Assignment 1

FIT5201: Data Analysis Algorithms
Due date: 21:59:59 13, Sep, 2019

Please note that,

1. 1 sec delay will be penalized as 1-3 days 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 150. 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 (e.g., the sum of the squares of the errors) as the measurement of training errors and testing errors.

Question 1 [KNN Regressor, 20 Marks]

I. Implement the KNN regressor function:

knn(train.data, train.label, test.data, K=4)

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 Euclidean distance function.

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

II. Plot the training and the testing errors versus $1/K$ for $K=1,..,20$ 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. Report (in your Jupyter Notebook file) the optimum value for K in terms of the testing error. Discuss the values of K corresponding to *underfitting* and *overfitting* based on your plot in the previous part (Part II).

Question 2 [L-fold Cross Validation, 15 Marks]

I. Implement a L-Fold Cross Validation (CV) function for your KNN regressor:

cv(train.data, train.label, K, numFold=10)

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

II. Using the training data in Question 1, run your L-Fold CV where the numFold is set to 10. Change the value of $K=1,..,15$ in your KNN regressor, and for each K compute the average 10 error numbers you have got. Plot the average error numbers versus $1/K$ for $K=1,..,15$ in your KNN regressor. Save the plot in your Jupyter Notebook file for Question 2.

III. Report (in your Jupyter Notebook file) the optimum value for K based on your plot for this 10-fold cross validation in the previous part (Part II).

Section B. Prediction Uncertainty with Bootstrapping

This section is the adaptation of Activity 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. Modify the code in Activity 2 to handle bootstrapping for KNN regression.

II. Load **Task1B train.csv** and **Task1B test.csv** sets. Apply your bootstrapping for KNN regression with times = 100 (the number of subsets), size = 25 (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 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 2 of Module 1. But the error is measured in different ways compared with the KNN classifier.

III. 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. Load **Task1B train.csv** and **Task1B test.csv** sets. Apply your bootstrapping for KNN regression with K=10 (the neighbourhood size), size = 25 (the size of each subset), and change times = 10, 20, 30,.., 200 (the number of subsets). Now create a boxplot where the x-axis is 'times', and the y-axis is the average error (and the uncertainty around it) corresponding to each value of 'times'. Save the plot in your Jupyter Notebook file for Question 3.

V. Based on the plot in the previous part (Part IV), how does the test error and its uncertainty behave as the number of subsets 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 two 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 and one blue box. In the red box we have 2 apples and 6 oranges, whilst in the blue box we have 3 apples and 1 orange. 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 blue box?

Note that the chance of picking the red box is 60% 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 1 of Module 2). This section assesses your mathematical skills (derivation) and programming skills.

Question 5 [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. 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 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 provided.

III. Now let's compare SGD and BGD implementations of ridge regression from Step II:

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

   b. Set the termination criterion as maximum of 18 weight updates for BGD, which is equivalent to 18 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 in your

Jupyter Notebook file for Question 5. 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 18x N.

d. Explain (in your Jupyter Notebook file) 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.

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 1 of Module 3 into a multiclass classifier.

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 y_n to represent the predicted label of x_n

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_n != t_n do
      - $\boldsymbol{w}_{y_n} := \boldsymbol{w}_{y_n} - \eta \boldsymbol{x}_n$
      - $\boldsymbol{w}_{t_n} := \boldsymbol{w}_{t_n} + \eta \boldsymbol{x}_n$

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

Question 6 [Multiclass Perceptron, 20 Marks]

I. Load **Task1D_train.csv** and **Task1D_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 .09, 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 include it in your Jupyter Notebook file for Question 6.

Section F. 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 behavior of learning curves of typical generative and discriminative models.

Question 7 [Discriminative vs Generative Models, 25 Marks]

I. Load **Task1E train.csv** and **Task1E 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; include the plot in your Jupyter Notebook file for Question 7.

III. Explain your observations in your Jupyter Notebook file.:

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** and **your answers** for questions {1,2,3,5,6,7} with the extension ".ipynb". The file names should be in the following format **STUDNETID_assessment_1_qX.ipynb** where 'X=1,2,3,5,6,7' 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.zip
   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 (-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%)