COMP219: Artificial Intelligence

Lecture 25: Supervised Learning

Overview

- Last time
  - Planning in the real world; scheduling with time and resource constraints; critical path method; minimum slack; HTN
- Today
  - Types of learning
  - Supervised learning
    - Learning decision trees
- Learning outcomes covered today:

  Identify or describe the major approaches to learning in AI and apply these to simple examples

Why do we want an agent to learn?

- Cannot anticipate all situations – unknown environments (e.g. navigating new space)
- Cannot predict changes over time (e.g. react to the stock market)
- Don’t know how to design some solutions (e.g. recognising faces) or it’s too time consuming to do so
- Learning modifies the agent’s decision mechanisms to improve performance

A Learning Agent

- An agent is learning if it improves its performance on future tasks after making observations about the world
- Any component of an agent can be improved by learning, but the choice of technique depends on:
  - What the component is
  - What prior knowledge the agent has
  - How the data and component are represented
  - What feedback is available to learn from

Example: Training a Taxi Driver Agent

- When the instructor shouts “Brake” the agent may learn a condition-action rule for when to brake; agent also learns when the instructor does not shout
- By seeing camera images which it is told are buses, the agent learns to recognise buses
- By trying actions and observing the results (e.g. braking hard on a wet road), agent can learn effects of actions
- When it receives no tip from passengers after driving wildly, it can learn a component of its utility function

Three main types of learning

- Supervised learning
  - Agent learns a function from observing example input-output pairs
  - e.g. taxi agent told “that’s a bus”
- Unsupervised learning
  - Learn patterns in the input without explicit feedback
  - Most common task is clustering
  - e.g. taxi agent notices “bad traffic days”
- Reinforcement learning
  - Learns from a series of reinforcements: rewards or punishments
  - e.g. 2 points for a win in chess

Supervised Learning

- There are many supervised learning methods:
  - **Decision trees**
  - **Linear regression**
  - **Linear classification**
  - Logistic regression
  - Neural networks
  - **Non-parametric** models, e.g. **nearest neighbours** and locally weighted regression
  - Support vector machines
- We will introduce just a few of these – entire modules on machine learning do not cover all of them

Supervised Learning Applications

- Classification problems
- Facial recognition
- Handwriting recognition
- Speech recognition
- Spam detection
- …

Congrats: You have won $1 million…

The Supervised Learning Task (I)

Given a training set of N example input-output pairs

$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N),$

where each y_j was generated by an unknown function

$y=f(x)$,

discover a function h that approximates the true function f.

- Note:
  - x and y can be any value (not just numbers)
  - h is a hypothesis

The Supervised Learning Task (II)

- Learning is a search through the space of possible hypotheses for one that performs well, even on new examples beyond the training set
- Test the function by dividing the examples into a test set and a training set
  - Learn a function from the training set
  - Test its accuracy by applying to the (unseen) test set
- Hypothesis generalises well if it correctly predicts y from novel examples

Learning Problem

- When y is **discrete**, i.e. one of a finite set of values (e.g. sunny, cloudy, yes, female) we have a classification problem
- When y is **continuous**, such as a number (e.g. tomorrow's temperature, age) we have a regression problem

Supervised Learning Issues

- Choosing between multiple consistent hypotheses
  - Ockham's razor: choose the simplest hypothesis consistent with the data

- Lack of labelled data
- Data noise – labels may not be accurate
  - e.g. Learning ages from photos of faces – take photos and ask age. Some people may lie about their age – systematic inaccuracy not random noise
- Semi-supervised learning:
  - Agent given a few labelled examples
  - Must learn a large collection of unlabelled sample

Learning Decision Trees (I)

- A decision tree is a simple representation for classifying examples, which is a natural representation easily understood by humans
- Decision tree learning is one of the most successful techniques for supervised classification learning
- A decision tree represents a function that takes an input *vector* of attribute values and returns a “decision” – a single output value or class
- Input and output values can be discrete or continuous

Learning Decision Trees (II)

- In a decision tree:
  – Each internal (non-leaf) node is labelled as an input attribute; it tests a single attribute value
  – Arcs are labelled with possible attribute values
  – Leaves are labelled with a class/value to return
- To classify an example, filter it down the tree:
  – For each node, follow the arc representing the example’s attribute value
  – When a leaf is reached, return the classification

Example Decision Tree Problem

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)

Attribute-based Representations

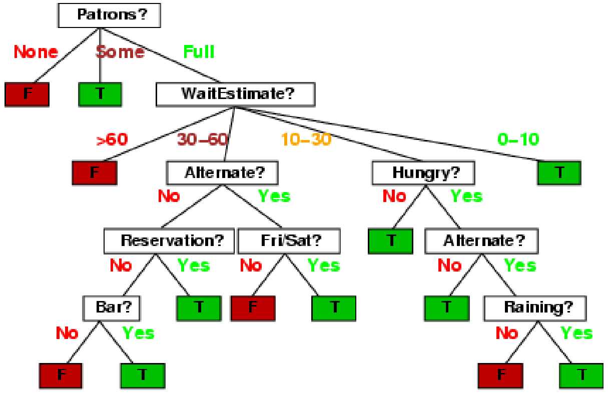
- Examples described by attribute values (Boolean, discrete, continuous)
- e.g. situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)
- Must learn a definition for the Boolean goal predicate *Wait*

Restaurant Example Decision Tree

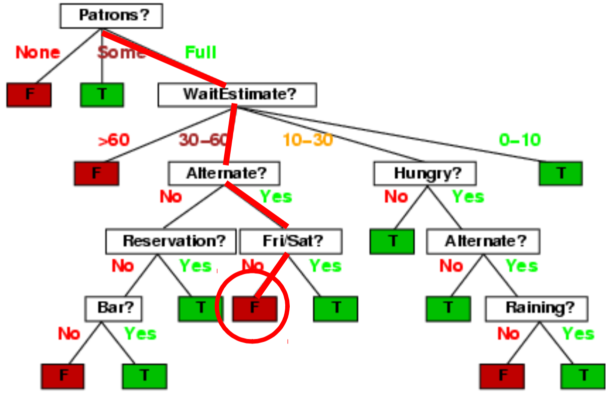
- One possible representation for hypotheses
- e.g. Here is the “true” tree for deciding whether to wait:



- NB: doesn’t use *Price* and *Type* attributes

Tracing Example X_2

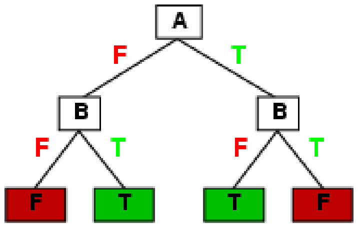
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F



Expressiveness of DTs

- 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



- Any function in propositional logic can be expressed as a DT:

  Goal ⇔ *(Path1* ∨ *Path2* ∨ *…)*

  where each path is a conjunction of attribute-value tests:

  Path=(Patrons=Full ∧ *WaitEstimate=0-10)*

Decision Tree Learning (I)

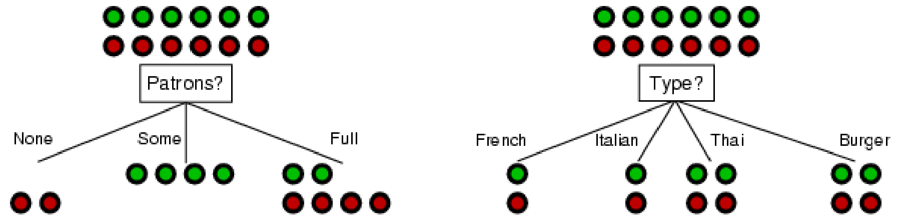
- Aim: find a small tree consistent with the training examples
- Training set for a Boolean DT is (x,y) pair where x is the input vector and y is the Boolean output
- Greedy divide-and-conquer strategy: (recursively) choose “most significant” attribute as root of (sub)tree
  - Divides the problem into smaller sub-problems
  - Always choose the most significant attribute first: the one that makes the most difference to classification in the training set
  - Hope to classify by the smallest number of tests – then the tree will be shallow and all paths short

Decision Tree Learning (II)

- Four cases to consider for recursive DT problems:

1. If remaining examples all one class, STOP
2. If examples are a mix of class, choose best attribute to split them
3. If no examples remaining (i.e. no examples observed for this combination of attribute values) return default value
4. If no attributes left but examples of each class, then the examples have the same description but different classifications, because:
   - Error or noise in data
   - Non-deterministic domain
   - Can't observe an attribute that distinguishes examples

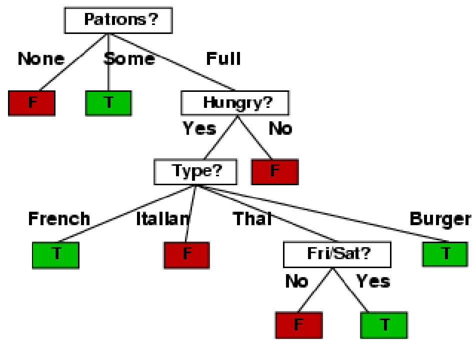
Choosing an Attribute

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

- *Patrons?* is a better choice as it separates more examples

Restaurant Example cont'd

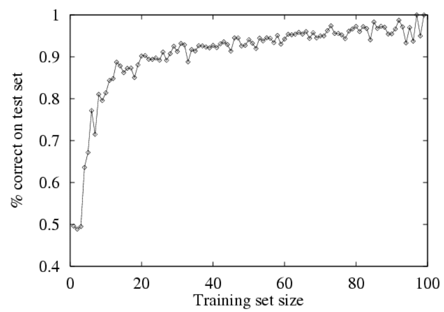
- Decision tree learned from just 12 examples:



- Substantially simpler than "true" tree – a more complex hypothesis isn't justified by small amount of data

Evaluating Accuracy: Learning Curve

- How do we know that $h \approx f$?
- Try h on a new test set of examples:
  - Randomly split the example set into a training set and a test set
  - Learn h then test its accuracy by applying to test set
  - Repeat (e.g. 20 trials) using different size of training set, then plot
- Learning curve = % correct on test set as a function of training set size



Can we broaden the application of Decision Trees?

- Must overcome several issues:
  - *Missing data*: how to classify?
  - *Multi-valued attributes*: usefulness?
  - *Continuous and integer-valued attributes*: split point?
  - *Continuous-valued output attributes*: e.g. for numerical output we use a regression tree - each leaf has a linear function rather than a value

Summary

- Learning needed for unknown environments, 'lazy' designers
  - Different types of learning
  - For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
    - One method: decision tree learning

- Next time
  - Linear models for supervised learning