Assignment
Implementation of K mean clustering

Akash Sharma
UBIT-as475
as475@buffalo.edu
Department of Computer Science and Engineering
University at Buffalo
Buffalo, NY 14214

14 November 2021

1 Introduction

1.1 K Mean clustering

K mean clustering is a algorithm use for unsupervised data for machine learning. It takes inference from the input data without any output data.It finds the similar points and make a cluster of points with the help of centroids. K 'means' referring to the average of data which means finding the centroids.

1.2 Data set and its features

Cifer 10 dataset consist of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Dataset has 5 training batch and one test batch each with 10000 images. Classes of dataset consist of following
1. Airplane
2. Automobile
3. Bird
4. Cat
5. Deer
6. Dog
7. Frog
8. Horse
9. Ship
10. Truck

All classes are mutually exclusive.

2 Steps to implement K Mean clustering

1. Load the Cifer-10 data and processed according to requirement.
2. Select the centroids randomly.
3. Assign the each to the closest cluster by calculating euclidean distance of each point with the center.
4. Choose the centroids corresponding to minimum distance.
5. Repeat the steps 3 until the new centroids do not change.
6. Find the Silhouette score and Dunn Index to test the accuracy.

3 Explanation

3.1 Accessing the Libraries

```
[16] #IML Project 2
     import tensorflow as tf
     tf.keras.datasets.cifar10
     from keras.datasets import cifar10
     from matplotlib import pyplot as plt
     import numpy as np
     import random as rd
     from matplotlib.pyplot import cm
     from sklearn.metrics import silhouette_samples,silhouette_score
     from sklearn.preprocessing import StandardScaler
     from validclust import ValidClust
     from validclust import dunn
     from sklearn.metrics import pairwise_distances
```

Libraries are loaded according to the requirement of the function used in program.

3.2 Loading the data

```
def load_data():
    (Train_X, Train_Y), (Test_X, Test_Y) = cifar10.load_data()
    Train_X = Train_X.astype('float32') / 255
    Test_X = Test_X.astype('float32') / 255
    #Flattening the images
    Train_X = Train_X.reshape((-1, 3072))
    Test_X = Test_X.reshape((-1, 3072))

    return Test_X
```

Above method is used for loading the data from the Cifer-10 database which consist of 60000 images with 10 features. After the loading, data should be processed so that it can be use in the program.

3.3 Methods used to calculate the K mean

3.3.1 KMean Method

```
def Kmean():
    n_clusters=10
    iteration=100

    return n_clusters, iteration
```

Kmean method is used to assign the number of cluster and the value of iteration. Cluster value and iteration value can be updated with the help of this method.

3.3.2 Update cluster method

```
def update_clusters(Train_X, centroids):
  clus_group = []
  distances= []
  #SSE = []

  for r in Train_X:
    sum =0
    for centroid in centroids:
      distances.append(np.sqrt(np.dot(r-centroid,r-centroid)))
    less_distance = min(distances)
    index_pos = distances.index(less_distance)
    clus_group.append(index_pos)
    distances.clear()


  return np.array(clus_group)
```

Above Update cluster method is used to find the distances between the centroids and the data points. Distance is calculated as a euclidean distance and it is calculated row by row , from every centroid to every data point. After the distance calculation less distance opted to assign the closest cluster and indexes of the distance which is near to the cluster listed in list(clust_group).

3.3.3 Change Centroids method

```
def change_centroids(Train_X, clus_group):
  n_centroids=[]
  cluster_no=np.unique(clus_group)

  for no in cluster_no:
    n_centroids.append(Train_X[clus_group == no].mean(axis=0))

  return np.array(n_centroids)
```

Above Change centroids method is used assign centroids by calculating the means which comes from the indexes from cluster group list.

3.3.4 Predict method

```
def predict():
  Train_X=load_data()
  n_clusters, iteration=Kmean()
  #rd_index=rd.sample(range(0,Train_X.shape[0]),n_clusters)
  #print(rd_index)
  #centroids=Train_X[rd_index]
  centroids=Train_X[[8509, 9991, 9992, 9993, 9994, 9995, 9996, 9997, 9998, 8999]]

  for i in range(iteration):
    clus_group=update_clusters(Train_X, centroids)
    old_centroids=centroids
    centroids=change_centroids(Train_X, clus_group)
    if(old_centroids==centroids).all():
      break

  print("silhouette_score")
  print(silhouette_score(Train_X, clus_group))
  dist = pairwise_distances(Train_X)
  print("Dunn Index")
  print(dunn(dist, clus_group))
```

Above predict method is used to load data and assign first values of centroid. This method is used to iterate to get the centroid value and after calculating the final centroids, silhouette score and Dunn index can be calculated.

4 Result

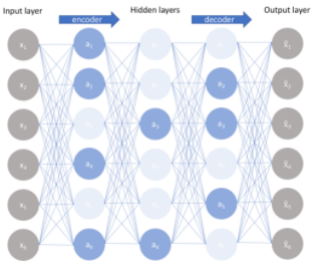
silhouette_score=0.058223516
Dunn Index=0.11481418

5 Part:2 Implementation of Autoencoder

5.1 What is Autoencoder?

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible.

5.2 Architecture of Autoencoder



Above layer is architecture of autoencoder. It is divided into 3 layers. Input layer as Encoder, hidden layer as code and the outout layer as Decoder. They are Encoder, Decoder, and Code. The encoder and decoder are completely connected to form a feed forwarding mesh—the code act as a single layer that acts as per its own dimension. To develop an Autoencoder, you have to set a hyperparameter, you have to set the number of nodes in the core layer. The decoder's output network is a mirror image of the input encoder in a more detailed manner. The decoder produces the desired output only with the help of the code layer.

5.3 Steps to implement Autoencoder

1. Load the Cifer-10 data and processed according to requirement.
2. Divide the data into input layer as encoder , encoded as encoded layer and output layer as decoder.
3. Compile the input and output layer by calling model function.
4. Training the autoencoder by training images and predict the test images by test images
5. Calculate the kmean and silhouette score on encoded image.

5.4 Input and Output images

In above images first line shows the input images of the test data and second line shows the decoded images which is reconstructed in the autoencoder.

6 Result

silhouette_score=0.0465229
Silhouette score has a range between .01 to .09

7 References

1. https://www.cs.toronto.edu/~kriz/cifar.html
2. https://medium.com/analytics-vidhya/machine-learning-algorithm-k-nearest-neighbors-and
3. https://validclust.readthedocs.io/en/latest/validclust.html
4. https://medium.com/@cmukesh8688/silhouette-analysis-in-k-means-clustering-cefa9a7ad111
5. https://mayankdw.medium.com/k-means-clustering-and-dunn-index-implementaion-from-scrat
6. https://www.youtube.com/watch?v=MFraC1JObUo&t=1735s
7. https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-
8. https://paperswithcode.com/method/sparse-autoencoder#
9. https://www.educba.com/autoencoders/
10. https://www.youtube.com/watch?v=m2AyljDHYes&t=988s