R249/419C

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8

BSc. (Hons) in Computer Science

Stage 4

SUPPLEMENTAL EXAMINATIONS 2013

*** *SOLUTIONS* ***

ARTIFICIAL INTELLIGENCE II

Dr. John Kelleher
Dr. Deirdre. Lillis
Mr. R. Walshe

Duration: 2 Hours

Question 1 is **compulsory**

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

any 2 Other Questions (30 marks each).

*** *SOLUTIONS* ***

SOLUTIONS

*** *SOLUTIONS* ***

SOLUTIONS

1. (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) Describe the differences between **lazy learners** and **eager learners**, giving examples of each.

(10 marks)

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

Lazy learners do not try to build a model from the training data, but simply use it at classification time

Eager learners build a mode from the training data during training, and use only this model at classification time, ignoring the original data.

Key differences:

- Lazy methods may consider query instance when deciding how to generalise beyond the training data D; eager methods cannot since they have already chosen global approximation when seeing the query.
- **Efficiency** lazy learners require less training times but more time at prediction; eager learners require more training times by less time for prediction
- **Accuracy** lazy learners effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function; eager learners must commit to a single hypothesis that covers the entire instance space.
- It is easier for lazy learners to handle **concept drift**

Examples:

Lazy learning example : case based reasoning

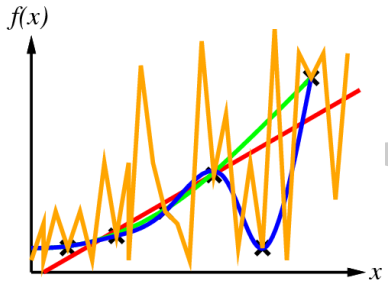
Eager learning example : Decision=tree, neural networks, support vector machines

(c) Inductive machine learning is often referred to as an **ill-posed problem**. What is meant by this?

(10 marks)

SOLUTIONS

Inductive machine learning algorithms essentially search through a hypothesis space to find a the best hypothesis that is consistent with the training data used. It is possible to find multiple hypotheses that are consistent with a given training set (i.e. agrees with all training examples). It is for this reason that inductive machine learning is referred to as an ill-posed problem as there is typically not enough information in the training data used to build a model to choose a single best hypothesis. Inductive machine learning algorithms must somehow choose one of the available hypotheses as the *best*. An example like that shown in the figure below would be useful at this point



(d) Why is it difficult to select the correct inductive bias for a machine learning algorithm?

(15 marks)

SOLUTIONS

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

- Choosing the correct inductive bias for a particular inductive learning problem is difficult because it involves getting the balance right in the **tradeoff** between the expressiveness of a hypothesis space and the complexity of finding a simple, consistent hypothesis within that space. In other words, there is a tradeoff between complex hypotheses that fit the training data well and simpler hypotheses that may **generalise** better (How well a model trained on a training set predicts the right output for new instances is called **generalization**).
- If the hypothesis class ˝ is less complex than the true function we are trying to model, we have **underfitting**. In other words the inductive bias is to restrictive on the search and the learning algorithm is not able to generate hypotheses that are complex enough. In the case of underfitting, as we increase the complexity allowed by the inductive bias, (i.e., increase the complexity of ˝) the training error decreases.
- If our model selection has chosen a ˝ that is too complex, the data is not enough to constrain it and we may end up with a bad hypothesis. 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. This problem is called **overfitting**. Overfitting becomes more likely as the hypothesis space and the number of input attributes grows, and less likely as we increase the number of training examples.

SOLUTIONS

Table 1: A dataset showing the behaviour of two individuals in an online shop. A 1 indicates that the person bought the item a 0 indicates that they did not.

Person ID	Item 107	Item 498	Item 7256	Item 28063	Item 75328
A	1	1	1	0	0
B	1	0	0	1	1

Table 2: A query instance from the same domain as the examples listed in Table 1. A 1 indicates that the person bought the item a 0 indicates that they did not.

Person ID	Item 107	Item 498	Item 7256	Item 28063	Item 75328
Q	1	0	1	0	0

2. (a) You have been given the job of building a recommender system for an large online shop that has a stock of over 100,000 items. In this domain the behaviour of individuals is captured in terms of what items they have bought or not bought. Table 1 lists the behaviour of two individuals in this domain for a subset of the items that at least one of the individuals has bought. Table 2 lists the behaviour of a customer that you want to generate recommendations for. Table 3 lists 3 different models of similarity that work on binary data, similar to the data in this domain (**Russell-Rao**, **Sokal-Michener**, and **Jaccard**).

$$\begin{aligned} \text{Russell-Rao(X,Y)} &= \frac{CP(X,Y)}{P} \\ \text{Sokal-Michener(X,Y)} &= \frac{CP(X,Y)+CA(X,Y)}{P} \\ \text{Jaccard(X,Y)} &= \frac{CP(X,Y)}{CP(X,Y)+PA(X,Y)+AP(X,Y)} \end{aligned}$$

Table 3: Similarity Metrics for Binary Data.

(i) Given that there are over 100,000 items available in the store which of these models of similarity (**Russell-Rao**, **Sokal-Michener**, or **Jaccard**) is most appropriate for this domain. Give an explanation for your choice.

(5 marks)

In a domain where there are 100,000's of items co-absenses aren't that meaningful. For example, you may be in a domain where there are so many items most people haven't seen, listened to, bought or visited the vast majority of them and as a result the majority of features will be co-absenses. The technical term to describe dataset where most of the features have zero values is **sparse data**. In these situations you should use a metric that ignore co-absenses and if your features are binary then you should use the **Jaccard similarity** index.

(ii) Assuming that the recommender system uses the similarity metric you selected in Part (i) and that the system will recommend to person Q the items that the person most similar to person Q has already bought but that person Q has not bought, **which item or items will the system recommend to person Q?** Support you answer by showing your calculations and explaining your analysis of the results.

SOLUTIONS

(10 marks)

Using a similarity metric the higher the value returned by the metric the more similar the two items are.
Assuming the student chose the **Jaccard** similarity metric then Person A is more similar to Q than Person B: $Jaccard(Q, A) = \frac{2}{2+1} = 0.6667$, $Jaccard(Q, B) = \frac{1}{4} = 0.25$. As a result the system will recommend item **498**.
If the student selected one of the other similarity metrics for part (a), the supporting calculations should be:

- Russell-Rao(Q,A)= $\frac{2}{5} = 0.4$
- Russell-Rao(Q,B)= $\frac{1}{5} = 0.2$
- Sokal-Michener(Q,A)= $\frac{4}{5} = 0.8$
- Sokal-Michener(Q,B)= $\frac{2}{5} = 0.4$

As is evident from these calculations regardless of which similarity metric is used Person A is more similar to Q than Person B. So the system will recommend item **498** regardless of which similarity metric is used.

(b) Table 4, on the next page lists a data set with of 6 examples described in terms of 3 binary descriptive features (**A**, **B**, and **C**) and a target label (**Target**). You are asked to create a decision tree model using this data. **Which of the descriptive features will the ID3 decision tree induction algorithm choose as the feature for the root node of the decision tree?** Support you anwer with appropriate calculations and dicussions of your results. Note that Table 5, also on the next page, lists some equations that you may find useful for this question.

(15 marks)

SOLUTIONS

The ID3 decision tree induction algorithm selects the decriptive feature with the highest information gain as the feature for the root node of the decision tree. The first step in calculating information gain is to calculate the entropy for the entire dataset:

$$\begin{aligned} H(DS) &= -\sum_{v \in \{C1, C2\}} p_v \, log_2 p_v \\ &= -\left(\frac{3}{6} \, log_2 \frac{3}{6}\right) + -\left(\frac{3}{6} \, log_2 \frac{3}{6}\right) \\ &= 1.00 \; bits \end{aligned}$$

The table below shows the calculation of the infromation gain for each of the descriptive features in the dataset:

Split By	Feature Value	Partition	Examples	Entropy of Partition	Remainder	Info. Gain
A	1	DS_1	1,2,3	0.9183	0.9183	0.0817
	0	DS_2	4,5,6	0.9183		
B	1	DS_3	2,4,5,6	0.8113	0.5409	0.4591
	0	DS_4	1,3	0		
C	1	DS_5	1,2,3,4,6	0.9709	0.8091	0.1909
	0	DS_6	5	0		

From this table we can see the feature **B** has the highest information gain and consequently the ID3 algorithm will chose this feature as the feature tested at the root node of the tree.

ID	A	B	C	Target
1	1	0	1	C1
2	1	1	1	C2
3	1	0	1	C1
4	0	1	1	C2
5	0	1	0	C1
6	0	1	1	C2

Table 4: Dataset for the ID3 Algorithm Question

$$\begin{aligned} \text{Entropy(DS)} &= -\textstyle\sum_{i=1}^{k} p_i \times log_2(p_i) \\ \text{Remainder(F)} &= \textstyle\sum_{v \in Domain(F)} \frac{|DS_v|}{|DS|} Entropy(DS_v) \\ \text{InformationGain(F,DS)} &= Entropy(DS) - Remainder(F) \end{aligned}$$

Table 5: Equations from information theory.

SOLUTIONS

Table 6: Spam and Ham Dataset

Spam	**Ham**
Offer is Free	*Great Learning Fun*
Free Learning Link	*Great Machine Learning*
Click Free Link	*Free Learning Event*
	Learning is Fun
	Learning Costs Money

Table 7: Query Title

Fun is Free

3. Table 6 lists a dataset of email subjects. Table 7 lists the title of a query instance that we would like to classify as being either a spam or ham email based on its title.

   (a) Using **Laplacian smoothing**, where

$$p(x = v) \quad = \quad \frac{count(x = v) + k}{count(x) + (k \times |Domain(x)|)}$$

   with **k=1** and a **vocabulary size of 12** calculate the following probabilities:

   (i) $P(Spam) = ?$

   (2 marks)

   $P(Spam) = \frac{3+1}{8+(1\times2)} = \frac{4}{10} = 0.4$

   (ii) $P(Ham) = ?$

   (2 marks)

   $P(Ham) = \frac{5+1}{8+(1\times2)} = \frac{6}{10} = 0.6$

   (iii) $P('Fun'|Spam) = ?$

   (2 marks)

   $P('Fun'|Spam) = \frac{0+1}{9+(1\times12)} = \frac{1}{21} = 0.0476$

   (iv) $P('Fun'|Ham) = ?$

   (2 marks)

   $P('Fun'|Ham) = \frac{2+1}{15+(1\times12)} = \frac{3}{27} = 0.1111$

   (v) $P('is'|Spam) = ?$

   (2 marks)

   $P('is'|Spam) = \frac{1+1}{9+(1\times12)} = \frac{2}{21} = 0.0952$

   (vi) $P('is'|Ham) = ?$

   (2 marks)

   $P('is'|Ham) = \frac{1+1}{15+(1\times12)} = \frac{2}{27} = 0.0741$

SOLUTIONS

(vii) $P('Free'|Spam) =?$

(2 marks)

$P('Free'|Spam) = \frac{3+1}{9+(1\times12)} = \frac{4}{21} = 0.1905$

(viii) $P('Free'|Ham) =?$

(2 marks)

$P('Free'|Ham) = \frac{1+1}{15+(1\times12)} = \frac{2}{27} = 0.0741$

(b) Calculate the probability of the query title in Table 7 belonging to the Spam class under the **Naive Bayes assumption** and using the **smoothed probabilities** you calculated in Part (a):

$$P(Spam|'Fun\ is\ Free') =?$$

(8 marks)

$P(Spam|'FunisFree')$

$$= \frac{P('Fun'|Spam)P('is'|Spam)P('Free'|Spam)P(Spam)}{(P('Fun'|Spam)P('is'|Spam)P('Free'|Spam)P(Spam)) + (P('Fun'|Ham)P('is'|Ham)P('Free'|Ham)P(Ham))}$$

$$= \frac{0.0476 \times 0.0952 \times 0.1905 \times 0.4}{(0.0476 \times 0.0952 \times 0.1905 \times 0.4) + (0.1111 \times 0.0741 \times 0.0741 \times 0.6)}$$

$$= 0.48543863$$

(c) Calculate the probability of the query title in Table 7 belonging to the Spam class under the **Naive Bayes assumption** and using **maximum likelihood** probabilities (i.e. the probabilities we would get if we did not use Laplacian smoothing):

$$P(Spam|'Fun\ is\ Free') =?$$

(6 marks)

Because the word 'Fun' does not appear in any of the Spam titles the maximum likelihood (i.e., unsmoothed) probability of $P('Fun'|Spam) = 0$. As a result the maximum likelihood probability of $P(Spam|'Fun\ is\ Free') = 0$. Showing the complete calculation:

P(Spam—'Fun is Free')

$$= \frac{P('Fun'|Spam)P('is'|Spam)P('Free'|Spam)P(Spam)}{(P('Fun'|Spam)P('is'|Spam)P('Free'|Spam)P(Spam)) + (P('Fun'|Ham)P('is'|Ham)P('Free'|Ham)P(Ham))}$$

$$= \frac{\frac{0}{9} \times \frac{1}{9} \times \frac{3}{9} \times \frac{3}{8}}{(\frac{0}{9} \times \frac{1}{9} \times \frac{3}{9} \times \frac{3}{8}) + (\frac{2}{15} \times \frac{1}{15} \times \frac{1}{15} \times \frac{5}{8})}$$

$$= 0.0$$

SOLUTIONS

4. (a) The following model is commonly used for continuous prediction tasks:

$$y(x) = w_0 + w_1 x_1 + \ldots + w_D x_D$$

(i) Provide the name for this model and explain all its terms.

(3 marks)

Students should explain that this is a simple linear regression model which can be effectively used to make predictions. x is a vector of feature values for a query instance and w is a vector of feature weights. An diagram of a simple one dimensional linear function would help.

(ii) Briefly describe a technique for finding optimal values for the terms $w_0, w_1, \ldots, w_D$ in the model based on a historical training set.

(3 marks)

Students should explain that given a training set the performance of a particular linear regression model can be measured using an appropriate evaluations function, e.g. the *sum of squares error* as follows:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n) - t_n)^2$$

where t_n is the *true* answer for training instance x_n and $y(x_n)$ is the prediction made by the model for instance x_n. Optimal values for w are found by minimising E(w).

(b) Figure 1, on the next pages, shows a backprogation network that is currently processing the training vector $[1.0, 0.9, 0.9]$ which has an associated target vector $[0.1, 0.9, 0.1]$. 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, carry out the calculations listed below. Note that Table 8, also on the next page, lists some equations that you may find useful when doing this question.

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

(10 marks)

SOLUTIONS

$$
\begin{aligned}
a_i(in_i) &= \frac{1}{1+\exp^{-((W_{BD}\times a_B(in_B)+(W_{CD}\times a_C(in_C)))}} \\
&= \frac{1}{1+\exp^{-((-0.3\times 0.6)+(0.9\times 0.8))}} \\
&= \frac{1}{1+\exp^{-0.54}} \\
&= 0.632
\end{aligned}
$$

$$
\begin{aligned}
a_j(in_j) &= \frac{1}{1+\exp^{-((W_{BE}\times a_B(in_B)+(W_{CE}\times a_C(in_C)))}} \\
&= \frac{1}{1+\exp^{-((-0.6\times 0.6)+(0.1\times 0.8))}} \\
&= \frac{1}{1+\exp^{-(-0.44)}} \\
&= 0.392
\end{aligned}
$$

$$
\begin{aligned}
a_k(in_k) &= \frac{1}{1+\exp^{-((W_{BF}\times a_B(in_B)+(W_{CF}\times a_C(in_C)))}} \\
&= \frac{1}{1+\exp^{-((0.4\times 0.6)+(1.2\times 0.8))}} \\
&= \frac{1}{1+\exp^{-1.2}} \\
&= 0.769
\end{aligned}
$$

(ii) Calculate the Δ error for each output unit (to 3 decimal places).

(6 marks)

$$
\begin{aligned}
\Delta_D &= (target_D - a_D(in_D)) \times a_D(in_D) \times (1 - a_D(in_D)) \\
&= (0.1 - 0.632) \times 0.632 \times (1 - 0.632) \\
&= -0.124
\end{aligned}
$$

$$
\begin{aligned}
\Delta_E &= (target_E - a_E(in_E)) \times a_E(in_E) \times (1 - a_E(in_E)) \\
&= (0.9 - 0.392) \times 0.392 \times (1 - 0.392) \\
&= 0.121
\end{aligned}
$$

$$
\begin{aligned}
\Delta_F &= (target_F - a_F(in_F)) \times a_F(in_F) \times (1 - a_F(in_F)) \\
&= (0.1 - 0.769) \times 0.769 \times (1 - 0.769) \\
&= -0.119
\end{aligned}
$$

(iii) Calculate the Δ error for each hidden unit B and C. (to 3 decimal places)

(8 marks)

SOLUTIONS

$$
\begin{aligned}
\Delta_B &= a_B(in_B) \times (1 - a_B(in_B)) \times \left(\sum_{x \in \{D,E,F\}}^{n} W_{Bx} \times \Delta_x \right) \\
&= a_B(in_B) \times (1 - a_B(in_B)) \times (((W_{BD} \times \Delta_D) + (W_{BE} \times \Delta_E) + (W_{BF} \times \Delta_F))) \\
&= 0.6 \times (1 - 0.6) \times (((-0.3 \times -0.124) + (-0.6 \times 0.121) + (0.4 \times -0.119))) \\
&\approx -0.020
\end{aligned}
$$

$$
\begin{aligned}
\Delta_C &= a_C(in_C) \times (1 - a_C(in_C)) \times \left(\sum_{x \in \{D,E,F\}}^{n} W_{Cx} \times \Delta_x \right) \\
&= a_C(in_C) \times (1 - a_C(in_C)) \times (((W_{CD} \times \Delta_D) + (W_{CE} \times \Delta_E) + (W_{CF} \times \Delta_F))) \\
&= 0.8 \times (1 - 0.8) \times (((0.9 \times -0.124) + (-0.1 \times 0.121) + (1.2 \times -0.119))) \\
&\approx -0.043
\end{aligned}
$$

SOLUTIONS

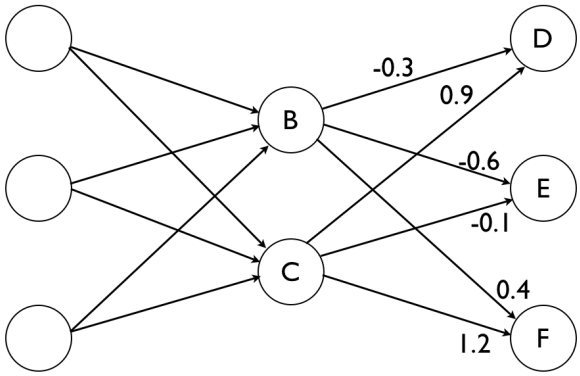


Figure 1: Example Neural Net

Weighted sum of inputs for unit i with j inputs:	$in_i = \sum_j W_{ji} a_j(in_j)$
Activation Function (Logistic) for unit i:	$a_i(in_i) = \frac{1}{1+\exp^{-in_i}}$
Perceptron weight update rule for link $j \rightarrow i$	$w_{ji} = w_{ji} + \eta\,(t_i - a_i(in_i)) \times a_j(in_j)$
Hebbian Weight Update Rule for link $j \rightarrow i$	$w_{ji} = \eta \times a_j(in_j) \times a_i(in_i)$
Partial Derivative for Logistic Activation Function	$\frac{\delta a_i(in_i)}{\delta in_i} = a_i(in_i) \times (1 - a_i(in_i))$
Error for an output unit i	$error_i = target_i - a_i(in_i)$
Delta Error for an output unit i	$\Delta_i = error_i \times a_i(in_i) \times (1 - a_i(in_i))$
Delta Error for a hidden unit j feeding into n units	$\Delta_j = (\sum_{i=1}^{n} W_{ji} \times \Delta_i) \times a_j(in_j) \times (1 - a_j(in_j))$
Delta Weight Update Rule for link $x \rightarrow k$	$W_{x,k} = W_{x,k} + (\eta \times a_x(in_x) \times \Delta_k)$

Table 8: Equations used in Perceptron and Neural Network training.

SOLUTIONS