Project 3: Clustering

Sambo Dutta
University at Buffalo
UBIT Name: sambodut
UB PERSON NUMBER: 50318667
sambodut@buffalo.edu

Abstract

Unsupervised Learning is a type of machine learning technique where we need to draw inferences from a dataset which is not labelled. The main objective is to group the data in certain groups so that the data in the same group are more alike then in different groups.The most common form of unsupervised learning are the clustering techniques.In this project we have used K-means clustering and Gaussian Mixtute Models to cluster data of the Fashion MNIST data.An autoencoder model is also implemented to reduce the dimension of the data so that clustering models can run more efficiently

1 Introduction

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance.

The most common form of clustering technique is K-Means clustering which involves specifying the no of clusters as input, and the algorithm returns the clusters from the dataset. This form of clustering technique is called hard clustering in which each of the individual data points are assigned to only one cluster.

To overcome the problem of hard clustering the Gaussian Mixture Models are used where each data point is assigned a probability of it belong to a particular cluster. This form of clustering has given better results of our project which involves clustering on the Fashion MNIST dataset. Since in the original dataset there are 784 features so we have trained an auto encoder model to perform dimension reduction

2 Architectures

2.1 K-Means Clustering

The K-means clustering technique is simple, and we begin with a description of the basic algorithm. We first choose K initial centroids, where k is a user- specified parameter, namely, the number of clusters desired. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster. The centroid of each cluster is then updated based on the points assigned to the cluster. We repeat the assignment and update steps until no point changes clusters, or equivalently, until the centroids remain the same.

Algorithm
1. Select K points as initial centroids.
2. Repeat
3. Form K clusters by assigning each point to its closest centroid.
4. Recompute the centroid of each cluster.
5. Until Centroids do not change.

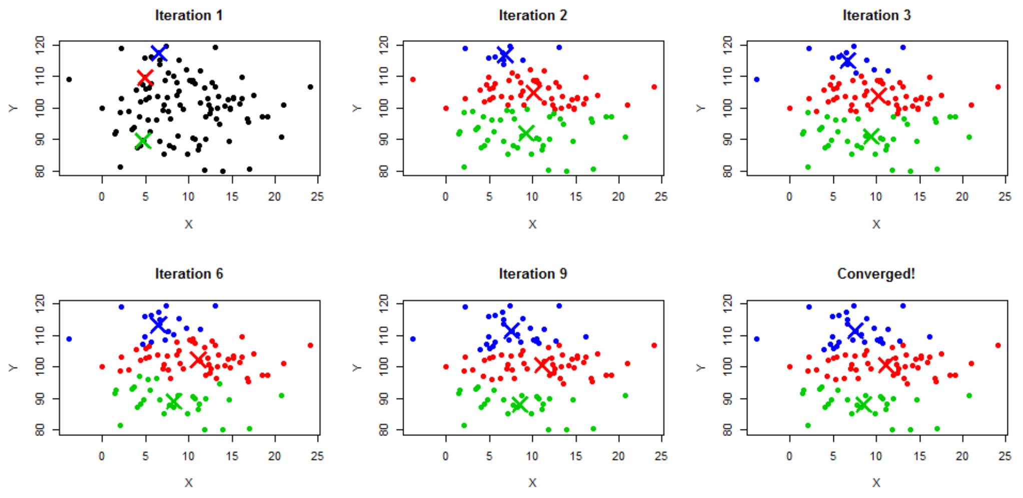


Figure 1: K-Means Algorithm

2.2 Gaussian Mixture Models

There are some issues with K-means algorithm. K-means often does not work when clusters are not round shaped because of it uses some kind of distance function and distance is measured from cluster center. Another major problem with K-Means clustering is that Data point is deterministically assigned to one and only one cluster, but in reality there may be overlapping.

For address these problems gaussian mixture model was introduced. A probabilistic approach to clustering addressing many of these problems. In this approach we describe each cluster by its centroid (mean), covariance , and the size of the cluster(Weight). Here rather than identifying clusters by "nearest" centroids, we fit a set of k gaussians to the data. And we estimate gaussian distribution parameters such as mean and Variance for each cluster and weight of a cluster. After learning the parameters for each data point we can calculate the probabilities of it belonging to each of the clusters.
The algorithm is as follows:
1. Initialize the mean μ_k, the covariance matrix Σ_k and the mixing coefficients π_k for k gaussian distribution.
2. Recompute all the parameters using the current data point distribution for each gaussian distribution.
3. Compute log-likelihood function.
4. Put some convergence criterion

2.3 Autoencoders

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible.
Autoencoders consists of 4 main parts:
1.Encoder: In which the model learns how to reduce the input dimensions and compress the input

data into an encoded representation.
2.Bottleneck: which is the layer that contains the compressed representation of the input data. This is the lowest possible dimensions of the input data.
3.Decoder: In which the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.
4.Reconstruction Loss: This is the method that measures measure how well the decoder is performing and how close the output is to the original input.

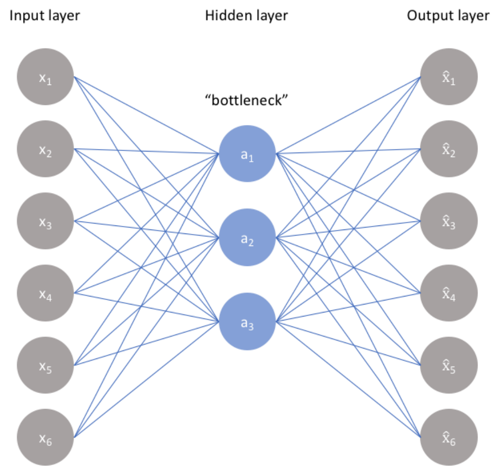


Figure 2: Autoencoder Model

3 Experiments

3.1 Dataset

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. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

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, and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

3.2 Normalization

Before the splitting the dataset into training,testing and validation,we are normalizing the dataset.The dataset has 784 unique features in which individual features may have in its own ranges.While some of these features may have a very low range, others which have a higher range might overpower its weightage and might become a more dominating feature just because of its higher range. So in order to avoid these kind of situations we are normalizing the data in the range 0 to 1 for each particular feature, so that each features have equal weightage.We have used the equation below to normalize each feature:

$$x_{new} = \frac{x_{old}}{255} \tag{1}$$

since 255 is the maximum value of a pixel.

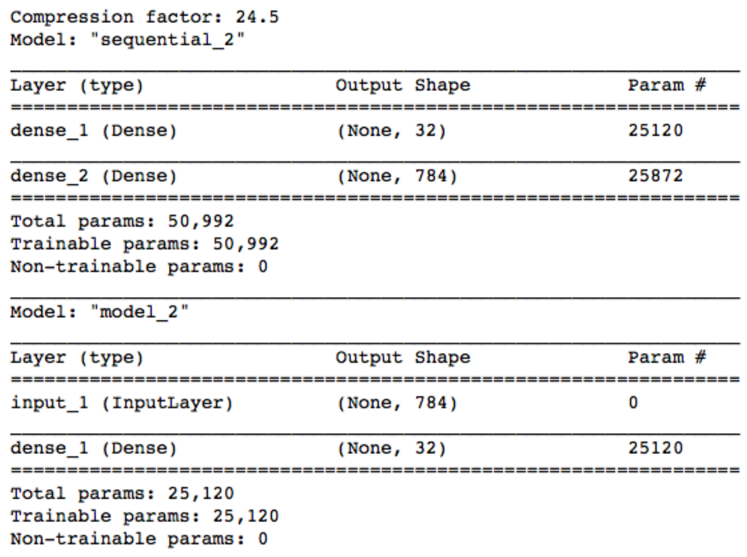
4 Results

We have performed three separate experiments. First we have conducted a K-Means clustering algorithm on the Fashion MNIST dataset. The Keras and sklearn libraries were used to perform this clustering algorithm. The accuracy of the clustered data is evaluated using Normalized mutual info score (NMI) and adjusted mutual info score (AMI) of the sklearn library. The accuracies were noted to be 51.2 50 percent respectively.

For our second experiment we have trained three variants of an autoencoder model on this dataset as shown in the figure 3,4,5 respectively. The training and validation loss for the respective architectures are plotted in Figures 6, 7 and 8.

The following table shows the accuracy, NMI and AMI of all the clustering algorithms using the three different autoencoders.

Clustering Algorithm	Accuracy	NMI	AMI
K-Means	48.28	51.2	50
K-Means+Simple Encoder	47.21	44.28	43.7
Gaussian Mixture + Simple Encoder	51.7	57.5	55.8
K-Means+Deep Encoder	44.8	45.1	44.37
Gaussian Mixture + Deep Encoder	49.6	58.37	56.25
K-means+ Convoluted Encoder	45.75	54.66	52.82
Gaussian Mixture + Convoluted Encoder	51.67	58.27	56.81

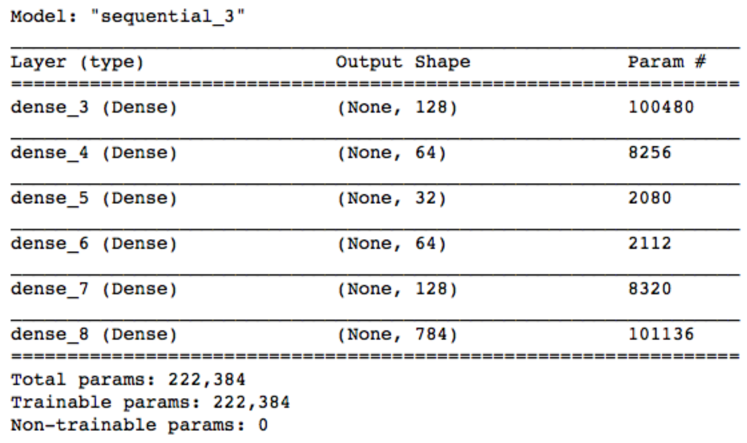


```
Compression factor: 24.5
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                25120
_________________________________________________________________
dense_2 (Dense)              (None, 784)               25872
=================================================================
Total params: 50,992
Trainable params: 50,992
Non-trainable params: 0
_________________________________________________________________
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 784)               0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                25120
=================================================================
Total params: 25,120
Trainable params: 25,120
Non-trainable params: 0
```

Figure 3: Simple Architecture

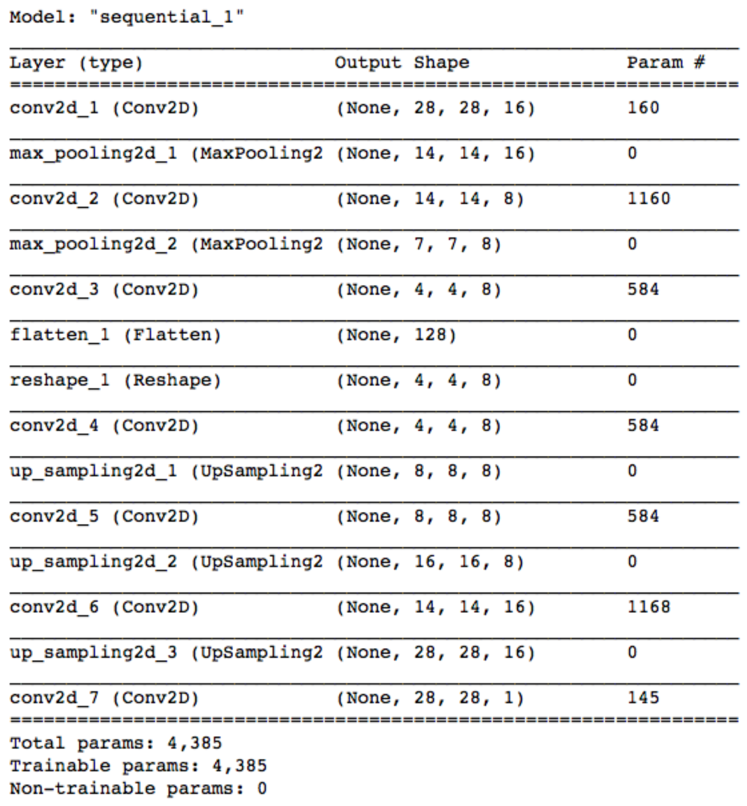
5 Conclusion

In this project using Fashion MNIST dataset unsupervised learning algorithms have been implemented. Firstly baseline K-Means algorithm was performed and ts accuracy was noted. Then to overcome the shortcomings of K-Means like cluster shape and hard clustering, Gaussian Mixture Model algorith was employed. However since the dimension of the dataset was high, the encoder portion of the autoencoders was used to reduce the dimensions. The comparison between all types of autoencoders have been shown in the result section.



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 128)               100480
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_6 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_7 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_8 (Dense)              (None, 784)               101136
=================================================================
Total params: 222,384
Trainable params: 222,384
Non-trainable params: 0
```

Figure 4: Deep Architecture



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 16)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 8)         1160
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 8)           0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 4, 4, 8)           584
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
reshape_1 (Reshape)          (None, 4, 4, 8)           0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 8)           584
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 8, 8, 8)           0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 8)           584
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 8)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 14, 14, 16)        1168
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 28, 28, 16)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 28, 28, 1)         145
=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
```

Figure 5: Convoluted Architecture

6 Reference

[1] https://medium.com/clustering-with-gaussian-mixture-model/clustering-with-gaussian-mixture-model-c695b6cd60da
[2]https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726
[3] https://ramhiser.com/post/2018-05-14-autoencoders-with-keras/

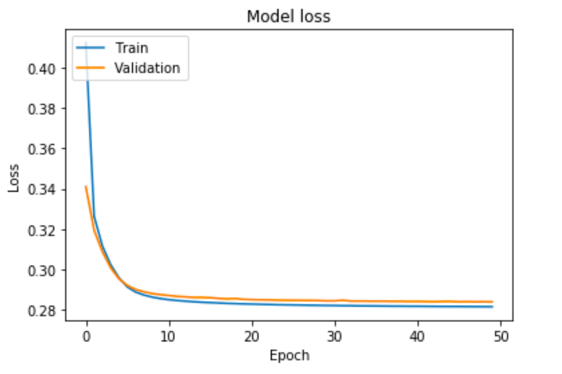


Figure 6: Training vs Validation for Simple Model

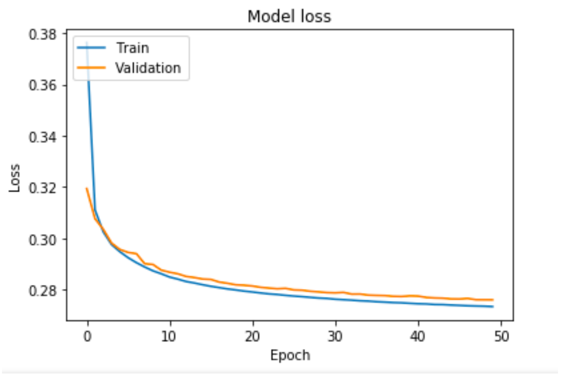


Figure 7: Training vs Validation for Deep Model

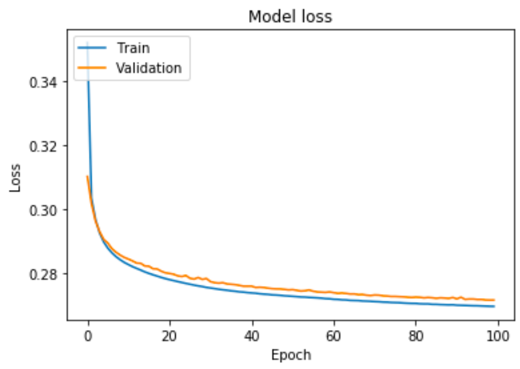


Figure 8: Training vs Validation for Convoluted Model