COL 341: Assignment 2

Notes:

- This assignment has two parts - Neural Network and Naive Bayes.
- You should submit all your code (including any pre-processing scripts written by you) and any graphs that you might plot.
- You are advised to use vector operations (wherever possible) for best performance.
- Include a report of maximum 5 pages which should be a brief description explaining what you did. Include any observations and/or plots required by the question in the report.
- You should use Python/R for all your programming solutions.
- Your assignments will be auto-graded, make sure you test your programs before submitting. We will use your code to train the model on training data and predict on test set.
- Input/output format, submission format and other details are included on README. Your programs should be modular enough to accept specified parameters.
- You should submit work of your own. You should cite the source, if you choose to use any external resource. You will be awarded F grade or DISCO in case of plagiarism.
- You can use total of 7 buffer days across all assignments.
- You can download data from this link.

1. **Neural Networks (65 points, Due date: Sep 7, 2018)**
In this problem, we'll train neural networks to classify Devanagari Handwritten Characters. You have been provided a CSV data file. Each row in the data file corresponds to an image of size 32x32. First entry of each row corresponds to the label associated with the image followed by the vector of gray-scale pixel intensities of the image. The data belongs to 46 different classes: 36 Devanagari characters and 10 Devanagari numerals.

   (a) (25 points) Write a program implementing a general neural network architecture. Implement the back-propagation algorithm to train the network. You should train the network using mini Batch Gradient Descent (BGD) with adaptive learning rate $\eta_t = \frac{\eta_0}{\sqrt{t}}$. Your implementation should be generic enough so that it can work with different architectures. Assume a fully connected architecture i.e., each unit in a hidden layer is connected to every unit in the next layer. Refer to this file for reference . You should implement the algorithm from first principles and not use any existing python/R modules. Use Mean Squared Error(MSE) as the loss function. Use **sigmoid** as the activation function for the units in output layer.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}\|Y_n - \hat{Y_n}\|^2 \tag{1}$$

   Note that Y_n and $\hat{Y_n}$ are vectors. Here Y is a one-hot vector where only one value (corresponding to true class index) is non-zero. N is the number of samples in the batch.

(b) (20 points) Use Your Implementation to train a neural network on the given Devanagari Data Set. Experiment with different types of architectures. Vary the number of hidden layers. Try different number of units in each layer, different loss functions etc. What happens as we increase the number of units or the number of hidden layers? Comment on your observation and report your best performing architecture.

Note:

- Use holdout method to find the best architecture. Split the data set into two: a set of examples to train with, and a validation set. Train the model on the training set. Use the prediction on the validation set to determine which architecture to use.
- Use sigmoid as the activation unit.

(c) (20 points) In the previous part pixel values are being used as features of the image. Can some features other than the absolute pixel values be used? Experiment with various feature extraction techniques such as Gaber Filters, DCT, FFT, HOG, wavelet etc. to extract feature from the images. Try non linearities other than the sigmoid. How does these changes affect your accuracy compared to the previous parts. Comment on your observations and report your best performing design.

Evaluation:

- For code, you can get 0 (error), half (code runs fine but predictions are incorrect within some predefined threshold) and full (works as expected).
- For part-b and part-c, marks will be given based on training time and accuracy on test data-set. There will be relative marking for this part.
- For part-b and part-c marking will be done in two parts: code (75%) and report(25%).

2. **Text Classification using naive bayes (35 points, Due date: Sep 7, 2018)**
In this problem, we will use the Naive Bayes algorithm for text classification. The dataset for this problem is a subset of the Amazon product data. First entry of each row is the rating followed by the review. Given a review, task is to predict the rating given by the reviewer. A review comes from one of the five categories (class label). Here, class label represents rating given by the user along with the review.

(a) (10 points) Implement the Nave Bayes algorithm to classify each of the articles into one of the given categories. Report the accuracy over the training as well as the test set. Notes:

i. Use unigrams as features.
ii. Make sure to use the Laplace smoothing in Naive Bayes to avoid any zero probabilities. Use c=1.
iii. You should implement your algorithm using logarithms to avoid underflow issues.
iv. You should implement Naive Bayes from the first principles and not use any existing R/python modules.

(b) (10 points) The dataset provided is in the raw format i.e., it has all the words appearing in the original set of articles. This includes words such as of, the, and etc. (called stopwords). Presumably, these words may not be relevant for classification. In fact, their presence can sometimes hurt the performance of the classifier by introducing noise in the data. Similarly, the raw data treats different forms of the same word separately, e.g., eating and eat would be treated as separate words. Merging such variations into a single word is called stemming.

Perform stopword removal and stemming before using naive bayes. You can use libraries (nltk for python, tm for R) to perform stopword removal and stemming.

(c) (15 points) Feature engineering is an essential component of Machine Learning. It refers to the process of manipulating existing features/constructing new features in order to help improve the overall accuracy on the prediction task. For example, instead of using each word as a feature, you may treat bi-grams (two consecutive words) as a feature. Come up with at least two alternative features and learn a new model based on those features. Explain your best performing model in the report.

Evaluation:

- For code: you can get 0 (error), half (code runs fine but predictions are incorrect within some predefined threshold) and full (works as expected).

- For part-c, marks will be given based on training time and macro-Fscore on test data-set. There will be relative marking for this part.
- For part-c marking will be done in two parts: code (10) and report(5).