Student ID: **Solution Outline** _____

First Name: _____

Last Name: _____

University of Auckland

FINANCE 781 – Financial Machine Learning

Final Test

Semester 1, 2019

→ DO NOT TURN OVER THIS PAGE UNTIL INSTRUCTED TO DO SO ←

Time: 2 hrs

Total marks: 60 (so 2 minutes per mark)

- ***PLEASE WRITE YOUR NAME AND STUDENT ID AT THE TOP OF THIS PAGE***
- **Complete all the questions in Section A and Section B**
- **Show your workings and note any assumptions you make**
- **This test is OPEN BOOK**
- **Calculators allowed**
- **An additional page for workings is available at the back**
- **GOOD LUCK!**

→ DO NOT TURN THIS PAGE UNTIL INSTRUCTED TO DO SO ←

Section A: Multiple Choice + Explanation (Attempt all)

30 marks (10 questions, 3 marks per question) – Circle the correct answer and motivate your answer in the space provided by showing your workings or explaining your reasoning.

Question 1) Which of the following is not a common way to distinguish between different machine learning algorithms?

A. Batch vs Online
B. Classification vs Regression
C. Instance-based vs Model-based
D. NP complete vs NP incomplete
E. Supervised vs Unsupervised

D: NP completeness is a concept from theoretical computer science which is not commonly used to categorise machine learning approaches.

Question 2) You are training a model using gradient descent and early stopping (when the validation error starts going up). However your algorithm does not ever stop early, it just keeps going until you stop it manually after a few hours… Which of the following could account for what you observe?

A. You are doing your calculations in QBASIC on a really slow IBM PC from 1985
B. Your validation set is a sub-set of your training set
C. You calculated `valid_error = np.sqrt(mean_squared_error(y_train, y_train_pred)`
D. Line 781 of your code contains a typo: `y_valid = y_train`
E. It could be any of the above

E: all of the above could explain why you are not encountering early stopping.

Question 3) A neural network has an input layer with 5 nodes, followed by a hidden layer with 7 nodes, followed by another hidden layer with 3 nodes, followed by a single output node. Each non-input node is fully connected to all the nodes in the layer just above it (with a weight associated with each connection) and in addition each non-input node has a bias. What is the total number of trainable parameters in this network?

A. 11
B. 59
C. 70
D. 74
E. None of the above

C. (5*7+7)+(3*7+3)+(1*3+1) = 70

Remember bias terms are associated with the destination node, not the source node. So the input layer has no biases associated with it, but all the lower layers do, including the output layer.

Question 4) Giu, Kelly and Xiu (2018) ("Empirical Asset Pricing via Machine Learning") use the Huber loss function instead of the more common Mean Square Error (MSE) loss function for some machine learning approaches. Which of the following are true regarding the Huber loss function?

1. The Huber loss function is increasing in the absolute magnitude of its input.
2. The Huber loss function is always equal to or less than the MSE loss function for the same input
3. The Huber loss function has an associated hyper-parameter, while the MSE loss function does not have one.

A. 1 and 3 only
B. 1, 2 and 3
C. 1 and 2 only
D. 2 and 3 only
E. None of the above

B. 1 and 2 can be shown with a bit of high school math*, 3 is true since the Huber function has a parameter epsilon, which is a hyper parameter. BTW, the magnitude of x = the absolute of x = |x| = abs(x) = x with the negative sign stripped away if it has one.

* You did not have to do formal proofs, but here they are in case you want to see them…

Definition: H(x, e) = if |x|<=e then x^2 else 2e|x|-e^2, with hyper parameter epsilon = e >= 0.

Proof of 1): x^2 is increasing in |x| and 2e|x|-e^2 is increasing in |x| since e is positive. QED.

Proof of 2): Show H(x,e) <= x^2 for any x

First, assume |x|<= e. Then H(x,e) = x^2 <= x^2 trivially.

Second, assume |x|>e. Need to show that:

2e|x|-e^2 <= x^2

If |x|>e then there exists positive k>0 such that |x| = e+k; sub this into the above...

2e(e+k)-e^2 <= (e+k)^2

2e^2+2ek-e^2 <= e^2+2ek+k^2

(e^2+2ek) <= (e^2+2ek)+k^2

Which is clearly true since k^2 > 0 if k>0, which it is. QED.

Question 5) Consider the following dataset containing the occurrence of actual bankruptcies along with predictions of the ex-ante risk of bankruptcy based on data from one year earlier. Assume you use a threshold of **0.7** to convert the predicted bankruptcy risk into a binary classification of whether bankruptcy will actually take place (0 = no bankruptcy, 1 = bankruptcy).

ID	Actual Bankruptcy	Predicted	Predicted Bankruptcy Risk	Actual-Pred
1021	0	0	0.549	0-0 =>TN
1027	1	0	0.443	1-0 =>FN
1033	0	0	0.493	0-0 =>TN
1039	1	0	0.565	1-0 =>FN
1045	0	0	**0.687**	0-0 =>TN
1051	1	1	**0.703**	1-1 =>TP
1057	1	1	0.790	1-1 =>TP
1063	1	1	0.780	1-1 =>TP
1069	0	1	0.927	0-1 =>FP
1075	1	1	0.817	1-1 =>TP

Based on your predicted classification, what would the recall and precision be? (rounded to 2 decimal places):

A. Recall = 0.4, Precision = 0.4
B. Recall = 0.8, Precision = 0.67
C. Recall = 0.67, Precision = 0.8
D. Recall = 0.7, Precision = 0.8
E. Recall = 0.67, Precision = 0.7

C: See extra columns in table above.

So TN= 3/10=0.3, FN = 2/10=0.3, FP = 1/10=0.3, TP = 4/10=0.4.

	Pred=0	*Pred=1*
Actual=0	TN = 0.3	*FP = 0.1*
Actual=1	**FN = 0.2**	***TP = 0.4***

Recall = TP/(FN+TP) = 0.4/(0.2+0.4) = **0.67,** Precision = TP/(FP+TP) = 0.4/(0.1+0.4) = **0.8**

Question 6) Consider the following Python code to predict credit losses for individual borrowers. The matrix X already contains borrower features, while the vector y already contains subsequent credit losses of the borrower, expressed as a decimal percentage.

```
import numpy as mynumpy
from sklearn.model_selection import test_train_split
from sklearn.metric import mean_absolute_error
X_train, X_val, y_train, y_val = train_test_split(X, y)
booster = GradientBoostingRegressor(max_depth=1,n_estimators=1)
booster.fit(X_train, y_train)
y_val_predicted = booster.predict(X_train)
mae_val = mean_absolute_error(y_val, y_val_predicted)
```

Consider the following statements:

1) "`import numpy as mynumpy`" will raise an error.
2) Variable "`mae_val`" will contain the mean absolute validation error.
3) The implemented model can handle feature interactions.
4) "`booster`" needs to be replaced with "`gbrt`" for the code to work.

Select the correct statement(s):

A. Only 1 and 4 is true
B. Only 2 and 3 is true
C. Only 2 is true
D. All the statements are true
E. None of the statements are true

E: 1) will not raise an error, it is simply unusual. 2) won't contain the validation mae since it was predicted using the training (not validation) dataset. 3) is not true as the trees are limited to depth 1 and only 1 tree will be grown – a tree of depth 1 cannot handle interactions, since it can only model a single feature. 4) is not true as it is just a variable name, which can be changed to any valid Python name.

Question 8) You are using early-stopping as a regularisation approach for a particular learning algorithm. You are using RMSE as your loss metric. D_train contains the training set and D_dev contains the validation set. The algorithm terminates with the following output:

```
Early stopping, best iteration is:
[954] D_train's rmse: 0.4069     D_dev's rmse: 0.5767
Model best iteration 954
Wall time: 3min 16s
```

Select the correct answer:

A. Bias = 0.4069, Variance = 0.5767, Total Error = 0.9836
B. Bias = 0.4069, Variance = 0.1698, Total Error = 0.5767
C. Bias = 0.5767, Variance = 0.4069, Total Error = 0.9836
D. Bias = 0.5767, Variance = -0.1698, Total Error = 0.4069
E. None of the above

B or E: Technically, the total error can be decomposed into the sum of

1) Irreducible error

2) Bias

3) Variance

Variance will be the difference between the error on the validation set and the training set, which is 0.1698, so that rules out A, C and D. Also, the total error should be equal to the error on the validation dataset (0.5767). So that again rules out A, C and D. So **B**. However, technically the bias is distinct from the irreducible error, so if you assume non-zero irreducible error, then the bias has to be smaller than the error on the training set (since irreducible error + bias = error on training set). So **E** (None of the above) is correct under this assumption.

Question 9) Consider the model below:

```
tree = DecisionTreeRegressor(min_samples_leaf=5, max_depth=5)
```

If the model is overfitting the training, which of the following changes would help address overfitting?

A. tree=DecisionTreeRegressor(min_samples_leaf=10, max_depth=2)
B. tree=DecisionTreeRegressor(min_samples_leaf=2, max_depth=5)
C. tree=DecisionTreeRegressor(min_samples_leaf=5, max_depth=10)
D. tree=DecisionTreeRegressor(min_samples_leaf=2, max_depth=10)

A: To reduce overfitting, you could **increase** min_samples_leaf or **decrease** max_depth or both (as in A). The other choices would all increase overfitting, not decrease it.

Question 10) Consider a dataset of features consists of the monthly returns of 10,000 individual bonds over 10 years (120 months). You wish to use it to predict the return of a global bond index in the following month, for which you also have 10 years of monthly returns.

Only one of the following approaches is actually sensible and feasible. Which one? (All the others suffer from flaws ranging from serious to fatal.)

A. Predict the next month bond index return using Ordinary Least Squares (OLS) implemented via the normal equation.
B. First reduce the dimensionality of the dataset by obtaining the first 100 principal components using PCA, then use the correlation matrix of the 100 principal components to cluster the principal components into a smaller number of clusters using agglomerative hierarchical clustering using average linkage. Then use the average return of the clusters to predict the bond index return in the next month.
C. Expand the set of features in the dataset by applying PolynomialFeatures with degree 8, then apply Ridge Regression to the new set of expanded features to predict the returns to a bond index in the next period.
D. Use Partial Least Squares to extract a lower-dimensional representation of the dataset to predict the bond index.
E. Convert each month's data into a 100 x 100 image to represent all the bond returns in that month (with the scaled individual bond returns representing pixel brightness). Then train a convolutional neural network (CNN) to predict the bond index in the next month using the images.

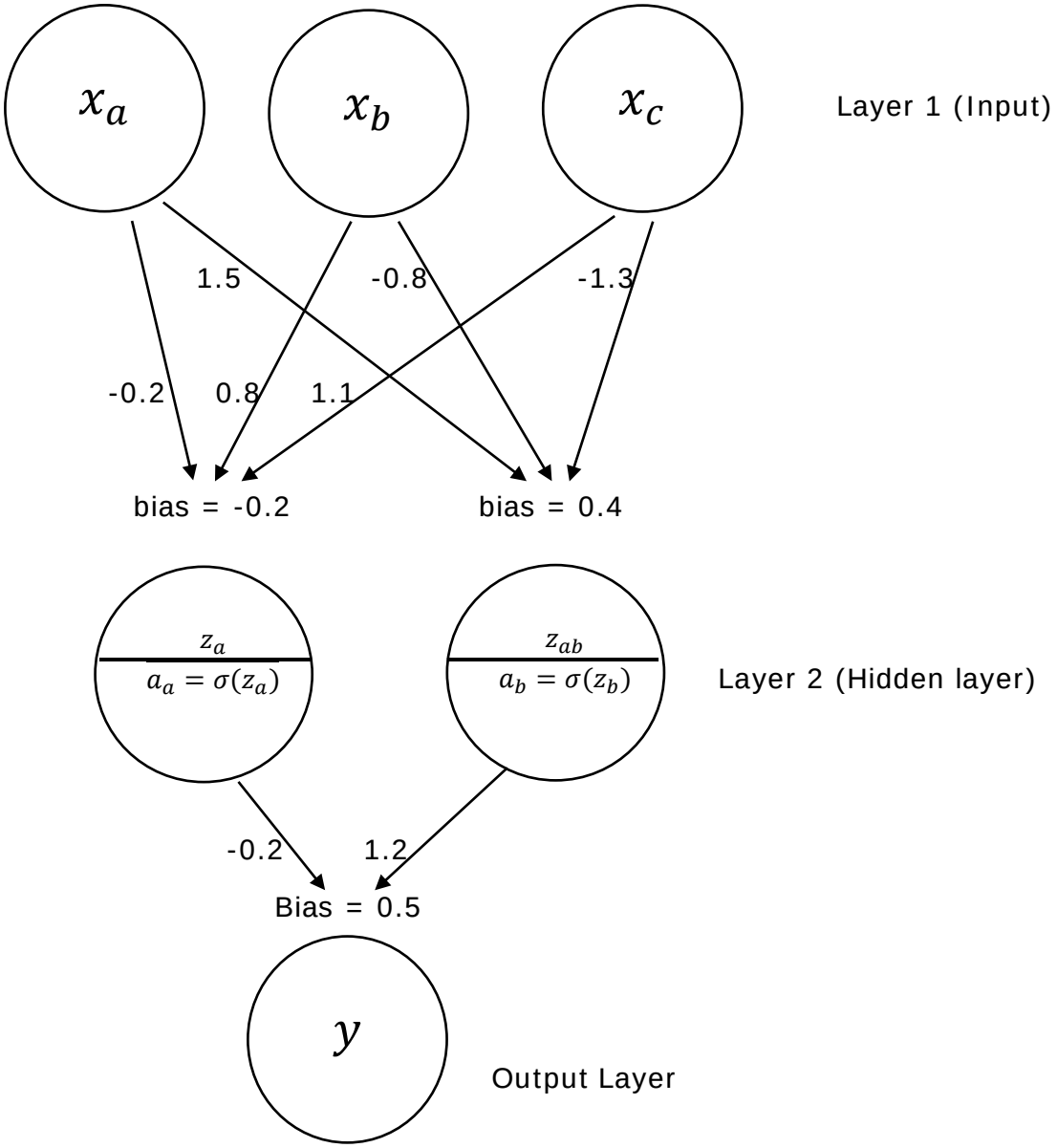
D: (A) won't work as the number of features >> the number of instances, which means the VCV is not of full rank and OLS normal equation will fail (intuitively, there are more coefficients than datapoints, so the datapoints cannot be determined using the normal equation – but it could, if you were using gradient descent instead). (B) won't work, since the correlation matrix of principal components are all zero's off the diagonal, and hence cannot be used as the base for a clustering dissimilarity measure – the dissimilarity would be identical for any pair of bonds. (C) will generate 10,000^8 = 10^32 features = 10^23 gigabytes at even at one byte per feature. Good luck with your ridge regression… (E) will yield a dataset of 120 images, not really enough for training a convolutional neural network which typically needs millions of images…So **D** seems the only feasible approach, even if it is not perfect.

Section B: Long questions (3 questions; attempt all)

Question B.1 [10 Marks]

Consider the following neural network with one hidden layer. It is intended to be used for regression tasks, not classification tasks. It consists of an input layer (X), a hidden layer (Z) with an associated activation (A) and an output layer (Y).

Each node is connected to the node in the layer below it. Each connection has a weight, which is indicated directly on the diagram below. In addition, a bias is added as also indicated on the diagram below. The activation function is $\sigma(z) = \max(z, 0)$, also known as a Rectified Linear Unit or ReLU.



a) For a single input vector $[x_a, x_b, x_c] = [3, -1, 2]$ calculate the output of the neural network. **[4 Marks]**

xa = 3, xb = -1, xc = 2

za = -0.2 + -0.2xa + 0.8xb + 1.1xc = -0.2 + -0.2(3) + 0.8(-1) + 1.1(2) = 0.6

zb = 0.4 + 1.5xa + -0.8xb + -1.3xc = 0.4 + 1.5(3) + -0.8(-1) + -1.3(2) = 3.1

aa = max(za,0) = max(0.6,0) = 0.6

ab = max(zb,0) = max(3.1,0) = 3.1

y = 0.5 + -0.2aa + 1.2ab = 0.5 + -0.2(0.6) + 1.2(3.1) = **4.1**

b) For an input matrix **X** with 3 columns (features) and 200 rows (instances), write out the linear algebra equation that will calculate **A**, the matrix of activations for all the nodes in the hidden layer for all of instances. Indicate the dimensions of each matrix in the equation. **[6 Marks]**

Z (200,2) = **X** (200,3) x **W** (3,2) + **b** (1,2)

Note, there are two nodes in the second layer, hence Z has two columns, so there are two biases and two weights per second-layer nodes -> that accounts for the 2 columns in W and b in the above

The first layer has three nodes; that accounts for the 3 rows in W. Note that W and b do NOT change across instances.

Let **K** (200,2) = **X** (200,3) x **W** (3,2) (the matrix product of X and W), then

Z (200,2) = **K** (200,2) + **b** (1,2)

Note that this is not conformant, so b (1,2) is “broadcast” to b (200,2) [see Geron p 492 solution to Chapter 10 question.]

So **Z** (200,2) = **K** (200,2) + **b** (200,2) which is conformant.

Then **A** (200,2) = σ(**Z**) = max(**Z**,0) (element-wise)

Question B.2 [10 Marks]

You have been engaged by Quantum Strategies LLC to work on a project to predict long-horizon (5-year) returns for individual US stocks. These predictions will be used as inputs into a proprietary portfolio optimiser. The data available to you includes 20 years' worth of monthly past returns, trading volumes and fundamental accounting data (at a quarterly frequency), all aligned in time when first available to investors. There are around 2,000 stocks in the cross section, although it varies through time.

Required: Write an email in which you outline the approach you intend to take, specifically indicating:

- How you intend to split the available data into training, validation and test data sets
- The machine learning approach you intend to use (and why)
- Which hyper-parameters you intend to use, and how you intend to select them
- How you will prevent over-fitting
- The performance measures you will use to evaluate the success of your approach

Important: *Please be specific in terms of which data set (training, validation or test) you will use for estimating model parameters, choosing hyper-parameters, evaluating model performance, etc.*

Question B.3 [10 Marks]

In their working paper Gu, Kelly and Xiu (2018) - "*Empirical Asset Pricing via Machine Learning*" consider 13 different machine leaning approaches (see Table 1, p. 25). The detail implementation of each is approach is described in Appendix C (starting at p 53).

a) Select *one* of these approaches and explain the approach in your own words in language that could be understood by a B.Com finance graduate without any particular background in machine learning. **[5 Marks]**

b) Describe the approach Gu, Kelly and Xiu (2018) use to split data into training, validation and test datasets. Why do they consider it necessary to use this approach, rather than simply allocating instances to the different datasets randomly, as is commonly done in other machine learning contexts? **[5 Marks]**