

HCAI

- We have AI that can search, and represent knowledge, and plan actions, and play games.
 - So where does the human factor come into all this?
- AI has practical applications for human-computer interaction (HCI), as well as for autonomous behaviour
 - For example, Bell's automated directory service:
 - "For what city?"
 - "For what name?"
 - More interesting though, is the creation of an agent that can represent expert knowledge



Expert Systems

- Programs that represent a human expert's knowledge in a certain domain, with the ability to analyze a situation, and possibly recommend a course of action
- First devised in 1970's, was in vogue in industry applications throughout the 1980's. Still used today, in specialized applications
- Example #1: HelloYellow (310-YELO)
 - voice-driven Yellow Pages searching application
 - "conversational marketing"
 - uses business types and location to narrow down recommendations for restaurants, shops, etc.

More Expert Systems

- Example #2: Mycin (1970's)
 - medical expert system, diagnosed infection blood diseases
 - correct diagnosis rate of about 65%, above most non-specialists and only slightly below specialist rates (~80%)
 - never actually used in practice, due to liability issues
- Example #3: Microsoft troubleshooter
 - solves problems by working with user to diagnose symptoms
 - product's effectiveness is sometimes questionable, but it allows the Help Center to reduce the number of trivial support cases that it has to deal with
- Example #4: Autopilots



Expert System Components

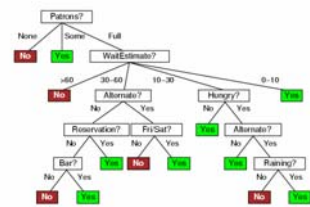
- Knowledge Base
 - stores the **attributes** that affect the problem domain, as well as possible **classifications** and **solutions** to the situation
 - stores rules that connect factors with solutions, usually in conjunctive if-then form
 - rules are either set manually by a domain expert or generated automatically from data
- Interface
 - obtains information about current situation from user or world
 - usually prompts user for information on the factors that would narrow down the possible situations most effectively
 - continues to prompt for information until the possible situations all belong to the same **class** of problems

Expert System Tools

- Expert systems are typically stored as a set of rules, through which a satisfiability search is performed after obtaining each new piece of information
- Example: **CLIPS**
 - C Language Integrated Production System
 - NASA-sponsored expert system software, which automatically creates an expert system based on user-defined facts and rules
- Question: What information should be obtained first, to classify the problem the fastest?
- Organization of questions can be represented as a **decision tree**

Decision Trees

- Example decision tree for waiting for a table (from Russell & Norvig, p. 654)



Decision Trees (cont'd)

- Decision tree components:
 - Internal nodes of decision tree represent tests of one of the attributes of the situation
 - Branches of tree represent the possible values of the test used in making the decision
 - Leaf nodes represent the classification of this problem
- Simplification rules:
 - Assume that branches represent discrete values (continuous values are an extension)
 - assuming boolean values is a further simplification
 - Classifications are either positive or negative (multiple assessment possibilities are also an extension)

Decision Tree Features

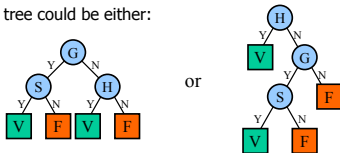
- Advantages of expert systems & decision trees:
 - Industrial benefits
 - reduced worker demand
 - less downtime while waiting on scarce expert resources
 - Simple to comprehend and interpret (white box model)
 - Robustness
 - can process large dataset without pre-processing
 - can be verified statistically against other test datasets
- Disadvantages of expert systems & decision trees:
 - Data needs to be very well-specified
 - Ordering of tests can lead to very bad decision trees

Bad Decision Trees

- Decision trees can be bad for much the same reason that binary search trees can be bad:
 - Example: Given the following data...

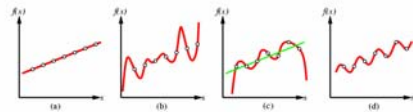
Example	Smooth?	Green?	Hollow?	Type
Lime	No	Yes	No	Fruit
Cucumber	Yes	Yes	No	Veg
Apple	Yes	No	No	Fruit
Pepper	Yes	No	Yes	Veg

... the tree could be either:



Bad Decision Trees (cont'd)

- Other risk of decision trees is overfitting the data



- Sometimes it's better to have one or two misclassified values than to have the decision tree branch too far down, just to capture the data
 - sparse data problem: some categories might only have one or two elements, very prone to error or noise
 - Occam's Razor: solution to a situation is usually the simplest one available (within reason)

Decision Tree Strategies

- One strategy is to keep most **informative** nodes at root (nodes whose attribute splits the data the best)
- Measurement for information about a node is **entropy**

$$I(P(C_1), P(C_2), \dots, P(C_n)) = \sum_{i=1}^n -P(C_i) \log_2 P(C_i)$$

- Gives a measurement in **bits**.
- Nodes with equal probability for two possibilities
 - $I(1/2, 1/2)$ (fair coin toss)
 - transmit 1 bit of information:
 - $I(1/2, 1/2) = -1/2 \log_2 1/2 - 1/2 \log_2 1/2 = 1$ bit
- Nodes with 99% of getting one value (e.g. heads) only transmits 0.08 bits of information from a decision

Choosing Attribute Tests

- As the probability of the possible classification categories nears 0 and 1, the entropy test will approach 0 overall (highest entropy value is 1)
- Selection strategy:
 - keep attributes that minimize the entropy in the nodes that result from the data split (greedy selection strategy)
 - stop selecting attributes when entropy is zero (leaf node condition)
- In fruit/vegetable example:
 - entropy(G) = 1 bit
 - entropy(S) = entropy(H)
 - $= -1/4 \log_2 1/4 - 3/4 \log_2 3/4$
 - $= -1/4(-2) - 3/4(-0.415)$
 - $= 0.915$ bits
 - Only choose G for root attribute if you need to guarantee a 2-attribute decision tree. Otherwise, S or H are better.

Information Gain

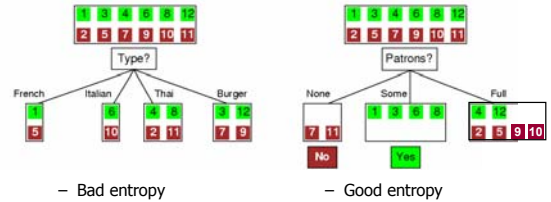
- Another attribute test is the **gain** in information that comes from choosing an attribute to split the decision tree cases.
- The gain is the difference between the information needed at the node where an attribute is chosen, and the information needed by the nodes that result from choosing the attribute.

$$\text{Gain}(C,A) = \text{entropy}(C) - \sum_{v \in V(A)} P(A=v) \text{entropy}(C|A=v)$$

- C is the classification category, A is the variable for the attribute, V is the set of attribute values, and v is a particular value from this set.

Entropy Examples

- Picking a restaurant to go to:



Training & Testing

- To show how more data eliminates the problems of sparse data and noise, separate data into **training** and **test** sets.
 - After creating model based on examples in training set, put test cases through decision tree and record the percentage that get classified accurately.
 - The result is that performance improves as training size increases, although this might be a result of **peeking** during training (allowing test set to gradually influence training set).

Decision Tree Pruning

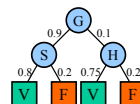
- To keep the decision tree simple and avoid overfitting the data, we can prune the less relevant attributes from the tree
 1. First, put the tree into rule-based form.
 2. (Rules from the fruit example would be:
 - if (green && smooth) then Vegetable.
 - if (green && ~smooth) then Fruit ...).
 3. Construct a contingency table for the rules, that measures the number of occurrences for an attribute in each rule
 4. Calculate the **expected value** for each value of an attribute, and see how much the occurrences of these value deviates from the expected value → the χ^2 "chi-squared" test
 5. Values with low deviation can be eliminated from the decision tree
 6. Rebuild the tree, using the modified attribute list

Decision Tree Algorithms

- ID3
 - basic algorithm; uses entropy measurement to select attributes for decision tree nodes
 - chooses attributes to minimize the entropy in the resulting nodes
- CART (Classification and Regression Trees)
 - relies on the Gini impurity test ($1 - \sum \text{frequencies}^2$) to check if the leaf categories are homogenous or not
- C4.5 & C5.0
 - based on ID3 algorithm
 - prunes trees to lower decision tree height
 - also considers cases with missing attribute data, varying costs and continuous values

Decision Tree Variations

- Branch costs
 - attribute tests aren't always 100% certain.
 - by placing confidence values on each branch, the expert system can model uncertainty in its decision-making
 - a key ability for when data doesn't classify neatly into categories
 - result is a confidence value for all leaf categories, based on an overall calculation.



- e.g. object is green, smooth and not hollow

$$C(V) = (0.9)*(0.8) + (0.1)*(0.75) = 0.795$$

$$C(F) = (0.9)*(0.2) + (0.1)*(0.25) = 0.205$$

Decision Tree Variations (cont'd)

- Continuous values
 - more difficult to ascertain attribute divisions for continuous values than for discrete values, but still possible
 - rather than testing the attribute's one or two possible values, intervals are chosen along the continuous range of the attribute, to see which reduces the entropy of the system the most
 - that attribute division is then compared to the other attribute values of the system

Ensemble Learning

- To help train the decision tree faster, we can create several decision trees and have them build a stronger model by weighting them
- The training examples can be weighted as well, so that examples that were misclassified earlier can be weighted more heavily in later training runs
- This technique is called boosting
- Idea behind this is that a single weak decision tree might misclassify a situation, but several classifiers are less likely to misclassify in the exact same way
 - get majority opinion when applying label

