CS412 Machine Learning, Spring 2024: Homework 1

March 6, 2024

Instructions

- Please submit your code as a zipped notebook. Name your submission as CS412-HW1-YourName-Code.zip, replacing 'YourName' with your actual first and last names.
- In preparing your notebook, please ensure that your code is well-organized and not confined to a single cell. Use separate cells for different sections of the code, explanations, and visualizations. This organization will help in making your notebook more readable.
- The responsible TA for this homework is Mansur Kiraz. But first ask your questions on the Homework Forum.
- You are required to code in Python. If the question is asking for you to implement, you should not use any libraries when in doubt about what to use or what not to use please write to the responsible TA (Mansur for this homework)
- If you are submitting the homework late (see the late submission policy in the syllabus), we will grade your homework based on the time stamp of submission and your remaining late days.

1 Is your nearest neighbor close to you in high-dimensional space?

Important Note: We strongly recommend writing your functions in a vectorized form using NumPy for efficiency. Vectorized operations significantly reduce computation time.

In question, we will explore the behavior of distance metrics in high-dimensional spaces, something that we discussed in class. You will plot *closest_to_average_ratio* for data points when the number of dimensions is varied and you will explore the pairwise distances for data points with low and higher dimensions. Follow these steps:

1. **Data Generation:** Write a function `random_points(num_points, num_dimensions)` to generate numbers from a standard Normal distribution, `num_points` random points in a `num_dimensions`-dimensional space.
2. **Average Distance of A Selected Point:** Write a function `average_distance_sp(points, selected_point)` to calculate the average distance of the `selected_point` to all other points in `points`.
3. **Minimum Distance of a Selected Point:** Write a function `minimum_distance_sp(points, selected_point)` to calculate the minimum distance of the `selected_point` and to all the other points in `points`.
4. **Closest-to-Average Ratio:** Write a function `closest_to_average_ratio(min_dist, avg_dist)` to calculate the ratio of the closest distance to the average distance.
5. **Experimentation and Visualization:**
   (a) Perform the following steps. For a given dimension d:

i. Generate 1000 d-dimensional data points, let's call this set S_d

ii. Select a random point from S_d and calculate the average distance, minimum distance, and closest-to-average ratio. Perform this for 100 different randomly selected points. Call this set A_d

iii. Aggregate the results and compute the mean and standard deviation across the A_d.

Repeat (i)-(iii) for all dimensions $d = 1, 2 \ldots 100$

(b) The mean average distance, the mean minimum distance, and the mean the closest-to-average ratio as line plots against the number of dimensions. Show the standard deviation of the distributions as well. The x-axis is varying d from 1 to 100. The y-axis is the metric value calculated.

(c) Plot a histogram of all pairwise distances for cases where d is 2, 5, 10, and 100 dimensions.

(d) Analyze, interpret, and discuss the trends observed in the plots.

2 k-NN Classifier In Action

2.1 Dataset

The **Fashion-MNIST** dataset is a collection of 28x28 grayscale images of 70,000 fashion products from 10 different categories, each represented by 7,000 images with each pixel value ranging from 0 to 255.

Figure 1: Samples from the Fashion MNIST

To download the Fashion-MNIST dataset, you will use the `Keras`[1] library. The dataset comprises 60,000 training samples and 10,000 test samples. You will split the training data into two sets: a development set for training your models and a validation set for testing the performance of your models during development.

You will reserve **20% of the training data for validation** and use the **remaining 80% for training** your models. Before splitting the data, ensure that it is shuffled to maintain the representativeness of both the training and validation sets. This step helps in preventing any bias that might be introduced due to the order of the data.

It is important to note that the official test set of 10,000 samples **should not be used for model selection or hyperparameter tuning during development**. This test set should only be used at the end of your project to evaluate the final performance of your chosen model.

2.2 Find the Best K

Note on Data Preprocessing for k-NN: The Fashion-MNIST images provided by Keras are in a 3D array format (samples, 28, 28), representing the number of samples and the dimensions of each image. However, the k-NN classifier implemented in the Scikit-learn library requires input data in a 2D array format (samples, features). Therefore, before training your k-NN model, you must reshape the dataset from 3D arrays into

[1] https://keras.io/api/datasets/fashion_mnist/

2D arrays. This process, often referred to as 'flattening ', converts each 28x28 image into a 784-dimensional vector.

Your task is to build **k-NN classifier**[2] using the `Scikit-learn` library. You will train the k-NN classifier using the training set and tune the hyperparameters to optimize its performance on the validation set. Specifically, you will determine the optimal number of nearest neighbors (referred to as n neighbors in the documentation) to use.

To find the optimal value of k neighbors, you should experiment with the following values: [**1, 3, 7, 12, 20, 30, 50, 75, 100**]. For each value, evaluate the performance of the classifier on the **validation set**. Print out the optimal value of k. Plot the validation accuracy for the different values of k neighbors, creating a graph where the x-axis represents the values of k neighbors, and the y-axis represents the corresponding validation accuracy. This visualization will aid in selecting the most suitable value for k neighbors. Utilize the `matplotlib` library to generate the plot.

Now that you have determined the optimal value of k neighbors, you should retrain the k-NN classifier by combining the training and validation sets. Then, assess its performance on the test set, and print out the test accuracy. This will provide an estimate of the test accuracy and how well your classifier is expected to perform on new, unseen data.

Finally, plot the confusion matrix for the final model's predictions across the ten classes. Discuss any patterns or insights you can draw from this visualization. For example, which classes have the highest number of misclassifications? Are there specific pairs of classes that are commonly mistaken for one another?

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`