PROJECT 3. CLUSTER ANALYSIS WITH KMEANS AND GAUSSIAN MIXTURE MODEL

Thana Shree Jeevanandam
University of Buffalo
thanashr@buffalo.edu

Abstract

Unsupervised Learning, a type of self-organized learning that helps find previously unknown patterns in data set without pre-existing labels is studied here. The topic under study in this project is to recognize images and to cluster them. The K-Means and Gaussian Mixture Models with Auto Encoders are used to handle the challenge. The implementations are done in python. K-Means, Gaussian Mixture Models and Neural Network are implemented using open-source neural-network library, Keras. The validation and testing are done to check the prediction accuracy. The different evaluation metrices are used to check the effectiveness of the solution. On the whole, the system is trained on this dataset to cluster them without predefined labels and their accuracy of predicting labels for each cluster is evaluated.

1.INTRODUCTION

Fashion MNIST serves 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. The images are clustered with unsupervised learning methods. The goal is to increase the prediction accuracies of cluster labels. Hard Clustering is done with K-Means. After the task 1, the KMeans is performed on an autoencoded output to improve the accuracy of prediction of labels. The task 3 comes with performing a Gaussian Mixture Model on the encoded values.

2. DATASET

The Fashion-MNIST is a dataset of Zalando's article images used for the prediction task consists of 70,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Each pixel of the 784 pixels, has a single pixel-value, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. The Dataset is used to populate the Training, Validation and Testing sets. The Dataset contains,

Samples total	70,000
Pixel value	0 - 255
Height of image (in pixel)	28
Width of image (in pixel)	28
Total pixel/image	784

The 10 test classes are:

a) T-shirt/top
b) Trouser
c) Pullover
d) Dress
e) Coat
f) Sandal
g) Shirt
h) Sneaker
i) Bag
j) Ankle Boot

The task at hand is to predict the classes for each example using Neural Networks.

3. PREPROCESSING OF DATASET

The project starts with preprocessing of data. As a preprocessing measure, the process initiates reading the contents from the dataset in python. The original CSV data file is processed into a Pandas Dataframe. The steps in preprocessing of data includes,

a. Extracting data.
b. Splitting the dataset.

3.1 EXTRACTING DATA

Fashion MNIST dataset is downloaded and processed into a Numpy array that contains the feature vectors and a Numpy array that contains the labels using fashion mnist reader notebook present inside the scripts folder.

3.2 SPLITTING THE DATASET

The next step of preprocessing is partitioning datasets into Training, Validation and Testing sets. The Training data comprises of 80% of the overall dataset; Validation set comprises of 10% and Testing comprises of the last 10%. The partition should be such that there is no overlapping of data in any of the datasets. The model is then trained on the Training dataset and tested on the Validation and Testing sets.

4. PROCESSING OF DATASET

The preprocessed dataset is all set to get into the process. In task1, the KMeans clusters the dataset and predicts labels for the clusters. In task2, Auto Encoder Neural Network with K Means and in task3, Auto Encoder Neural Network with Guassian Mixture Model

are used. They are implemented using Keras with a group of hyperparameters. The steps in the processing include,

4.1 TASK 1

4.1.1 KMEANS CLUSTERING

The KMeans algorithm searches for a pre – determined number of clusters within an unlabeled multidimensional dataset. It accomplishes this using a simple conception of what the optimal clustering looks like:

- The "cluster center" is the arithmetic mean of all the points belonging to the cluster.
- Each point is closer to its own cluster center than to other cluster centers.

These assumptions make the basis of KMeans Algorithm. The K Means Algorithm is as follows,

```
Initialize k means with random values
For a given number of iterations:
    Iterate through items:
        Find the mean closest to the item
        Assign item to mean
        Update mean
```

The KMeans in Sklearn library has the parameters:

- Number of Clusters
- Number of initializations
- Maximum Iterations etc.,

The number cluster is iterated for the values 7 to 13.

kmeans10=KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300, n_clusters=10, n_init=10, n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001, verbose=0)

4.1.2 ELBOW POINT

One method to validate the number of clusters is the elbow method. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of *k* (7 to 13), and for each value of *k,* we calculate k_inertia (Sum of squared distances of samples to their closest cluster center.)

They are plotted against each other and is found that the graph looks like an arm, the "elbow" on the arm is the value of *k* that is the best. The idea is that we want a small SSE, but that the

SSE tends to decrease toward 0 as we increase *k* (the SSE is 0 when *k* is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster). So, our goal is to choose a small value of *k* that still has a low SSE, and the elbow usually represents where we start to have diminished returns by increasing *k*.

4.2 TASK 2

4.2.1 AUTO – ENCODER NETWORK

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. Autoencoder, by design, reduces data dimensions by learning how to ignore the noise in the data. The compression and decompression functions are data-specific, lossy, and learned automatically from examples rather than engineered by a human.

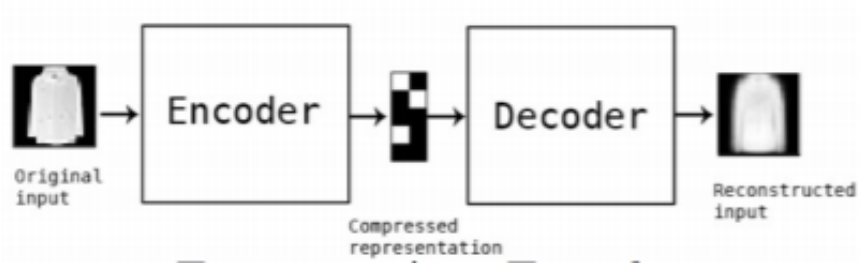

Fig 4.1 AUTO ENCODER

4.2.2 CREATING K-MEANS CLUSTERING LAYER

The task-2 performs K-Means on the auto encoded images. Here, the images are compressed into latent floating points by the Auto Encoder. K – Means is used to generate the clusters and predict the test labels as performed in task 1.

Thus, the auto encoded K-Means is expected to give more accuracy than the K-Means alone in task-1.

4.3 TASK 3

4.3.1 CREATING GMM CLUSTERING LAYER

The task-3 performs Gaussian Mixture Model on the auto encoded images. Here, the images are compressed into latent floating points by the Auto Encoder. Gaussian Mixture Model is used to generate the clusters and predict the test labels as performed in task 1.

A *Gaussian Mixture* is a function that is comprised of several Gaussians, each identified by $k \in \{1, \ldots, K\}$, 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 center.
- 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.

The first processing step is to take only the important features from the images. Convolutional NN serves this aspect.

5. ARCHITECTURE

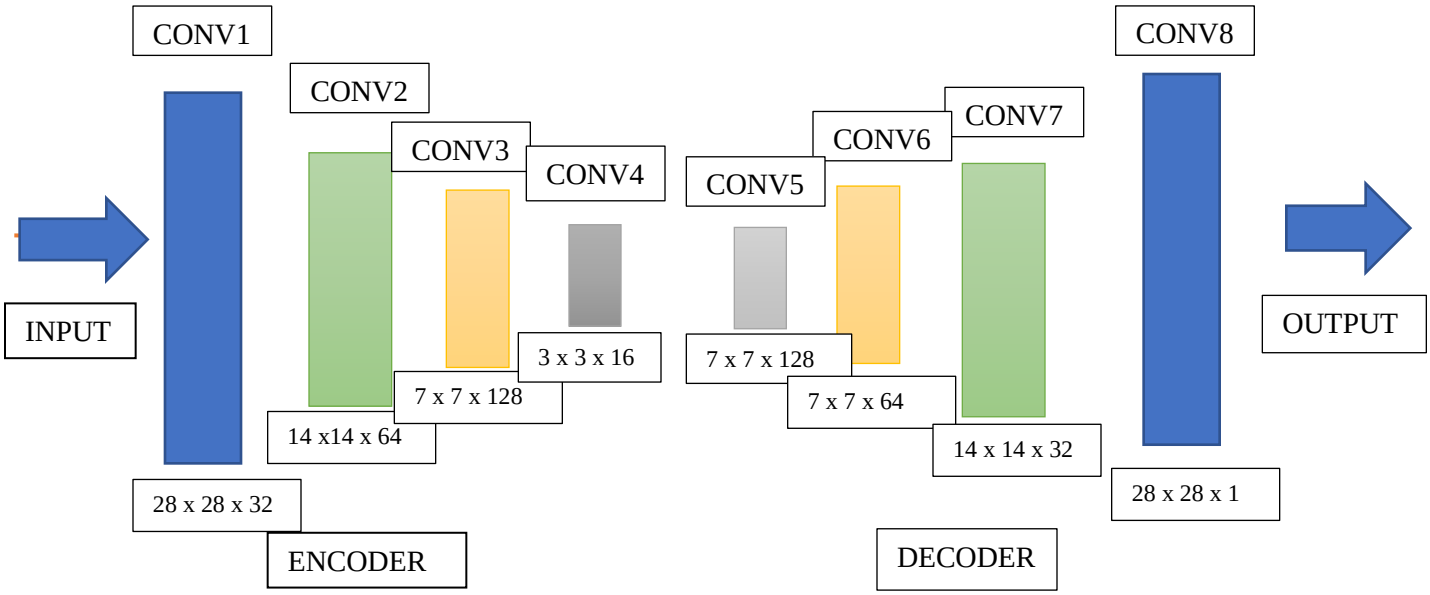

Fig 5.1 ARCHITECTURAL DIAGRAM of AUTO ENCODER

6. RESULT

```
ConfusionMatrix
[[587  29   6   0  34  93 245   1   5   0]
 [ 50 890   0   0   9  22  29   0   0   0]
 [ 19   4   4   0 566  61 342   0   4   0]
 [277 503   2   0  10  92 113   0   3   0]
 [136  27   4   0 627  42 159   0   5   0]
 [  0   0   0  45   0 650   6 227   0  72]
 [189  12  15   0 311 115 358   0   0   0]
 [  0   0   0   2   0  62   0 785   0 151]
 [  3   6 351   1  62  84  36  40 408   9]
 [  0   0   0 423   0  29   4  23   2 519]]
accuracy 0.4828
```

Fig 6.1 CONFUSION MATRIX (Task-1) with ACCURACY=48.28%

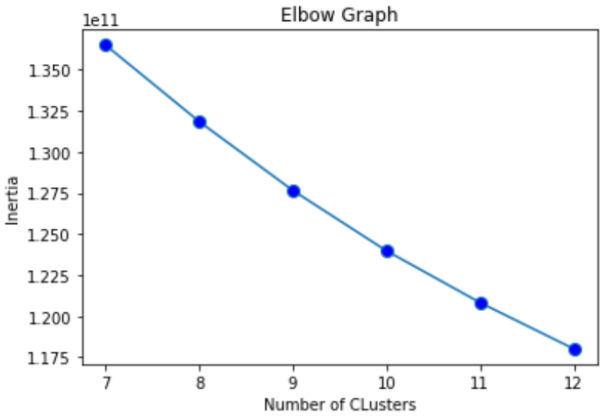

Fig 6.2 ELBOW POINT and ACCURACY (Task - 1)

6.2 TASK-2 RESULT

```
Task 2 Confusion Matrix
[[562   0  12  94  13   3 308   1   7   0]
 [  8 830   3  53  11   0  95   0   0   0]
 [ 21   0 316  12 283   2 365   0   1   0]
 [ 20  12   2 647  24   0 294   0   1   0]
 [  2   1 216 111 506   1 157   0   5   1]
 [  0   0   0   0   0  62 406 487   0  45]
 [130   0 145  80 197   4 436   0   5   3]
 [  0   0   0   0   0 179   7 811   0   3]
 [  2   1  51   4  10 285 142  18 444  43]
 [  1   0   0   2   1 454  39  20   1 482]]
(Task2) Accuracy of K-Means: 0.5096
```

Fig 6.3 CONFUSION MATRIX (Task-2) with ACCURACY=50.96%)

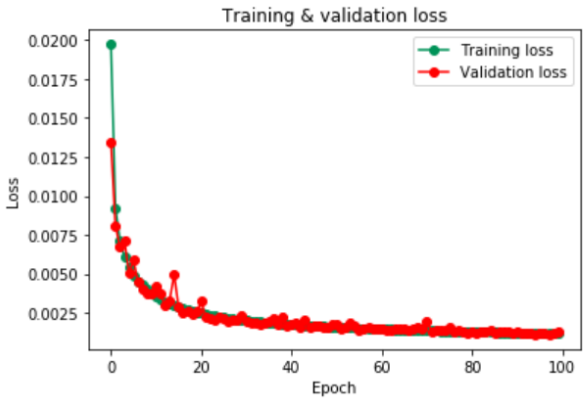

Fig 6.4 LOSS vs EPOCH (Task-2)

6.3 TASK-3 RESULT

```
Task 3 Confusion Matrix
[[534   0  15  99   5  20 282   0  45   0]
 [  2 769   2 111  10   8  92   0   6   0]
 [ 23   0 346  10 247   4 318   0  52   0]
 [ 14   2   3 670  17  16 261   0  17   0]
 [  2   1 245 119 478   4 126   0  25   0]
 [  0   0   0   0   0 762   1  94 133  10]
 [112   0 129  91 207  19 368   0  74   0]
 [  0   0   0   0   0 109   0 707  14 170]
 [  1   0  10   5   4 158  12   7 782  21]
 [  0   0   0   0   0  61   0  16 332 591]]
(Task3) Accuracy of GMM: 0.6007
```

Fig 6.5 CONFUSION MATRIX (Task-3) with ACCURACY=60.07%

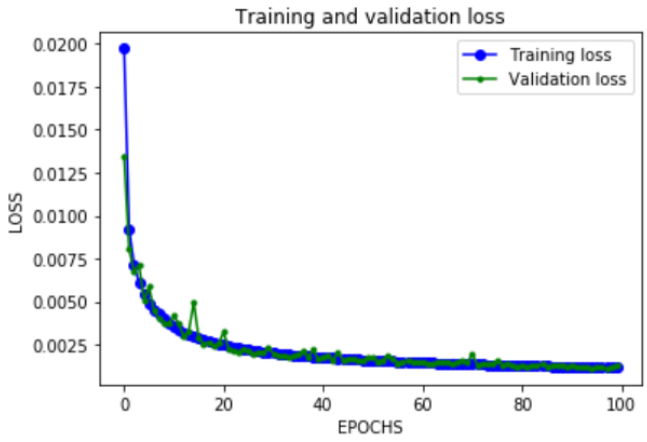

Fig 6.6 LOSS vs EPOCH (Task-3)

7. CONCLUSION

Unsupervised learning was performed on the dataset and the 10 clusters were predicted to examine the accuracy with the test labels. Among the 3 models used, Gaussian Mixture Model with Auto Encoder Neural Network has given the highest accuracy than the other two models. Thus, the classes were predicted with 60.07% accuracy.

8. REFERENCES

[1] https://scikit-learn.org/stable/modules/mixture.html - GMM

[2] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[3] https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 - KMeans Clustering