RS228/406

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8

BSc (Hons) in Computer Science

Stage 4

SUPPLEMENTAL EXAMINATIONS 2009

*** *SOLUTIONS* ***

ARTIFICIAL INTELLIGENCE 2

Dr. John Kelleher
Prof. B. O'Shea
Dr. I. Arana

Duration: 2 Hours

Answer Question 1 (40 marks) **and**

any 2 Other Questions (30 marks each).

*** *SOLUTIONS* ***

*** *SOLUTIONS* ***

1. (a) In the context of machine learning, explain what is meant by **overfitting** the training data.

(5 marks)

Overfitting occurs when classifiers make decisions based on accidental properties of the training set that will lead to errors on the test set (or new data). As a result, whenever there is a large set of possible hypotheses, one has to be careful not to use the resulting freedom to find meaningless "regularity" in the data.

(b) In the context of inductive learning explain what is meant by a **consistent hypothesis**.

(5 marks)

A hypothesis is consistent if it agrees with the true function on all examples that we have.

(c) Distinguish between **classification learning** and **regression learning**.

(5 marks)

- learning a discrete-valued function is called classification learning
- learning a continuous function is called regression.

(d) Briefly describe how a Decision Tree works.

(5 marks)

A decision tree takes as input an object or situation described by a set of attributes and returns a decision the predicted out value for the input. A decision tree reaches its decision by performing a sequence of tests. Each internal node in the tree corresponds to a test of the value of one of the properties, and the branches from the node are labeled with the possible values of the test. Each leaf node in the tree specifies the value to be returned if that leaf is reached.

(e) In the context of inductive logic learning, what is meant by the **extension** of a hypothesis?

(5 marks)

Each hypothesis predicts that a certain set of examples, namely those that satisfy the hypothesis definition, are examples that satisfy the goal predicate. This set of examples is called the extension of the hypothesis. For example, assuming a standard interpretation, the extension of the predicate $digit(X)$ is $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$

(f) Distinguish between the **generalisation** and **specialization** of a logical predicate.

(5 marks)

The generalisation of a logical predicate results in the broadening of the extension of the predicate. Some of the operations that can be used to generalise a logical predicate include: converting conjunctions in the predicate to disjunctions or dropping conditions in the predicate. You would generalise a predicate in response to recognising a false negative. The specialisation of a logical predicate results in the narrowing of the extension of the predicate. Some of the operations that can be used to generalise a logical predicate include: converting disjunctions in the predicate to conjunctions or adding conditions in the predicate. You would specialise a predicate in response to recognising a false positive.

(g) In the context of machine learning distinguish between **false negatives** and **false positives**.

(5 marks)

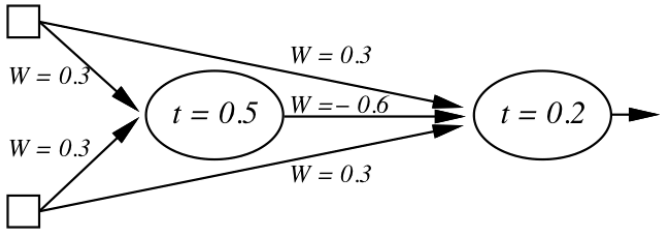
False negative an example can be a false negative for the hypothesis, if the hypothesis says it should be negative but in fact it is positive.

False positive an example can be a false positive for the hypothesis, if the hypothesis says it should be positive but in fact it is negative.

(h) Construct by hand a neural network that computes the XOR function of two inputs. Make sure to specify what sort of units you are using.

(5 marks)

XOR (in fact any Boolean function) is easiest to construct using step-function units. Because XOR is not linearly separable, we will need a hidden layer. It turns out that just one hidden node suffices. To design the network, we can think of the XOR function as OR with the AND case (both inputs on) ruled out. Thus the hidden layer computes AND, while the output layer computes OR but weights the output of the hidden node negatively. The image below illustrates the network. Note the t values in the image define the thresholds of the units.



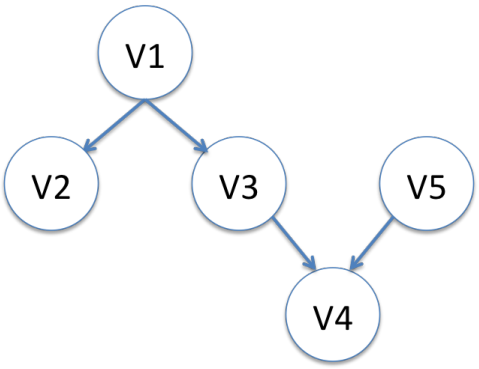


Figure 1: An example Bayesian network.

2. (a) Given that $P(a|b) = 0.5$, $P(a) = 0.3$, $P(b) = 0.4$ calculate $P(b|a)$.

(5 marks)

$P(b|a) = \frac{P(a|b) \times P(b)}{P(a)} = \frac{0.5 \times 0.4}{0.3} = 0.67$

(b) Express the joint probability distribution for the network in Figure 1 using the chain rule.

(10 marks)

The chain rule is: $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1)$. In a properly constructed bayesian network: $parents(X_i) \subseteq \{x_{i-1}, \ldots, x_1\}$. Therefore, in a properly constructed Bayesian network: $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_{i-1}, \ldots, x_1) = \prod_{i=1}^{n} P(x_i|parents(X_i))$. So the joint probability distribution for the given network is: $P(V4|V3, V5)P(V3|V1)P(V2|V1)P(V5)P(V1)$

(c) (i) In the context of inference in probabilistic temporal models define, in your own words. what is meant by the term **filtering**.

(5 marks)

Informally: filtering is the task of computing the belief state – the posterior distribution over the current state, given all the evidence to date.

(ii) Provide the equation that defines a recursive formulation of filter estimation and explain its components.

(10 marks)

Formally: filtering involves computing the probability distribution $\mathbf{P}(X_t|e_{1:t})$ where that $\mathbf{X}_t$ denotes the world state at time slice t and $\mathbf{e}_{1:t}$ denotes the stream of evidence arriving from our sensors beginning at time $t = 1$. This probability distribution can be recursively computed using:

$$\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{e}_{1:t+1}) =$$
$$\alpha \underbrace{\mathbf{P}(\mathbf{e}_{t+1}|\mathbf{X}_{t+1})}_{Sensor\ Model.} \sum_{x_t} \underbrace{\mathbf{P}(\mathbf{X}_{t+1}|x_t)}_{Transition\ Model.} \underbrace{P(x_t, |\mathbf{e}_{1:t})}_{Current\ State\ Distribution.}$$

- The **current state distribution** is the probability distribution that we believe over timeslice x_t given the evidence from timeslice 1 to t - $\mathbf{e}_{1:t}$.
- The **transition model** describes how the world state evolves over time.
- The **sensor model** describes how the "sensors" – that is evidence variables – are affected by the actual state of the world.

3. (a) Explain what is meant by **inductive learning**.

(5 marks)

Inductive Learning involves the process of learning by example where a system tries to induce a general rule from a set of observed instances

(b) Suppose we generate a training set from a decision tree and then apply decision-tree learning to the training set. Is it the case that the learning algorithm will eventually return the correct tree as the training set size goes to infinity? Why or why not?

(5 marks)

The algorithm may not return the"correct" tree, but it will return a tree that is logically equivalent, assuming that the method for generating examples eventually generates all possible combinations of input attributes. This is true because any two decision tree defined on the same set of attributes that agree on all possible examples are, by denition, logically equivalent. The actual form of the tree may differ because there are many different ways to represent the same function. (For example, with two attributes A and B we can have one tree with A at the root and another with B at the root.) The root attribute of the original tree may not in fact be the one that will be chosen by the information gain heuristic when applied to the training examples.

(c) The following sets express the mappings between predicates r, p, q, s, $class1$ and $class2$: $r \rightarrow \{a1, a2, a5, a6\}$, $p \rightarrow \{a2, a3, a5, a7\}$, $q \rightarrow \{a1, a2, a6\}$, $s \rightarrow \{(a2, f), (a1, 1), (a6, f)\}$, $class1 \rightarrow \{a2\}$, $class2 \rightarrow \{a2, a6\}$.

(i) Given the above sets give a specialisation of the rule $class1(X) \leftarrow r(X) \wedge p(X)$ such that the rule is only satisfied by $class1$ members.

(5 marks)

$class1(X) \leftarrow r(X) \wedge p(X) \wedge q(X)$

(ii) Given the above sets give a rule that will correctly classify only members of $class2$.

(5 marks)

$class2(X) \leftarrow s(X, f)$

(d) The FOIL inductive logic programming algorithm is considering adding a literal l to a rule r. The extension of the rule r before adding l is the following set of positive and negative examples $\{+, +, +, +, +, +, -, -, -, -\}$. The extension of the rule after the literal is added (i.e. the extension of $r + l$) is $\{+, +, +, +, +, +, -, -\}$. What is the information gain of adding the literal l to the rule r? (Note you do not need to calculate the logs in your answer ,i.e. you can express your answer as an equation containing logs).

(10 marks)

$$Foil\_Gain(L,R) \equiv t\left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0}\right)$$

Where

- L is the candidate literal to add to rule R
- p_0 = number of positive bindings of R
- n_0 = number of negative bindings of R
- p_1 = number of positive bindings of $R+L$
- n_1 = number of negative bindings of $R+L$
- t is the number of positive bindings of R also covered by $R+L$

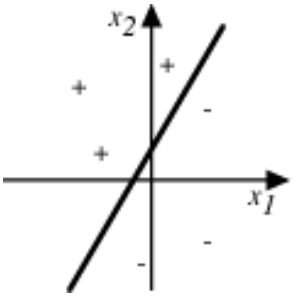
In this instance

- p_0 = 6
- n_0 = 4
- p_1 = 6
- n_1 = 2
- t = 6

$$Foil\_Gain(L,R) \equiv 6\left(\log_2 \frac{6}{6+2} - \log_2 \frac{6}{6+4}\right)$$

4. (a) Explain why linear threshold perceptrons can represent the AND and OR functions but not the XOR function.

(10 marks)



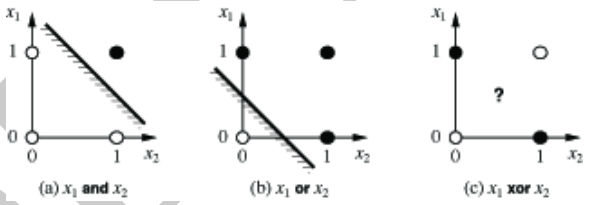
We can think of a 2 input threshold perceptron as representing a line separator in 2D input space.In other words, the function $\sum_j W_j a_j > 0$ or $\vec{w} \bullet \vec{a} > 0$ defines a line in the input space and the perceptron outputs a 1 for instances lying on one side of the line and -1 for instances lying on the other side of the line.

The figure above illustrates this concept. The equation for the line in this image is $\vec{w} \bullet \vec{a} > 0$. The inputs into the perceptron are x_1 and x_2; i.e. $\vec{a} = \{x_1, x_2\}$.

This concept of a linear perceptron defining a line scales up into higher dimensional input space. In 3D and higher inputs space (i.e. in situations where the linear perceptron has 3 or more inputs), the linear activation function $\vec{w} \bullet \vec{a} > 0$ defines a hyperplane decision surface in the n-dimensional space of inputs.

Data-sets of positive and negative examples that can be separated by a hyperplane are called *linearly separabe*.

Of course, not all data-sets of positive and negative examples are linearly separable. The XOR function is one example of a non-linearly separable function.



The figure above illustrates this by comparing it with the AND and OR function which are linearly separable. In this figure black dots represent a point in the input space where the value of the function is 1, and white dots indicate a point where the value is 0.

As is evident from the images, in the AND and OR inputs spaces it is possible to draw a line that separates the black dots from the white dots. However, in the XOR space no such line exists. *As a result, a threshold perceptron cannot represent the XOR function.*

(b) Describe the perceptron training rule?

(10 marks)

Perceptrons learn by modifying the weights associated with their inputs. Consequently, the learning problem faced by perceptron training is to determine a weight vector that causes the perceptron to produce the correct $+-1$ output for each of the given training examples.

One way to learn an acceptable weight vector is to begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.

Weights are modified at each step according to the *perceptron training rule*, which revises the weight w_i associated with input a_i according to the rule:

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t-o)a_i$$

where t = target output, o = observed output, η is a positive constant called the *learning rate*. The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g.0.1) and is sometimes made to decay as the number of weight-training iterations increases.

(c) Why does the perceptron learning rule converge toward successful weight values?

(10 marks)

If the training example is *correctly classified* $(t-o)=0 \rightarrow \Delta w_i = 0$ so *no weights are updated*.

If the case of a *false negative* (o=0 and t=1) we want to make the perceptron output a 1 instead of a 0 so the weights must be altered to increase the value of $\vec{w} \bullet \vec{a}$. Notice that in this case *the rule will increase* w_i because $(t-o), \eta$ and a_i are all positive.

On the other hand, in the case of a *false positive* (o=1 and t=0) then *the weights associated with* a_i *will be decreased*.