S228/406

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8

BSc (Hons) in Computer Science

Stage 4

SEMESTER 2 EXAMINATIONS 2008

*** *SOLUTIONS* ***

ARTIFICIAL INTELLIGENCE 2

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

Duration: 2 Hours

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

any 2 Other Questions (30 marks each).

*** *SOLUTIONS* ***

*** *SOLUTIONS* ***

1. (a) Distinguish between **supervised** and **unsupervised** learning.

(5 marks)

The distinction is that with **supervised learning** we know the actual label or category for each piece of data on which we train, whereas with **unsupervised learning** we do not know the classification of the data in the training sample. Unsupervised learning can thus often be viewed as a **clustering** task, while supervised learning can usually be seen as a **classification** task, or equivalently as a function-fitting task where one extrapolates the shape of a function based on some data points.

(b) 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.

(c) Explain what is meant by the term **abductive reasoning**.

(5 marks)

Abductive reasoning allows the antecedent (head) of a rule to concluded with the conclusion is true provided that doing so is consistent. Abductive reasoning is primarily diagnostic; given the effect find the likely cause.

(d) What does it mean if two classes C_1 and C_2 are described as **linearly separable**.

(5 marks)

This means that for each class C_i there exists a hyperplane H_i such that on its positive side lie all $x \in C_i$ and on its negative side lie all $x \in C_j, j \neq i$

(e) Let us say we have three classification algorithms. How can we order these three from best to worst?

(20 marks)

This is a discursive question so giving a precise answer is not appropriate. However, key points that the student should touch on include:

- Predictive accuracy
- Speed and scalability
  - Time to construct the model
  - Time to use the model
- Robustness (handling noise and missing values)
- Scalability
- Interpretability (understanding and insight provided by the model)

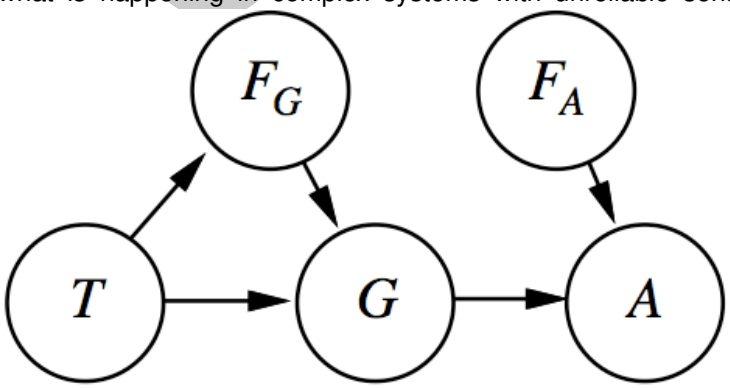
it should be noted also, that these evaluation criteria are application dependent.

2. (a) In your local power station, there is an alarm that senses when a temperature gauge exceeds a given threshold. The gauge measures the temperature of the core. Consider the Boolean variables A (alarm sounds), F_A (alarm is faulty), and F_G (gauge is faulty) and multivalued nodes G (gauge reading) and T (actual core temperature).

(i) Draw a Bayesian network for this domain, given that the gauge is more likely to fail when the core temperature gets too high.

(5 marks)

The key aspects are: the failure nodes are parents of the sensor nodes, and the temperature node is a par ent of both the gauge and the gauge failure node. It is exactly this kind of correlation that makes it difcult for humans to understand what is happening in complex systems with unreliable sensors.



(ii) Suppose there are just two possible actual and measured temperatures: normal and high. The probability that the gauge gives the correct temperature is x when it is working, but y when it is faulty. Give the conditional probability

table associated with node G.

(5 marks)

Note the semantics of F_G, which is true when the gauge is faulty, i.e., not working.

	$T = Normal$		$T = High$	
	F_G	$\neg F_G$	F_G	$\neg F_G$
$G = Normal$	y	x	$1-y$	$1-x$
$G = High$	$1-y$	$1-x$	y	x

(b) After your yearly checkup, the doctor has bad news and good news. The bad news is that you tested positive for a serious disease and that the test is 99% accurate (i.e., the probability of testing positive when you do have the disease is 0.99, as is the probability of testing negative when you don't have the disease). The good news is that this is a rare disease, striking only 1 in 10,000 people of your age.

(i) Why is it good news that the disease is rare?

(10 marks)

What the patient is concerned about is $P(disease|test)$. Now $P(a|b) = \frac{P(b|a)P(a)}{P(a)} \rightarrow P(disease|test) = \frac{P(test|disease)P(disease)}{P(test)} \rightarrow$ roughly speaking, the reason it is a good thing that the disease is rare is that $P(disease|test)$ is proportional to $P(disease)$, so a lower prior for disease will mean a lower value for $P(disease|test)$. Roughly speaking, if 10,000 people take the test, we expect 1 to actually have the disease, and most likely test positive, while the rest do not have the disease, but 1 percent of them (about 100 people) will test positive anyway, so $P(disease|test)$ will be about 1 in 100. The moral is that when the disease is much rarer than the test accuracy, a positive test result does not mean the disease is likely. A false positive reading remains much more likely.

(ii) What are the chances that you actually have the disease?

(10 marks)

We are given the following information: $P(test|disease) = 0.99$
$P(\neg test|\neg disease) = 0.99$
$P(disease) = 0.0001$
and the observation $test$.
$P(a|b) = \frac{P(b|a)P(a)}{P(a)}$
$P(disease|test) = \frac{P(test|disease)P(disease)}{P(test)} \rightarrow$
$P(disease|test) = \frac{P(test|disease)P(disease)}{P(test|disease)P(disease)+P(test|\neg disease)P(\neg disease)} \rightarrow$
$\frac{0.99 \times 0.0001}{(0.99 \times 0.0001)+(0.1 \times 0.9999)}$
$P(disease|test) = .009804$

3. (a) In the context of Decision Tree Learning define what is meant by the following terms:

   (i) entropy

   (5 marks)

   For c classification categories the entropy E is defined as: $E = \sum_{i=1}^{c} -p_i \, log_2 \, p_i$ where p_i is the probability of category i occurring.

   (ii) information gain

   (5 marks)

   The information gain for an attribute is the expected reduction in entropy if the examples were to be partitioned according to that attribute and is defined as: $Gain(T, A) = E(T) - \sum_{j=1}^{v} \frac{|T_j|}{|T|} E(T_j)$ where T is a set of training examples and T_j is a subset of examples having value j for attribute A

   (b) Using the ID3 algorithm we never test the same attribute twice along one path in a decision tree. Why not?

   (5 marks)

   In standard decision trees, attribute tests divide examples according to the attribute value. Therefore any example reaching the second test already has a known value for the attribute and the second test is redundant. In some decision tree systems, however, all tests are Boolean even if the attributes are multivalued or continuous. In this case, additional tests of the attribute can be used to check different values or subdivide the range further (e.g., rst check if $X > 0$, and then if it is, check if $x > 10$).

   (c) 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.

   (d) Discuss the advantages and disadvantages of k-**Nearest Neighbour** classification.

   (10 marks)

Strengths

(i) No training involved lazy learning

(ii) New data can be added on the fly

(iii) Some explanation capabilities

(iv) Robust to noisy data by averaging k-nearest neighbors

Weaknesses

(i) Not the most powerful classification (generally its accuracy will be lower than an ANN or SVM model)

(ii) Slow classification

(iii) Curse of dimensionality (as you increase the number of features you need more and more examples to cover the problem space - kNN are particularly susceptible to this issue as they do not do any feature selection).

4. Figure 1 shows a backprogation network that is currently processing the training vector $[1.0, 0.9, 0.9]$ which has an associated target vector $[0.1, 0.9, 1.0]$. Given that the output from unit B is 0.6 and from C is 0.8, and assuming that the activation function used at all nodes in the network is the logistic function (i.e., $f(x) = \frac{1}{1+\exp^{-x}}$):

   (a) Calculate the actual output vector (to 3 decimal places).

   (5 marks)

Output of unit $i = f(\sum_{j=1}^{n} W_{j,i} \times activation_j)$
First output unit input = -0.3 x 0.6 + 0.9 x 0.8 = 0.54 → f(0.54) = 0.632
Second output unit input = -0.6 x 0.6 + -0.1 x 0.8 = -0.44 → f(-0.44) = 0.392
Third output unit input = 0.4 x 0.6 + 1.2 x 0.8 = 1.2 → f(1.2)= 0.769

   (b) Calculate the error for each output unit.

   (5 marks)

Error = target - output
First output unit = (0.1 - 0.632) = - 0.532
Second output unit = (0.9 - 0.392) = 0.508
Third output unit = (0.1 - 0.769) = - 0.696

   (c) Calculate the error for each hidden unit B and C.

   (10 marks)

Each hidden node j is responsible for some fraction of the error Err_i of each of the output units i to which it connects. Thus the Err_i values are divided according to the strengths of the connection between the hidden node and the output nodes and are propagated back to the hidden nodes. Where a hidden node feeds-forward into more than 1 output node the errors propagated back to it are summed: $Err_j = \sum_{i=1}^{n} W_{ji} \times Err_i$:
$Err_B = (-0.3 \times -0.532) + (-0.6 \times 0.508) + (0.4 \times -0.696) = 0.1596 + -0.3048 + -0.2784 = -0.4236$
$Err_C = (0.9 \times -0.532) + (-0.1 \times 0.508) + (1.2 \times -0.696) = -0.4788 + -0.0508 + -0.8352 = -1.3648$

(d) Calculate the new weight for the connection from unit B to the output unit D after the training example has been processed. Use a learning rate of $\alpha = 0.25$ and momentum of zero.

(10 marks)

The weight update rules is:
$W_{j,i} = W_{j,i} + (\alpha \times activation_j \times Err_i \times \frac{\delta f(e_i)}{\delta e_i})$
where $e_i = \sum_{j=0}^{n} W_{j,i} \times activation_j)$ and $f()$ is the network activation function. When $f(e_i)$ is a sigmoidal function (as in this question) $\frac{\delta f(e_i)}{\delta e_i} = f(e_i) \times (1 - f(e_i))$.
From part 1 $e_D = 0.54$ and $f(e_D) = 0.3682$
$W_{B,D} = W_{B,D} + (\alpha \times activation_B \times Err_D \times f(e_D) \times (1 + f(e_D) \rightarrow$
$W_{B,D} = -0.3 + (0.25 \times 0.6 \times -0.532 \times 0.632 \times (1 + 0.632)) \rightarrow$
$W_{B,D} = -0.3 + (0.25 \times 0.6 \times -0.532 \times 0.632 \times (1.632)) \rightarrow$
$W_{B,D} = -0.3 + (0.25 \times 0.6 \times -0.532 \times 1.031424) \rightarrow$
$W_{B,D} = -0.3 + (0.25 \times 0.6 \times -0.548717568) \rightarrow$
$W_{B,D} = -0.3 + (0.25 \times -0.3292305408) \rightarrow$
$W_{B,D} = -0.3 + (-0.0823076352) \rightarrow$
$W_{B,D} = -0.3823076352 \rightarrow$
$W_{B,D} = -0.3823$ rounded

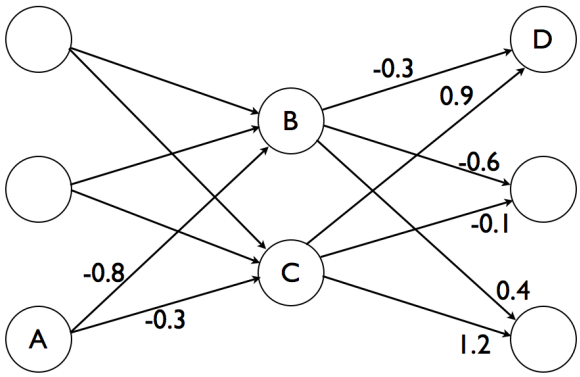


Figure 1: Example Neural Net