# UNSUPERVISED MACHINE LEARNING CLUSTERING ON FASHION MNIST DATASET

**Aashna Mahajan**
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
*aashnama@buffalo.edu*

## Abstract

The purpose of this project is to create the model which efficiently recognizes the image. The image is recognized as one of the clusters and this is achieved by applying KMeans clustering and Auto-encoding algorithm which helps in classifying the data by creating clusters. The number of clusters, epochs and batch size affect the accuracy of the model, hence modifications in them can help us in reaching the desired output in the experiments.

## 1 Introduction

### 1.1 Unsupervised Learning

Unsupervised Machine learning contains only the input and no labelled output. The model learns about the data from the structure of the data. Different patterns are found out from the dataset without the labels or references to the outputs or the outcomes, which help in determining the results. Unsupervised learning deals with clustering and association problems.

### 1.2 KMeans Clustering

The KMeans algorithm is an unsupervised algorithm used to cluster the data in n groups or clusters with almost equal variance. In the algorithm, we try to minimize the inertia and the sum-of-squares.

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

Figure 1: Computing minimal sum-of-squares

We specify the number of clusters using n_clusters parameter. The init parameter defines the method for initialization which is set to random since we initially take n centroids randomly. In the k-means algorithm we initially use the randomly initialized centroids to form clusters. The n_init defines the number of times that we run the k-means algorithm for different centroids. The algorithm parameter helps us specify the KMeans algorithm to be used. We keep changing the clusters according to the mean values achieved from each cluster. We stop the process once the centroids of the clusters do not change. The algorithm works for large dataset and can also be used for a large scale of applications.

### 1.3 Auto-Encoder

Autoencoding is an algorithm used for data compression. The auto-encoder learns about the compression and decompression functions which are implemented with neural networks. It is a types of feedforward neural network where the input and the output are the same. An encoding function, a decoding function, and a distance function is used to build an encoder. The amount of information loss between the compressed representation of your data and the decompressed representation that is the loss function is calculated using the distance function. Auto-Encoders are Data-specific (only able to meaningfully compress data similar to what they have been trained on) , lossy (The input and output of the autoencoder will not be exactly the same, it will be a close but a degraded representation) and unsupervised (Auto-encoders don't need explicit labels).

#### 1.3.1 Auto-Encoder with KMeans Clustering

The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion. The measure of internally coherent clusters are recognized by Inertia. Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very highdimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called curse of dimensionality). Running a dimensionality reduction algorithm such as Principal component analysis (PCA) or Auto-encoder prior to k-means clustering can alleviate this problem and speed up the computations.

#### 1.3.1 Auto-Encoder with GMM Clustering

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. The Gaussian Mixture object implements the expectation-maximization (EM) algorithm for fitting mixture-of-Gaussian models. It can also draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. A Gaussian Mixture fit method is provided that learns a Gaussian Mixture Model from train data. Given test data, it can assign to each sample the Gaussian it mostly probably belong to using the Gaussian Mixture predict method.

## 2 Dataset

The Fashion-MNIST dataset is used for training and testing the clustering models. The Fashion-MNIST is a dataset of which consists of a training set of 60,000 examples images and a test set of 10,000 examples images. Each image is a 28x28 grayscale image. Each image in the dataset is of 28 pixels in height and 28 pixels in width which results in a total of 784 pixels. Each pixel indicates the lightness or darkness of that pixel. The higher numbers indicate darker. This pixel-value is an integer between 0 and 255. The training and test data sets will have 784 columns. The Fashion MNIST dataset can be imported using the keras library.

## 3 Pre-processing

Data pre-processing is an important step before feeding the data into the real model. It cleans the data and turns it into an enhanced form from the raw data. The pre-processed data makes it easier for the model to learn.

### 3.1 Import libraries

Libraries consist of built-in modules which help in simplifying the functionalities. Python provides a great set of libraries which increase the productivity of the developers. Libraries such as Numpy, SkLearn, Matplotlib, Keras are imported.

### 3.2 Partitioning the Dataset

The dataset is partitioned into the training set that has 60000 images and the testing set which has 10000 images. Then cross validation is performed on the set to get validation set and the testing set with 50 : 50 ratio. The dataset can be split or partitioned using the **train_test_split** function. The dataset is then concatenated since we do not require the test data, since we are working on unsupervised learning.

### 3.3 Normalization

Normalization is a pre-processing method which normalizes the data i.e. brings the uneven or varied ranged data to a common scale without actually distorting the difference in the range of the values. The redundant data can be eliminated and good quality clusters can be generated using Normalization of the data. This further improves the efficiency of the clustering algorithms.

## 4 Architecture

Computational graphs are useful in representing mathematical equations graphically. Following figure is the computational graph of KMeans clustering and Auto-Encoders.

### 4.1 KMeans Clustering

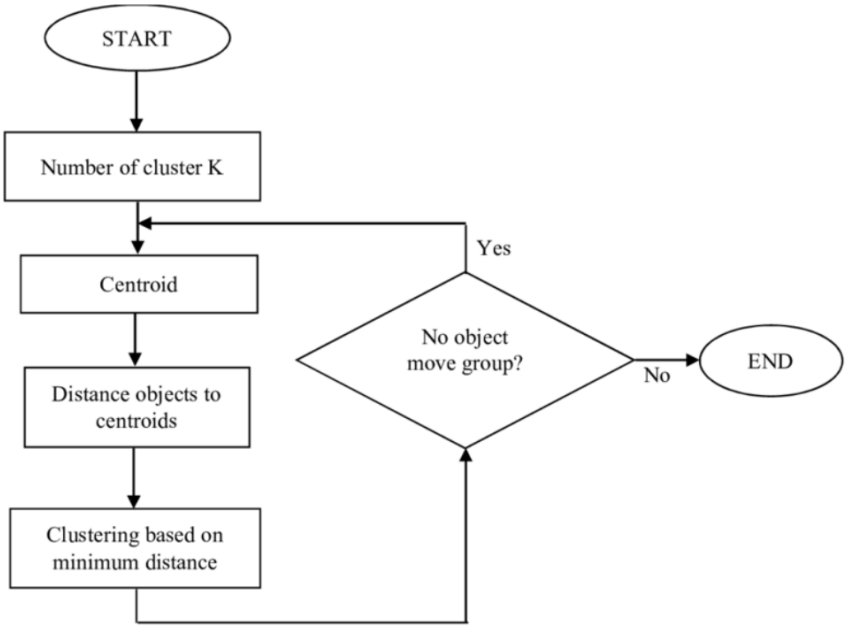


Figure 2: Computational graph of KMeans Clustering

The Clustering Algorithm intends to group the data into clusters based on the similarities. Initially the number of clusters is set to 10. Randomly 10 centroids are assigned, the algorithm evaluates the distance between each point from the centroids, obtaining the minimum distance, to put them into clusters. We then find the centroids of the new clusters formed and find the distances of the centroids with each point again, re-allotting the points to the clusters. This allows us to move the centroids and form the efficient clusters. We keep calculating the new centroids till the centroids change no more.

## 4.2 Auto-Encoder

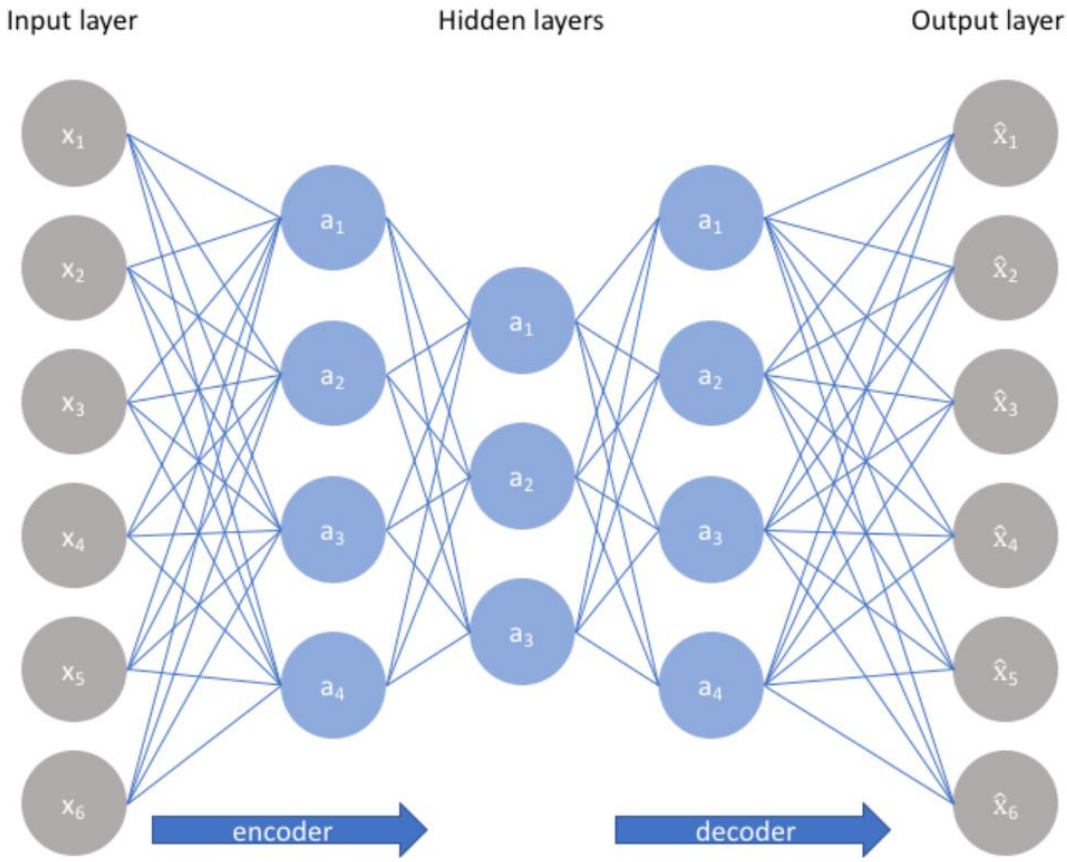


Figure 3: Computational graph of Auto Encoders

The Auto-encoders are forward-feed neural networks, which gives the same output as the input. The input is compressed, and this reduces the features, into a latent space representation. Then the output is reconstructed using this representation. The network consists of an encoder and a decoder. To obtain the useful features from the undercomplete auto-encoding, which constrains the latent representation(h) to have a smaller dimensions than the input(x). Then we train the undercomplete autoencoder to learn the important features from the training data. We can reach an overcomplete case, if the copying task is performed without getting any useful information, here the latent represention(h) goes higher than the input(x). The main purpose of the auto-encoders is dimentionality reduction and data denoising.

# 5 Results

## 5.1 KMeans Clustering

| | |
|---|---|
| **n_clusters** | **10** |
| **init** | **random** |
| **n_init** | **10** |
| **Max_iter** | **200** |
| **tol** | **1e-04** |
| **random_state** | **0** |
| **algorithm** | **elkan** |
| **Accuracy** | **0.5398** |

Figure 4: Parameters and Accuracy of KMeans clustering

```
[[4001  199  208   23   47  508 1977    3   34    0]
 [ 285 6298   30    2   55  148  182    0    0    0]
 [  98   12 2168   23 2178  473 2020    1   27    0]
 [1987 3636   20    5   66  487  787    0   11    1]
 [ 817  173 1096   15 3561  230 1077    0   31    0]
 [   3    1    0   13    0 4680   58 1672   12  561]
 [1224   61 1071   56 1404  729 2428    6   19    2]
 [   0    0    0    6    0  724    1 5929    0  340]
 [  24   24  346 2508   45  496  379  333 2837    8]
 [   8    3    2    7   11  235   69  777    5 5883]]
```

Figure 5: Confusion Matrix for KMeans Clustering

These parameters give high accuracy for the model to give correct results. 0.5398 Accuracy of the model is obtained. 10 clusters are formed. The initial centroids are chosen randomly. We change the centroids n_init times i.e. 10 and it is run for about 200 iterations. The KMeans algorithm that is used is ‘elkan’. This gives is efficient cluster formations.

## 5.2 Auto-Encoder with KMeans clustering and GMM

**0.5519** Accuracy is obtained with Auto-encoder with KMeans Clustering and **0.5932** Accuracy is obtained with Auto-encoding with GMM. Using the Encoder the dimensions are reduced and using the decoder the dimensions are reconstructed. Using multiple Convolution layers, max-pooling and normalization techniques and relu activation functions, we obtain an efficiently trained dataset.

| | |
|---|---|
| Auto-encoding with KMeans Clustering | 0.5519 |
| Auto-encoding with GMM | 0.5932 |

Figure 6: Auto-encoder accuracy

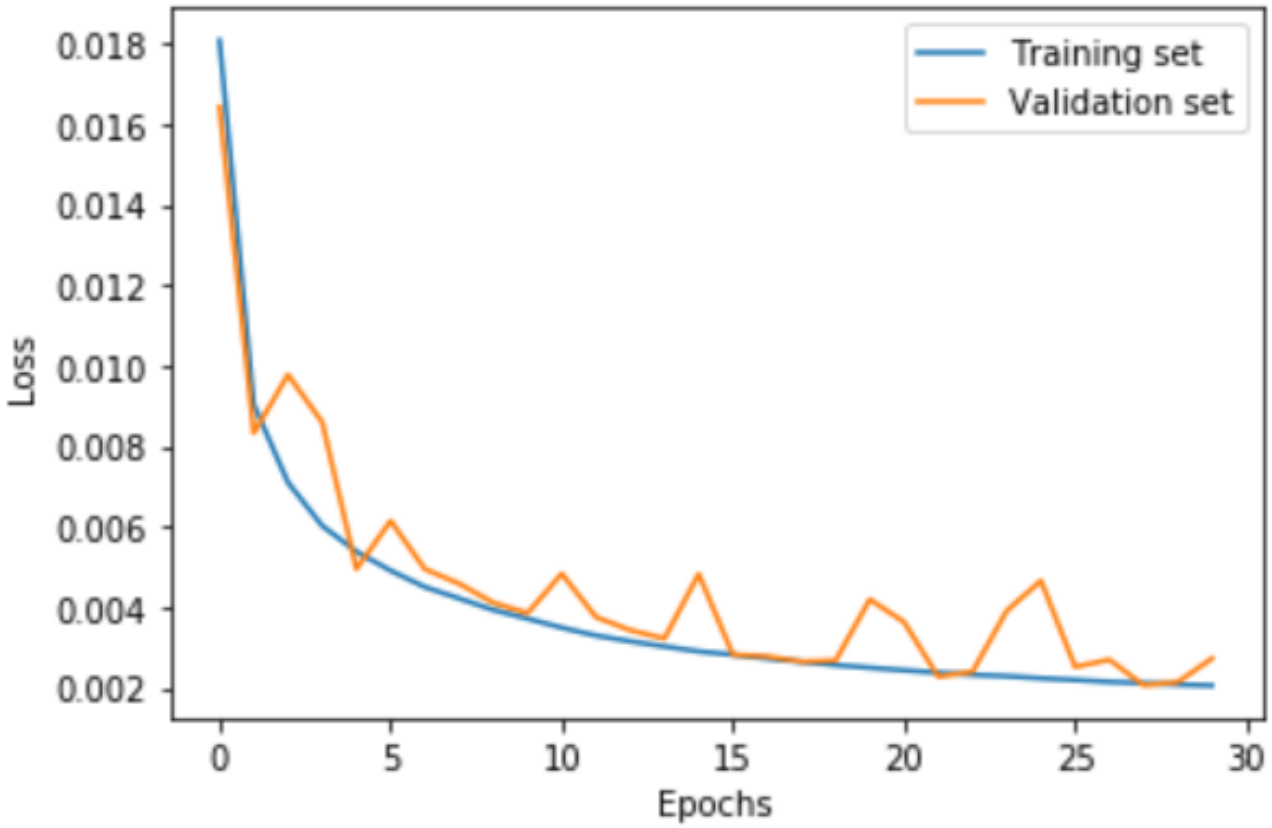


Figure 10: Loss versus epochs graph for auto-encoder

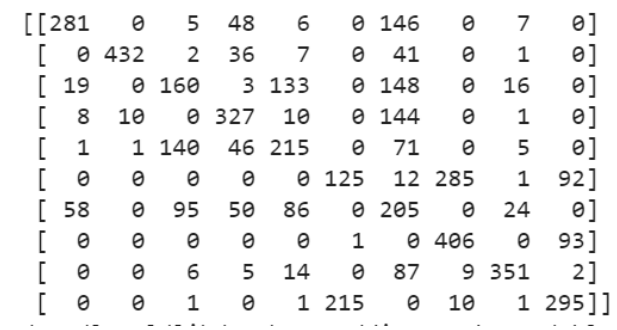


Figure 11: Confusion Matrix for auto-encoder **KMeans**

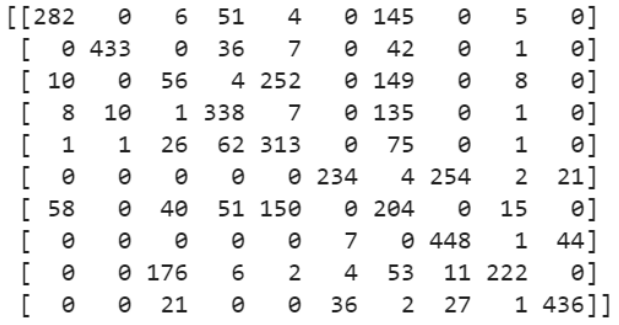


Figure 11: Confusion Matrix for auto-encoder **GMM**

## 6 Conclusions

Clustering is a highly efficient algorithm in classifying data into multiple clusters and identifying the data to be a part of at least one cluster. Pre-processing the data by normalizing it, arranges the data in a common range which makes it easier to access the data. The diagonal in the confusion matrix represents the true positives.

It turns out that Using auto-encoders provides an efficient algorithm to identify the clusters. Auto-Encoding with GMM gives the highest accuracy on the Fashion-MNIST dataset i.e. 59.32%, while with auto-encoding using KMeans gives an accuracy of 55.19% and Clustering using KMeans gives an accuracy of 53.98%.

### References

[1] https://www.datarobot.com/wiki/unsupervised-machine-learning/

[2] https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[4] Requirements of the Project

[5] https://scikit-learn.org

[6] https://en.wikipedia.org/wiki/Keras

[7] https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

[8] https://arxiv.org/ftp/arxiv/papers/1503/1503.00900.pdf

[9] https://www.kaggle.com/s00100624/digit-image-clustering-via-autoencoder-kmeans/notebook

[10] https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

[11] https://smorbieu.gitlab.io/accuracy-from-classification-to-clustering-evaluation/

[12] https://www.datacamp.com/community/tutorials/autoencoder-classifier-python