Assignment 2

Latent Variables and Neural Networks

Due Date: 21:59:59 23 May 2021

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 consists of three parts (A,B,C), which cover latent variables models and neural networks (Modules **4** and **5**). The total marks of this assessment are 100 and will contribute 16% to your final score.

Part A. Document Clustering

In this part, you solve a document clustering problem using unsupervised learning algorithms (i.e., soft and hard Expectation Maximization) for document clustering**.**

Question 1 [EM for Document Clustering, 40 Marks]

I. Derive **Expectation** and **Maximization** steps of the hard-EM algorithm for Document Clustering, show your work in your submitted PDF report. In particular, include all model parameters that should be learnt and the exact expression (using the same math convention that we saw in the Module 4) that should be used to update these parameters during the learning process (ie., E step, M step and assignments).

II. Implement the hard-EM (you derived above) and soft-EM (derived in Chapter **5** of Module **4**). Please provide enough comments in your submitted code.

Hint: If it helps, feel free to base your code on the provided code for EM algorithm for GMM in Activity **4.1** or the codebase provided in the Moodle).

III. Load **Task2A.txt** file and necessary libraries (if needed, perform text preprocessing similar to what we did in Activity **4.2**), set the number of clusters K=4, and run both the soft-EM and hard-EM algorithms on the provided data.

IV. Perform a PCA on the clusterings that you get based on the hard-EM and soft-EM in the same way we did in Activity **4.2**. Then, visualize the obtained clusters with different colors where x and y axes are the first two principal components (similar to Activity **4.2**). Attach the plots to your PDF report and report how and why the hard and soft-EM are different, based on your plots in the report.

Part B. Neural Network vs. Perceptron

In this part, you apply a 3-layer Neural Network on a synthetically generated data to compare its performance with Perceptron. Here, we are looking for your explanation about the differences between perceptron and NN that leads to different results.

Question 2 [Neural Network's Decision Boundary, 30 Marks]

I. Load **Task2B_train.csv** and **Task2B_test.csv** sets, plot the training data with classes are marked with different colors, and attach the plot to your PDF report.

II. Train two perceptron models on the loaded training data by setting the learning rates η to .01 and .09 respectively, using a code from Activity **3.1**. Calculate the test errors of two models and find the best η and its corresponding model, then plot the test data while the points are colored with their estimated class labels using the best model that you have selected; attach the plot to your PDF report.

Hint: Note that you must remove NA records from the datasets (using "complete.cases()' function). You may also choose to change the labels from [0, 1] to [-1, +1] for your convenience. If you decided to use the code from Activity 3.1, you may need to change some initial settings (e.g., epsilon and tau.max). Finally, remember

that perceptron is sensitive to initial weights. Therefore, we recommend to run your code a few times with different initial weights.

III. For each combination of K (i.e, number of units in the hidden layer) in {5, 10, 15, …, 100} and μ (learning rate) in {0.01, 0.09}, run the 3-layer Neural Network given to you in Activity **5.1** and record testing error for each of them (40 models will be developed, based on all possible combinations). Plot the error for μ 0.01 and 0.09 vs K (one line for μ 0.01 and another line for μ 0.09 in a plot) and attach it to your PDF report. Based on this plot, find the best combination of K and μ and the corresponding model, then plot the test data while the points are colored with their estimated class labels using the best model that you have selected; attach the plot to your PDF report.

Hint: In case you choose to use the provided examples in Activity 5.1, you may need to transpose the dataset (using "t()" function) and use different values for parameter settings (e.g., lambda).

IV. In your PDF report, explain the reason(s) responsible for such difference between perceptron and a 3-layer NN by comparing the plots you generated in Steps II and III.

Hint: Look at the plots and think about the model assumptions.

Part C. Self-Taught Learning

In this part, you implement self-taught learning for Neural Network using the Autoencoder that provided in Activity **5.2** and a 3-layer NN (from Activity **5.1** or **H2O** package)

Question 3 [Self Taught Neural Network Learning, 30 Marks]

I. Load **Task2C labeled.csv**, **Task2C unlabeled.csv** and **Task2C test.csv** data sets and required libraries (e.g., H2O). Note that we are going to use **Task2C labeled.csv** and **Task2C unlabeled.csv** for training the autoencoder. We are going to use **Task2C labeled.csv** for training the classifier. Finally, we evaluate the trained classifier on the test **Task2C test.csv.**

II. Train an autoencoder (similar to Activity **5.2**) with only one hidden layer and change the number of its neurons to: 20, 40, 60, 80, …, 440 (i.e. from 20 to 440 with a step size of 20).

III. For each model in Step II, calculate and record the reconstruction error which is simply the average (over all data points while the model is fixed) of Euclidian distances between the input and output of the autoencoder (you can simply use "h2o.anomaly()" function). Plot these values where the x-axis is the number of units in the middle layer and the y-axis is the reconstruction error. Then, save and attach the plot to your PDF report. Explain your findings based on the plot in your PDF report.

IV. Build the 3-layer NN from Activity 5.1 or "h2o.deeplearning" function (make sure you set " autoencoder = FALSE") to build a classification model using all the original attributes from the training set and change the number of its neurons to: 20, 40, 60, 80, .., 440 like Step II. For each model, calculate and record the test error.

V. Build *augmented* self-taught networks using the models learnt in Step II. For each model:

   A. Add the output of the middle layer of an autoencoder as extra features to the original feature set.
   B. Train a 3-layer NN using all features (original + extra) and varying the number of hidden neurons (like Step IV) as well. Then calculate and record the test error.

   For example, each model should be developed as follows:

   Model 1: 20 hidden neurons + extra 20 features (from an autoencoder),
   Model 2: 40 hidden neurons + extra 40 features (from an autoencoder),
   …,
   Model 22: 440 hidden neurons + extra 440 features (from an autoencoder).

VI. Plot the error rates for the 3-layer neural networks from Step IV and the augmented self-taught networks from Step V, while the x-axis is the number of hidden neurons and y-axis is the classification error. Save and attach the plot to your PDF report. In your pdf, explain how the performance of the 3-layer neural networks and the augmented self-taught networks is different and why they are different or why they are not different, based on the plot.

Hint: Since the dataset for this task is large and high-dimensional, running the whole experiments several times is very time consuming. Therefore, it is recommended to only use a small portion of your data when you develop or debug your code.

Hint: If you can combine Step II, IV and V (so learn each autoencoder only once), you may save a great portion of the execution time.

Hint: If you don't see the expected behaviour in your plots, you may need to check that the data is clean, i.e. it doesn't have NA entries, it's normalised etc. Moreover, you may need to check that your implementation of the model and training/decoding algorithms is correct.

Submission & Due Date:

The files that you need to submit are:

1. Jupyter Notebook files containing the code for questions {1,2,3} with the extension ".ipynb". The file names should be in the following format: **STUDNETID_assessment_2_qX.ipynb** where 'X=1,2,3' 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.

Assessment Criteria:

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

- Ability to understand the fundamentals of neural networks and latent variable models.

- Working code: The code executes without errors and produces correct results.

- Quality of report: You report should show your understanding of the latent variable models and neural networks 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%)