Cluster Analysis Using Unsupervised Learning

Sanidhya Chopde
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
schopde@buffalo.edu

Abstract

The aim is to perform cluster analysis on Fashion-MNIST dataset using unsupervised learning. We will be using KMeans clustering to create clusters of the dataset.

1 Introduction

In this project we have to cluster images and identify it as ne of many clusters. The task is to train the unsupervised model using the Fashion MNIST dataset. We have to use the KMeans algorithm to the cluster the Fashion MNIST dataset, train the data and then find the accuracy of the model.

2 Dataset Definition

For this project we will be using the Fashion-MNIST dataset for training, testing and validation. The dataset is of Zalando's article images consisting of a training set of 60000 examples and a testing set of 10000 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 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. Each training and test example is assigned to one of the labels as 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. for Fashion-MNIST dataset

3 Pre-Processing

3.1 KMeans Clustering

The preprocessing phase of part 1 involved doing the following things:

- We have to flatten the training labels and test labels to make them compatible for further computations required by the KMeans model.
- Then, we have to normalize the data by dividing every single example of the training and testing image data with 255. Since every single pixel is of a value between 0 & 255, this step is done so as to get the values of the examples between 1 and 0 which makes it easy to classify.

3.2 Auto-Encoder with KMeans clustering

- We have to reshape the input training and testing image data to make it of the shape 60000*28*28*1 for further computations.
- We then have to normalize the data by dividing every single example of the training and testing image data with 255. Since every single pixel is of a value between 0 & 255, this step is done so as to get the values of the examples between 1 and 0 which makes it easy to classify.

3.3 Auto-Encoder with GMM Clustering

- We have to normalize the data by dividing every single example of the training and testing image data with 255. Since every single pixel is of a value between 0 & 255, this step is done so as to get the values of the examples between 1 and 0 which makes it easy to classify.

4 Architecture

4.1 KMeans Clustering

The first step is to download and process the Fashion-MNIST dataset. We have to process it into a numpy array which contains the feature vectors and the labels. We then have to use the Sklearns library to cluster the data into 10 clusters using KMeans clustering.

AndreyBu says that "the objective of K-means is simple: group similar data points together and discover underlying patterns. To achieve this objective, K-means looks for a fixed number (k) of clusters in a dataset."

To get the optimal accuracy of the model we have to tweak some parameters in the KMeans function provided by the Sklearns library. We had to do KMeans clustering here for 10 clusters. The other parameters that were tweaked to attain the optimal accuracy are: init, random_state and max_iter.

```
kmeans = KMeans(init = "random", n_clusters=10,random_state=5, max_iter = 27)
```

4.2 Auto-Encoder with KMeans clustering

Running a dimensionality reduction algorithm such as Auto-encoder prior to k-means clustering can alleviate the problem of Euclidean distances being inflated in very high dimensional spaces due to the curse of dimensionality and speed up the computations.

We start by creating an autoencoder. Since our inputs are images, we will use convolutional neural networks as encoders and decoders as they give better results for images. The encoder will consist of Conv2D and Maxpooling2D layers and the decoder will consist of Conv2d and UpSampling2D layers. We then will compile and train the model using the fit() function.

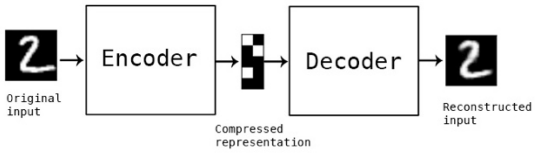

Figure 1: Auto-Encoder representation

We will the extract the encoder, encode the training set and finally cluster it using KMeans clustering from the Sklearns library of python.

4.3 Auto-Encoder with GMM Clustering

In this method we will use the same autoencoder that we created in the previous part involving KMeans clustering but use GMM i.e. Gaussian Mixture Model instead of KMeans clustering. 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 belongs to, using the GaussianMixture.fit_predict method.

5 Result

5.1 KMeans Clustering

The testing accuracy for KMeans clustering method is: 54.46% with n_clusters =10, max_iter = 29, init = "random" and random_state = 5.
The confusion matrix for KMeans clustering is:

```
[[538   3   1   0  41  73   6  29 104 205]
 [  2   0   0   0  10  13   1 843 119  12]
 [  7   3   0   0 577  53   3   5  27 325]
 [ 13   2   0   0  14  70   2 451 387  61]
 [  1   6   0   0 645  39   2  18 163 126]
 [  0   2 275 122   0 596   1   1   0   3]
 [ 99   1   0   0 336 107  15  21 109 312]
 [  0   0 880  35   0  85   0   0   0   0]
 [  1 416  32   1  65  73 395   5   3   9]
 [  0   2 157 801   2  31   1   1   4   1]]
```

Figure 2: Confusion matrix for KMeans Clustering

5.2 Graph for Auto-Encoder

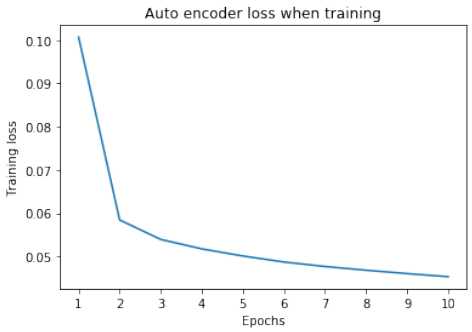

Figure 3: training loss and validation loss vs number of epochs

5.3 Auto-Encoder with KMeans clustering

The testing accuracy for Auto-Encoder with KMeans clustering method is: 57.53% with epochs =10, max_iter =27 and n_clusters =10.
The confusion matrix for Auto-Encoder with KMeans clustering is:

```
[[861   2   9  16   5   2 101   0   4   0]
 [  2 972   1  18   3   0   4   0   0   0]
 [ 14   0 756  13 111   0 105   0   1   0]
 [ 29   4  10 882  46   1  24   0   4   0]
 [  1   0  63  14 837   0  84   0   1   0]
 [  0   0   0   1   0 954   0  20   1  24]
 [126   0  51  23  57   0 734   0   9   0]
 [  0   0   0   0   0  11   0 972   0  17]
 [  4   0   2   2   6   2   4   3 977   0]
 [  0   0   0   0   0   4   1  38   0 957]]
```

Figure 4: Confusion matrix for Auto-Encoder with KMeans clustering

5.4 Auto-Encoder with GMM Clustering

The testing accuracy for Auto-Encoder with GMM clustering is: 55.72% with epochs =10, max_iter = 28 and n_components = 10.
The confusion matrix for Auto-Encoder with GMM clustering is:

```
[[ 45   0   9  50  99   0 679  33  77   8]
 [ 77   0   1   2   4   0   8  24 884   0]
 [  6   0   4 727 169   0  14  73   7   0]
 [201   0   0  15  20   0 159  29 574   2]
 [ 14   0   8 694  79   0  72  74  59   0]
 [  3 564   3   0   1 143   0   0   0 286]
 [ 45   0  17 426 176   0 209  83  39   5]
 [  0 958   0   0   0  34   0   0   0   8]
 [ 55   5 774  48  23   0   2  32   0  61]
 [  3 166   1   0   1 813   1   3   0  12]]
```

Figure 5: Confusion matrix for Auto-Encoder with GMM clustering

6 Conclusion

In this project we performed cluster analysis on the Fashion MNIST dataset using unsupervised learning. We used the KMeans function from the Sklearns library in python as the base and also implemented autoencoder with KMeans clustering and autoencoder with GMM clustering. We found the accuracy and confusion matrices of these methods

Acknowledgements

I am extremely grateful to Professor Sargur Srihari for teaching all the necessary concepts related to logistic regression. I would also like to thank all the TA's of the course for helping me at every step during the course of this project.

References

[1] Professor Sargur Srihari's lecture slides.
[2] Autoencoder for KMeans (https://www.kaggle.com/s00100624/digit-image-clustering-via-autoencoder-kmeans)
[3] Fashion-MNIST with tf.keras (https://medium.com/tensorflow/hello-deep-learning-fashion-mnist-with-keras-50fcff8cd74a)
[4] Keras Documentation(https://keras.io/)
[5] Sklearns (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html)
[6] Accuracy: from classification to clustering evaluation (https://smorbieu.gitlab.io/accuracy-from-classification-to-clustering-evaluation/)

[7] KMeans clustering (https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1)