PA 01: kNN Classification

CS535/EE514 Machine Learning

Deadline: Friday, 11:55 PM, 22nd October 2021

Instructions

- The aim of this assignment is to give you hands-on experience with a real-life machine learning application.
- You will be analyzing the sentiment of tweets using k-NN classification.
- You can only use the Python programming language and Jupyter Notebook.
- Please use procedural programming style and comment your code thoroughly.
- There are two parts of this assignment. In part 1, you can use **NumPy**, **Pandas**, **Matplotlib**, and any other standard Python libraries. You are not allowed to use **NLTK**, **scikit-learn**, or any other machine learning toolkit. You can only use **scikit-learn** in part 2.
- **Carefully read the submission instructions, plagiarism and late days policy.**

Submission Instructions

You should submit both your notebook file (`.ipynb`) and python script (`.py`) on LMS. Please name your file `Name_RollNo_Assignment1`. Zip these files in a folder and name the folder `Name_RollNo_Assignment1`. If you don't know how to save `.ipynb` as `.py` see this. Failing to submit any one of them might result in the reduction of marks.

Plagiarism Policy

The code MUST be done independently. Any plagiarism or cheating of work from others or the internet will be immediately referred to the DC. If you are confused about what constitutes plagiarism, it is your responsibility to consult with the instructor or the TA in a timely manner. **PLEASE DO NOT LOOK AT ANYONE ELSE'S CODE NOR DISCUSS IT WITH THEM**.

Late Days Policy

The deadline for the assignment is final. However, in order to accommodate all the 11th-hour issues, there is a late submission policy i.e. you can submit your assignment within 3 days after the deadline with a 25% deduction each day.

Part 1: Implementing kNN classifier from scratch (75 marks)

You are not allowed to use scikit-learn or any other machine learning toolkit for this part. You have to implement your own k-NN classifier from scratch. You may use Pandas, NumPy, Matplotlib, and other standard Python libraries.

Problem:

The purpose of this assignment is to get you familiar with the k nearest neighbor classification. You are given the **Apple Sentiment Tweets** dataset that contains around 1630 tweets about Apple labeled as positive, neutral and negative in the form of 1, 0, and -1 respectively. Your task is to implement the k nearest classifier and use it for predicting the sentiments of the tweets about Apple.

Dataset:

The dataset contains around 1,630 tweets. There are only two columns in the dataset:

Text: Contains the text of the tweet.

Sentiment: Contains the sentiment of the tweet which is divided into three classes: 1 (positive), -1 (negative), and 0 (neutral).

Divide this dataset into training and testing sets based on an 80-20 split using python.

Preprocessing:

In the preprocessing step, you're required to remove the stop words, punctuation marks, numbers, unwanted symbols, hyperlinks, and usernames from the tweets and convert them to lower case. You may find the string and regex module useful for this purpose. A stop word list is provided within the assignment.

Feature Extraction:

In the feature extraction step, you'll represent each tweet as a bag-of-words (BoW), that is, an unordered set of words with their position ignored, keeping only their frequency in the tweet. For example, consider the below tweets:

T1 = Welcome to machine learning!

T2 = kNN is a powerful machine learning algorithm.

The bag-of-words representation (ignoring numbers, case, and punctuation) for the above tweets are:

Vocabulary	welcome	to	machine	learning	knn	is	a	powerful	algorithm
T1	1	1	1	1	0	0	0	0	0
T2	0	0	1	1	1	1	1	1	1

Note: We only use the training set to construct the vocabulary for the BoW representation.

Tasks:

After you have preprocessed the data and created a bag of words, you have to implement the following tasks:

1. Create your own k-Nearest Neighbors classifier function by performing the following tasks:
   - For a test data point, find its distance from all training instances.
   - Sort the calculated distances in ascending order based on distance values.
   - Choose k training samples with minimum distances from the test data point.
   - Return the most frequent class of these samples. (Your function should work with Euclidean distance as well as Manhattan distance. Pass the distance metric as a parameter in the k-NN classifier function. Your function should also be general enough to work with any value of k.)
2. Implement an evaluation function that calculates the classification accuracy, F1 score, and the confusion matrix of your classifier on the test set.
3. Use 5- fold cross-validation on your training data. (In cross-validation, you divide the training data set into 5 parts. 4 parts will be used for training and 1 part will be used for validation. Then you will take a different part of your data as a validation data set and train your algorithm on the rest of the data set.) Run your k-NN function for this data for the values of k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. Do this for both the Euclidean distance and the Manhattan distance for each value of k. Formulas for both are listed below:

   Euclidean Distance:

   $$d(\vec{p}, \vec{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 + ... + (p_n - q_n)^2}$$

   Manhattan Distance:

   $$d(\vec{p}, \vec{q}) = |(p_1 - q_1)| + |(p_2 - q_2)| + |(p_3 - q_3)| + ... + |(p_n - q_n)|$$

4. For the even values of k given in the above task, break ties by backing off to the k-1 value. (For example, if you have k = 6 nearest neighbors and three of them have

the label 'positive' and three have the label 'negative, then you will break off the tie by taking k=5 nearest neighbors. On the other hand, let's say if you have k = 6 nearest neighbors where two have the label 'positive', two have the label 'negative', and two have the label 'neutral'. In that case, k =5 will still lead to two labels having a draw in which case you will continue decreasing k until there is a clear winner.)

5. Run your evaluation function for each value of k for both distance metrics, Report classification accuracy, F1 score, and confusion matrix.

6. Present the results as a graph with k values on the x-axis and classification accuracy on the y-axis. Use a single plot to compare the two versions of the classifier (one using Euclidean and the other using Manhattan distance metric). Make another graph but with the F1 score on the y-axis this time. The graphs should be properly labeled.

7. Comment on the best value of k you have found for both distance metrics using cross-validation.

8. Finally, use the best value of k for both distance metrics and run it on the test data set. Find the F1 score, classification accuracy, and confusion matrix and print them.

Part 2: k-NN classifier using scikit-learn (25 marks)

In this part, you have to use scikit-learn's k-NN implementation to train and test your classifier on the dataset used in Part 1. Repeat the tasks you have done in Part 1 but this time using scikit-learn. Use your bag of words to do cross-validation and run the k-NN classifier for values of k = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 using both Euclidean and Manhattan distance. Use scikit-learn's `accuracy_score` function to calculate the accuracy, `classification_report` to calculate macro-average (precision, recall, and F1), and `confusion_matrix` function to calculate confusion matrix for test data. Also present the results as a graph with k values on the x-axis and performance measures on the y-axis just like you did in part 1. Use a single plot to compare the two versions of the classifier (one using Euclidean and the other using Manhattan distance metric). Finally, print the best values of k for both distance metrics. Then use this value of k on the test data set and print the evaluation scores.