Unsupervised Learning using ML
(K-means Clustering and Autoencoder)

Rushabh Shah
(50375759)

1 Abstract:

The goal of this project is to train a machine learning model using unsupervised learning. For the stated purpose

1) We use the technique of K-means Clustering and use it on the Cifar-10 dataset to group the examples most similar to each other. The problem takes 60000(32x32) images as input.
2) We perform K-means Clustering on sparse representations generated by the Auto-Encoder

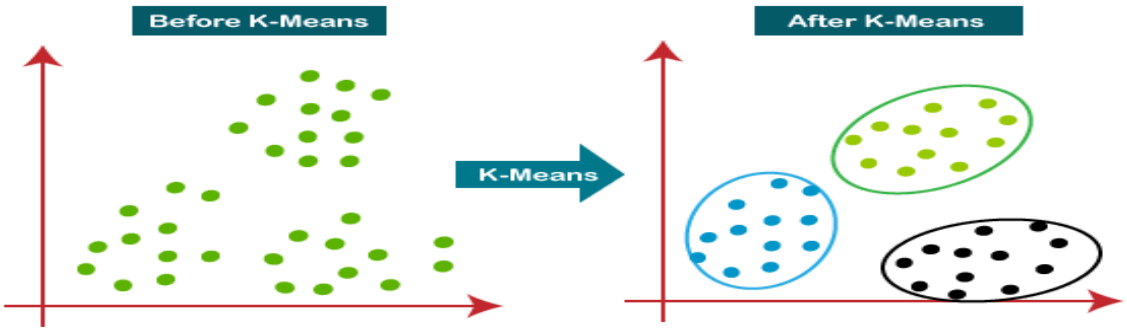
2 Dataset:

Cifar-10 dataset. The total number of training examples are 60000(32x32) in the form of images.

3 Introduction

3.1 K-Means Clustering

K-Means is a powerful technique to perform unsupervised learning. K-Means groups samples which are most similar to each other. The groups are formed in the form of clusters centered around a centroid. Number of centroids can vary. K-means algorithm works without referring to the known samples or labels and hence it is classified as an unsupervised learning technique.



Applications:

1) Diagnostic systems
2) Search engines
3) Sensor networks
4) Fraud detection

3.2 Autoencoder

Autoencoder is an unsupervised learning technique where we use neural networks to encode and decode images. In autoencoders, we map input to an output using an intermediate representation.

Some properties of autoencoders:-

1) Data specific
2) Lossy
3) Different for different types of data

The aim of an autoencoder is to learn a lower-dimensional representation, typically for dimensionality reduction, by training the network to capture the most important parts of the input image.

Types of autoencoders:-

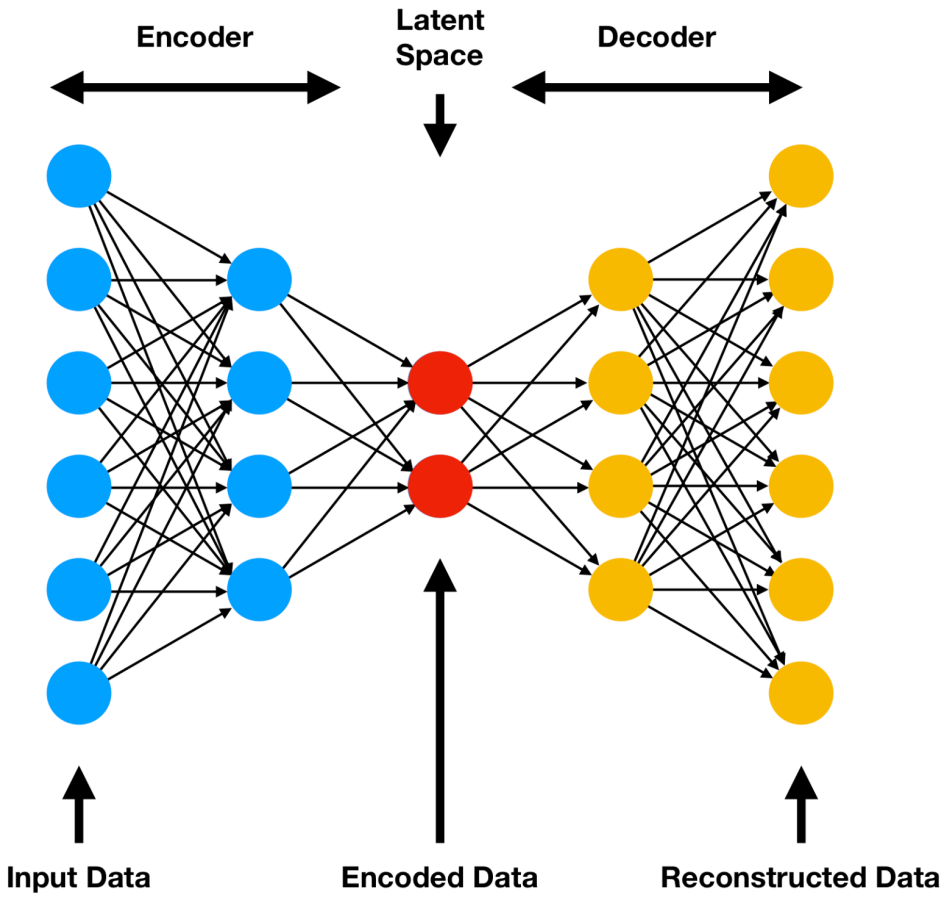
1) Sparse Autoencoder
2) Denoising Autoencoder
3) Deep Autoencoder
4) Convolutional Autoencoder
5) Undercomplete Autoencoder
6) Regularized Autoencoder
7) Stochastic Autoencoder

Autoencoder architecture

Encoder: A module that compresses the input data into sparse representation.

Code: A module that contains the latent representations.

Decoder: A module to “decompress” the latent representations and reconstruct the data back.



Applications:

1) Dimensionality Reduction
2) Denoising
3) Compressing images
4) Feature Extraction
5) Image generation

4 Implementation

4.1 Preprocessing

The dataset used in this project is Cifar-10. The total number of features are 1024 each representing a pixel in the range of 0 to 255 :-

The data has been divided into train and test datasets with 50000 and 10000 examples respectively.

In this project, we make use of the test dataset for implementing K-means from scratch and training dataset for autoencoders.

Initially, the data has been converted to grayscale from RGB using cv2 library. Also, we scale the data to a normalized scale to get better results. Hence, the input data after preprocessing is of the form 10000x1024.

4.2 K-Means Implementation

The steps performed in K-Means algorithm are as follows:-

1) Define the initial K clusters and randomly select centroids for each. For each data point x, calculate the euclidean distance between the data point and each centroid. The euclidean distance is given by -

   D = sqrt((X1 - X2)**2)
2) After finding the euclidean distance, assign the data point to the cluster based on the minimum distance to all the centroids.
3) Adjust the centroid of each cluster by taking the mean of all the data points belonging to the cluster.
4) Repeat the process until convergence is achieved or desired result is obtained.

In order to break out of the loop, we use a technique as follows

If the absolute difference between the current and the previous centroids falls below 0.003 which means that the positions of centroids are not changing much and hence we break out of the loop.

Results:

In this project, we have made use of two evaluation metrics ASC(Average Silhouette Coefficient) and Dunn's index to assess the quality of the clusters.

Dunn’s index:-
It is calculated as the lowest intercluster distance (ie. the smallest distance between any two cluster centroids) divided by the highest intracluster distance (ie. the largest distance between any two points in any cluster).

ASC:-
The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample.

On running the algorithm, the ASC and Dunn's index values are 0.06017 and 0.089 respectively.

```
[38]
      dist = pairwise_distances(X_test)
      dunn(dist, points)

      0.08923394585125889

[39]  silhouette_score(X_test, points)

      0.060171946112711774
```

4.3 Autoencoder

We start by preprocessing, first by converting the data into grayscale and then normalizing.

Then the autoencoder model is built using a Convolutional neural network. Here, the encoded images are in the form of sparse representations of size 4x4x16 which is flattened to a vector of length 256. The flatten image is reshaped back to 4x4x16 and up-sampling is done to get the reconstructed images.

The filter size used is 3 x 3 and the activation functions used is “Relu” for all the layers except for the output layer where “Sigmoid” function is used.

A model named “encoder” is constructed using the input and flattened images as the output. This model, which gives sparse representations as output, is used to train the K-Means model.

Model Used: Sequential model is used for training the Autoencoder

Optimizer and Loss function: Adam optimizer and Mean squared error loss function

Autoencoder Architecture:-

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 64)        640

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 32)        18464

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 16)          4624

 flatten (Flatten)           (None, 256)               0

 reshape (Reshape)           (None, 4, 4, 16)          0

 conv2d_3 (Conv2D)           (None, 4, 4, 16)          2320

 up_sampling2d (UpSampling2D  (None, 8, 8, 16)         0
 )

 conv2d_4 (Conv2D)           (None, 8, 8, 32)          4640

 up_sampling2d_1 (UpSampling  (None, 16, 16, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 16, 16, 64)        18496

 up_sampling2d_2 (UpSampling  (None, 32, 32, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 32, 32, 1)         577

=================================================================
Total params: 49,761
Trainable params: 49,761
Non-trainable params: 0
```

Plot for input images and decoded images

Plotting images

```
n = 10
plt.figure(figsize=(16, 3))
for i in range(n):

    ax = plt.subplot(2, n, i + 1)
    plt.imshow(X_train[i].reshape(32, 32))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)



    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_images_train[i].reshape(32, 32))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```

Results:

Evaluation of the model is done based on ASC.

ASC:-
The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample.

The silhouette score for Test images and Train images is `0.05522129430544721` and `0.05674460927257531` respectively.

References:

1) https://www.coursera.org/learn/machine-learning/home/welcome
2) https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/
3) https://en.wikipedia.org/wiki/K-means_clustering#Description
4) https://dzone.com/articles/10-interesting-use-cases-for-the-k-means-algorithm
5) https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html
6) https://keras.io/api/optimizers/
7) https://keras.io/api/layers/convolution_layers/convolution2d/

8) https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html