Cluster Analysis on Fashion-MNIST Dataset using Unsupervised Learning

Sankeerth Tella
50317364
Department of Computer Science
The State University of New York at Buffalo
Buffalo, NY, 14221
stella3@buffalo.edu

Abstract

In this project, I performed clustering analysis on Fashion-MNIST clothing image using unsupervised learning technique. In this I performed three clustering techniques. First, KMeans algorithm was used to cluster original data space of Fashion-MNIST model using Sklearns library. Secondly, I performed Auto-Encoder based K-Means clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library. Finally, I performed Auto-Encoder based Gaussian Mixture Model clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.

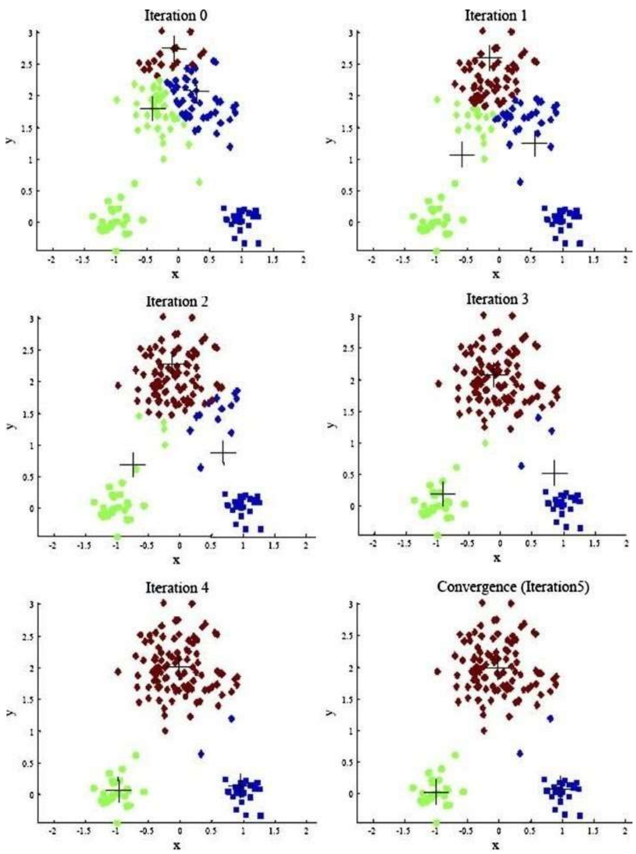
1 INTRODUCTION

1.1 Clustering:

Clustering is a Machine Learning technique that involves the grouping of data points. Given a set of data points, we can use a clustering algorithm to classify each data point into a specific group. In theory, data points that are in the same group should have similar properties and/or features, while data points in different group should have highly dissimilar properties and/or features. Clustering is a method of unsupervised learning and is a common technique for statistical data analysis used in many fields. In Data Science, we can use clustering analysis to gain some valuable insights from our data by seeing what groups the data points fall into when we apply a clustering algorithm.

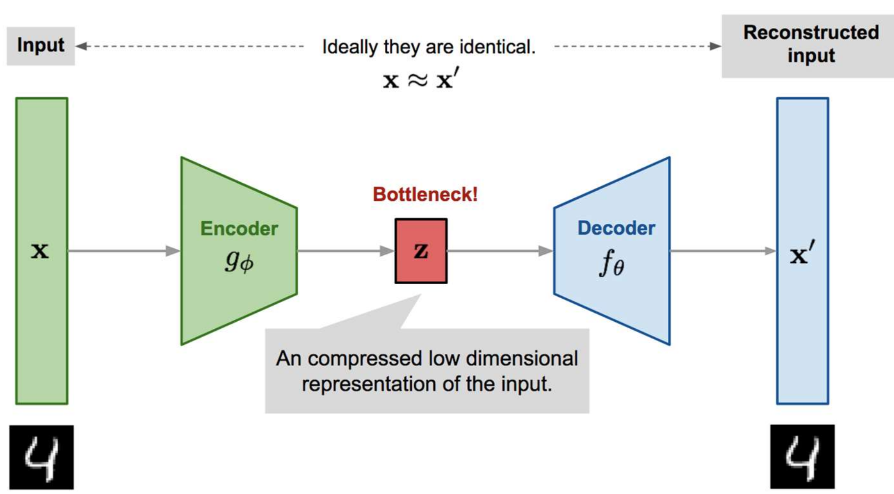
1.2 k-Means clustering:

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster.These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as center of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more.



1.3 Auto Encoder:

Autoencoder is a data compression algorithm where there are two major parts, encoder, and decoder. The encoder's job is to compress the input data to lower dimensional features. For example, one sample of the 28x28 MNIST image has 784 pixels in total, the encoder we built can compress it to an array with only ten floating point numbers also known as the features of an image. The decoder part, on the other hand, takes the compressed features as input and reconstruct an image as close to the original image as possible. Autoencoder is unsupervised learning algorithm in nature since during training it takes only the images themselves and not need labels.



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 optimize to minimize the reconstruction loss, using Stochastic Gradient Descent.

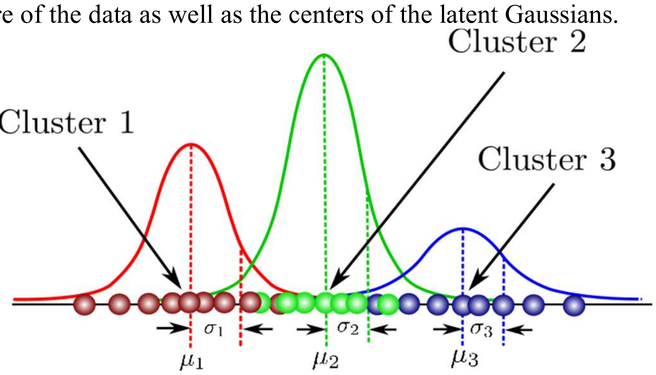
1.4.1 Auto-Encoder with K-Means Clustering:

The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum-of-squares criterion. 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 inated (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.

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

1.4.2 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.



2 DATASET:

For training and testing of our classifiers, we used the Fashion-MNIST dataset using keras load dataset function. The Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels (see above), and represents the article of clothing. The rest of the columns contain the pixel values of the associated image.

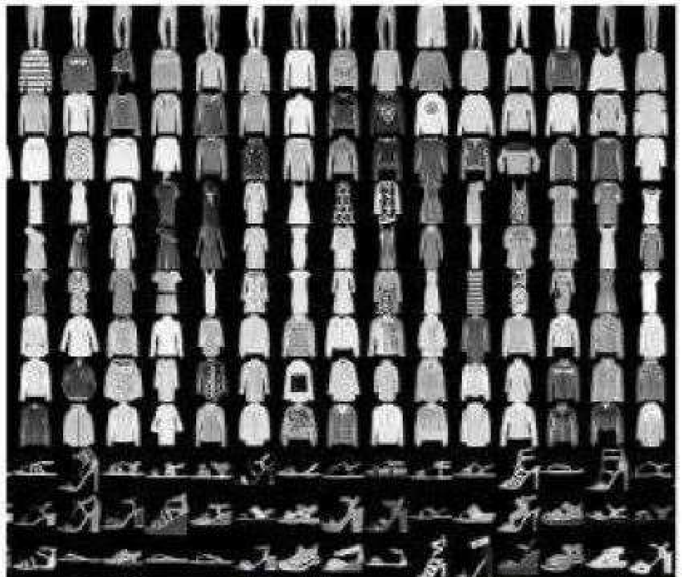


Figure 1: Example of how the data looks like.

Each training and test example is assigned to one of the labels as 108 shown in table 1.

1	T-shirt/top
2	Trouser
3	Pullover
4	Dress
5	Coat
6	Sandal
7	Shirt
8	Sneaker
9	Bag
10	Ankle Boot

Table 1: Labels for Fashion-MNIST dataset

3 PREPROCESSING:

In our given dataset, we have gray scale images whose values are in range of o to 255. So, I need to apply normalization technique. Generally, normalization is changing the range of values of data without distorting the data. In this case the data can be normalized by dividing the data with 255 so that whole data can be normalized between 0 to 1.

4 ARCHITECTURE:

4 .1 K-Means Clustering:

In the k-Means clustering task, I used k-means function provided by sklearn library. This function takes as input the number of clusters which I initialized to 10(as there are 10 groups in the data set given). I used the initialization method k-means++ and the number of iterations as 10 and max-iterations as 350. Then I set the tolerance to 1e-10.

I clustered my train data using the above model and tested it with using test data and got accuracy of 55.326% and got the confusion matrix.

4 .2 Auto Encoder based K-Means Clustering:

The auto encoder is a symmetric model that encodes the images and decodes it. The Auto encoding technique is used and it can be implemented using deepnet auto encoder which used dense layers to build the encoder and decoder. The architecture of my dense auto-encoder is as follows.

```
autoencoder.summary()
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 784)               0
_________________________________________________________________
dense_17 (Dense)             (None, 1500)              1177500
_________________________________________________________________
dense_18 (Dense)             (None, 1000)              1501000
_________________________________________________________________
dense_19 (Dense)             (None, 500)               500500
_________________________________________________________________
dense_20 (Dense)             (None, 10)                5010
_________________________________________________________________
dense_21 (Dense)             (None, 500)               5500
_________________________________________________________________
dense_22 (Dense)             (None, 1000)              501000
_________________________________________________________________
dense_23 (Dense)             (None, 1500)              1501500
_________________________________________________________________
dense_24 (Dense)             (None, 784)               1176784
=================================================================
Total params: 6,368,794
Trainable params: 6,368,794
Non-trainable params: 0
_________________________________________________________________
```

By training the auto-encoder, the model has now learned to compress each image into latent floating-point values. Now, the test set is passed through the network and the output (encoded images) is taken from the encoder, and the K-Means algorithm is applied on this output to generate the cluster centroids.

4 .3 Auto Encoder based Gaussian mixture model Clustering:

I used the same encoder described above and sent the encoded result obtained to a GMM object and performed clustering.

```
from sklearn.mixture import GaussianMixture
op_gmm = GaussianMixture(n_components=10,tol=1e-05,max_iter=300,random_state=0)
output = op_gmm.fit_predict(encoded_image)
```

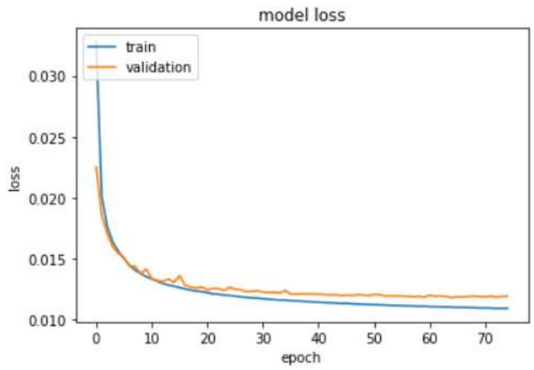
5 Results:

5.1 Accuracy of K-Means base line model:

```
kmeans_acc_baseline = metrics.accuracy_score(y_train, predicted_Y)
print(kmeans_acc_baseline)
```

```
0.5532666666666667
```

5.2 Graph of training loss and validation loss vs number of epochs while training for auto-encoder:



5.3 Accuracy for Auto-Encoder based K-Means clustering prediction:

```
[20]
    from sklearn import metrics
    kmeans_acc = metrics.accuracy_score(train_y, predicted_Y)
    print(kmeans_acc)
```

```
0.5764333333333334
```

5.4 Confusion matrix for Auto-Encoder based K-Means clustering:

```
[[4271  374   49 1118   31    0    0   15  135    7]
 [  41 5699    0  109  116    0    0   28    4    3]
 [ 278   23 3008  139 2422    0    0    6  118    6]
 [ 184 3157   36 2524   42    0    0   21   32    4]
 [  37   29 1443 1293 2936    0    0    4  252    6]
 [  13   19   69    0    4    0    0 2744   39 3112]
 [1422  125 2484  874  816    0    0   10  254   15]
 [   0    0    0    0    0    0    0 4676    8 1316]
 [ 140   78    2   41   24    0    0   77 5553   85]
 [   0    5    0    1    0    0    0   55   20 5919]]
```

5.5 Accuracy for Auto-Encoder based GMM clustering:

```
[24] print(gmm_acc)

0.6035833333333334
```

5.6 Confusion matrix for Auto-Encoder based GMM clustering:

```
[25] from sklearn.metrics import confusion_matrix
     cm=metrics.confusion_matrix(train_y, gmm_predicted_Y)
     print(cm)

[[4699  373  123  607   36    3    0    1  157    1]
 [ 137 5660    8   84  103    0    0    4    4    0]
 [ 168    7 3545  126 2087    0    0    1   66    0]
 [ 493 2820   63 2516   50    1    0    1   56    0]
 [  86   15 1165 1163 3542    0    0    0   29    0]
 [  25    0   25    1   10 2250    0 2449   32 1208]
 [1525  172 2783  535  774    0    0    1  209    1]
 [   1    0    0    0    0 1366    0 4552   21   60]
 [ 144   52   52   28   43   17    0   16 5643    5]
 [   5    0    5    5    4 2114    0   45   14 3808]]
```

6 Conclusion:

I observe that the auto-encoder based GMM is greater than that of auto-encoder based K-Means and this is greater than that of k-means base line model accuracy.

7 References:

1) https://www.researchgate.net/figure/Structure-of-clustering-model-with-autoencoder-and-K-means-combination_fig2_332368916
2) https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algorithm
3) https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html
4) https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
5) https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
6) https://blogs.oracle.com/datascience/introduction-to-k-means-clustering
7) https://www.dlology.com/blog/how-to-do-unsupervised-clustering-with-keras/