Unsupervised Learning using K-Means Clustering & Gaussian Mixture Model augmented by Deep Auto-Encoders

Gaurav Madhav Pai
Department of Computer Science
University at Buffalo (SUNY)
Buffalo, NY 14226 *gmadhavp@buffalo,edu*

Abstract

In this project, K-NN and Gaussian Mixture Models are used to identify clusters in the Fasion MNIST dataset. Further, the performance of the algorithms are increased by reducing the dimensionality of data. The data is encoded and the dimensionality of the data is reduced using Deep Auto Encoders. The results of the algorithms are compared.

1. Introduction

K-Nearest Neighbours is a unsupervised learning algorithm which assumes similar things exist in close proximity. In other words, similar things are near to each other. K-Means Clustering algorithm is used in this project to group the 60000 images into 10 clusters corresponding to 10 different classes.

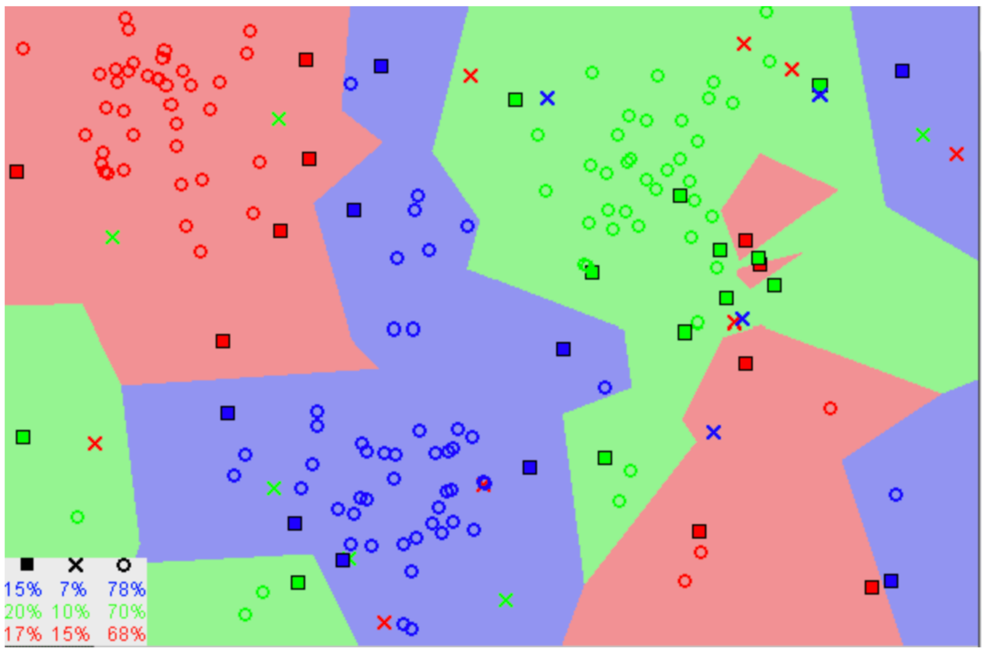


Image showing how similar data points typically exist close to each other

Fig1 Clustering Visualization of K-Means Clustering Algorithm

Further, Deep Auto Encoder is implemented using Artificial Neural Network. Auto Encoders carry out the process of encoding and decoding of data. In the encoding part, the data is encoded and dimensionality is reduced, and the opposite is done in the decoding part. The Auto Encoder is trained using 60000 images of the training data and use the trained weights to perform only encoding of the test images. These encoded images are sent as input to K-Means Clustering algorithm which should theoretically improve the clustering accuracy.

Gaussian Mixture Models are also used for clustering the data. A Gaussian Mixture is a function that is comprised of several Gaussians, each identified by k ∈ {1,…, K}, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.

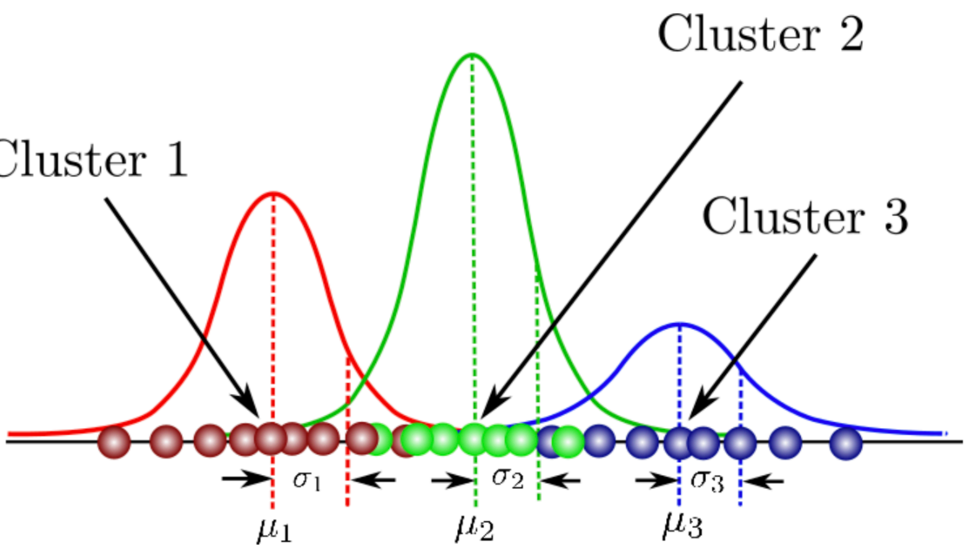
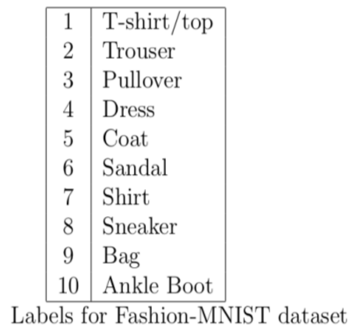


Fig 2. Components in Gaussian Mixture Models

A comparison of the performances of the above mentioned algorithms is discussed below.

2. Dataset

For training and testing of our classifiers, we will use the Fashion-MNIST dataset. 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.



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

Labels for Fashion-MNIST dataset

Figure 3: List of Classes in the dataset

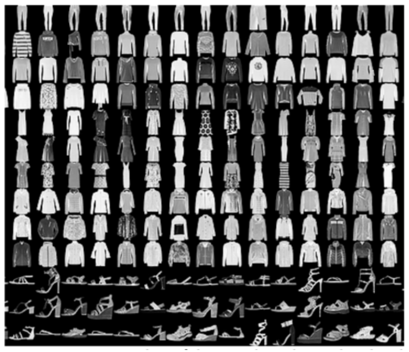
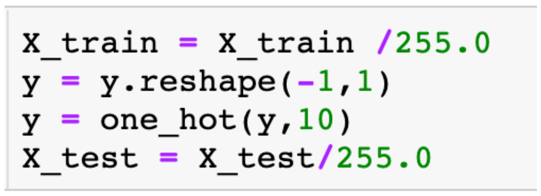


Figure 4: Example of how the data looks like.

3. Pre-processing

Since the pixel values of each image is between 0 and 255, it is normalized by dividing each pixel by 255 to bring the values down to the range 0 and 1.



```
X_train = X_train /255.0
y = y.reshape(-1,1)
y = one_hot(y,10)
X_test = X_test/255.0
```

Figure 5– Code for normalization and one hot encoding

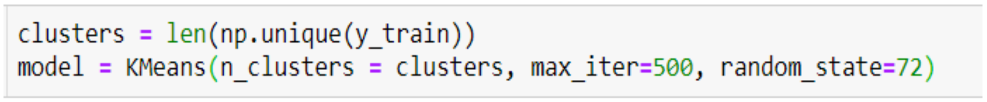
4. Architecture

4.1 K-Nearest Neighbours

The K-means algorithm identify k number of centroids and then allocate every data point to the nearest cluster while keeping the centroids as small as possible. The 'means' in the K-means refers to averaging of the data; that is, finding the centroid. To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids.
The algorithm halts creating and optimizing clusters when either:

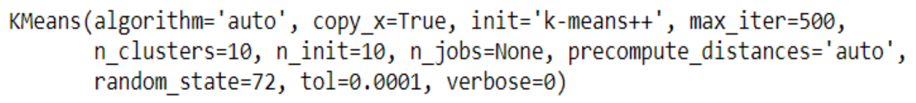
• The centroids have stabilized — there is no change in their values because the clustering has been successful.
• The defined number of iterations has been achieved.

The following figure shows that K-Means Clustering is used for finding out 10 clusters (n_clusters = 10).



```
clusters = len(np.unique(y_train))
model = KMeans(n_clusters = clusters, max_iter=500, random_state=72)
```

Fig6. Code for creating K-Means Object

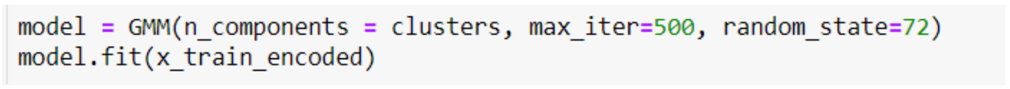


```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=500,
       n_clusters=10, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=72, tol=0.0001, verbose=0)
```

Fig 7. K-Means Clustering Model Architecture

4.2 Gaussian Mixture Models

In this project the Gaussian Mixture Model is used to find out 10 Gaussians (n_components = 10). It is shown in the following figure.



```
model = GMM(n_components = clusters, max_iter=500, random_state=72)
model.fit(x_train_encoded)
```

Fig 8. Code to Create GMM Model Object

4.3 Deep Auto Encoders

Artificial Neural Network is used to implement Deep Auto Encoder. The choice of ANN vs CNN for the implementation of Deep Auto Encoder was made considering the trade-off between training time and amount of improvement in the accuracy of the clustering models.

A Neural Network consists of an input later some hidden layer and an output layer. Each layer consists of a neuron which compute the result of a linear equation (figure 4) and is followed by an activation function. The input enters from the input layers and goes through a hidden layers and arrives at the output layers as predictions.

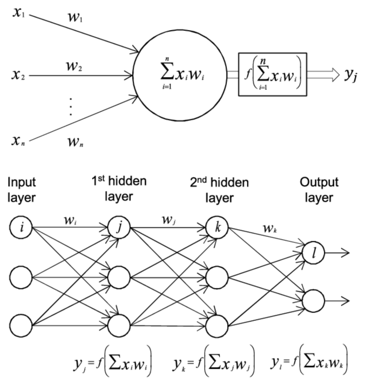
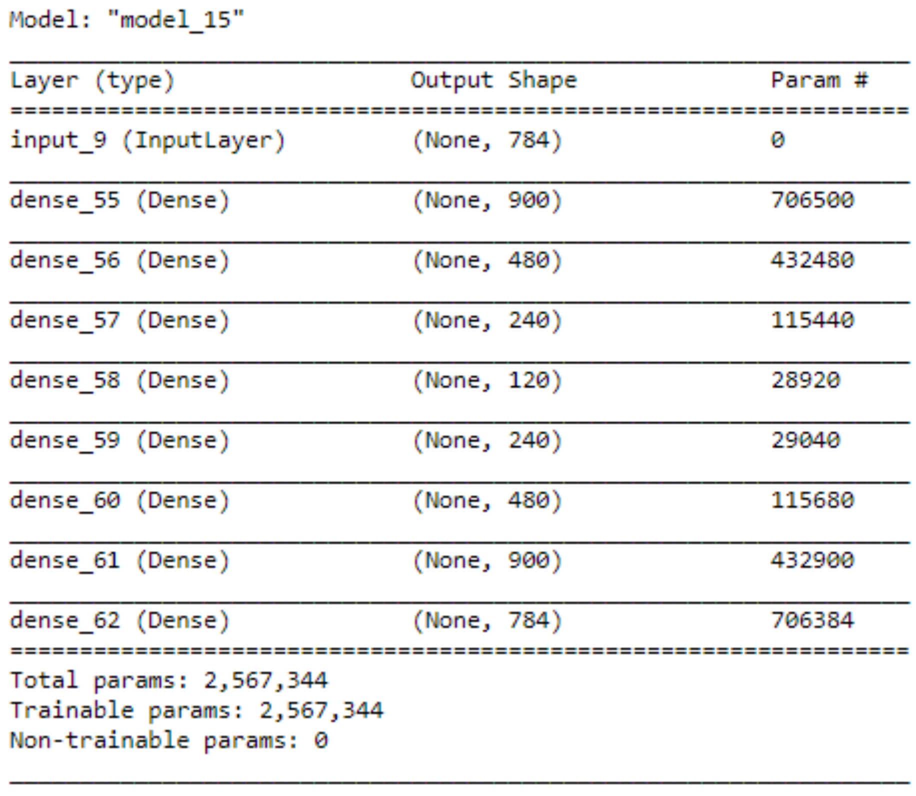


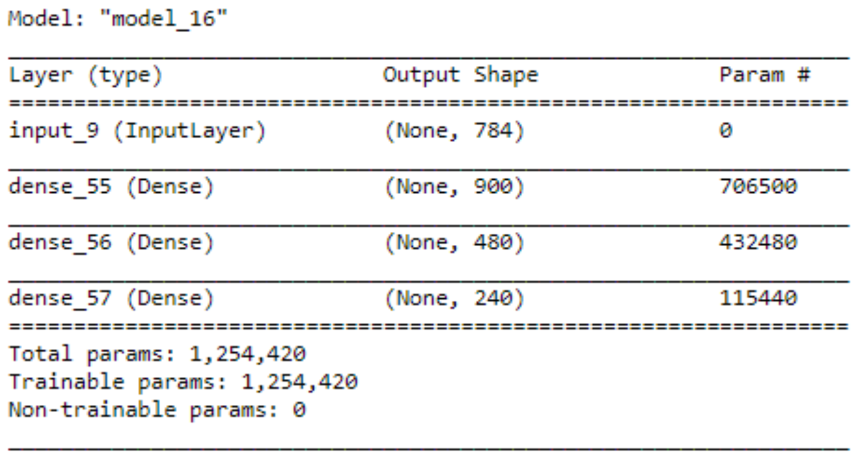
Figure 9 – A typical Neural Network



```
Model: "model_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         (None, 784)               0
_________________________________________________________________
dense_55 (Dense)             (None, 900)               706500
_________________________________________________________________
dense_56 (Dense)             (None, 480)               432480
_________________________________________________________________
dense_57 (Dense)             (None, 240)               115440
_________________________________________________________________
dense_58 (Dense)             (None, 120)               28920
_________________________________________________________________
dense_59 (Dense)             (None, 240)               29040
_________________________________________________________________
dense_60 (Dense)             (None, 480)               115680
_________________________________________________________________
dense_61 (Dense)             (None, 900)               432900
_________________________________________________________________
dense_62 (Dense)             (None, 784)               706384
=================================================================
Total params: 2,567,344
Trainable params: 2,567,344
Non-trainable params: 0
_________________________________________________________________
```

Fig10 – Auto Encoder Architecture

The above figure shows the architecture of the Artificial Neural Network used to implement the Deep Auto Encoder. The first 4 layers perform the encoding and last 4 layers perform the decoding. “Categorical Cross Entropy” loss is used to train the model. After training the Auto-Encoder, the first 4 layers with the trained weights are used to encode the test images and these are sent to K-Means Clustering and GMM models to increase the efficiency. The encoding model architecture is shown below.



```
Model: "model_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         (None, 784)               0
_________________________________________________________________
dense_55 (Dense)             (None, 900)               706500
_________________________________________________________________
dense_56 (Dense)             (None, 480)               432480
_________________________________________________________________
dense_57 (Dense)             (None, 240)               115440
=================================================================
Total params: 1,254,420
Trainable params: 1,254,420
Non-trainable params: 0
_________________________________________________________________
```

Fig 11 Encoder Architecture

4.4 Feedforward

The neural network essentially consists of 2 main steps. The first of which is feedforward. In this is part, the data makes a complete pass from the first layer to the last layer of the network to give a prediction.

4.5 Backpropagation

Backpropagation in neural network is the second step. After the forward pass, in order to improve the weights, backpropagation uses the Gradient Descent algorithm to optimize the weights and biases in the network. This process of feedforward and backpropagation is done until the networks gives a good result on the task it was intended to do. Backpropagation algorithm requires the gradients of the activation functions.

5. Results

5.1 Baseline K-Means Clustering

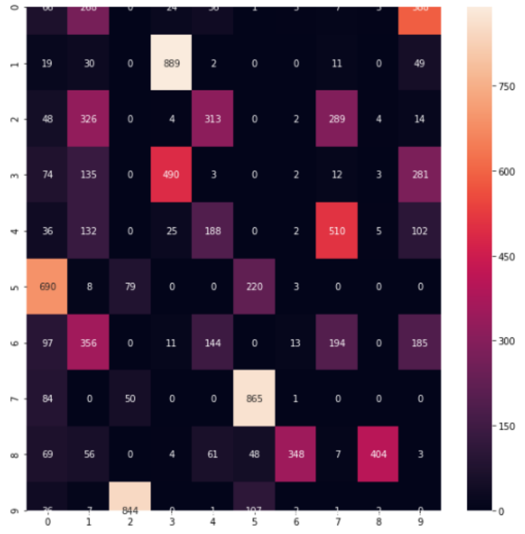


Fig 12 – Confusion Matrix for Baseline K-Means Clustering

Accuracy after adjusting clustering labels : 55%

5.2 K-Means with encoded images

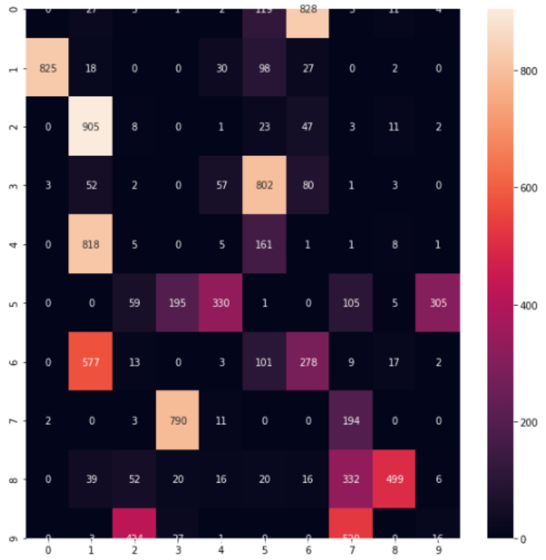
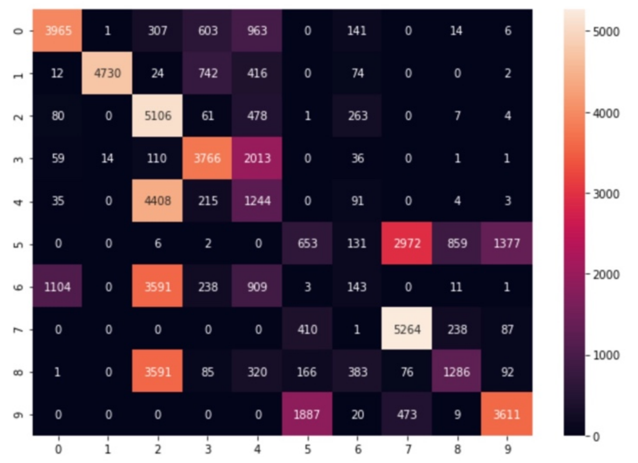


Fig 13 – K-Means Clustering with Encoded Images

Accuracy after adjusting cluster labels : 56%

5.3 Gaussian Mixture Models with Encoded Inputs

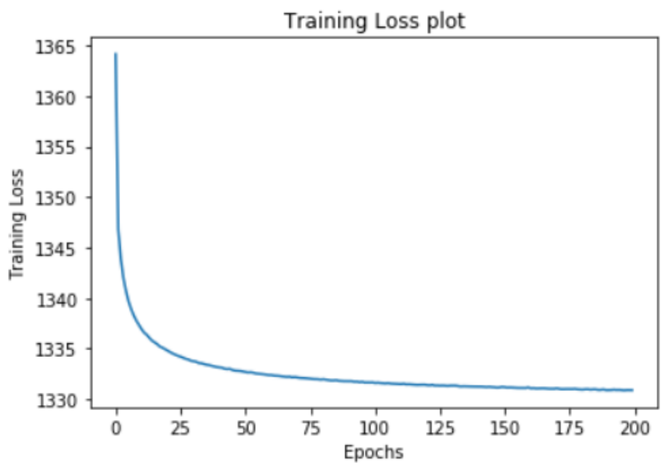


	0	1	2	3	4	5	6	7	8	9
0	3965	1	307	603	963	0	141	0	14	6
1	12	4730	24	742	416	0	74	0	0	2
2	80	0	5106	61	478	1	263	0	7	4
3	59	14	110	3766	2013	0	36	0	1	1
4	35	0	4408	215	1244	0	91	0	4	3
5	0	0	6	2	0	653	131	2972	859	1377
6	1104	0	3591	238	909	3	143	0	11	1
7	0	0	0	0	0	410	1	5264	238	87
8	1	0	3591	85	320	166	383	76	1286	92
9	0	0	0	0	0	1887	20	473	9	3611

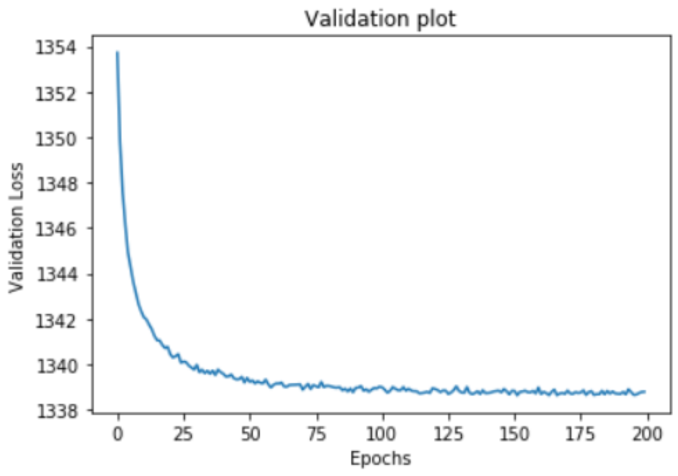
Fig 14: GMM with Encoded Images

Accuracy after adjusting: 61%

Training Loss for AutoEncoder



Validation Loss for AutoEncoder



6. Conclusion

Using Encoded inputs to clustering models increases the performance of the clustering. The this project GMM gives us the best clustering.
The Algorithms can be further improved by tuning the hyperparameters.