CSE 474/574: Introduction to Machine Learning (Fall 2021)

Checkpoint Date: October 24, Sunday, 11:59pm
Due Date: November 7, Sunday, 11:59pm

Sargur N. Srihari
University at Buffalo, The State University of New York
Buffalo, New York 14260
Contact: 716-645-6162 (O), srihari@buffalo.edu

October 10, 2021

1 Introduction

The task of the project is to perform unsupervised learning on Cifar 10 dataset. There are two task, the first is to perform K-means clustering on the raw data from scratch. The second is to perform K-means clustering on a representation generated by the Auto-Encoder method using library functions. The code should be written in Python using Keras.

2 Dataset

The Cifar 10 dataset has a training set of 50,000 examples, and a test set of 10,000 examples. Each example is a 32x32 image, associated with a label from 10 classes. Each image is 32 pixels in height and 32 pixels in width, for a total of 1024 pixels in total. This pixel-value is an integer between 0 and 255. The training and test data sets have 1025 columns including the labels.

3 Part 1: Implement K-Means Clustering[40 points]

Note : For Part 1, you are not allowed to use any Python libraries or built-in functions that directly perform K-Means Clustering. Use of ML libraries like sklearn will result in 0 points for the assignment.

3.1 Processing

K-means is based on the Euclidean distance. Use only the Cifar-10 dataset's test dataset for training the model. The implementation of K-means algorithm can be divided into four steps:

- Define the initial k clusters, and their centers, a = $a_1, a_2 ... a_k$.
- As for every sample x_i, calculate the distance between x_i and every cluster center, and classify x_i into the cluster possessing the shortest distance.
- As for every cluster a_j, re-calculate the cluster center, $a_j = \frac{1}{C_i} \sum x$
- Repeat step 2 & 3 to reach a terminate condition.

3.2 Tasks

1. Using K-Mean generate clusters and access the quality of the clusters using ASC (Average Silhouette Coefficient) and DI (Dunn's Index) evaluation metrics.

4 Part 2: Implement Auto-Encoder[40 points]

Note : For Part 2, you are allowed to use any Python libraries or built-in functions.

4.1 Processing

The general idea of Auto-Encoders is that we have to set an encoder and a decoder as neural networks and to learn the best encoding-decoding scheme using an iterative optimisation process. So, at each iteration we feed the autoencoder architecture (the encoder followed by the decoder) with some data, we compare the encoded-decoded output with the initial data and back-propagate the error through the architecture to update the weights of the networks. Use only the Cifar-10 dataset's train dataset for training the model

4.2 Tasks

1. Use Auto-Encoders to generate sparse representations of the input image.
2. Using K-Mean generate clusters from the sparse representations generated by the Auto-Encoders and access the quality of the clusters using ASC (Average Silhouette Coefficient) and DI (Dunn's Index) evaluation metrics.

5 Deliverables

There are two deliverables: report and code. After finishing the project, you may be asked to demonstrate it to the TAs, particularly if your results and reasoning in your report are not clear enough.

1. **Report (20 points)**

   The report should be delivered as a separate pdf file, and it is recommended for you to use the NIPS template to structure your report. You may include comments in the Jupyter Notebook, however you will need to duplicate the results in the report. The report should describe your results, experimental setup and comparison between the results obtained from different setting of the algorithm and dataset.

2. **Code (80 points)**

   The code for your implementation should be in Python only. The name of the Main file should be main.ipynb or main.py. Please provide necessary comments in the code.

6 Checkpoint Submission[Due Date: 24th October]

You only need to submit part 1 for the checkpoint submission to receive feedback and it will not be graded. Your Python code and Report should be packed in a ZIP file named `UBIT name_assignment1_checkpoint.zip(e.g. soumyyak_assignment1_checkpoint.zip)` and upload it to UBlearns in the Assignments section.

7 Final Submission[Due Date: 7th November]

You need to submit both part 1 and part 2 for the Final submission. Your Python code and Report should be packed in a ZIP file named `UBIT name_assignment1_Final.zip(e.g. soumyyak_assignment1_Final.zip)` and upload it to UBlearns in the Assignments section.