LAHORE UNIVERSITY OF MANAGEMENT SCIENCES
Syed Babar Ali School of Science and Engineering

EE514/CS535 Machine Learning
Spring Semester 2021

Programming Assignment 2 – k-Nearest Neighbors Classification

Issued: Tuesday 9th February, 2021

Total Marks: 100

Submission: 11:55 pm, Thursday 18th February, 2021.

Contribution to Final Percentage: 4%

Goal

The goal of this assignment is to get you familiar with k-NN classification and to give hands on experience of basic python tools and libraries which will be used in implementing the algorithm. You will learn the following:

- Feature engineering, i.e., extracting features from raw data using different techniques.
- Implementation of the k-NN algorithm from scratch.
- Classification of images using the k-NN algorithm.
- Evaluation of the performance of your classifier using different evaluation metrics such as confusion matrix, F1-score, accuracy.

Instructions

- Submit your code both as notebook file (.ipynb) and python script (.py) on LMS. The name of both files should be your roll number. Failing to submit any one of them will result in the reduction of marks.
- The code MUST be implemented independently. Any plagiarism or cheating of work from others or the internet will be immediately referred to the DC.
- 10% penalty per day for 3 days after due date. No submissions will be accepted after that.
- Use procedural programming style and comment your code properly.

Part 1: Implement k-NN classifier from scratch (35 Marks)

NOTE:

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.

Dataset:

The dataset contains 10,000 images of dogs and cats which have already been split (80 %, 20%) into training and test data. There are two top-level directories [test_set/, training_set/] corresponding to test set and training set respectively. Each of these directories further contains two directories [cats/, dogs/] comprising images of cats and dogs. The class labels of each of the images correspond to the directory they are contained in i.e., cat/dog.
Dataset: Dogs and Cats Images

Feature Extraction:

In the feature extraction step, firstly, you have to read the images which will give you a multi dimensional array containing RGB pixel intensities of the image. Using raw pixel values is the simplest way to create features from an image but for this part we will use HOG (Histogram of Oriented Gradients) feature descriptor to extract features from image data. The HOG descriptor focuses on the structure or shape of an object. It identifies if a pixel is an edge or not, as well as edge direction, by extracting the gradient and orientation (or magnitude and direction) of the edges. You can use skimage.feature library to extract HOG features from the image.

Tasks:

1. Create your own k-Nearest Neighbors classifier function by performing 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 k-NN classifier function. Your function should also be general enough to work with any value of k.)
2. Implement a evaluation function which calculates the classification accuracy, F1 score and the confusion matrix of your classifier on the test set. What significance does the F1 score hold, and why is it a better metric than accuracy?
3. Run your k-NN function for the values of $k = 1, 2, 3, 4, 5, 6, 7$. 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 $k-1$ value (Assume that you take the $k=4$ nearest neighbors of a particular image, and two of them have the label "cat" and the other two have the label "dog". In this case you will break tie by backing off to $k=3$ neighbors).
5. Present the results as a graph with k values on x-axis and F1 score on y-axis. Use a single plot to compare the two versions of classifier (one using Euclidean and the other using Manhattan distance metric). The graphs should be properly labelled.

Part 2: k-NN classifier using scikit-learn (20 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. Run the k-NN classifier again for values of $k = 1, 2, 3, 4, 5, 6, 7$ using both Euclidean and Manhattan distance. Use scikit-learn's accuracy_score function to calculate the accuracy, F1 score to calculate F1 score and confusion_matrix function to calculate confusion matrix for test data. Also present the results as a graph with k values on x-axis and F1 score on y-axis for both distance metrics in a single plot.

Part 3: Implement a k-NN classifier for a multi-class data set (45 marks)

Dataset:

You will be using the weather data set for this part, which can be found here. There are two top-level directories [Test_data/, Training_data/] corresponding to test set and training set respectively. There are 899 images in the training data and 224 images in the test data. Each of these directories further contains four directories [Cloudy/, Rain/, Shine/, Sunrise/] comprising images of relevant weather conditions. The class labels of each of the images correspond to the directory they are contained in i.e., Cloudy/Rain/Shine/Sunrise.

Feature Extraction:

In the feature extraction step, you have to read the images and then resize them to a fixed size (32, 32), which can be done using this function. After that, flatten the RGB pixel intensities of the images (which are multi-dimensional arrays obtained by reading the image) to a single list of numbers, i.e., a one dimensional array. Doing so, you will get a numpy array of shape (image_size*3,) for each image, which will serve as your feature vector for the particular image. You can use cv2 and numpy to implement these steps.

Tasks:

In this part you have to implement a generalized form of a k-NN classifier, which can classify a data set which has N classes. You have to repeat all the steps that you have implemented in Part 1, this time ensuring that all the steps are scaled up from a binary (classes $c = 2$) to a generalized form ($c = N$). Evaluation function should now provide macro average F1 score as an output output along with accuracy and confusion matrix.