Name of Candidate: ....................................................

Student id: ....................................................

Signature: ....................................................

THE UNIVERSITY OF NEW SOUTH WALES

Term 2, 2020

**COMP9417 Machine Learning and Data Mining – Sample Final Examination**

1. I ACKNOWLEDGE THAT ALL OF THE WORK I SUBMIT FOR THIS EXAM WILL BE COMPLETED BY ME WITHOUT ASSISTANCE FROM ANYONE ELSE.
2. TIME ALLOWED — 24 hours

3. OPEN BOOK EXAM - LECTURE NOTES, TUTORIALS, AND ONLINE RESOURCES ARE PERMITT RY.
4. SUBMISSION — YOU MU ERS FOR EACH QUE ON ON A NEW PAGE. MARKS MAY BE DEDUCTED FOR UNCLEAR WORK. YOU MAY TYPE YOUR SOLUTIONS USING LATEX, OR TAKE CLE RIT-TEN WORK. FOR QUESTIONS THAT REQUIRE MIT A .PY FILE (SEE TEMPLATE) CONTAINING YOU RATED PLOTS/TABLES MUST BE INCLUDED IN THE PDF.

5. DISCUSSION WITH OTHER STUDENTS IS STRICTLY PROHIBITED. CODE SUBMISSIONS WILL BE CHECKED FOR PLAGIARISM. CHEATING WILL RESULT IN A FAILING GRADE FOR THE COURSE AND POTENTIAL FURTHER DISCIPLINARY ACTION.
6. IF NEEDED, YOU ARE PERMITTED TO SEEK CLARIFICATION FROM COURSE STAFF ON THE WEBCMS FORUM. QUESTIONS SPECIFIC TO CONTENT WILL NOT BE ANSWERED.

**Question 1** is on **Linear Regression** and requires you to refer to the following training data:

| x | y |
|---|---|
| 4 | 2 |
| 6 | 4 |
| 12 | 10 |
| 25 | 23 |
| 29 | 28 |
| 46 | 44 |
| 59 | 60 |

We wish to fit a linear regression model to this data, i.e. a model of the form:

$$\hat{y}_i = w_0 + w_1 x_i.$$

We consider the Least-Squares loss function:



$$L(w_0, w_1) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where $n$ is the total num



**(a)** Derive the least squares estiamtes of $w_0$ and $w_1$, and compute them for the provided data.
**(b)** Based on your linear model, what is the prediction for a test poin $= 50$?
**(c)** Consider a new training point, $(x_{\text{new}}, y_{\text{new}}) = (40$ uares parameters if we include all training data (including this new point



**(d)** The new training point in (c) can be considered to be what kind of point? What does such a point mean for your estimated parameters? How could you remedy the situation?
**(e)** Are you comfortable coding this question up in Numpy? What about using the scikit learn implementation of Linear Regression?
**(f)** Compute the derivatives of the following functions (SHOW YOUR WORKING):

1. $f(x) = x^2 \ln(x)$
2. $g(x) = (1 + 2x^4)^3$
3. $h(x) = \frac{2\ln(x)+4x}{x^3}$

**Question 2** is on **Tree Learning** and requires you to refer to the following dataset containing a sample $S$ of ten examples. Each example is described using two Boolean attributes $A$ and $B$. Each is labelled (classified) by the target Boolean function.

| $A$ | $B$ | Class |
|---|---|---|
| 1 | 0 | + |
| 0 | 1 | - |
| 1 | 1 | - |
| 1 | 0 | + |
| 1 | 1 | - |
| 1 | 1 | - |
| 0 | 0 | + |
| 1 | 1 | + |
| 0 | 0 | + |
| 0 | 0 | - |

**(a)** What is the entropy of these examples with respect to the given classification?



**(b)** What is the Information gain of attribute $A$ on sample $S$ above?

**(c)** What is the informat

**(d)** What would be chos romation gain splitting criterion? W



**(e)** What are ensembles? Discuss one example in which decision trees are used in an ensemble.

**Questions 3** is on **Perceptron Training** and requires you to refer to the following training data:

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| -2 | -1 | -1 |
| 2 | -1 | 1 |
| 1 | 1 | 1 |
| -1 | -1 | -1 |
| 3 | 2 | 1 |

**(a)** Apply the Perceptron Learning Algorithm with starting values $w_0 = 5$, $w_1 = 1$ and $w_2 = 1$, and a learning rate $\eta = 0.4$. Be sure to cycle through the training data in the same order that they are presented in the table.

**(b)** Consider a new point, $x_\star = (-5, 3)$. What is the predicted value and predicted class based on your learned perceptron for this point?

**(c)** Consider adding a new point to the data set, $x_\star = (2, 2)$ and $y_\star = -1$. Will your perceptron converge on the new dataset? How might you remedy this?



**(d)** Consider the following three logical functions:

1. $A \wedge \neg B$
2. $\neg A \vee B$
3. $(A \vee B) \wedge (\neg A \vee \neg B)$



Which of these functions can a perceptron learn? Explain. What are tw extend a perceptron to learn all three functions ?

**Questions 4** covers **Unsupervised Learning** and require you to refer to the following information.

In these two questions you will apply the $k$-MEANS algorithm. You will use a univariate (one-variable) dataset containing the following 12 instances:

$$\text{Dataset} = \{\ 2.01,\ 3.49,\ 4.58,\ 4.91,\ 4.99,\ 5.01,\ 5.32,\ 5.78,\ 5.99,\ 6.21,\ 7.26,\ 8.00\ \}$$

Use the *Manhattan* or *city-block* distance, i.e., the distance between two instances $x_i$ and $x_j$ is the absolute value of the difference $x_i - x_j$. For example, if $x_i = 2$ and $x_j = 3$ then the distance between $x_i$ and $x_j$ is $|2 - 3| = 1$. Use the arithmetic mean to compute the centroids.

Apply the $k$-MEANS algorithm to the above dataset of examples. Let $k = 2$. Let the two centroids (means) be initialised to $\{3.33, 6.67\}$. On each iteration of the algorithm record the centroids.

After two iterations of the algorithm you should have recorded two sets of two centroids.

| **Centroids** | **After 1 iteration** | **After 2 iterations** |
|---|---|---|
| Centroids 1 | { 2.75, 5.81 } | { 3.24, 6.01 } |
| Centroids 2 | 4.00, 6.22 | 4.17, 6.43 } |
| C | | } |
| C | | } |
| C | | } |



**(a)** After applying your algorithm to the dataset for two iterati centroids in the table above has been learned ?
(select the row of the table with values closest to your centroids)

(a) Centroids 1
(b) Centroids 2
(c) Centroids 3
(d) Centroids 4
(e) Centroids 5

Now apply the algorithm for one more iteration. Record the new centroids after iteration 3 and answer the following question.

**(b)** After 3 iterations it is clear that:

(a) due to randomness in the data, the centroids could change on further iterations

(b) due to randomness in the algorithm, the centroids could change on further iterations

(c) $k$-MEANS converges in probability to the true centroids

(d) the algorithm has converged and the clustering will not change on further iterations

(e) the algorithm has not converged and the clustering will change on further iterations

**Question 5** is on **Learning Theory** and requires you to apply a mistake-bounded learner to the following dataset.

This dataset has 6 binary features, $x_1, x_2, \ldots x_6$. The class variable $y$ can be either 1, denoting a positive example of the concept to be learned, or 0, denoting a negative example.

| Example | $\mathbf{x_1}$ | $\mathbf{x_2}$ | $\mathbf{x_3}$ | $\mathbf{x_4}$ | $\mathbf{x_5}$ | $\mathbf{x_6}$ | **Class** |
|---|---|---|---|---|---|---|---|
| 1) | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 2) | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 3) | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 4) | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 5) | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

Apply the Winnow2 algorithm to the above dataset of examples **in the order in which they appear**. Use the following values for the Winnow2 parameters: threshold $t = 2$, $\alpha = 2$. Initialise all weights to have the value 1.



| **Learned Weights** | $\mathbf{w_1}$ | $\mathbf{w_2}$ | $\mathbf{w_3}$ | $\mathbf{w_4}$ | $\mathbf{w_5}$ | $\mathbf{w_6}$ |
|---|---|---|---|---|---|---|
| Weight vector 1 | 2.000 | 1.000 | 1.000 | 0.000 | 2.000 | 1.000 |
| Wei | | | | | | |
| Wei | | | | | | |
| Wei | | | | | | |
| Weight vector 5 | 2.000 | 0.250 | 0.500 | 0.500 | 4.0 | |



**(a)** After one epoch, i.e., one pass through the dataset, which of 5 ... tions has been learned ?



(a) Weight vector 1
(b) Weight vector 2
(c) Weight vector 3
(d) Weight vector 4
(e) Weight vector 5

**(b)** On which of the examples did the algorithm ***not*** make a mistake ?

(a) Examples 1), 2) and 5)
(b) Example 5)
(c) Example 4)
(d) Examples 4) and 5)
(e) None of the above

**(c)** The algorithm has learned a consistent concept on the training data:

(a) True
(b) False
(c) It is not possible to determine this

**(d)** Assume the target concept from which this dataset was generated is defined by exactly two features. The worst-case mistake bound for the algorithm on this dataset is approximately:

(a) 1.79
(b) 2.58
(c) 3.58
(d) 4.67
(e) 10.75

**Question 6** is on Boosting Theory and requires you to implement the Adaptive Boosting Algorithm from lectures. Use the following code to generate a toy binary classification dataset:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import make_blobs

np.random.seed(2)
n_points = 15
X, y = make_blobs(n_points, 2, centers=[(0,0), (-1,1)])
y[y==0] = -1       # use -1 for negative class instead of 0

plt.scatter(*X[y==1].T, marker="+", s=100, color="red")
plt.scatter(*X[y==-1].T, marker="o", s=100, color="blue")
plt.show()
```



Your data should look like:



**(a)** By now, you should be familiar with the scikitlearn DecisionTreeClassifier class. Fit Decision trees of increasing maximum depth for depths ranging from 1 to 9. Plot the decision boundaries of each of your models in a $3 \times 3$ grid. You may find the following helper function useful:

```python
def plotter(classifier, X, y, title, ax=None):
```

```python
# plot decision boundary for given classifier
plot_step = 0.02
x_min, x_max = X[:, 0].min() - 1, X[:,0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:,1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
            np.arange(y_min, y_max, plot_step))
Z = classifier.predict(np.c_[xx.ravel(),yy.ravel()])
Z = Z.reshape(xx.shape)
if ax:
  ax.contourf(xx, yy, Z, cmap = plt.cm.Paired)
  ax.scatter(X[:, 0], X[:, 1], c = y)
  ax.set_title(title)
else:
  plt.contourf(xx, yy, Z, cmap = plt.cm.Paired)
  plt.scatter(X[:, 0], X[:, 1], c = y)
  plt.title(title)
```

**(b)** Comment on your results in (a). What do you notice as you increase the depth of the trees? What do we mean when we say that trees have low bias and high variance?



**(c)** We now restrict attention to trees of depth 1. These are the most basic decision trees and are commonly referred to as decision stumps. Consider the adaptive boosting algorithm presented in the ensemble m uild a model composed of $t$ we pick a model from the set of weak learners t



$$\epsilon_t = \sum_{i=1}^{n} w_{t-1,i} \mathbb{I}\{y_i \neq \hat{y}_i\}$$



where $w_{t-1,i}$ is the weight at the previous step for observation $i$, and $\mathbb{I}\{y_i \neq \hat{y}_i\}$ is equal to 1 if $y_i \neq \hat{y}_i$ and zero otherwise. We do this for a total of $T$ steps, which gives us a boosted model composed of $T$ base classifiers:

$$M(x) = \sum_{t=1}^{T} \alpha_t M_t(x)$$

where $\alpha_t$ is the weight assigned to the $t$-th model. Classification is then carried out by assigning a point to the positive class if $M(x) > 0$ or to the negative class if $M(x) < 1$. Here we will take the class of weak learners to be the class of Decision stumps. You may make use of the 'sample_weight' argument in the 'fit()' method to assign weights to the individual data points. Write code to build a boosted classifier for $T = 15$. Demonstrate the performance of your model on the generated dataset by printing out a list of your predictions versus the true class labels. (**note:** you may be concerned that the decision tree implementation in scikit learn does not actually minimise $\epsilon_t$ even when weights are assigned, but we will ignore this detail for the current question).

**(d)** In this question, we will extend our implementation in (c) to be able to use the plotter function in (b). To do this, we need to implement a boosting model class that has a 'predict' method. Once you do this, repeat (c) for $T = [2, \dots, 17]$. Plot the decision boundary of your 16 models in a $4 \times 4$ grid. The following template may be useful:

```
class boosted_model:
    def __init__(self, T):
      self.alphas = # YOUR CODE HERE
      # YOUR CODE HERE

    def predict(self, x):
        # YOUR CODE HERE
```

**(e)** Discuss the differences between bagging and boosting.

**Question 7** Some more suggestions - Make sure you are comfortable with the following

1. Naive Bayes Classification example from lectures/tutorials
2. Decision trees
3. Understanding the Bias Variance trade off
4. SVM calculation
5. K Means clustering
6. VC dimension
7. have worked through the labs and are comfortable with scikitlearn/numpy.