CLUSTER ANALYSIS WITH FASHION MNIST DATASET

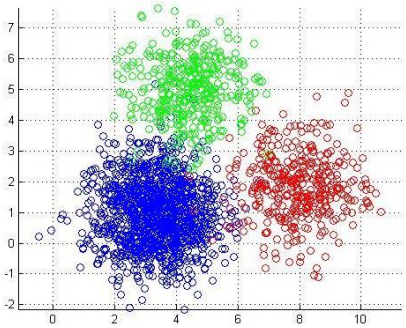
Rohith Kumar Gadalay
Department of Computer Science and Engineering
University at Buffalo
rohithku@buffalo.edu

ABSTRACT

The task of the project is to perform cluster analysis on Fashion MNIST dataset using unsupervised learning. Cluster analysis is one of the unsupervised machine learning technique which doesn't require labeled data. The goal is to train the unsupervised model using Fashion-MNIST clothing images. We perform three different tasks for clustering on Fashion MNIST dataset. The first one is using k-means algorithm to cluster the data using sklearns library, the second one is to build an Auto-Encoder based k-means clustering using keras and sklearns library and the third one is to build an Auto-Encoder based Gaussian Mixture Model clustering using keras and sklearns library.

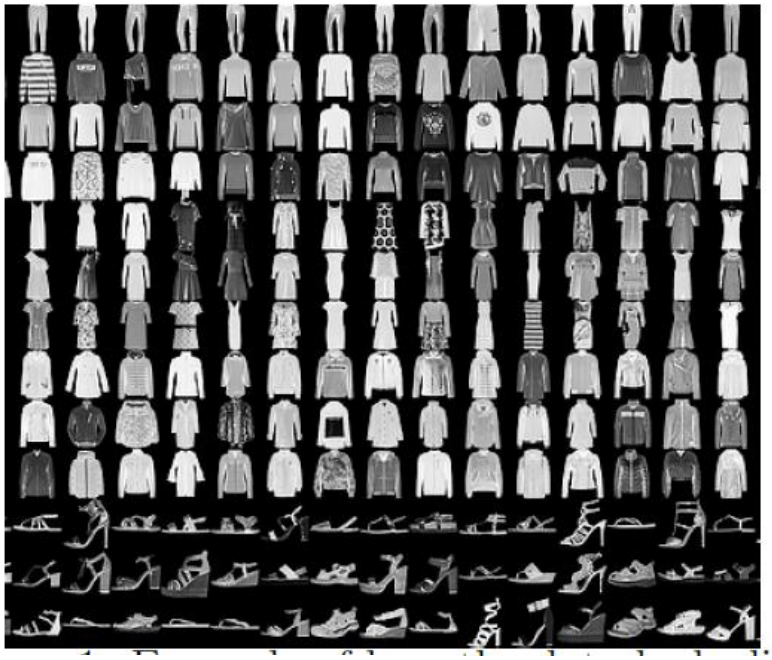
1 INTRODUCTION

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. K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. The traditional K-means algorithm is fast and applicable to a wide range of problems. However, their distance metrics are limited to the original data space, and it tends to be ineffective when input dimensionality is high. 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. 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.



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 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

3 PREPROCESSING

3.1 K-Means Clustering

1. The given dataset is read by importing from keras.datasets.fashion_mnist. The MNIST images are pre-arranged so that the first 60000 can be used for training and the last 10000 for testing. Map the retrieved data to training and test variables.
2. Reshape the training values from three-dimensional data to two-dimensional data and then normalize it appropriately.
3. Initialize the number of epochs and clusters to 10.

3.2 Auto-Encoder based K-means clustering

1.The training data is reshaped and normalized appropriately.
2.The training data retrieved is split in such a way that 80% data is used for training and 20% is used for validation purpose.
3.Initialize the number of epochs to 10 and number of clusters to 10.
4.Reshaping the data again to feed into encoder.

3.3 Auto-Encoder based Gaussian Mixture Model Clustering

1.The training data is reshaped and normalized appropriately.
2.The training data retrieved is split in such a way that 80% data is used for training and 20% is used for validation purpose.
3.Initialize the number of components to 10.

.

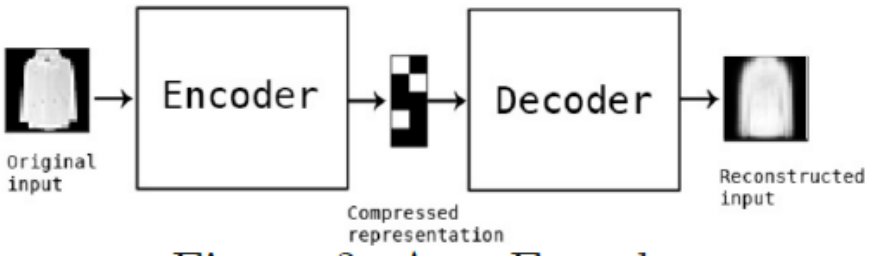
4 ARCHITECTURE

4.1 K-Means Clustering

The K-Means algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.
The k-means algorithm divides a set of samples into disjoint clusters, each described by the mean of the samples in the cluster. The means are commonly called the cluster centroids; note that they are not, in general, points from set of samples, although they live in the same space.

4.2 Auto-Encoder

"Autoencoding" is a data compression algorithm where the compression and decompression functions are data-specific, lossy, and learned automatically from examples rather than engineered by a human. Autoencoder uses compression and decompression functions which are implemented with neural networks To build an autoencoder, we need three things: an encoding function, a decoding function, and a distance function between the amount of information loss between the compressed representation of the data and the decompressed representation (i.e. a "loss" function). 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.



4.2.1 Auto-Encoder based K-Means 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.

4.2.2 Auto-Encoder based Gaussian Mixture Model 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 GaussianMixture.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 GaussianMixture.fit_predict method.

5 RESULTS

5.1 K-Means Clustering

I have used sklearn library to cluster the dataspace of Fashion-MNIST using k-means algorithm. I have used the number of clusters as 10 and obtained the accuracy as **54.19%**

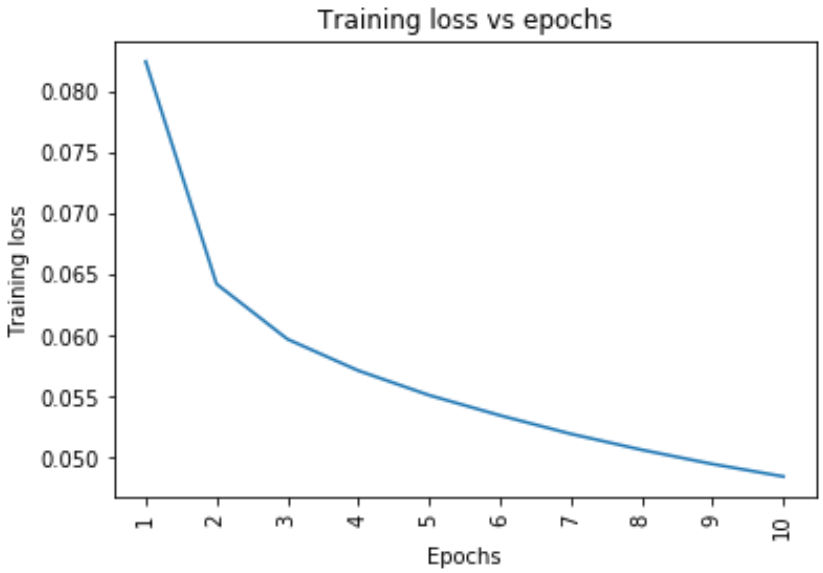
The confusion matrix obtained is:

```
[[3252    1   42   87    6    0  761    0    7    0]
 [   5 5178    1   32   10    0    4    0    2    0]
 [  79   66 3444   39 3414    0 1880    0   66    6]
 [ 489  239   12 3687  177    8  204    0  135   10]
 [ 342  216  294 1145 1346    0  465    0  156    7]
 [   0    0    0    0    0    0    0    0    0    0]
 [1699  293 1800  998  912 1472 2321    9 1224  156]
 [   5    0    2    0    0 3593    9 4642  219   98]
 [ 121    7  400   11  133    2  343    0 2922    1]
 [   8    0    5    1    2  925   13 1349 1269 5722]]
```

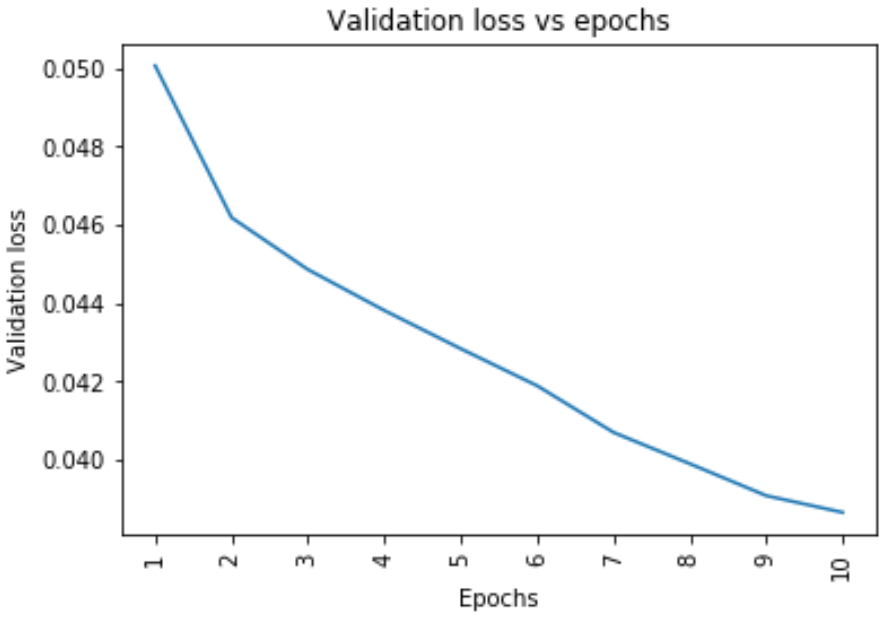
5.2 Auto-Encoder based K-means Clustering

I have used number of clusters as 10 and number of epochs as 10.
The accuracy obtained on the data is **56.39%**

The graph plotted for training loss versus the number of epochs is:



The graph plotted for validation loss versus the number of epochs is:



The confusion matrix is:

```
[[488   4   1   3   1 186 135  47  27 108]
 [  0   0   0   1   0  15  46   9 842  87]
 [  3   6   0   5   0 324  85 541   4  32]
 [  8   4   0   0   0  57 101  14 448 368]
 [  0   4   0   3   0 136  53 603  17 184]
 [  0   1 124   5 308   6 556   0   0   0]
 [ 87   1   0  12   0 295 156 318  18 113]
 [  0   0  43   4 881   0  72   0   0   0]
 [  1 404   1 359  62  30  71  64   5   3]
 [  0   1 829   4 134   3  24   3   0   2]]
```

5.3 Auto-Encoder based Gaussian Mixture Model Clustering

I have used number of clusters as 10.
The accuracy obtained on the data is **55.96%**

The confusion matrix is:

```
[[ 21  40   1  28   1  68 142 602  90   7]
 [  8   5   0   4   0 924   9   1  48   1]
 [431  44   0 239   0   5 250   7  16   8]
 [ 24  18   0   5   0 509  55   8 381   0]
 [601  32   0 141   0  14  83   0 123   6]
 [  0 244 600   0 146   0   7   0   0   3]
 [299  53   0 109   0  36 253 127 114   9]
 [  0   0 968   0  32   0   0   0   0   0]
 [  9  40  77  78   1  14 146   1   7 627]
 [  1   5 230   2 757   0   2   0   2   1]]
```

6 CONCLUSION

The clustering analysis has been performed on fashion MNIST dataset by using k-means, auto-encoder based clustering using k-means and auto-encoder gaussian mixture model. The accuracy, confusion matrix for different tasks are calculated. The graph for number of epochs vs training loss and number of epochs vs validation loss are plotted. It is observed that the accuracy and loss value varies as the epoch value is changed. Below are the accuracy obtained:

K-Means Clustering:54.19%

Auto-Encoder based K-Means Clustering:56.39%

Auto-Encoder based Gaussian Mixture Model Clustering:55.96%

7 References

* https://nips.cc/Conferences/2015/PaperInformation/StyleFiles
* https://blog.keras.io/building-autoencoders-in-keras.html
* https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a
* https://www.dlology.com/blog/how-to-do-unsupervised-clustering-with-keras/
* https://www.kaggle.com/s00100624/digit-image-clustering-via-autoencoder-kmeans