Learning and Decision Trees

COMP3411/9814: Artificial Intelligence

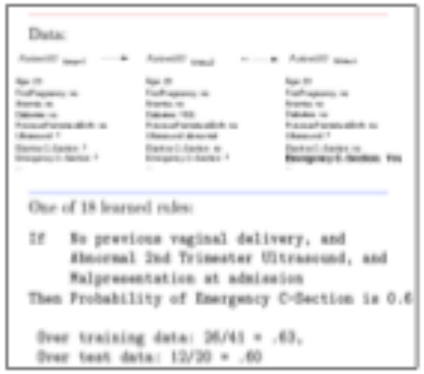
Lecture Overview

- Learning agents
- Inductive learning
- Decision tree learning

Turing’s Child Machine

“Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child-brain is something like a notebook as one buys from the stationers. Rather little mechanism, and lots of blank sheets... Our hope is that there is so little mechanism in the child-brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child.”

Machine Learning Applications



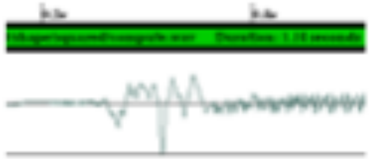
Data:

```
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

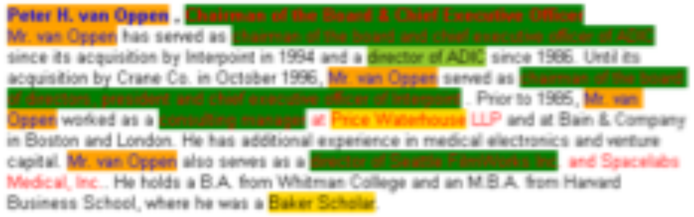


Speech Recognition

Control learning

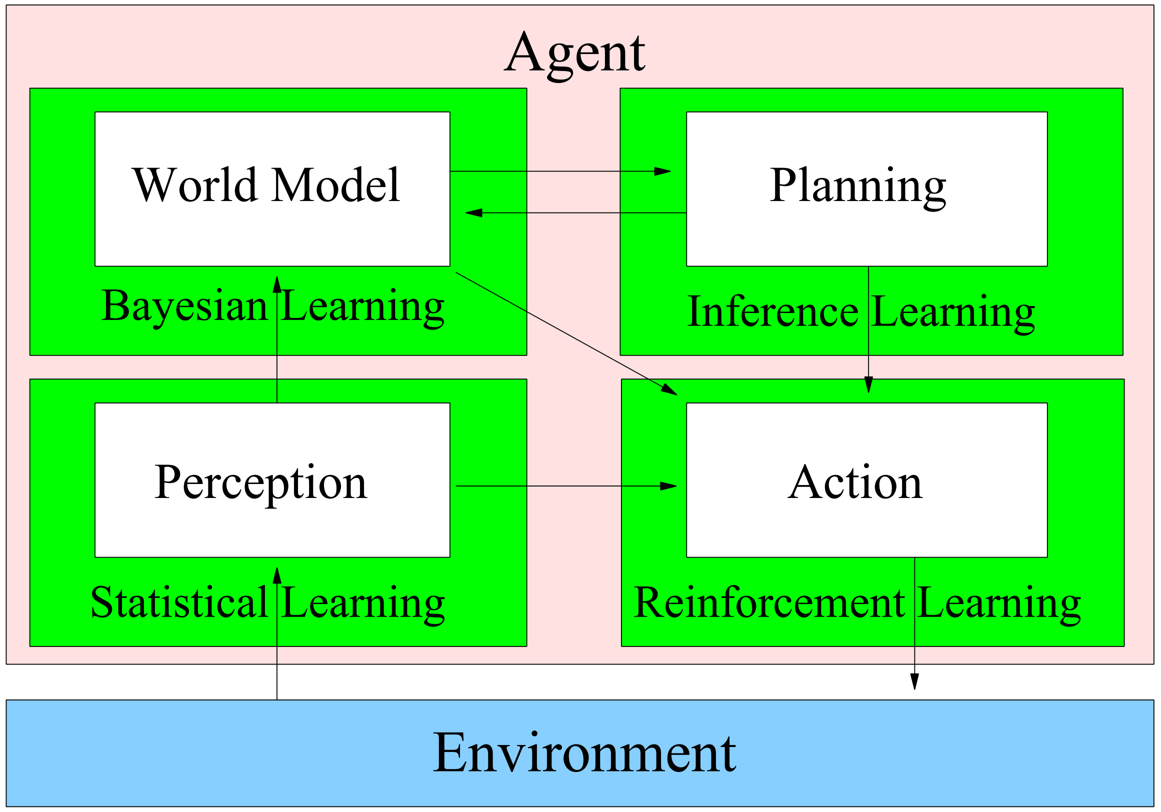
Object recognition

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.

Learning Agent



Learning Agents

- Improve performance on future tasks after making observations about the world
- 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

Types of Learning

- Supervised Learning
  - Agent given examples of inputs and outputs
  - Learns a function from inputs to outputs that agrees with training examples and generalises to new examples
- Unsupervised Learning
  - Agent only given inputs
  - Tries to find structure in inputs
- Reinforcement Learning
  - Agent given a reward
  - Learns to maximise (expected) rewards over time

Supervised Learning

- Given a training set and a test set
  - each has set of examples
  - example has attributes and a class, or target value

    $<x_1, x_2, \ldots, x_n, \mathrm{y}>$ y may be discrete or continuous

- Agent given attributes and class for each example in training set

  ➡ Try predict class of each example in test set

- Many supervised learning methods, e.g.:
  - Decision Tree
  - Neural Network
  - Support Vector Machine, etc.

Supervised Learning

- Training set:
    - Examples may be presented all at once (batch) or in sequence (online)
    - Examples 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 on predicting output for each example in test set

Supervised Learning Methodology

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

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*

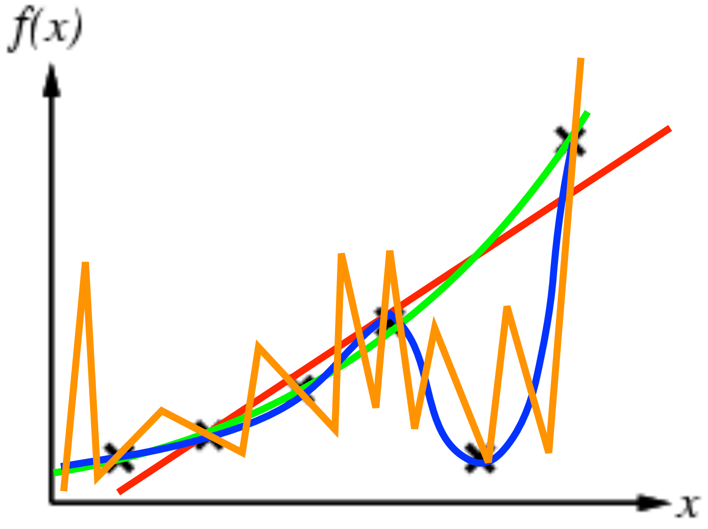
  such that $h \approx f$

  given a training set of examples

- Ignores prior knowledge
  - Assumes examples are given

Inductive Learning Method

- Construct *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* in all examples
- E.g., curve fitting
  - Which curve best fits data?

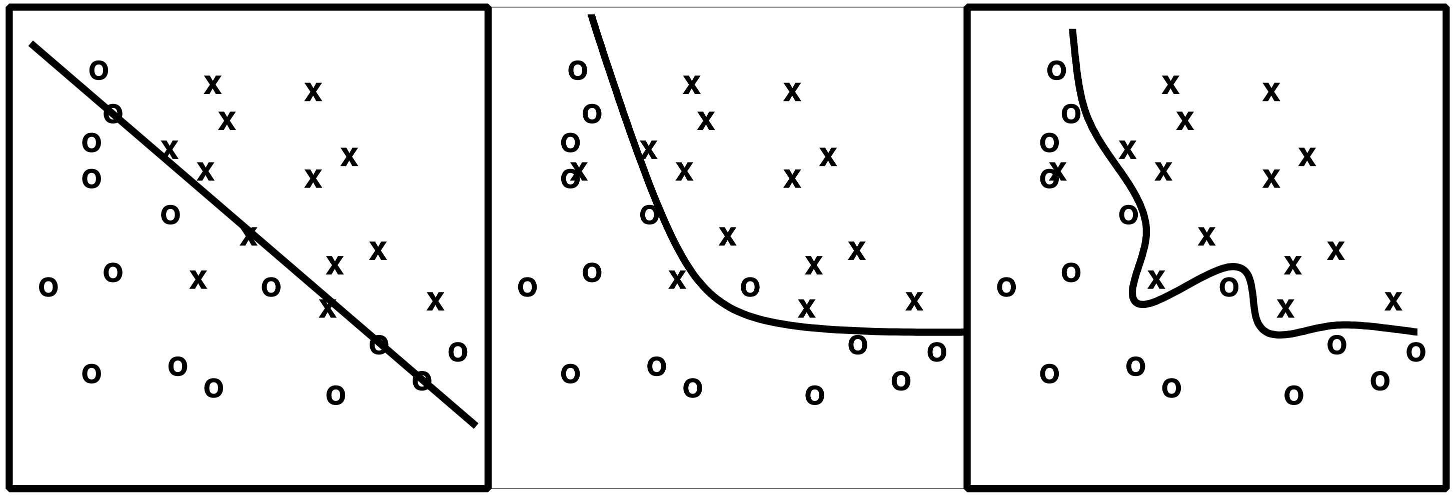


Ockham’s razor

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

Ockham’s razor

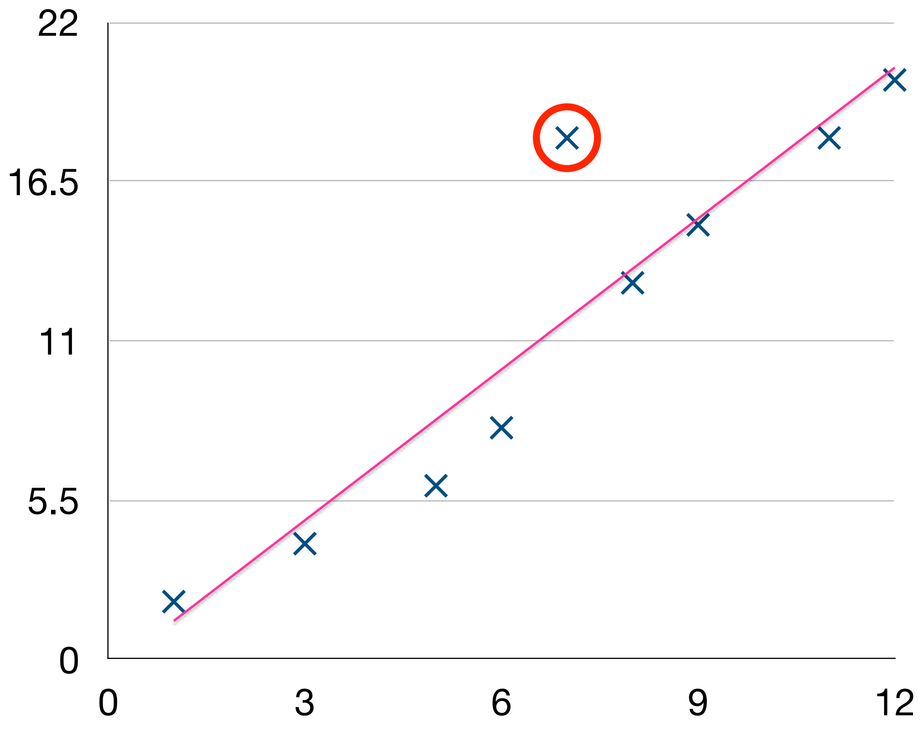
“The most likely hypothesis is the simplest one consistent with the data.”



inadequate good compromise over-fitting

- Can have noise in data
- Need to tradeoff simplicity and accuracy

Outliers



Supervised Learning (again)

Given:

- a set of training examples, each with a set of features and target value (or class):

$<x_1, x_2, \ldots, x_n, \ \mathrm{y}>$

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

predict the target value of the new example.

Decision Tree Learning

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

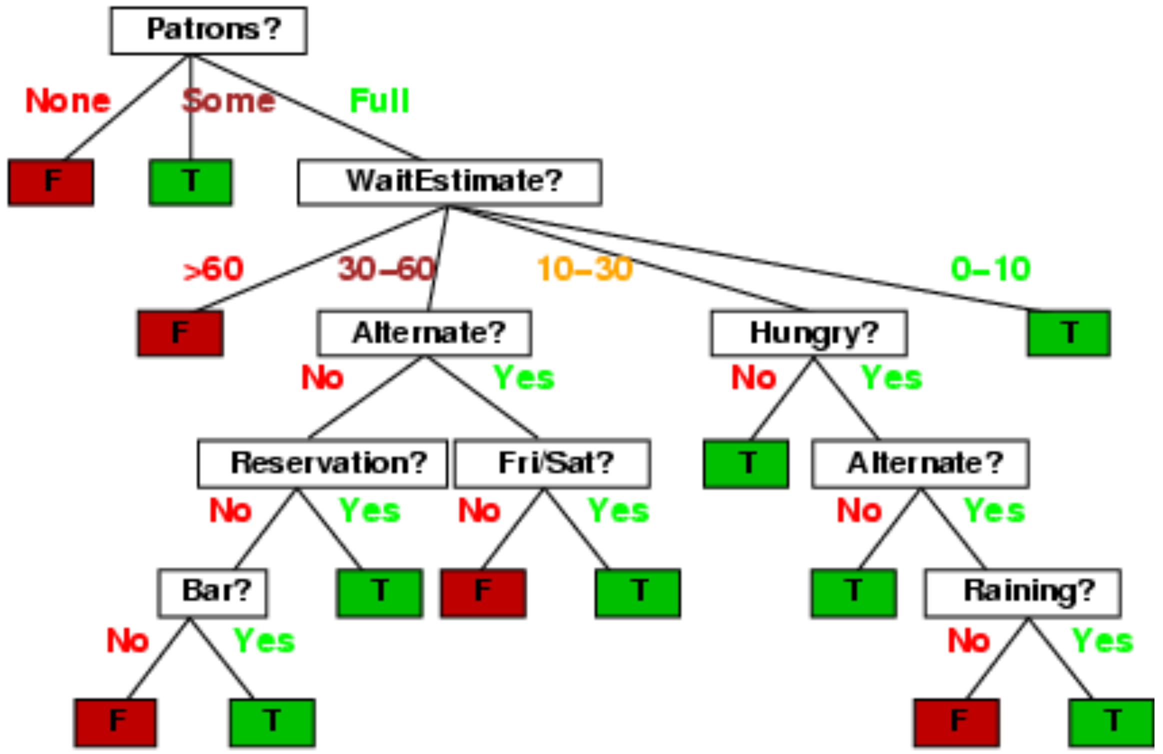
- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

class value

	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

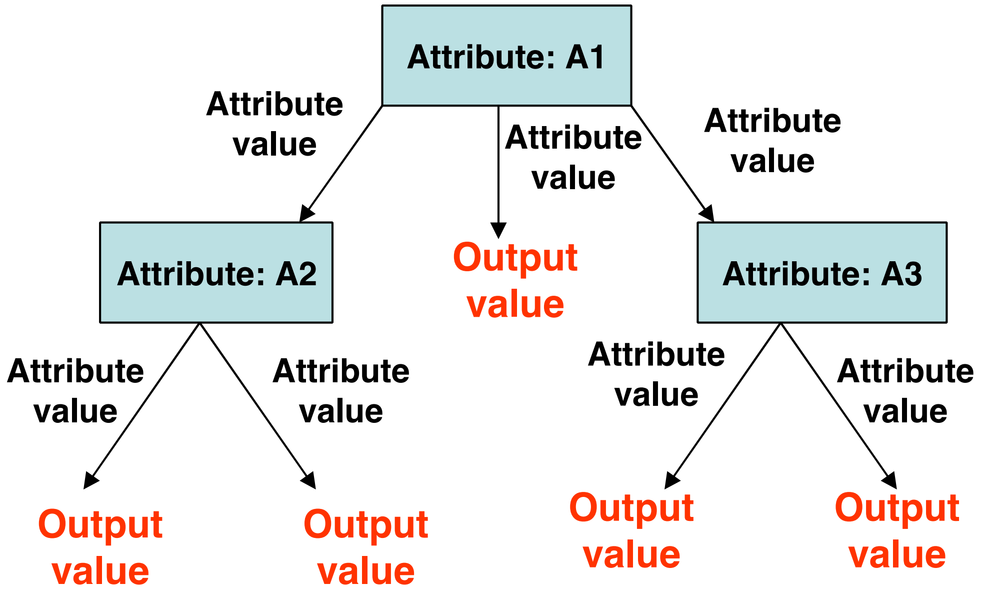
Decision Trees

- One possible representation for hypotheses



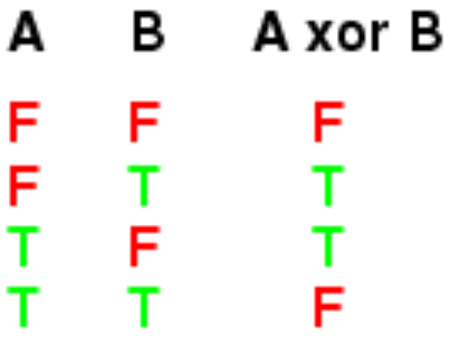
Decision Trees

- Output can be multi-valued, not just binary

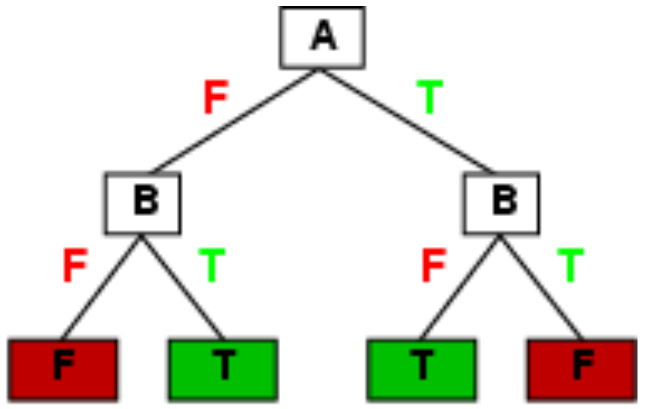


Expressiveness

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

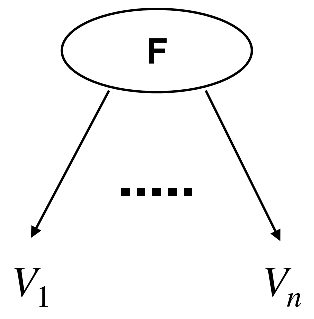


A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- 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

ID3 (Quinlan)

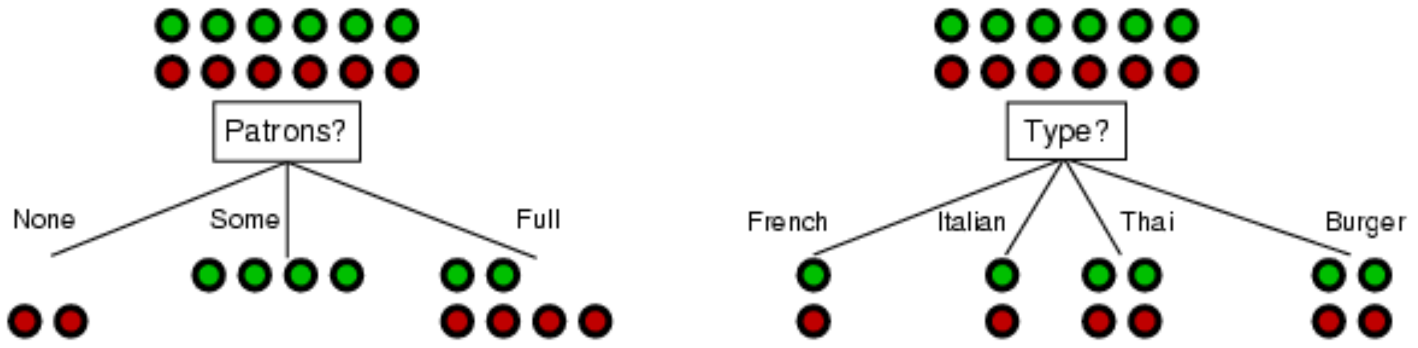


- The algorithm operates over a set of training instances, *C*.
- If all instances in *C* are in class *P*, create a node *P* and stop, otherwise select a feature *F* and create a decision node.
- Partition the training instances in *C* into subsets according to the values of *V*.
- Apply the algorithm recursively to each of the subsets C_i

Generalisation

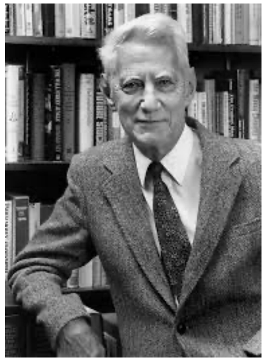
- Training data must not be inconsistent
    - see later how to handle inconsistent data
- Can split attributes in any order and still produce a tree that correctly classifies all examples in training set
- However, want a tree that is likely to generalise to correctly classify (unseen) examples in test set.
- Prefer simpler hypothesis, i.e. a smaller tree.
- How can we choose attributes to produce a small tree?

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”.
- “Informativeness” can be quantified using the mathematical concept of “entropy”.
- Tee can be built by minimising entropy at each step

Information Gain


Claude Shannon
(1916 – 2001)

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.

Choosing Attributes

- The order in which attributes are chosen determines how complicated the tree is.
- ID3 uses information theory to determine the most informative attribute.
- The information content of a message is the inverse of the probability of receiving the message

$$\text{information}(M) = -\ \text{probability}(M)$$

- Taking log (base 2) makes information correspond to the number of bits required to encode a message:

$$\text{information}(M) = -\log_2 \text{probability}(M)$$

Entropy

- Entropy is a measure of how much information we gain when the class value is revealed to us.
- In decision tree learning, when we partition a dataset by a particular attribute the resulting entropy is lower.
    - An entropy of 0 means all the examples have the same class value
    - An entropy of 1 means that they are randomly distributed
- If the prior probabilities of the n class values are $p_1, \ldots, p_n$ then the entropy is

$$H\left(\langle p_{1,} \ldots, p_{n,} \rangle\right) = \sum_{n=1} -p_i \log_2 p_i$$

Entropy and Huffmann Coding

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

- Suppose we want to encode, into a bit string 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.

- Example 1: $H\big(\langle 0.5,\ 0.5\ \rangle\big) = 1 \text{ bit}$

Entropy and Huffmann Coding

- 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 .
- Example 2: $H\big(\langle 0.5,\ 0.25,\ 0.25\ \rangle\big) = 1.5 \text{ bit}$

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

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

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

$$H(\langle p_{1,}\ldots, p_{n,}\rangle) = \sum_{n=1} -p_i \log_2 p_i$$

Entropy

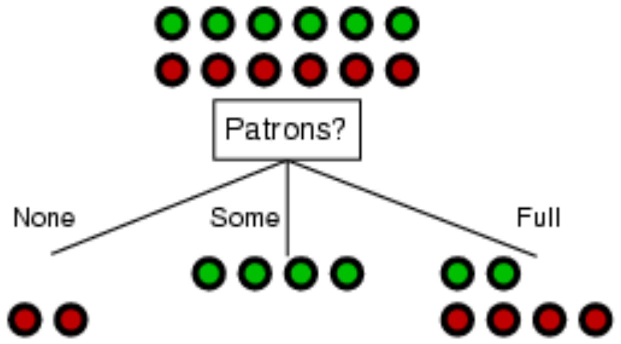
- Suppose we have p positive and n negative examples in a node.
- $H\left(\left\langle \frac{p}{p+n}, \frac{n}{p+n} \right\rangle\right)$ bits needed to encode a new example.
  - e.g. for 12 restaurant examples, $p = n = 6$ so we need 1 bit.
- An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification.
- Let E_i have p_i positive and n_i negative examples
  $H\left(\left\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} \right\rangle\right)$ bits needed to encode an example
  expected number of bits per example over all branches is

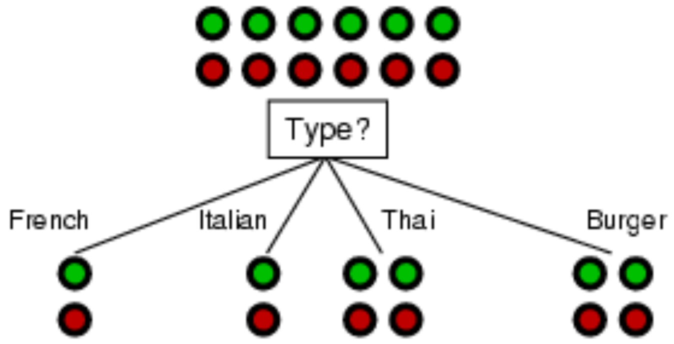
Probability of going down branch *i*

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

- For Patrons, this is 0.459 bits, for Type this is (still) 1 bit ➔ split on Patrons

Choosing and Attribute





$$\begin{aligned}
\text{For Patrons, Entropy} &= \frac{1}{6}(0) + \frac{1}{3}(0) + \frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3})\right] \\
&= 0 + 0 + \frac{1}{2}\left[\frac{1}{3}(1.585) + \frac{2}{3}(0.585)\right] = 0.459 \\
\text{For Type, Entropy} &= \frac{1}{6}(1) + \frac{1}{6}(1) + \frac{1}{3}(1) + \frac{1}{3}(1) = 1
\end{aligned}$$

Entropy

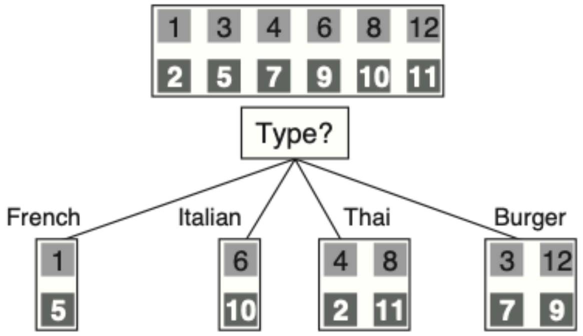
- If the prior probabilities of n attribute values are $p_1, \cdots, p_n$, then the entropy of the distribution is

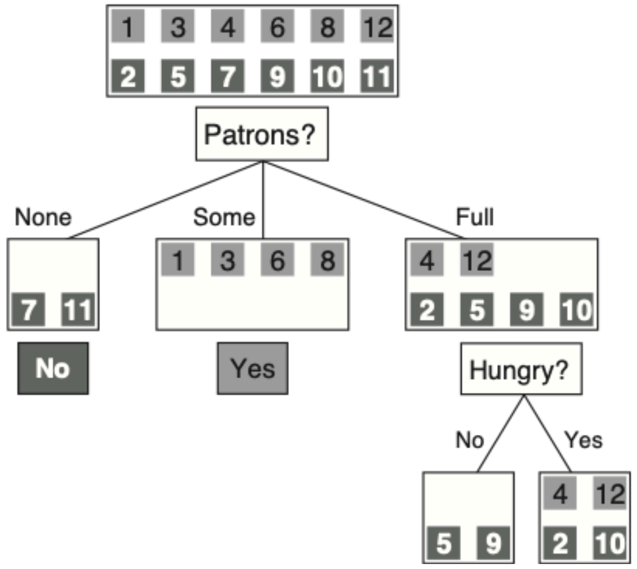
$$H\left(\langle p_{1,}\ldots, p_{n,}\rangle\right) = \sum_{n=1} -p_i \log_2 p_i$$

- 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

- 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

Choosing Next Attribute



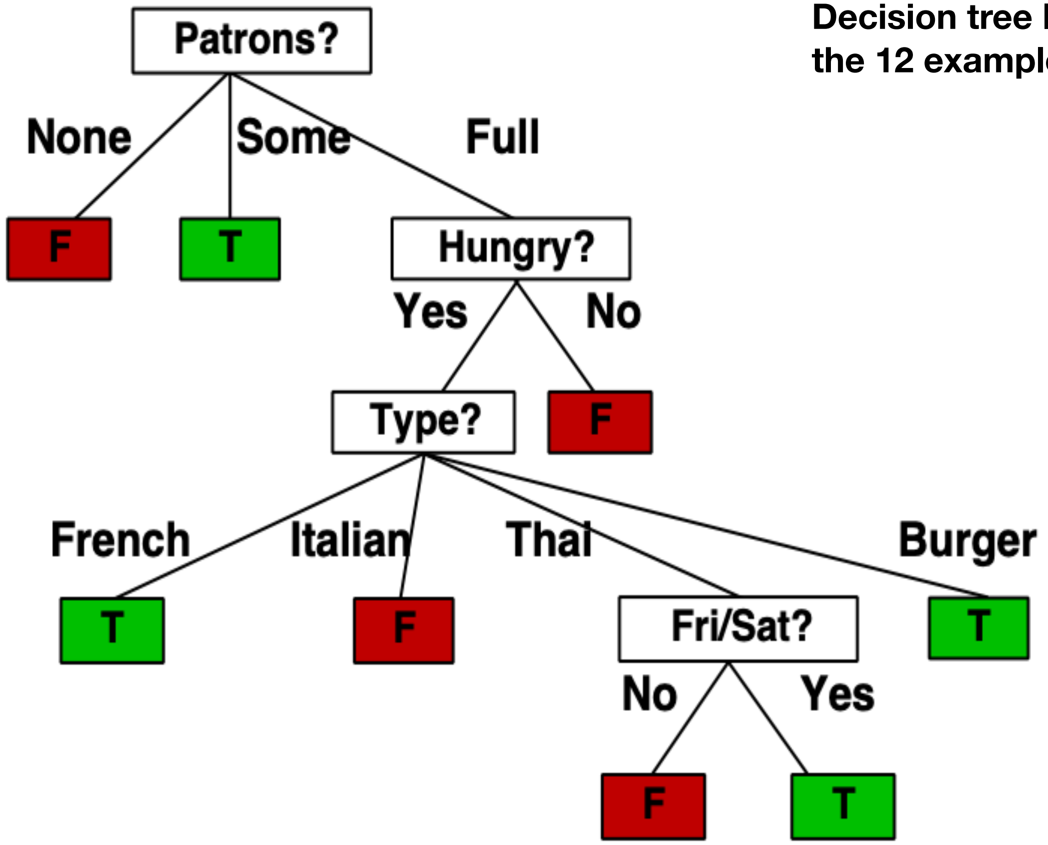


$$\sum_i \frac{p_i + n_i}{p + n} H\left(\left\langle \frac{p_i}{p_i + n}, \frac{n_i}{p_i + n_i} \right\rangle\right)$$

After splitting on Patrons, split on Hungry

	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

Induced Tree



Decision tree learned from the 12 examples.

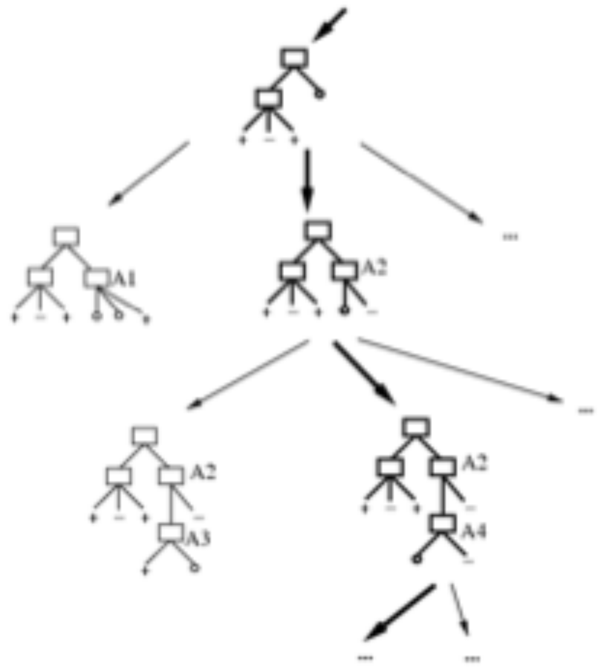
Decision Trees

- 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

Which Tree Should We Output?



- Decision tree learning performs heuristic search through space of decision trees
- Stops at smallest acceptable tree
- Occam’s razor: prefer simplest hypothesis that fits data

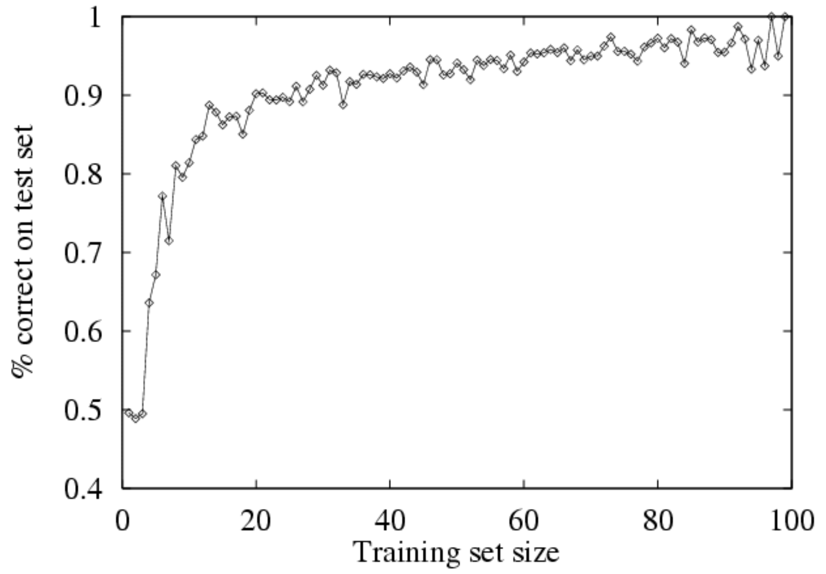
Performance measurement

- How do we know that h ≈ f ?

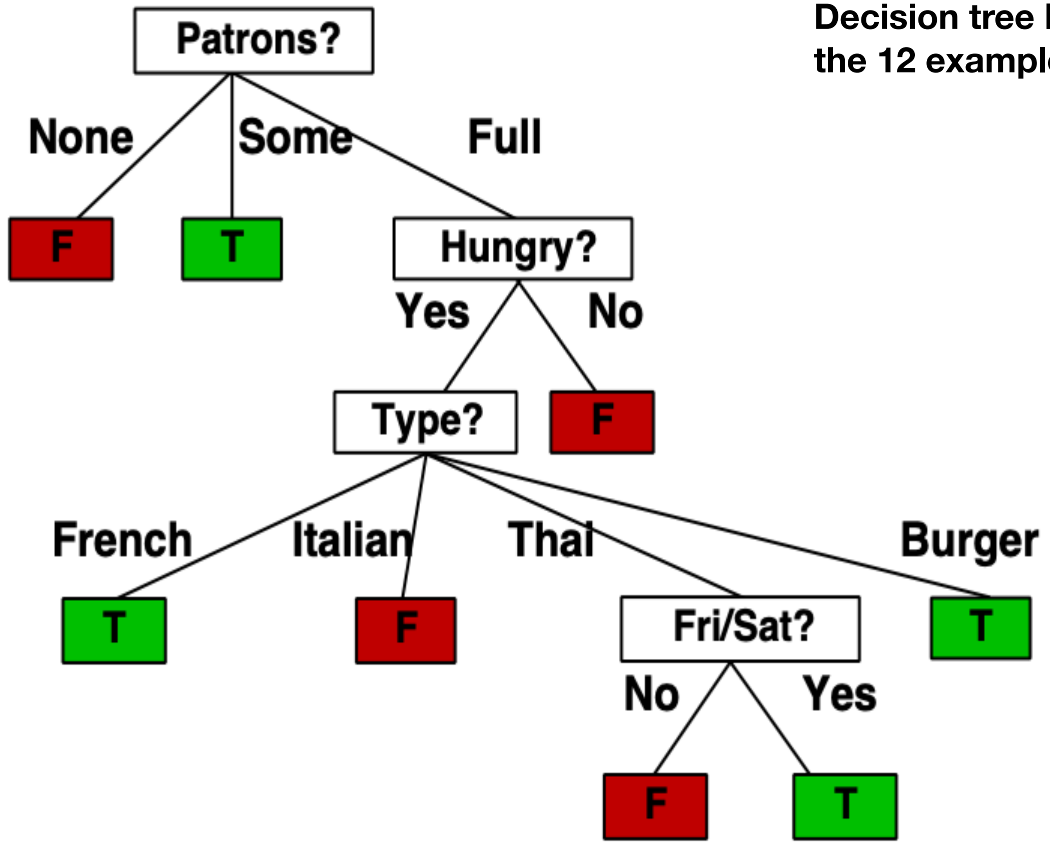
1. Use theorems of computational/ statistical learning theory

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



Induced Tree

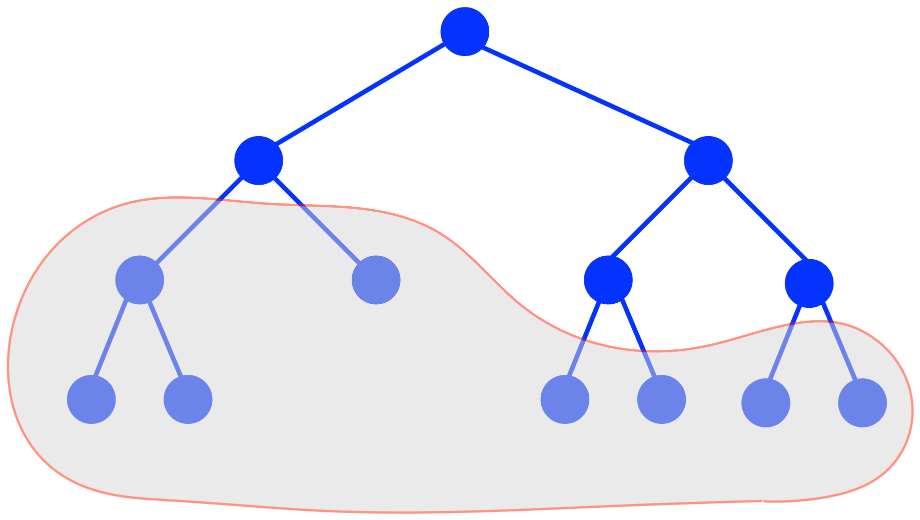


Decision tree learned from the 12 examples.

Overfitting

- What if training data are noisy?
  - Misclassified examples
  - Incorrect measurements
- Making decision tree classify every example (including noise) could make it less accurate
  - Tries to classify bad examples

  ➡ Misclassifies correct examples in test set
- This is called *overfitting*
- Can improve accuracy by pruning branches created by try to classify noise.



Tree Pruning To Minimise Error

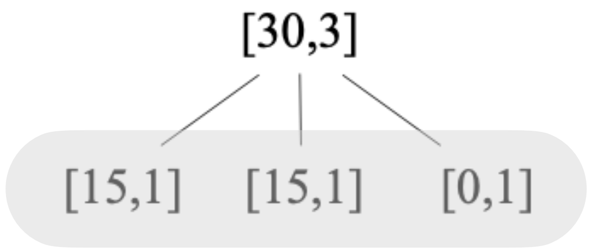
- Maximise expected accuracy, minimise expected probability of error
- For given tree, estimate its expected error
- Also estimate errors for variously pruned tree
- Choose tree with minimal estimated error

How to estimate error?

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

Tree Pruning

- The top node of this tree has
  - 30 +ve examples
  - 3 –ve examples



- It is split 3 ways.
- If we decide to prune it, the top node becomes a leaf node and the decision value is the majority class, i.e. +ve.
- Only prune if the expected error is less than the expected error with the children

Laplace Error and Pruning

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 examples

n = number of training examples 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.

How do we get the Laplace Error?

- Suppose a node as 99 +ve and 1 –ve: $\frac{n}{N} = \frac{99}{100}$ is the probability of the majority class.
- If we decide to prune, the expected error will be $1 - \frac{99}{100} = 0.01$
- This is a good estimate if N is large, but small better to rely on prior probability of class, i.e. $\frac{1}{k}$
- So the Laplace error adds a bias for small N. When N is large, correction is irrelevant

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

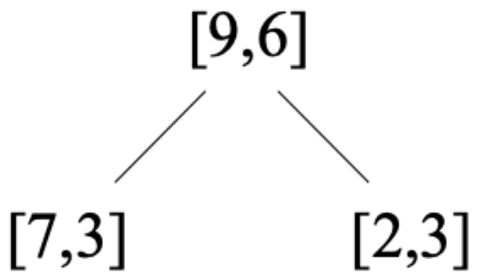
N = total number of training examples

n = number of training examples in the majority class

k = number of classes

Minimal Error Pruning

Should the children of this node be pruned or not?



Left child has class frequencies [7, 3]

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

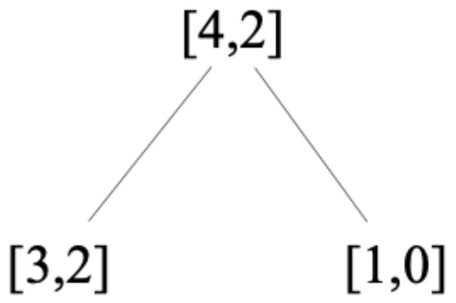
Right child has $E = 0.429$
Parent node has $E = 0.412$
Average for left and right child is:

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

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 $[3, 2]$

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

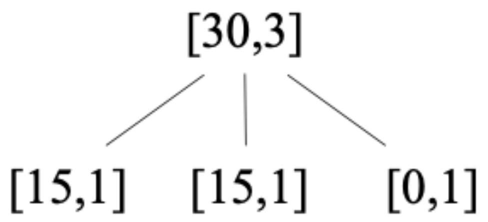
Right child has $E = 0.333$
Parent node has $E = 0.375$
Average for left and right child is:

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

Since $0.413 > 0.375$, children should be pruned

Minimal Error Pruning

Should the children of this node be pruned or not?



Left child has 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 and right child is:

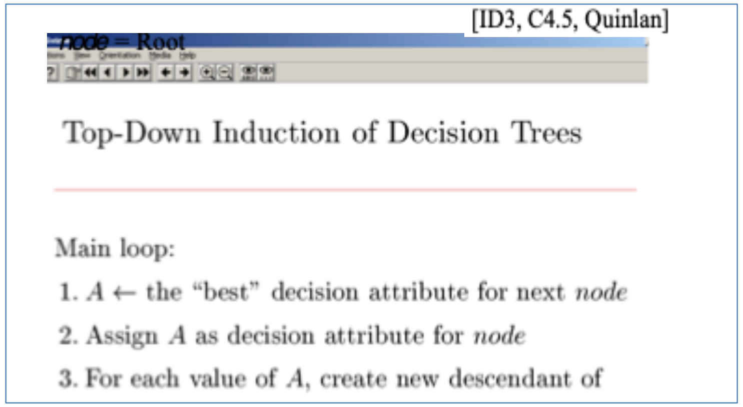
$$E = \frac{16}{33} \times 0.111 + \frac{16}{33} \times 0.111 + \frac{1}{33} \times 0.333 = 0.118$$

Since $0.118 > 0.114$, children should be pruned

Decision Tree Learning Algorithms

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

  .



[ID3, C4.5, Quinlan]

$node$ = Root

Top-Down Induction of Decision Trees

Main loop:

```
1. A ← 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
```

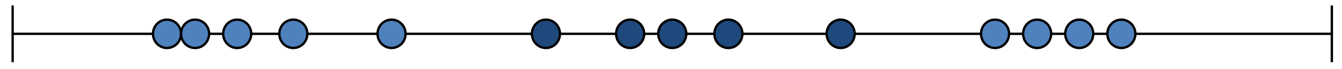
http://www.rulequest.com/

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

Numerical Attributes

- ID3 algorithm is designed for attributes that have discrete values.
- How can we handle attributes with continuous numerical values?

➡ Must discretise values.



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).
  - Easiest domains for decision tree learning is when each attribute takes on a small number of disjoint possible values (e.g., Hot, Mild, Cold).
  - Extensions 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

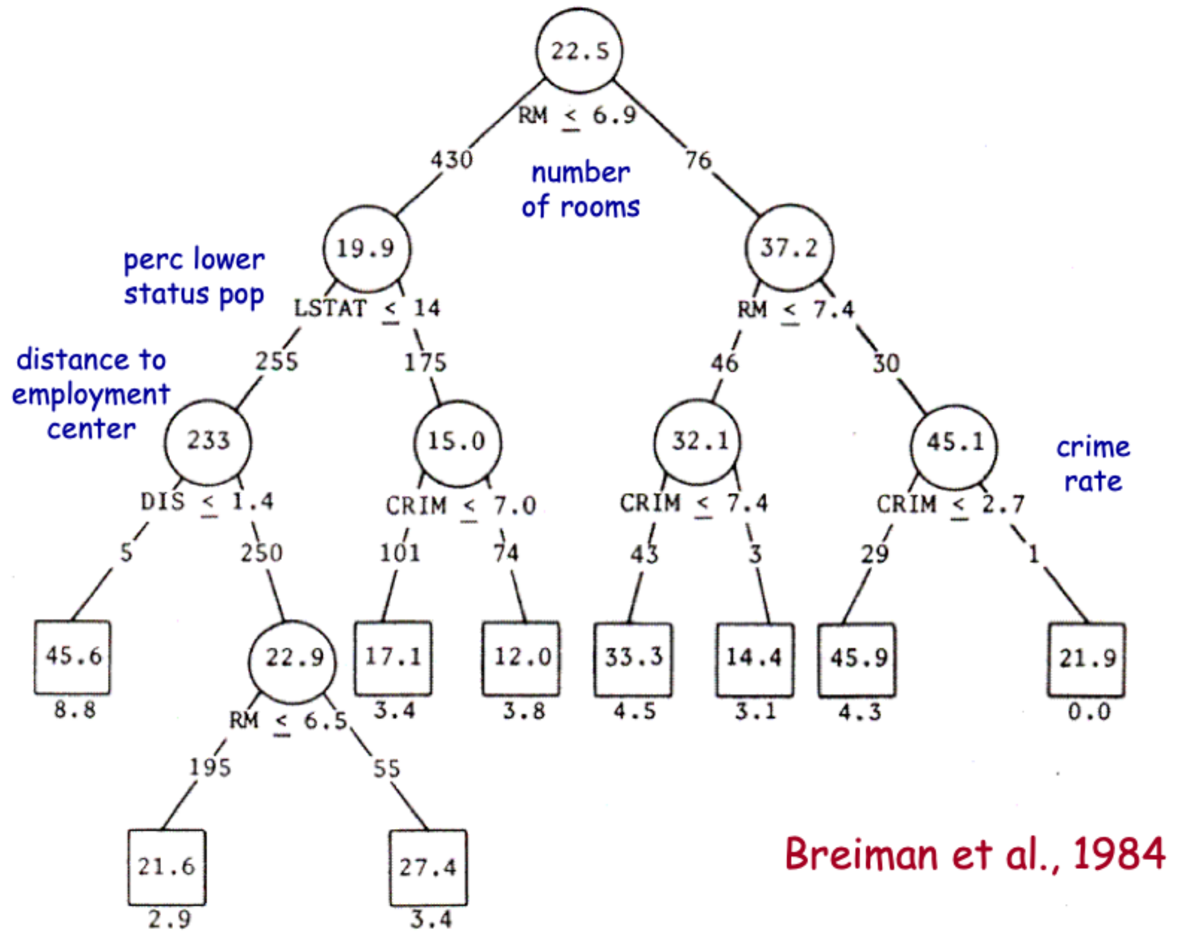
- ID3 (Quinlan 79)
- CART (Breiman et al. 84)
- Assistant (Cestnik et al. 87)
- C4.5 (Quinlan 93)
- C5 (Quinlan 97)
- Trees in WEKA (Witten, Frank 2000 - ...)
- scikit-learn

TDIDT - Top-Down Induction of Decision Trees

Inducing Regression Trees

- Like 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-...)

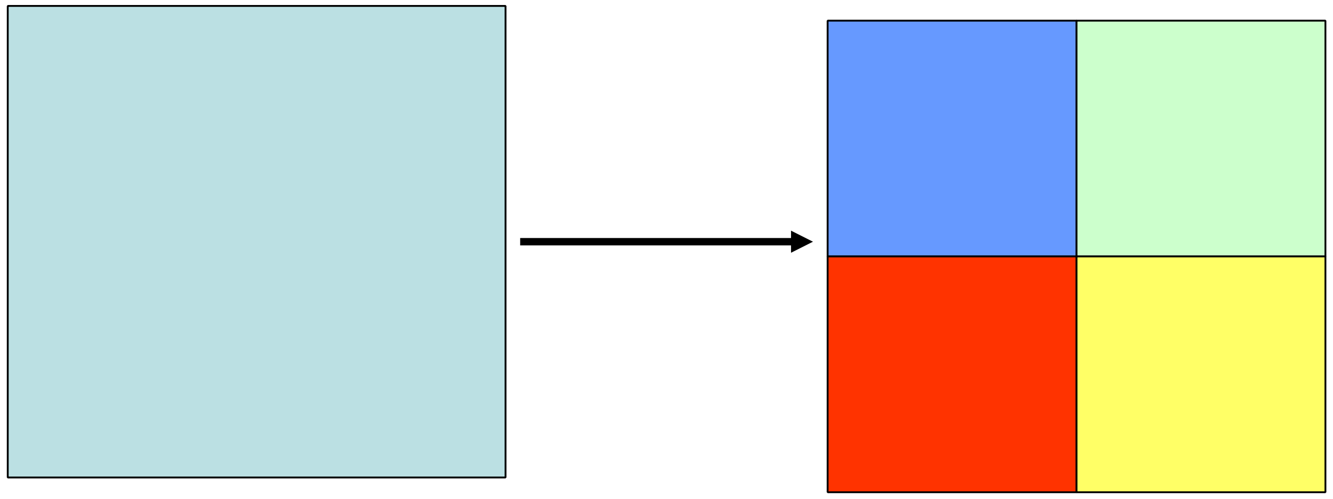
Training and Testing

- 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

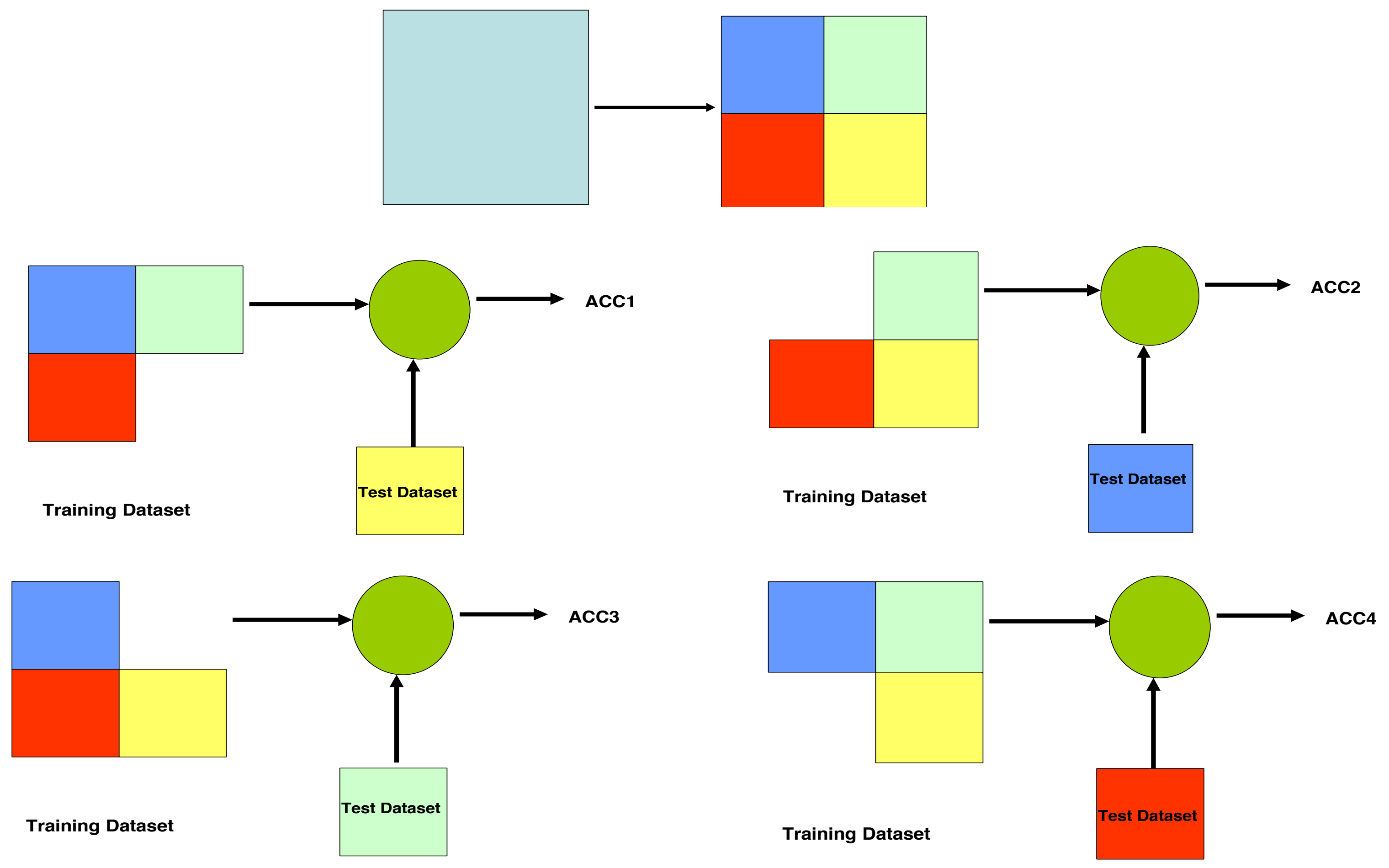
- Self-consistency test: when training and test dataset are the same
- Hold out strategy: reserve some examples for testing and use the rest for training (set part of that aside for validation, if required)
- K-fold Cross validation:
  - If don't have many examples
  - Partition dataset randomly into k equal-sized sets.
  - Train and test classifier k times using one set for testing and other $k-1$ sets for training

4-Fold Cross-validation



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

4-Fold Cross-validation



K-Fold Cross Validation

- 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** $k \in 1..K$:
    $testData \leftarrow k_{th}$ subset
    $h \leftarrow$ 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

- 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 $k \in 1..K$:
    $testData \leftarrow k_{th}$ subset
    $h \leftarrow$ 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
```

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

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