Cluster Analysis using Unsupervised Learning on Fashion MNIST Image Dataset

Bhakti Jadhav
Department of Computer Science
University at Buffalo
Buffalo NY, 14214
bhaktiha@buffalo.edu

Abstract

Fashion MNIST is an image dataset of Zalando's article which comprises of 70,000 images of fashion products from 10 categories so there are 7,000 images from each category. Each component is a 28 × 28 grayscale images. The dataset is divided such that training set consists of 60,000 images and test set consists of 10,000 images. Fashion MNIST is supposed to be a demanding dataset as a drop-in replacement for the original MNIST. Fashion MNIST is similar to original MNIST in terms of data format, image dimensions, training and testing set size and is used for exploring Machine Learning Algorithms. Outcome shows that Gaussian Mixture Model performed on data provided, performed well with a test accuracy of ≈63%.

1 Introduction

Machine Learning Algorithms can be classified as Supervised and Unsupervised learning algorithms. In unsupervised learning, we have a dataset but we don't have a target variable as in case of supervised learning. Here, we are supposed to observe the hidden patterns or features of the data and group them into clusters. Cluster in machine learning means the point in dataset which have some shared properties which falls into one alike group.

A simple implementation of unsupervised learning algorithm is clustering of news articles by Google news where each category like entertainment, politics, sports, science have 1000's of articles. Some more examples include market segmentation, social network analysis, recommendation systems etc.

For a broader view, we can categorize clustering into 3 different types, namely Hierarchical, Partitional and Bayesian.

- Hierarchical algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative("bottom-up") or divisive("top-down").
    - Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.
    - Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- Partitional algorithms typically determine all clusters at once but can also be used as divisive algorithms in the hierarchical clustering.
- Bayesian algorithms try to generate a posteriori distribution over the collection of all partitions of the data.

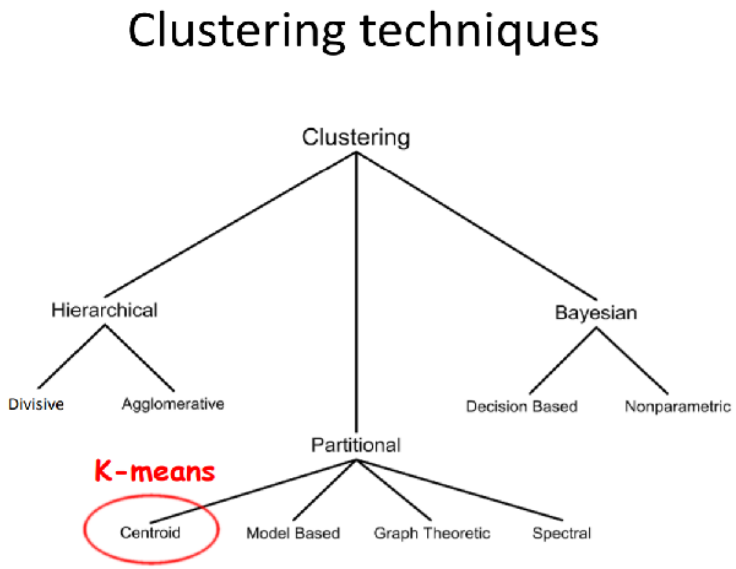


Figure1: One Hidden Layer Neural Network Model

In this project, we are going to cover the role of unsupervised learning algorithms by clustering Fashion-MNIST clothing images. Following are the three tasks performed:

1. Used KMeans algorithm to cluster original data space of Fashion-MNIST dataset using Sklearns library.
2. Built an Auto-Encoder based K-Means clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.
3. Built an Auto-Encoder based Gaussian Mixture Model clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.

2 Dataset

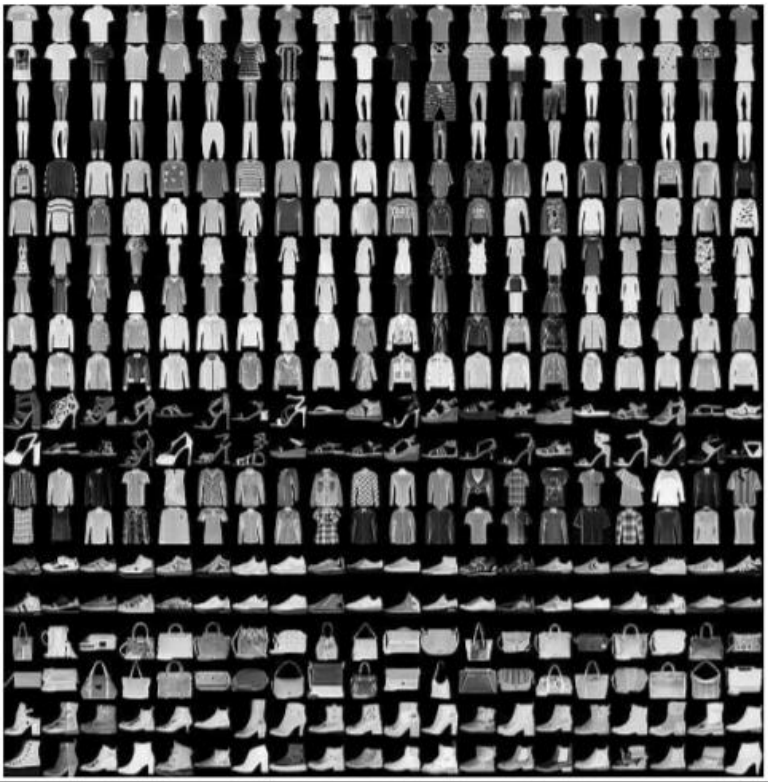
The dataset consists of training set(60,000 examples) and testing set(10,000 examples) along with labels. The images are 28 × 28 grayscale images with pixels between 0 to 255.

Table 1: Files contained in the Fashion-MNIST dataset.

Name	Description	# Examples	Size
train-images-idx3-ubyte.gz	Training set images	60, 000	25 MBytes
train-labels-idx1-ubyte.gz	Training set labels	60, 000	140 Bytes
t10k-images-idx3-ubyte.gz	Test set images	10, 000	4.2 MBytes
t10k-labels-idx1-ubyte.gz	Test set labels	10, 000	92 Bytes

Labels are an array of integers 0 to 9 representing 10 different classes of clothing. The training set receives 6000 random images of each category.

Table 2: Class names and example images in Fashion-MNIST dataset

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

3 Pre-Processing

The dataset is standardized before implementation of Neural networks in following ways:

3.1 KMeans Clustering

1. Re-shaping the images since, the first layer of the model expects input to be of the shape (60000 × 784) whereas the training set has dimensions 60000 × 28 × 28. So, reshaping will convert 28 × 28 dimensional into images of size 784.
2. Normalize the data to keep the gradient manageable in the range of [0,1].

3.2 Auto Encoder with KMeans Clustering:

1. Re-shaping the images since, the first layer of the model expects input to be of the shape (60000 × 784) whereas the training set has dimensions
   60000 × 28 × 28. So, reshaping will convert 28 × 28 dimensional into images of size 784.
2. Normalize the data to keep the gradient manageable in the range of [0,1].
3. Design and use an autoencoder to decrease the dimensionality of the data and extract useful information. This will then pass on the encoded images to the K-Means algorithm.

3.3 Auto Encoder with GMM Clustering:

1. Re-shaping the images since, the first layer of the model expects input to be of the shape (60000 × 784) whereas the training set has dimensions 60000 × 28 × 28. So, reshaping will convert 28 × 28 dimensional into images of size 784.
2. Normalize the data to keep the gradient manageable in the range of [0,1].
3. Design and use an autoencoder to decrease the dimensionality of the data and extract useful information. This will then pass on the encoded images to the K-Means algorithm.

4 Architecture

4.1 KMeans Clustering

Clustering the data points into a K number of mutually exclusive clusters each described by the mean of the data points in the cluster, minimizing a criterion knows as inertia or within-cluster sum-of-squares. Number of clusters K has to be specified. It works well with large number of samples and have been exploited across a range of applications in many different fields. The means are called as cluster centroids and they are not points from dataset, though they live in same space.

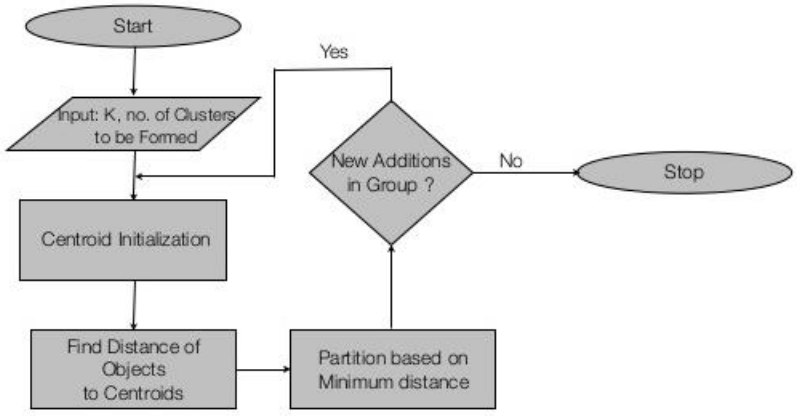


Figure2: KMeans Clustering Algorithm

4.2 Auto-Encoder

Auto-encoder is an unsupervised learning neural network that efficiently compress and encode the data. Then learns how to reconstruct the data back from reduced encoded representation to a representation that is close enough to original input. To build an autoencoder, you need three things:

- Encoding function
- Decoding function
- Distance function between the amount of information loss between the compressed representation of your data and the decompressed representation (i.e. a ”loss” function).

Bottleneck is a layer between encoder and decoder which contains the compressed representation of input data with lowest possible dimensions. The encoder and decoder will be chosen to be parametric functions (typically neural networks), and to be differentiable with respect to the distance function, so the parameters of the encoding/decoding functions can be optimized to minimize the reconstruction loss, using Stochastic Gradient Descent.

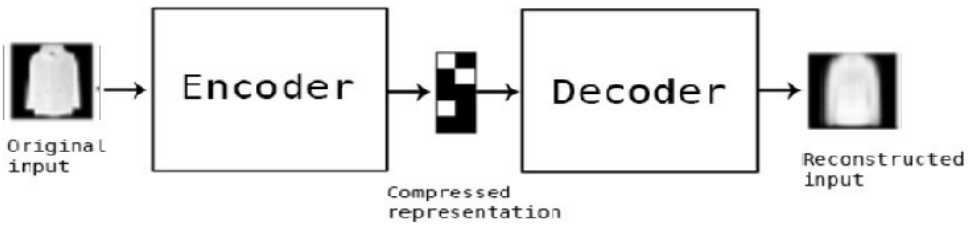


Figure3: Auto-Encoder

4.3 Auto-Encoder with KMeans Clustering

The K-means algorithm aims to choose centroids that minimize the inertia, or within-cluster sum-of squares criterion:

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

Inertia can be recognized as a measure of how internally coherent clusters are. Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high dimensional 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.

Table 3: Auto-Encoder with KMeans Clustering Model Summary

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 784)	0
dense_1 (Dense)	(None, 500)	392500
dense_2 (Dense)	(None, 500)	250500
dense_3 (Dense)	(None, 2000)	1002000
dense_4 (Dense)	(None, 10)	20010
dense_5 (Dense)	(None, 2000)	22000
dense_6 (Dense)	(None, 500)	1000500
dense_7 (Dense)	(None, 500)	250500
dense_8 (Dense)	(None, 784)	392784

Total params: 3,330,794
Trainable params: 3,330,794
Non-trainable params: 0

4.4 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 belongs to using the Gaussian Mixture.predict method.

Table 4: GMM Cluster Model Summary

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	(None, 784)	0
dense_13 (Dense)	(None, 256)	200960
dense_14 (Dense)	(None, 128)	32896
dense_15 (Dense)	(None, 10)	1290
dense_16 (Dense)	(None, 128)	1408
dense_17 (Dense)	(None, 256)	33024
dense_18 (Dense)	(None, 784)	201488

Total params: 471,066
Trainable params: 471,066
Non-trainable params: 0

5 Result

KMeans Clustering:

Test Accuracy: 59.74%

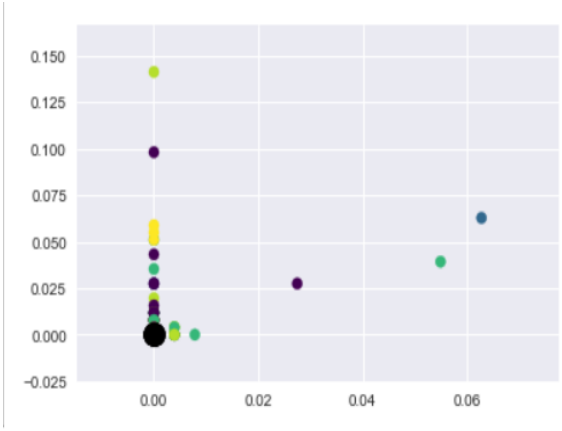
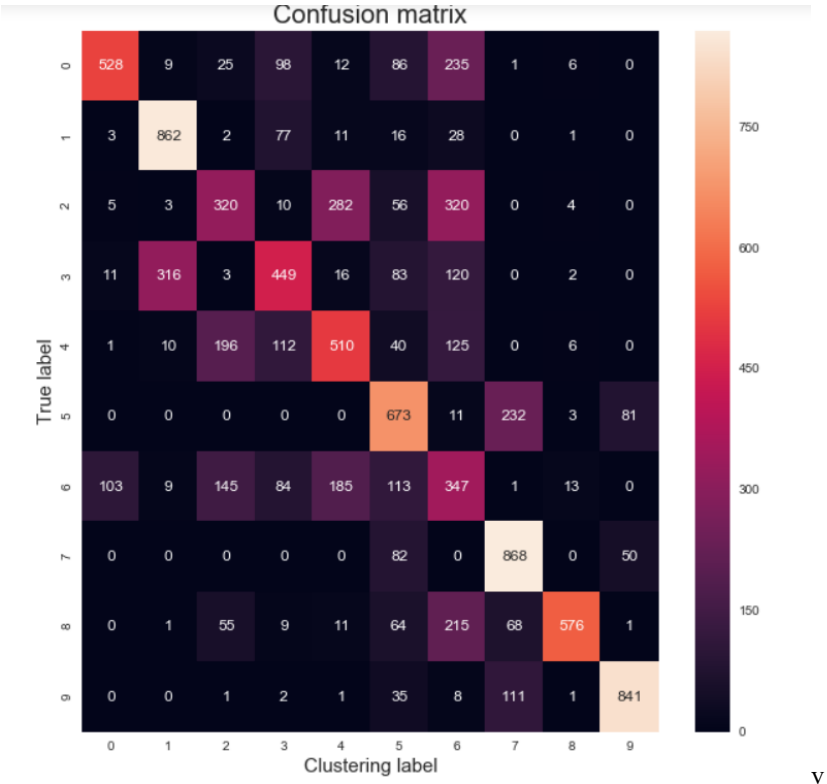


Figure4: KMeans Clustering



v

Figure5: Confusion Matrix for KMeans Clustering

Auto-Encoder with KMeans Clustering:

The best case is with Number of epochs = 150

Test accuracy: 61 %

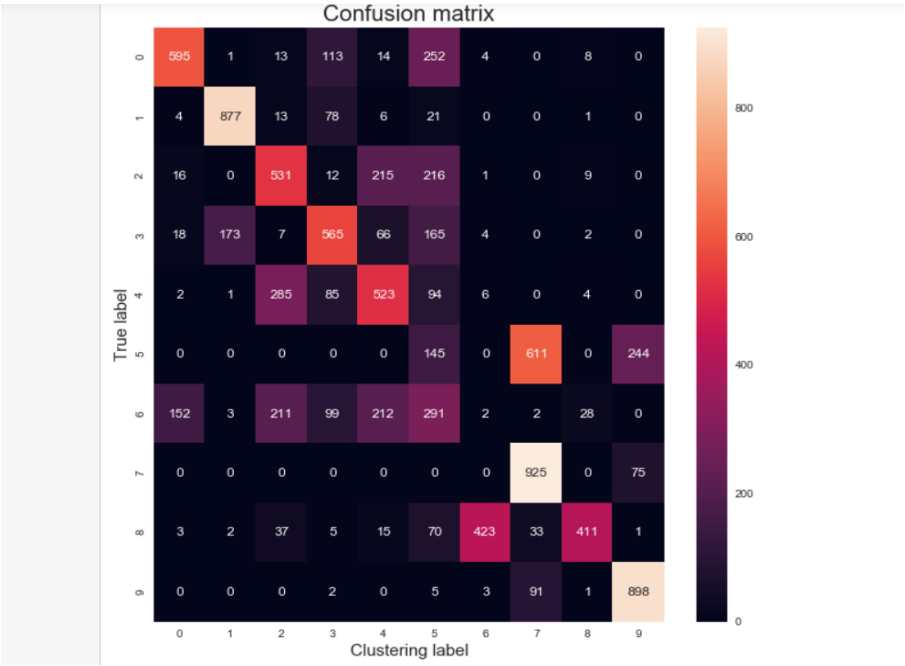


Figure6: Confusion Matrix for Auto Encoder with KMeans Clustering

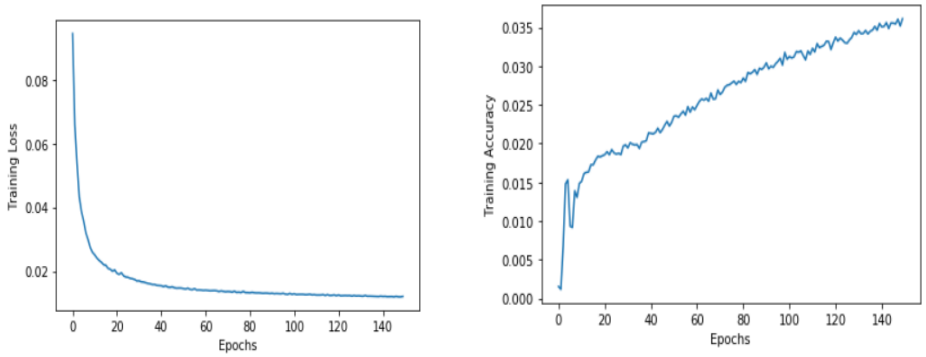


Figure7: Training Loss and Training Accuracy

Auto Encoder with GMM Clustering:

The best case is with Number of epochs = 200

Test accuracy: 63%

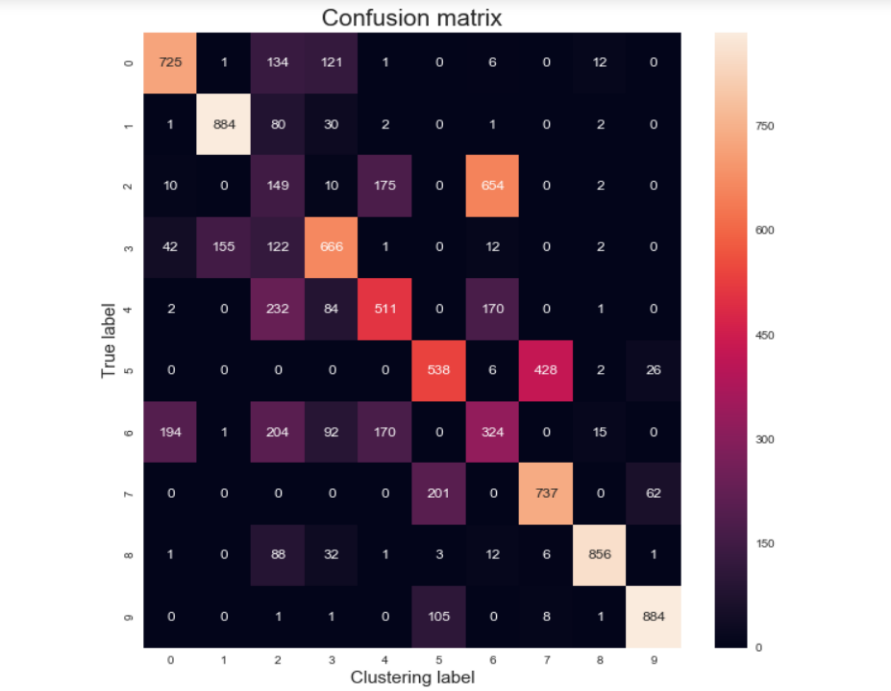


Figure9: Confusion Matrix for Auto Encoder with GMM Clustering

6 Conclusion

KMeans Clustering gives the least test accuracy among the 3 implemented unsupervised clustering models. In Auto Encoder with KMeans Clustering it is observed that higher the number of dense layers and number of epochs , higher is the accuracy. The Gaussian Mixture Model gives the best model with highest accuracy and least loss as expected.

Performance : KMeans Clustering < Auto-Encoder (KMeans) < Auto-Encoder(GMM)

References

[1] https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a

[2] https://www.dlology.com/blog/how-to-do-unsupervised-clustering-with-keras/

[3] https://www.analyticsvidhya.com/blog/2018/05/essentials-of-deep-learning-trudging-into-unsupervised-deep-learning/

[4] https://github.com/shoaibb/K-Means-Clustering/blob/master/K-Means%20Clustering.ipynb

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

[6] https://github.com/abriosi/mnist_medium_selu_gmm_mml/blob/master/article.ipynb

[7] https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726