# Assignment 2 – K-Means Clustering and Auto Encoding

Vishal Patil

14th Nov 2021

## 1.Assignment overview:

Goal of the assignment is to apply KMeans to cluster data based on certain features. For this purpose, I have made a KMeans algorithm without using libraries to cluster the image data.

## 2. Python Editor:

Jupyter Notebook IDE was used for all the coding implementation.

## 3. Data set:

CIFAR10 dataset is used with 10000 images in the test set and 50000 images in the train data set. The Image size is 32x32 pixels for each image.

## 4. PreProcessing of the Data:

Features based on which the clustering of the unlabelled dataset would take place would the pixel values of the images. So basically there would be 32x32 = 1024 features for the KMeans clustering algorithm to consider.

In preprocessing, each image in the data set is changed to a 1024 array. So the train data set would be an array of 50000x1024 values and the test data set would be an array of 10000x1024 values.

These values are then normalized by dividing each pixel value by 255, which is the max value of the pixel making the range of pixel values between 0 and 1. Images are also converted to Grayscale to reduce the amount of redundant information.

## 5. Implementation:

## 5.1 Part I: KMeans clustering:

Overview: Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way kmeans algorithm works is as follows:

Specify number of clusters K (10 in our case).

Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.

Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

Compute the sum of the squared distance between data points and all centroids.

Assign each data point to the closest cluster (centroid).

Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The score of the KMeans clustering for this case is calculated by the ASC Index and the Dunn's Index, results of which are posted below.

## 5.2 Part 2: Autoencoder:

An autoencoder is an Artificial Neural Network used to compress and decompress the input data in an unsupervised manner. Compression and decompression operation is data specific and lossy. The autoencoder aims to learn representation known as the encoding for a set of data, which typically results in dimensionality reduction by training the network, along with reduction a reconstruction side is also learned.

For the implementation, I have added 4 Dense layers in the encoder, with nodes 1024 in the first layer, 512 nodes in the second layer, 256 nodes in the 3rd layer and 32 nodes in the bottleneck layer. The decoder layer has 3 layers in the opposite order of appearance from that of the encoder except the bottleneck layer. The decoder output is a 1024 value array which can be rearranged to a 32x32 image after reversing the steps done to the image during the pre-processing step.

KMeans clustering, done in the first part of the assignment, on the 1024 feature space, is now done for the encoded data output which is a reduced (compressed) feature space. The score of the KMeans clustering for this case is calculated by the ASC Index, results of which are posted below.

# 6. Results:

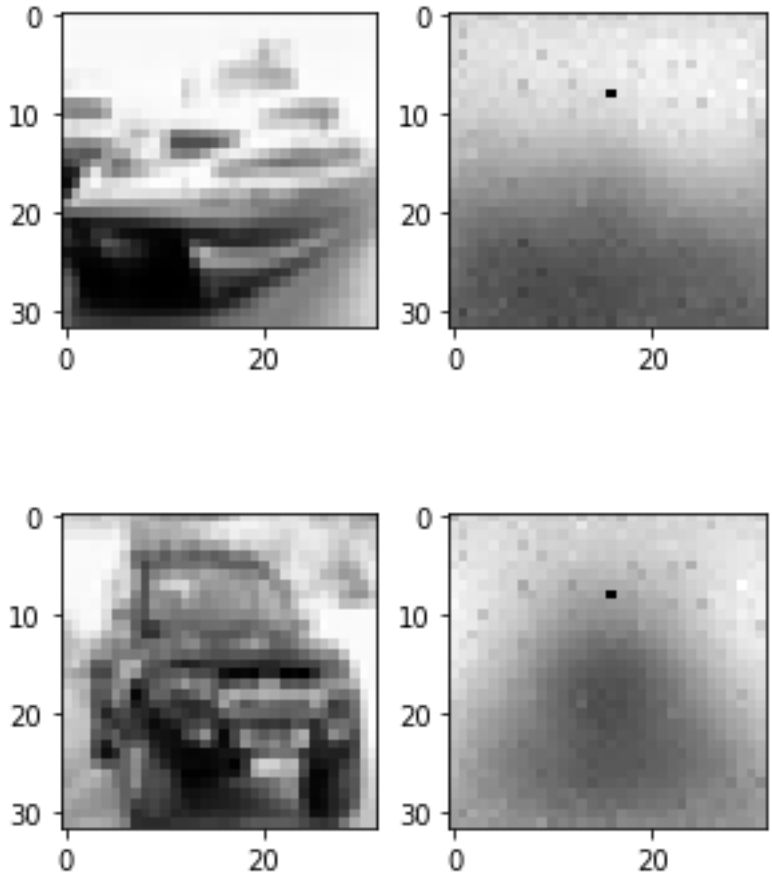
Part 1:

ASC Index: `0.0608802288657418`
Dunn's Index: `2.737724171154885`

Part 2:

ASC Index: `0.16624467`

`Input Image:` `Decoder output Image:`


0
10
20
30
0
20
0
10
20
30
0
20