Project: Sarcasm Detection

Instructions:

- The aim of this assignment is to give you an initial hands-on regarding real-life machine learning application.
- Use separate training and testing data as discussed in class.
- You can only use Python programming language and Jupyter Notebook.
- You can only use **numpy**, **matplotlib, gensim** and are not allowed to use **NLTK, scikit-learn or any other machine learning toolkit**.
- **Submit your code as one notebook file (.ipynb) on LMS. The name of file should be <roll number 1>_<roll number 2>. Only one submission is required per group.**
- Deadline to submit this project is: **Friday 15th May, 2020 11:55 p.m.**

Problem:

The purpose of this project is to get you familiar with word2vec, logistic regression, k nearest neighbor and perceptron classification. You are given with News Headlines Dataset for Sarcasm Detection that contains news headlines labeled for sarcasm. Your task is to implement a sarcasm detector for the news headlines.

Dataset:

The data set contains 28,616 headlines which are divided into two sets:

- train: 22,892 headlines
- test: 5,724 headlines

The format of the files is <label>,<headline>. Where <label> is 1 if the headline is sarcastic and 0 otherwise.

Preprocessing:

- Remove stop words and punctuation marks from the data set. A stop words list is provided with the data set.
- Represent the news headline as the average of all words in it. You'll use the pre-trained word2vec representations for this purpose.
    - Download Google's pre-trained 300-dimensional word2vec representations from here. (It's 1.5 GB! Don't wait till last date)
    - Install and import gensim to use the pre-trained representations.
    - To represent a sentence with 300-dimentional real valued vector, retrieve the vector representations of the words in it and then take the mean (average). You can ignore the words that are not in model's vocabulary.

Classification:

You’ll need to implement and compare three classification algorithms.

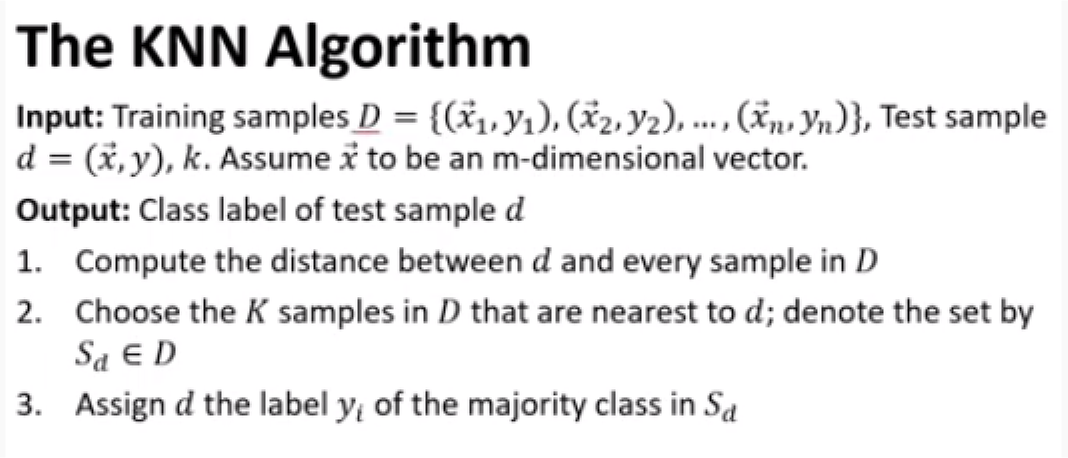
Logistic Regression

Implement Logistic Regression keeping in view all the discussions from the class lectures. Feel free to read Chapter 5 of Speech and Language Processing book to get in-depth insight of Logistic Regression classifier. Specifically, you’ll need to implement the following:

- Sigmoid function
- Cross-entropy loss function
- Mini-batch gradient descent with batch size of 32 samples
- Prediction function that predict whether the label is 0 or 1 for test set using learned logistic regression

k Nearest Neighbors

Implement kNN keeping in view all the discussions from the class lectures. Specifically, follow the steps shown in figure below.



The KNN Algorithm

Input: Training samples $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$, Test sample $d = (\vec{x}, y)$, k. Assume $\vec{x}$ to be an m-dimensional vector.

Output: Class label of test sample d

1. Compute the distance between d and every sample in D
2. Choose the K samples in D that are nearest to d; denote the set by $S_d \in D$
3. Assign d the label y_i of the majority class in S_d

Use Cosine Similarity as your similarity metric. You can either use sorting or Quickselect to choose k nearest neighbors. Make sure you code in generic enough that it can run with any value of k. Handle the ties by backing off to k-1 neighbors.

Perceptron

Implement Perceptron keeping in view all the discussions from the class lectures. Change class labels from [0, 1] to [-1, 1] and use the activation threshold of 0. Specifically, you’ll need to implement the following:

- Perceptron learning algorithm
- Prediction function that predict whether the label is -1 or 1 for test set using learned perceptron weights

Use the procedural programming style and comment your code thoroughly.

Evaluation:

You are required to provide a confusion matrix with values obtained by running your Logistic Regression, k Nearest Neighbor k = {1, 3, 5, 7, 10} and Perceptron classifier on the test set. Also report Precision, Recall, Accuracy and F1 score.