Artificial Intelligence

Lecture 13 - Learning

Outline

- What is learning
- Kinds of learning
  - rote learning
  - inductive learning
  - reinforcement learning
  - explanation-based learning
  - inductive logic programming
- Choice of representation
- Evaluating learning algorithms (PAC-learning)

Learning

- It is often impossible (or simply inconvenient) to provide a system with all the knowledge or reasoning capabilities it needs
- Learning is the acquisition of new knowledge (or, less often, reasoning capabilities)
- Many different kinds of learning and things which can be learnt, e.g.,
  - facts
  - rules
  - policies
  - inference procedures etc.

What Is Learning?

- Learning can be viewed as the problem of producing a mapping (function) from inputs to output
- Inputs are typically a description of an object or situation
- Output may be a property (e.g., a classification such as 'edible' or 'creditworthy'), a relation, which action to perform next (e.g., a particular move in chess), the outcome of an action, or something more abstract, e.g., a theory etc.
- The mapping can be represented in many ways, e.g., decision trees, neural networks, sets of formulas in first order logic

Kinds of Learning

- There are many machine learning approaches which differ in:
  - representation of the learned information – e.g., properties, relations, which action to perform next, etc.
  - whether the learning is supervised (correct answers are given during learning) or unsupervised (no answers given)
  - whether learning is 'knowledge free' or whether background information about the domain is assumed

Rote Learning

- The acquisition of new facts, rules etc. by “being told”, e.g., “*Paris is the capital of France*”, “*All men are mortal*” etc.
  - used by McCarthy's (1968) 'advice taker' system, where the user provides new information about a particular domain
- The acquisition of new facts produced by the system itself, e.g., remembering the result of a computation (speedup learning)
  - used in Samuels (1959) checkers program to learn the score associated with a given board position

Inductive Learning

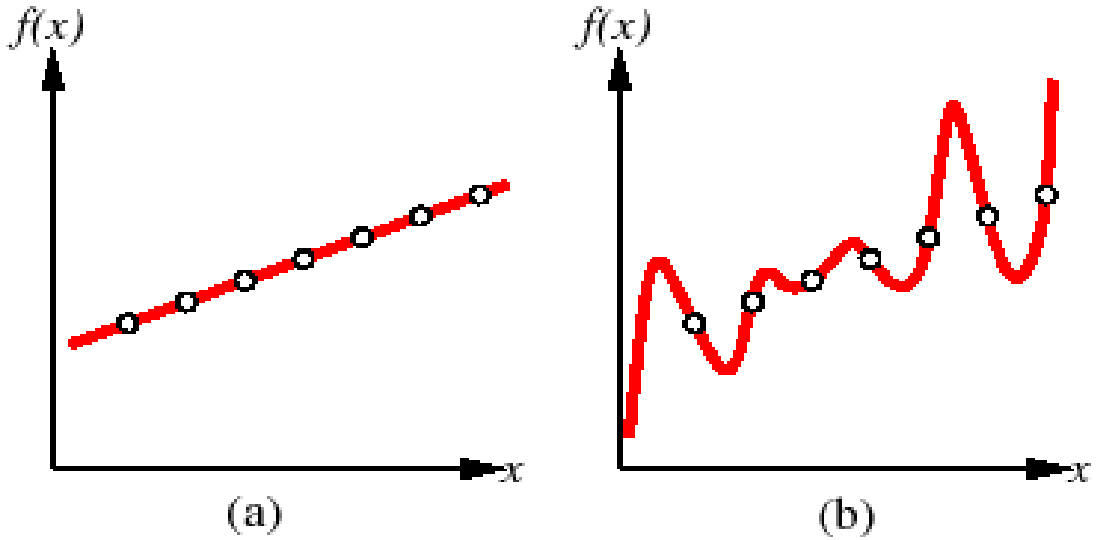
- A form of supervised learning, where we are given the correct (or approximately correct) value of the function for particular inputs
- Each example is of the form (x, $f(x)$), where x is the input(s) and $f(x)$ is the value of the function for x
- Given a set of examples (x, $f(x)$), generate a function $h(x)$ which approximates f
- h is called a hypothesis
- A hypothesis should be
  - consistent (i.e., should fit the data)
  - generalise well (predict unseen examples correctly)

Example: Credit Scoring

- Imagine a credit scoring system that decides whether a person should be given a loan
- Input data is historical information about previous loan applications (applicant's age, gender, address, job, credit history etc.)
- The value of the function is whether the loan should have been approved, e.g., whether the person defaulted on the loan
- Aim is to learn an *h* that predicts, for new loan applications, whether the applicant should be given a loan

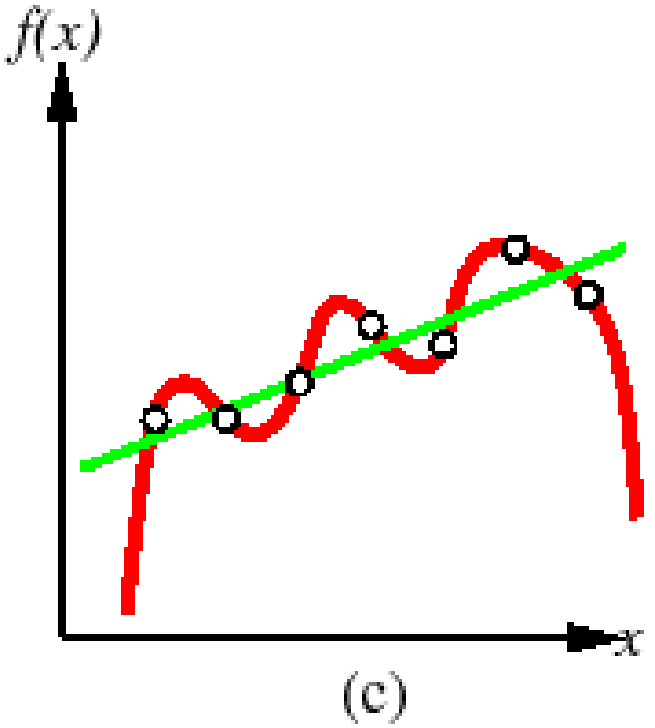
Simplest Hypothesis

- How to choose among multiple consistent hypotheses
- Prefer the simplest hypothesis which is consistent with the data
- e.g., if we can fit the data with a straight line, we should prefer this to a high degree polynomial which is also consistent with the data



Simplest Hypothesis

- In some cases we may prefer a simple hypothesis which does not fit the data perfectly to a more complex hypothesis which is consistent with the data
- e.g., a simple straight line may generalise better (predict new examples better) than a complex polynomial



Hypothesis Space

- The set of hypotheses we will consider is called the *hypothesis space*, e.g., the set of all polynomials of degree at least at most *k*
- The possibility of finding a simple, consistent hypothesis depends critically on the hypothesis space chosen
- A learning algorithm is *realisable* if the hypothesis space contains the true function; otherwise it is *unrealisable*
- e.g., trying to learn a sinusoidal function with a hypothesis space consisting of polynomials of finite degree is unrealisable, since they can't be represent a sinusoidal function accurately
- Since the true function is not known (it's what we are trying to learn), we often can't tell if a given learning problem is realisable

Decision Trees

- Decision tree learning is one of the simplest forms of inductive learning
- Decision trees represent boolean functions – input is an object or situation described by a set of properties; value is a yes/no decision
- Aim is to find a compact representation of the input examples that also correctly predicts on unseen cases
- Proceed by repeatedly choosing the most informative property to split the samples on the value
- In the credit scoring example, “having a job” might be the most informative property, and a test on this would form the root of the tree

Reinforcement Learning

- Form of unsupervised learning which uses reinforcement (rewards) to learn which action to perform next
- Agent is never explicitly told what the right action is
- Reward is typically associated with a *sequence* of actions
- Not clear which action(s) are 'responsible' for the reward, e.g., which moves causes the agent to win or lose a game
- May be model-based or model-free, though the later restricts the ability to learn in complex environments

Learning and (Prior) Knowledge

- Considers the problem of agents which already know something about the domain and are trying to learn more
- Learning problem is usually formulated in logical terms, as this makes it easier to specify *partial information* about the function to be learned
- e.g., if the aim to find a hypothesis which explains the classification of the examples given their descriptions, we have

$$Hypothesis \wedge Descriptions \models Classifications$$

Explanation-Based Learning

- Extracts general rules from a *single* example by *explaining* the example and generalising the explanation
- “explanation” is usually a logical proof, but it can be any form of reasoning or problem solving
- In EBL the background knowledge is assumed to be sufficient to explain the hypothesis (and hence the classification):

$$\textit{Hypothesis} \wedge \textit{Description} \models \textit{Classification}$$

$$\textit{Background-Knowledge} \models \textit{Hypothesis}$$

- Nothing factually new is learned from the example instance
- Instance can be seen as guiding the process of converting first-principles theories into useful, special-purpose knowledge

Inductive Logic Programming

- Finds inductive hypothesis that explain sets of observations using background knowledge

  $$Background\text{-}Knowledge \wedge Hypothesis \wedge Descriptions \models Classification$$

- Hypothesis must also be consistent with background knowledge – reduces the size of the hypothesis space
- Can learn relational knowledge that is not expressible using propositional attributes
- e.g., can generate new predicates to express generalisations

Choice of Representation

- All learning can be viewed as learning the representation of a function
- Choice of representation for the hypothesis, *h*, (i.e., the hypothesis space) is critical
- As with reasoning, there is a tradeoff between expressiveness and efficiency
- The more expressive the language (e.g., first order logic) the more computation and examples will be required to learn a compact representation

Evaluating Learning Algorithms

- Set of examples is split into two subsets: the *training* set and the *test* set
- Learning algorithm uses the training set to induce a hypothesis which is then evaluated against the test set
- Training and test sets are historical data and are typically fairly small: data are difficult to collect and using too many examples in the training set results in overfitting (poor generalisation)
- Function produced may not correctly predict all the examples in the test set
- Repeat for different sizes of training set and randomly selected of each size to obtain the average prediction quality of the learning algorithm as a function of the size of the training set (learning curve)

Computational Learning Theory

- How do we know that *h* is close to the target function *f*, if we don't know what *f* is?
- Assume that the training and test sets are drawn randomly from the same population of examples – i.e., that the future is like the past
- Any hypothesis that is consistent with a sufficiently large set of examples is unlikely to be wrong – *probably approximately correct*
- An hypothesis *h* is *approximately correct* if the probability that *h* is different from *f* on an example is less than a (small) constant
- With enough examples there is a high probability that all consistent hypotheses will be approximately correct

Approximately Correct Hypotheses

- The *error* of a hypothesis h with respect to the true function f is the probability that h is different from f on an example

$$error(h) = P(h(x) \neq f(x) \mid x \text{ drawn from } D)$$

  where D is the distribution from which examples are drawn

- A hypothesis is *approximately correct* if $error(h) \leq \varepsilon$ where ε is a small constant

How Many Examples are Needed?

- A hypothesis *h* has error at most ε with probability at least 1 – σ, if it is consistent with at least *N* examples, where *N* is given by

$$N \geq \frac{1}{\epsilon}\left(ln\frac{1}{\delta} + ln|\mathbf{H}|\right)$$

- σ is a small constant, and **H** is the set of possible hyptheses
- *h* is said to be probably approximately correct
- Inductive learning with no prior knowledge of the target function is generally *very* hard – similar results for neural networks

Other Problems ...

- Systems developed using machine learning techniques do not necessarily reason in the same way as people, e.g.:
  - a decision tree may ignore what (to a human) appears to be “relevant” information
  - inductive logic programming may invent entirely new concepts which do not correspond to those used by humans when reasoning about the problem
- May be hard for users/managers to trust output of the system
- Similar problems to other AI techniques (e.g., expert systems, belief networks etc.)

Example: Recommender Systems

- A *recommender system* is a system which suggests products or services based on a user's previous choices or choices of other, similar users
- Now widely used in online retailing: e.g., Amazon's “New/ Recommended for you”, “other users who bought this also bought” etc.; iTunes “genius” feature etc.
- Other applications include search query prediction, collaborative email processing etc.
- Often developed using machine learning techniques

Case Study: the 5lb ham ...

- Some years ago, Net Perceptions was installing its recommender software at a major catalog retailer
- Recommender system was tuned to produce high quality recommendations that were successful against historical sales data
- Recommendations were produced in real time to call centre staff while processing customer calls
- e.g., if the customer orders a pink dressing gown, the system might suggest fuzzy pink slippers to go with it, based on prior sales experience

Case Study: the 5lb ham ...

- The day arrives for the big test when the system is used “live” for the first time
- The first call comes in: the customer orders a complete two week diet package – all the powdered food you need to eat to lose 10lb in two weeks
- When the order is entered into the system, top of the list of recommendations is a *5lb tinned ham ...*
- Developers and company representatives are stunned and nearly cancel the test, but the call centre agent asks the customer if they want a ham as part of the order
- What did the customer do?

Case Study: the 5lb ham ...

- Customer says “sure, that would be great!”
- Software developers get the contract with the catalog retailer
- But no one understands why that particular recommendation was produced …
- Users have to trust the system

Summary

- Learning is hard – at least as hard as other AI problems
- Knowledge free learning works for simple problems but fails in harder cases
- We can make learning more tractable by incorporating knowledge, e.g., EBL and ILP
- However this requires both knowledge representation and reasoning, and assumes that we can do, e.g., the reasoning necessary to find an explanation in EBL