AIML - CS 335

Lab 11: Clustering

October 20, 2022

Lecturer: Abir De

Important: Please read the instructions mentioned in the questions carefully. We have NOT provided any boilerplate code however some utility scripts have been provided to make your job easier. In this lab, you will be graded based on your algorithm's final performance.

1 K-Means

For the first question of this assignment, you are provided with a json file *points_4D.json*. This file contains a list of points in 4-dimensions. It is known that these points mostly lie in a 2-dimensional subspace of the 4-dimensional real space. For this part, you are free to use library implementations of techniques whenever available.

1.1 Dimensionality Reduction and Visualization (project.py)

Your first task is to find this subspace and project these points on it. Use a suitable technique to make this transformation and store the resultant 2-D points in *points_2D.json*. A helper function for this is available in *utils.py*.

The helper script *visualize.py* will plot a scatter plot of these transformed points for you and store it in *scatter_2D.jpeg*.

1.2 Kernel Selection and KMeans (cluster.py)

If you have completed your first task succesfully, you will easily observe k different clusters in *scatter_2D.jpeg*. Note the value of k - you will perform K-Means clustering with this k later.

Although, the clusters might NOT be linearly seperable. Your next task is to project these points further into a **finite kernel space**[1] where they will be linearly seperable. Use K-Means clustering in this kernel space to assign labels to the points.

Finally, output the value of k and the labels obtained (which should be $0 <= label[i] <= k-1$) in *labels.json*. A helper function for this is available in *utils.py*. Make sure that you use this function because it disambiguates permutations of label assignments.

We will use *scatter_2D.jpeg* and *labels.json* to grade your implementation.

[1] Yes, directly project the points in your code. We are not doing Kernel-KMeans in this question (:

2 Latent Clusters

In this question, you are provided with latent representations of 1000 MNIST digits, which were created using an auto-encoder. These 1000 images were randomly sampled from the MNIST dataset, and their labels are unknown to us. The only information provided to you is that the sampled images contained digits from all the 10 classes, i.e., we do not know how many images from a particular class were sampled. Your task is to cluster the representations of similar digits together. For this part, you are not to use library clustering functions.

The latent space representations of the points are provided in *mnist_samples.csv*. Code up a clustering algorithm in *main.py* and store the resulting labels using the helper function store_labels() from *utils.py*. Once again, we will use a suitable performance metric to grade your submission.

3 Submission instructions

You should write code only in the files specified. You should not create any new files or folders. After you are done, perform the following two operations :

```
mv rollno_L11 <ROLL_NUMBER>_L11
tar -zcvf <ROLL_NUMBER>_L11.tar.gz <ROLL_NUMBER>_L11
```

Replace `<ROLL_NUMBER>` with your own roll number. If your roll number has alphabets, they should be in *"small"* letters. Submit the tar file on Moodle. Happy coding and happy Diwali!