Assessment 2

Linear Models for Regression and Classification

Objectives

This assignment assesses your understanding of linear models for regression and classification, covered in Modules 2 and 3. The total marks of this assessment is 100. This assessment constitute 35% of your final mark for this unit.

Part A. Ridge Regression

In this part, you develop Ridge Regression by adding the L2 norm regularization to the linear regression (covered in Activity **1** of Module **2**). This part assesses your mathematical skills (derivation) and programming skills.

Question 1 [Ridge Regression, 25 Marks]

I. Given the gradient descent algorithms for linear regression (discussed in Chapter **2** of Module **2**), derive weight update steps of stochastic gradient descent (SGD) as well as batch gradient descent (BGD) for linear regression with L2 regularisation norm. Show your work with enough explanation in your PDF report; you should provide the steps of SGD and BGD, separately.

   Hint: Recall that for linear regression we defined the error function *E* and set its derivation to zero. For this assignment, you only need to add an L2 regularization term to the error function and set the derivative of both terms (error term plus the regularization term) to zero. This question is similar to Activity **1** of Module **2**.

II. Using R (with no use of special libraries), implement SGD and BGD algorithms that you derived in Step I. The implementation is straightforward as you are allowed to use the code examples from Activity **1** in Module **2**.

III. Now let's compare SGD and BGD implementations of ridge regression from Step II:
   A. Load **Task2A_train.csv** and **Task2A_test.csv** sets,
   B. Set the termination criterion as maximum of 20 weight updates for BGD, which is equivalent to 20 x *N* weight updates for SGD (where *N* is the number of training data),
   C. Run your implementations of SGD and BGD while all parameter settings (initial values, learning rate etc) are exactly the same for both algorithms. During run, record training error rate every time the weights get updated. Create a plot of error rates (use different colors for SGD and BGD), where the x-axis is the number of visited data points and y-axis is the error rate. Save your plot and attach it to your report. Note that for every *N* errors for SGD in

the plot, you will only have one error for BGD; the total length of the x-axis will be 20 x *N.*

D. Explain your observation based on the errors plot you generated in Part C. Particularly, discuss the convergence speed and the fluctuations you see in the error trends.

Part B. Bias-Variance Analysis

In this part, you conduct a bias-variance study on the ridge regression that you have developed in Part A. This task assesses your analytical skills, and is based on Chapter **6** of Module **2**. You basically recreate Figure 2.6.3 of Module **2** using your implementation of Ridge regression (with SGD) from Part A.

Question 2 [Bias-Variance for Ridge Regression, 25 Marks]

I. Load **Task2B_train.csv** and **Task2B_test.csv** sets,

II. Sample 50 sets from the provided training set, each of which having 100 randomly selected data points (with replacement).

III. For each lambda in {0, 0.2, 0.4, 0.6, …, 5} do:
   A. Build 50 regression models using the sampled sets
   B. Based on the predictions of these models on the test set, calculate the (average) test error, variance, $(\text{bias})^2$, and variance + $(\text{bias})^2$.

   Plot the (average) test error, variance, $(\text{bias})^2$, and variance + $(\text{bias})^2$ versus log lambda, and attach the plot to your report.

IV. Based on your plot in the previous step (III), what's the best value for lambda? Explain your answer in terms of the bias, variance, and test error.

Part C. Multiclass Perceptron

In this part, you are asked to demonstrate your understanding of linear models for classification. You expand the binary-class perceptron algorithm that is covered in Activity **1** of Module **3** into a multiclass classifier.

Background. Assume we have N training examples $\{(\mathbf{x}_1,t_1),\dots,(\mathbf{x}_N,t_N)\}$ where t_n can get K discrete values $\{C_1, ..., C_K\}$, i.e. a K-class classification problem.

Model. To solve a K-class classification problem, we can learn K weight vectors $\mathbf{w}_k$, each of which corresponding to one of the classes.

Prediction. In the prediction time, a data point **x** will be classified as $argmax_k\ \mathbf{w}_k . \mathbf{x}$

Training Algorithm. We train the multiclass perceptron based on the following algorithm:

- Initialise the weight vectors randomly $\mathbf{w}_1,..,\mathbf{w}_K$
- While not converged do:

```
○ For n = 1 to N do:
    ■ $y = \text{argmax}_k \mathbf{w}_k . \mathbf{x}_n$
    ■ If $y \neq y_n$ do
        ● $\mathbf{w}_y := w_y - \eta \mathbf{x}_n$
        ● $\mathbf{w}_{yn} := w_{yn} + \eta \mathbf{x}_n$
```

In what follows, we look into the convergence properties of the training algorithm for multiclass perceptron (similar to Activity 1 of Module 3).

Question 3 [Multiclass Perceptron, 25 Marks]

I. Load **Task2C_train.csv** and **Task2C_test.csv** sets,
II. Implement the multiclass perceptron as explained above. Please provide enough comments for your code in your submission.
III. Set the learning rate η to .1, and train the multiclass perceptron on the provided training data. After processing every 5 training data points (also known as a mini-batch), evaluate the error of the current model on the test data. Plot the error of the test data vs the number of mini-batches, and attach the plot to your report.
IV. Suppose we did not want to use multiclass Perceptron, and instead would be interested to use the one-versus-one approach to solve the multi-class classification problem (Chapter 2 in Module 3). The idea is to build *K(K−1)/2* classifiers for each possible pair of classes where *K* is the number of classes. Each point is then classified according to a majority vote among the discriminant functions.
    A. Train your *K(K-1)/2* perceptron binary classifiers using the training data.
    B. Predict the labels of the data points in the test set. Whenever there is a tie between two or more labels for a data point, call it a *confusion* event.
    C. Did you expect to see a confusion event in the one-versus-one approach? For how many test data points you have observed confusion? Why? Explain.

Part D. Logistic Regression vs. Bayesian Classifier

This task assesses your analytical skills. You need to study the performance of two well-known generative and discriminative models, i.e. Bayesian classifier and logistic regression, as the size of the training set increases. Then, you show your understanding of the behaviour of learning curves of typical generative and discriminative models.

Question 4 [Discriminative vs Generative Models, 25 Marks]

I. Load **Task2D_train.csv** and **Task2D_test.csv** as well as the Bayesian classifier (BC) and logistic regression (LR) codes from Activities **2** and **3** in Module **3**.
II. Using the first 5 data points from the training set, train a BC and a LR model, and compute their test errors. In a “for loop”, increase the size of training set (5 data points at a time), retrain the models and calculate their test errors until all training data points are used. In one figure, plot the test errors for each model (with different colors) versus the size of the training set; name the plot “Learning Curve” and add it to your report.
III. Explain your observations in your report:

A. What does happen for each classifier when the number of training data points is increased?
B. Which classifier is best suited when the training set is small, and which is best suited when the training set is big?
C. Justify your observations in previous questions (III.A & III.B) by providing some speculations and possible reasons.

Hint: Think about model complexity and the fundamental concepts of machine learning covered in Module **1**.

Submission & Due Date:

The files that you need to submit are:

1. Jupyter Notebook files containing the code for questions {1,2,3, 4} with the extension ".ipynb". The file names should be in the following format:
   STUDNETID_assessment_2_qX.ipynb
   where 'X=1,2,3, 4' is the question number. For example, the Notebook for Question 2 should be named STUDNETID_assessment_2_q2.ipynb
2. You must add enough comments to your code to make it readable and understandable by the tutor. Furthermore, you may be asked to meet (online) with your tutor when your assessment is marked to complete your interview.
3. A PDF file that contains your report, the file name should be in the following format:
   STUDNETID_assessment_2_report.pdf

You should replace STUDENTID with your own student ID. All files must be submitted via Moodle by 11:59pm (AEDT) Sunday Week 3.

Assessment Criteria:

The following outlines the criteria which you will be assessed against:

- Ability to understand the fundamentals of machine learning and linear models.
- Working code: The code executes without errors and produces correct results.
- Quality of report: You report should show your understanding of the fundamentals of machine learning and linear models by answering the questions in this assessment and attaching the required figures.

Penalties:

- Late submission (-20% of awarded marks for between 1-3 days late, -50% for between 4-6 days late. 0 marks awards after 6 days)
- Jupyter Notebook file is not properly named (-5%)
- The report PDF file is not properly named (-5%)