Assignment 2: - K-Means Clustering and Auto-Encoder

Rishi Jay Joshi
Department of Computer Science and Engineering
University at Buffalo, SUNY
New York, United States
rishijay@buffalo.edu

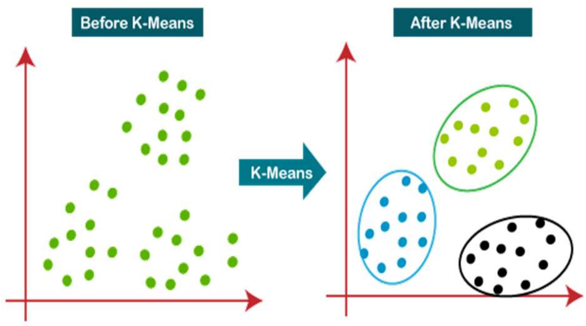
Abstract

Part 1 of the assignment deals with implementing K-Means clustering on the CIFAR-10 image dataset. K-Means is an unsupervised clustering algorithm that assigns data points to their respective clusters. Part 2 of the assignment deals with implementing Autoencoders. Autoencoders are neural networks that are used for compression of data.

1 Introduction

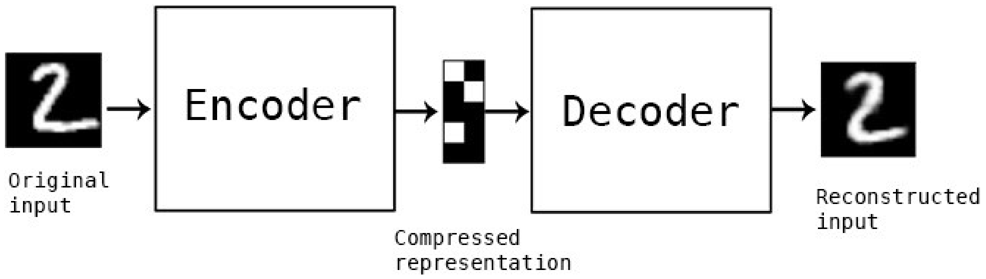
1.1 K-Means

[1] K-Means is one of the most popular "clustering" algorithms. K-means stores k centroids that it uses to define clusters. A point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid. K-Means finds the best centroids by alternating between (1) assigning data points to clusters based on the current centroids (2) chosing centroids (points which are the center of a cluster) based on the current assignment of data points to clusters.



1.2 Autoencoders

[3]"Autoencoding" is a data compression algorithm where the compression and decompression functions are 1) data-specific, 2) lossy, and 3) learned automatically from examples rather than engineered by a human. Additionally, in almost all contexts where the term "autoencoder" is used, the compression and decompression functions are implemented with neural networks.

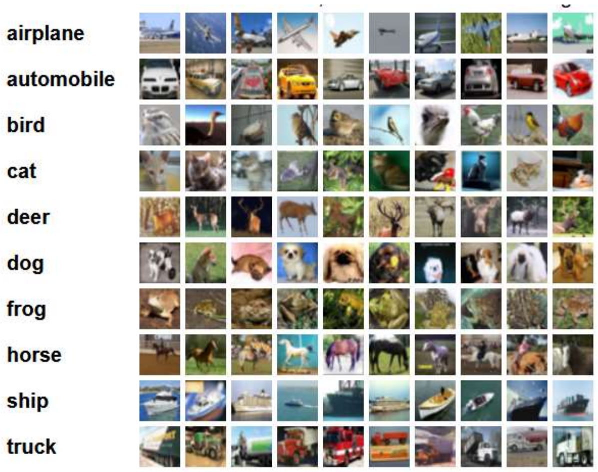


- Autoencoders are data-specific, which means that they will only be able to compress data similar to what they have been trained on.
- Autoencoders are lossy, which means that the decompressed outputs will be degraded compared to the original inputs
- Autoencoders are learned automatically from data examples, which is a useful property: it means that it is easy to train specialized instances of the algorithm that will perform well on a specific type of input. It doesn't require any new engineering, just appropriate training data.

2 Experiment

2.1 Dataset

CIFAR-10 dataset has been used for this assignment. [4]The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.



2.1 Python Editor

Google Colab based on Jupyter Notebook is used for implementation

2.3 Data Preparation

We first download the CIFAR-10 dataset using tensorflow and split it into training and testing. We then use CV2 to convert the images into grayscale. After the images have been converted into grayscale, we normalize the images.

2.4 K-Means Clustering

For Part 1, we have implemented K-Means Clustering on the CIFAR-10 dataset. The K-Means algorithm is as follows:-

- Define the initial k clusters, and their centers, a = a1, a2,…,ak.
- As for every sample xi, calculate the distance between xi and every cluster center, and classify xi into the cluster possessing the shortest distance.
- As for every cluster aj, re-calculate the cluster center, aj
- Repeat step 2 & 3 to reach a terminate condition.

[2] Given a set of observations (x1, x2, ..., xn), where each observation is a d-dimensional real vector, k-means clustering partitions the n observations into k (≤ n) sets S = {S1, S2, ..., Sk} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where $\boldsymbol{\mu}_i$ is the mean of points in Si. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \left\| \mathbf{x} - \mathbf{y} \right\|^2$$

The equivalence can be deduced from identity

$$\sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)^T (\boldsymbol{\mu}_i - \mathbf{y}).$$

Because the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in different clusters

2.5 Autoencoders

For Part 2, we have used autoencoders for sparse representation of our images. We have then used the autoencoder prediction on the K-Means Clustering algorithm. Autoencoders have been implemented using the Keras library.

[5] Our autoencoder has two parts: an encoder that maps the input to the code and a decoder that maps the code to a reconstruction of the input . An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions such that:-

$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$
$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$
$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \left\| \mathcal{X} - (\psi \circ \phi)\mathcal{X} \right\|^2$$

3 Results

3.1 Part 1 :– K-Means

After successfully running our K-Means Algorithm, we get the following Silhouette Score

```
Silhouette Score(n=10): 0.05105886980891228
```

And we get the following Dunn’s Index:-

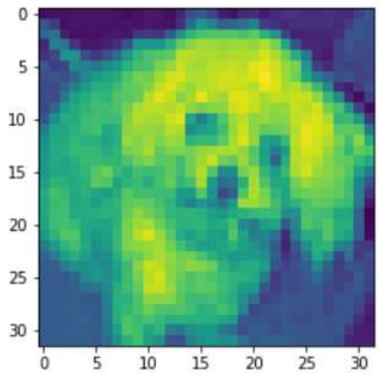
0.08936835

3.2 Part 2 :– Autoencoder

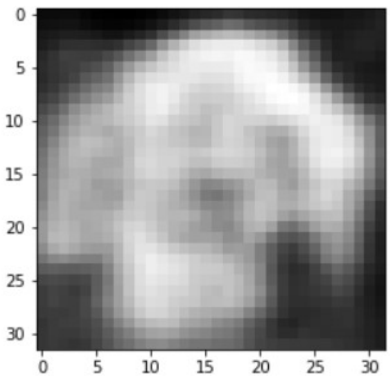
The model summary of our network is as follows:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32, 32, 1)]       0

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 256)               262400

 dense_1 (Dense)             (None, 1024)              263168

 reshape (Reshape)           (None, 32, 32, 1)         0

=================================================================
Total params: 525,568
Trainable params: 525,568
Non-trainable params: 0
_________________________________________________________________
```

After successfully training our autoencoder on our original images we get the sparse representation. An example is shown:-



The above image is the original image that is used for training. Shown below is the sparse representation of the image that was generated



After passing our predicted output into SKlearns's K-Means algorithm we get the following silhouette score:-

```
Silhouette Score(n=3): 0.05319574102759361
```

4 References

[1] https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

[2] https://en.wikipedia.org/wiki/K-means_clustering

[3]https://blog.keras.io/building-autoencoders-in-keras.html

[4] https://www.cs.toronto.edu/~kriz/cifar.html

[5] https://en.wikipedia.org/wiki/Autoencoder