PROJECT REPORT

K-MEANS CLUSTERING AND AUTO-ENCODER ON CIFAR-10 DATASET

SOUMIL GHOSH

EMAIL – SOUMILGH@BUFFALO.EDU

Introduction

Our task in this project was to perform K-Means Clustering on the CIFAR-10 dataset in two ways. In the first part, we applied K-Means clustering directly on the test images of CIFAR-10 dataset. In the second part, we used Auto encoder to generate sparse representation of the train images of the Cifar-10 dataset and applied K-Means Clustering on these sparse Representations which were generated.

Dataset

In this project we worked on the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60000 32*32 images each of which belong to either of the 10 classes namely,

- Airplane
- Automobile
- Bird
- Cat
- Deer
- Dog
- Frog
- Horse
- Ship
- Truck

Out of these 60000 images, there are 50000 images available for training and 10000 images available for testing.

Dataset Preprocessing

At first, we load the CIFAR-10 dataset from Datasets module of Keras. For implementing K-Means from scratch, we use the 10000 test images of the Dataset. For implementing Auto-Encoder and clustering the Sparse Representations using K-Means, we use the Train dataset of CIFAR-10. In both cases, first we must first convert the 3-diamensional image matrices to 1-Diamensional matrices. So, all the images are converted to matrices of size 3072*1 from matrices of size 32*32*3. Therefore, in the first case we get a train matrix of size 10000*3072 and in the second case we get a train matrix of size 50000*3072. Our next task is to normalize the train matrix. We do this by dividing each matrix element by 255. Now, we are prepared for training our respective models.

Methodology

K-Means Clustering

We are using only the test dataset of Cifar-10 for implementing K-Means Clustering. For implementing K-Means Clustering, at first, we must first assign 10 random centroids. For this purpose, we are first initializing 10 random indices over our dataset and then we are assigning the initial centroids from our training data using these random indices.

Steps for K-Means clustering:

1. We are iterating 50 times to ensure that better clusters are formed
2. In each iteration, we are first iterating over the entire dataset and calculating the Euclidian Distance of the data point from each of the centroids using *"np.linalg.norm"*. Then, we are finding out the centroid for which the distance is minimum and assigning the data point to the cluster corresponding to that centroid.
3. Next step is to calculate the new centroids. For this we are iterating over the list of clusters and for each of the clusters, we are finding out the data-points belonging to that cluster. Then for each cluster we calculate the new centroid and append it to the centroids updated matrix.

Auto-Encoder

Part-1- Generating Sparse Representations of Images

We have used 4 hidden layers with *3072, 1024, 512* and *128* neurons respectively to design the Encoder part of our Auto-Encoder model. In all these hidden layers*, RelU* activation function has been used.

To design our Decoder, we have used 2 hidden layers with *512* and *1024* neurons respectively. The decoder layers also have *RelU* as their activation function.

The last layer is the output layer containing *3072* neurons. We have used linear activation function in our output layer. We have used *Stochastic Gradient Descent* with a learning rate of *0.1* to better optimize our model. For loss evaluation, we have used *Mean Squared Error* loss function.

We trained our model for 10 epochs with a batch size of 64.

Part-2 – Performing K-Means on the Sparse Representations generated by the Auto-Encoder

For performing K-Means on the sparse representations generated by the Auto-Encoder we apply the K-Means Clustering module available in the Sklearn library on the output of layer 4 of our model which marks the end of the Encoder in our Auto-Encoder model.

Results

Evaluation Metrics

We used Silhouette Co-efficient and Dunn's Index as Evaluation Metrics. For calculating Silhouette Co-efficient, we used the Silhouette-Score and Silhouette-Samples modules of Sklearn library and for calculating Dunn's Index, we used the Validclust library.

K-Means Clustering

On applying K-Means clustering directly on the Cifar-10 images, we achieved Average Silhouette Co-efficient of *0.058* and Dunn's Index value of *0.114*.

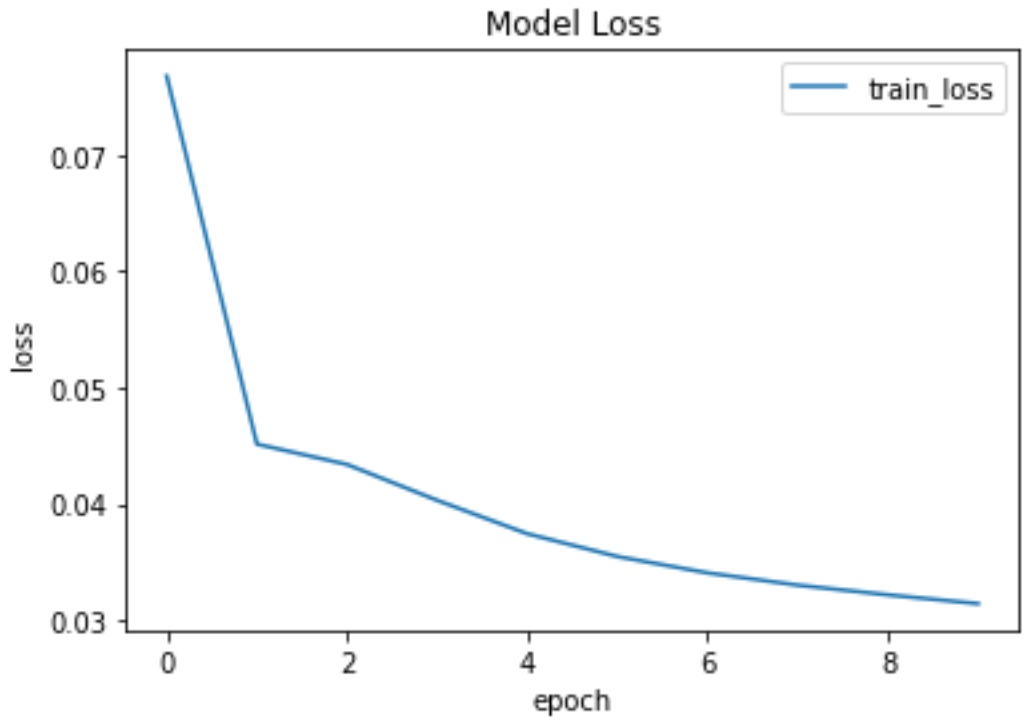
Auto-Encoder

After training our model for 10 epochs, we achieved a training loss of *0.0314*.

On applying K-Means clustering on the Sparse Representations of the Cifar-10 images generated by the Auto-Encoder, we achieved an Average Silhouette Coefficient score of approximately *0.0181*.

On comparing the results above, it is clear that Clustering on the Sparse Representations of the images was much better compared to the original images.

The loss curve for training the Auto-Encoder is shown below:



Conclusion

Through this project it became very clear to us that while performing Unsupervised Learning tasks, like Clustering using K-Means Clustering Algorithm, it is always better to first extract the necessary features and then perform K-Means clustering.

The raw images contain a lot of pixel data which are not useful in performing Clustering, these features instead confuse the algorithm and lead to bad results. Instead, if we extract only the necessary features and then perform Clustering, we can get much better results even if we use a larger dataset.