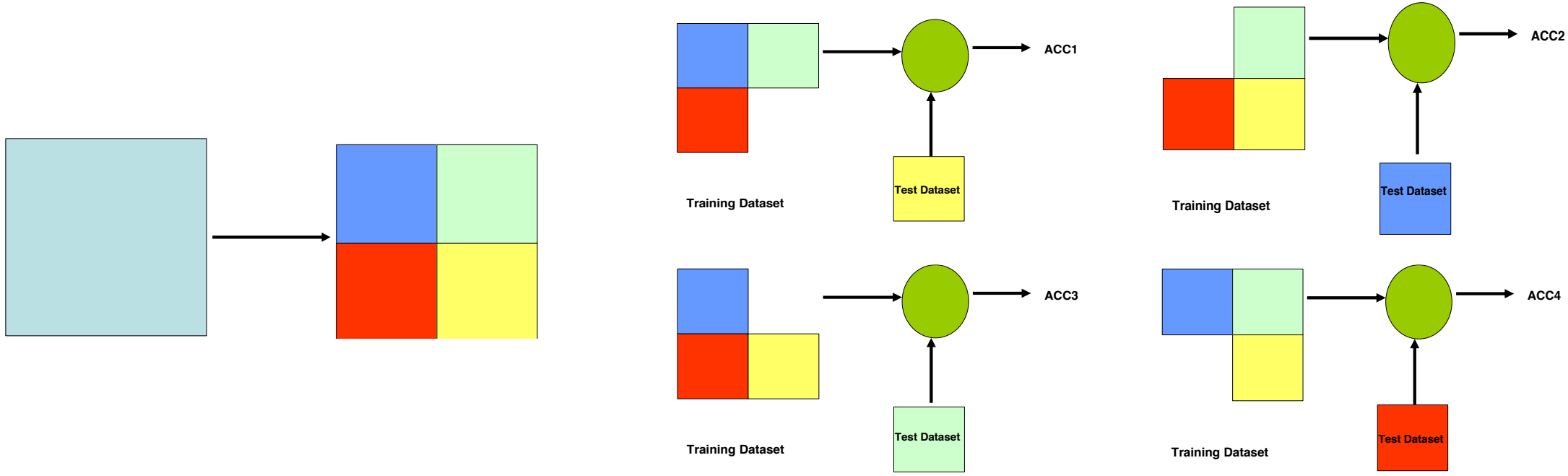
Partition data into K disjoint subsets
For k=1 to K
    testData = k_{th} subset
    h ← classifier trained on all data except for testData
    accuracy(k) = accuracy of *h* on testData
end
FinalAccuracy = mean of the K recorded test set accuracies

Leave-One-Out Cross Validation

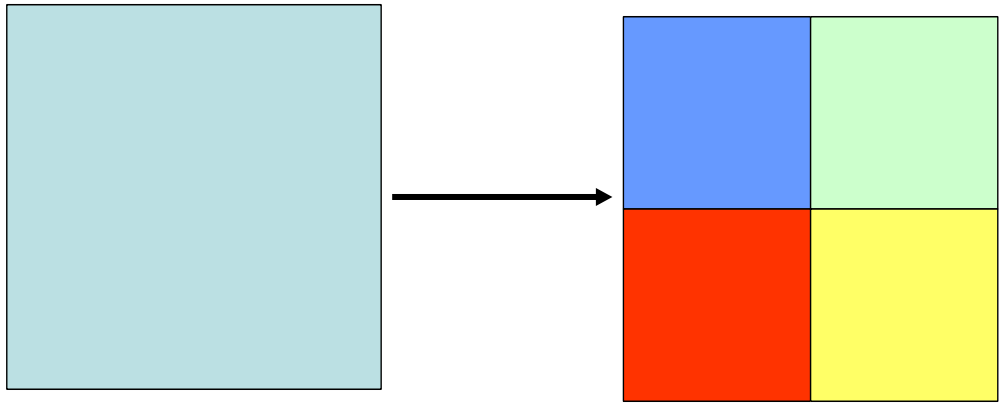
Idea: train multiple times, leaving out a disjoint subset of data each time for test. Average the test set accuracies.

Partition data into K disjoint subsets *each containing one example*
For k=1 to K
    testData = k_{th} subset
    h ← classifier trained on all data except for testData
    accuracy(k) = accuracy of *h* on testData
end
FinalAccuracy = mean of the K recorded test set accuracies

4-Fold Cross-validation



4-Fold Cross-validation



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

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
  - Other ways to prune Decision Trees

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

References

- Poole &Mackworth, Artificial Intelligence: Foundations of Computational Agents, Chapter 7.
- Russell & Norvig, *Artificial Intelligence: a Modern Approach*, Chapter 18.1, 18.2, 18.3