Assignment 1

FIT5201: Machine Learning

Please note that,

1. 1 sec delay will be penalized as 1 day delay. So please submit your assignment in advance (considering the possible internet delay) and do not wait until last minute.
2. We will not accept any resubmit version. So please double check your assignment before the submission.

Objectives

This assignment assesses your understanding of model complexity, model selection, uncertainty in prediction with bootstrapping, and probabilistic machine learning, and linear models for regression and classification, covered in Modules 1, 2, and 3. The total marks of this assignment is 125. This assignment constitutes 25% of your final mark for this unit.

Section A. Model Complexity and Model Selection

In this section, you study the effect of model complexity on the training and testing error. You also demonstrate your programming skills by developing a regression algorithm and a cross-validation technique that will be used to select the models with the most effective complexity.

Background. A KNN regressor is similar to a KNN classifier (covered in Activity 1.1) in that it finds the K nearest neighbors and estimates the value of the given test point based on the values of its neighbours. The main difference between KNN regression and KNN classification is that KNN classifier returns the label that has the majority vote in the neighborhood, whilst KNN regressor returns the average of the neighbors' values. In Activity 1 of Module 1, we use the number of mis-classifications as the measurement of training and testing errors in KNN classifier. For KNN regressor, you need to choose another error function as the measurement of training errors and testing errors.

Question 1 [KNN Regressor, 20 Marks]

I. [5 marks] Implement the KNN regressor function:

knn(train.data, train.label, test.data, K=3)

which takes the training data and their labels (continuous values), the test set, and the size of the neighborhood (K). It should return the regressed values for the test data points. Note that, you need to use a distance

function to choose the neighbors. The distance function used to measure the distance between a pair of data points is Manhattan distance function.

Hint: You are allowed to use KNN classifier code from Activity 1 of Module 1.

II. [5 marks] Plot the training and the testing errors versus $1/K$ for $K=1,..,35$ in one plot, using the **Task1A_train.csv** and **Task1A_test.csv** datasets provided for this assignment. Save the plot in your Jupyter Notebook file for Question 1. Report your chosen error function in your Jupyter Notebook file.

III. [10 marks] Report (in your Jupyter Notebook file) the optimum value for K in terms of the testing error. Discuss the values of K and model complexity corresponding to *underfitting* and *overfitting* based on your plot in the previous part (Part II).

Question 2 [Leave-One-Out Cross-Validation, 15 Marks]

I. [5 marks] A special case of L-Fold cross-validation is Leave-One-Out cross-validation where L (i.e., the number of folds/subsets) is equal to the size of the training dataset. In each iteration, one training data point is used as the validation set. Implement a Leave-One-Out cross-validation (CV) function for your KNN regressor:

cv(train.data, train.label, K)

which takes the training data and their labels (continuous values), K value (the number of neighbors), the number of folds, and returns errors for different folds of the training data.

II. [8 marks] Using the training data in Question 1, run your Leave-One-Out CV. Change the value of $K=1,..,20$ in your KNN regressor, and for each K compute the average error values you have got for folds. Plot the average of error values versus $1/K$ for $K=1,..,20$ in your KNN regressor. Save the plot in your Jupyter Notebook file for Question 2.

III. [2 marks] Report (in your Jupyter Notebook file) the optimum value for K based on your plot for this Leave-One-Out cross validation in the previous part (Part II).

Section B. Prediction Uncertainty with Bootstrapping

This section is the adaptation of Activity 1.2 from KNN classification to KNN regression. You use the bootstrapping technique to quantify the uncertainty of predictions for the KNN regressor that you implemented in **Section A**.

Background. Please refer to the background in **Section A**.

Question 3 [Bootstrapping, 25 Marks]

I. [5 marks] Modify the code in Activity 1.2 to handle bootstrapping for KNN regression.

II. [5 marks] Load **Task1B_train.csv** and **Task1B_test.csv** sets. Apply your bootstrapping for KNN regression with times = 30 (the number of subsets), size = 60 (the size of each subset), and change *K=1,..,20* (the neighbourhood size). Now create a boxplot where the x-axis is *K*, and the y-axis is the average test error (and the uncertainty around it) corresponding to each *K*. Save the plot in your Jupyter Notebook file for Question 3.

Hint: You can refer to the boxplot in Activity 1.2 of Module 1. But the error is measured in different ways compared with the KNN classifier.

III. [5 marks] Based on the plot in the previous part (Part II), how does the test error and its uncertainty behave as K increases? Explain in your Jupyter Notebook file.

IV. [5 marks] Load **Task1B_train.csv** and **Task1B_test.csv** sets. Apply your bootstrapping for KNN regression with K=5 (the neighbourhood size), times = 50 (the number of subsets), and change sizes = 5, 10, 15,..., 75 (the size of each subset). Now create a boxplot where the x-axis is 'sizes', and the y-axis is the average test error (and the uncertainty around it) corresponding to each value of 'sizes'. Save the plot in your Jupyter Notebook file for Question 3.

V. [5 marks] Based on the plot in the previous part (Part IV), how does the test error and its uncertainty behave as the size of each subset in bootstrapping increases? Explain in your Jupyter Notebook file.

Section C. Probabilistic Machine Learning

In this section, you show your knowledge about the foundation of the probabilistic machine learning (i.e. probabilistic inference and modeling) by solving one simple but basic statistical inference problems. Solve the following problems based on the probability concepts you have learned in Module 1 with the same math conventions.

Question 4 [Bayes Rule, 20 Marks]

Recall the simple example from Appendix A of Module 1. Suppose we have one red, one blue, and one yellow box. In the red box we have 3 apples and 1 orange, in the blue box we have 4 apples and 4 orange, and in the yellow box we have 5 apples and 3 oranges. Now suppose we randomly selected one of the boxes and picked a fruit. If the picked fruit is an orange, what is the probability that it was picked from the yellow box?

Note that the chances of picking the red, blue, and yellow boxes are 50%, 30%, and 20% respectively and the selection chance for any of the pieces from a box is equal for all the pieces in that box. Please show your work in your PDF report.

Hint: You can formulize this problem following the denotations in "Random Variable" paragraph in Appendix A of Module 1.

Section D. Ridge Regression

In this section, you develop Ridge Regression by adding the L2 norm regularization to the linear regression (covered in Activity 2.1 of Module 2) and study the effect of the L2 norm regularization on the training and testing errors.

This section assesses your mathematical skills (derivation), programming, and analytical skills.

Question 5 [Ridge Regression, 25 Marks]

I. [10 marks] Given the gradient descent algorithms for linear regression (discussed in Chapter 2 of Module 2), derive weight update steps of stochastic gradient descent (SGD) for linear regression with L2 regularisation norm. Show your work with enough explanation in your PDF report; you should provide the steps of SGD.

Hint: Recall that for linear regression we defined the error function E. For this assignment, you only need to add an L2 regularization term to the error function (error term plus the regularization term). This question is similar to Activity 2.1 of Module 2.

II. [5 marks] Using R (with no use of special libraries), implement an algorithm that you derived in Step I. The implementation is straightforward as you are allowed to use the code examples provided.

III. Now let's study the effect of the L2 norm regularization on the training and testing errors:

a. Load **Task1C_train.csv** and **Task1C_test.csv** sets.

b. [5 marks] For each lambda (the regularization parameter) in {0, 0.5, 1.0, …, 10}, build a regression model and compute the training and testing errors, using the provided data sets. While building each model, all parameter settings (initial values, learning rate, etc) are exactly the same, except a lambda value. Set the termination criterion as maximum of 20 x N weight updates (where N is the number of training data). Create a plot of error rates (use different colors for the training and testing errors), where the x-axis is log lambda and y-axis is the error rate. Save your plot in your Jupyter Notebook file for Question 5.

c. [5 marks] Based on your plot in the previous part (Part b), what's the best value for lambda? Discuss lambda, model complexity, and error rates, corresponding to underfitting and overfitting, by

observing your plot. (Include all your answers in your Jupyter Notebook file.)

Section E. Multiclass Perceptron

In this section, you are asked to demonstrate your understanding of linear models for classification. You expand the binary-class perceptron algorithm that is covered in Activity 3.1 of Module 3 into a multiclass classifier. Then, you study the effect of the learning rate on the error rate. This section assesses your programming, and analytical skills.

Background. Assume we have N training examples $\{(x_1,t_1),...,(x_N,t_N)\}$ where t_n can get K discrete values $\{C_1, ..., C_K\}$, i.e. a K-class classification problem. We use yy_{nn} to represent the predicted label of xx_{nn}

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

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

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

- Initialise the weight vectors randomly $\boldsymbol{w}_1,..,\boldsymbol{w}_K$
- While not converged do:
    - For n = 1 to N do:
        - y = $\text{argmax}_k \mathbf{w}_k \cdot \mathbf{x}_n$
        - If y != y_n do
            - $\mathbf{w}_y := \mathbf{w}_y - \eta \mathbf{x}_n$
            - $\mathbf{w}_{yn} := \mathbf{w}_{yn} + \eta \mathbf{x}_n$

In what follows, we look into the convergence properties of the training algorithm for multiclass perceptron (similar to Activity 3.1 of Module 3).

Question 6 [Multiclass Perceptron, 20 Marks]

I. Load **Task1D_train.csv** and **Task1D_test.csv** sets.

II. [10 marks] Implement the multiclass perceptron as explained above. Please provide enough comments for your code in your submission.

III. [10 marks] Train two multiclass perceptron models on the provided training data by setting the learning rates η to .1 and .01 respectively. Note that all parameter settings stay the same, except the learning rate, when building each model.
For each model, evaluate the error of the model on the test data, after processing every 5 training data points (also known as a mini-batch). Then, plot the testing errors of two models built based on the learning rates .1 and .01(with different colors) versus the number of mini-batches. Include it in your Jupyter Notebook file for Question 6.
Now, explain how the testing errors of two models behave differently, as the training data increases, by observing your plot. (Include all your answers in your Jupyter Notebook file.)

Submission & Due Date:

The files that you need to submit are:

1. Jupyter Notebook files containing the **code** and **your answers** for questions {1,2,3,5,6} with the extension ".ipynb". The file names should be in the following format **STUDNETID_assessment_1_qX.ipynb** where 'X=1,2,3,5,6' is the question number. For example, the Notebook for Question 2 should be named **STUDNETID_assessment_1_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_1_report.pdf** You should replace

STUDENTID with your own student ID. All files must be submitted via Moodle before the due date and time.

4. Zip all of your files and submit it via Moodle. The name of your file must be in the following format:
   STUDNETID_FirstName_LastName_assessment_1_report.pdf
   where in addition to your student ID, you need to use your first name and last name as well.

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 (students who submit an assessment task after the due date will receive a late-penalty of 10% of the available marks in that task per calendar day. Assessment submitted more than 7 calendar days after the due date will receive a mark of zero (0) for that assessment task.)

- Jupyter Notebook file is not properly named (-5%)

- The report PDF file is not properly named (-5%)