4th Assignment

Unsupervised Learning - Clustering

Machine Learning and Natural Language Processing

Christakakis Panagiotis

ID: aid23004

Postgraduate Studies Program in "Artificial Intelligence and Data Analytics", University Of Macedonia

Date: 15/12/2022

Unsupervised Learning - Clustering

Contents

Figures

Pictures

1 Introduction

In this assignment we'll try to implement cluster analysis on fashion MNIST dataset using unsupervised learning algorithms.

This dataset contains 60.000 training and 10.000 test images of various clothes and each one of them are assigned to one of the following labels:

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

Each image is 28x28 in grayscale and with K-Means, DBSCAN and OPTICS algorithms we'll try clustering. The evaluation of each clustering algorithm will be compared using Silhouette Coefficient, Calinski-Harabasz score, Davies-Bouldin score and Adjusted Mutual Info score. These metrics help us to choose the best parameters of each algorithm.

Using an auto encoder, the images will be compressed and then recreated-upscaled into the starting pixels. Last but not least, the results will be compared for normalized pixel values in 0 – 1 scale and values produced by the encoders part.

Autoencoder is an unsupervised machine learning and data compression algorithm that sets the target values to be equal to the inputs by using back propagation. It has two major parts, encoder and decoder. The job of the encoder is to compress the input data to lower dimensional features. For example, one sample of the 28x28 fashion MNIST image has 784 pixels in total, the encoder we built can compress it to an array with only ten floating point numbers also known as the feature of an image. The decoder part takes the compressed features as input and reconstruct an image as close to the original image. Autoencoder is unsupervised learning algorithm in nature because it takes only the images themselves and not the labels during the training period. The autoencoder built is one fully connected symmetric model.

2 Methods

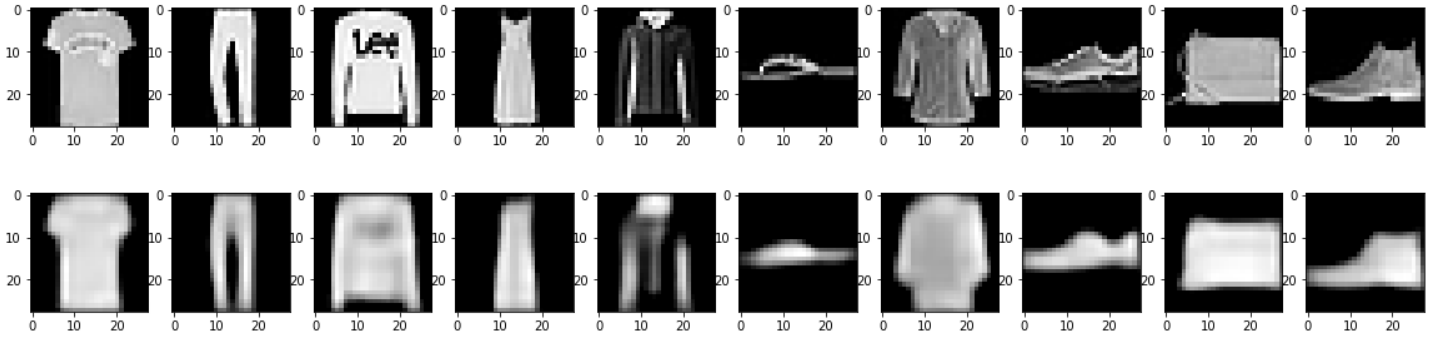
For implementing our methods we'll use python and appropriate libraries. Some of them are numpy, sklearn, tensorflow, random, keras etc. The experiments with 10 epochs run on CPU Intel Core i9-10900K, RAM 128GB DDR4, GPU RTX3090 and the experiments that are shown on Python Notebook run on Google Colab.

First of all, we need to import the fashion MNIST dataset. Then, splitting of data in training and validate is done through train_test_split function. We'll start the comparison with pixel values normalized in 0 – 1 scale.

The best number of clusters for K-Means is chosen by comparing the four metrics we calculated. We evaluate our metrics as following:

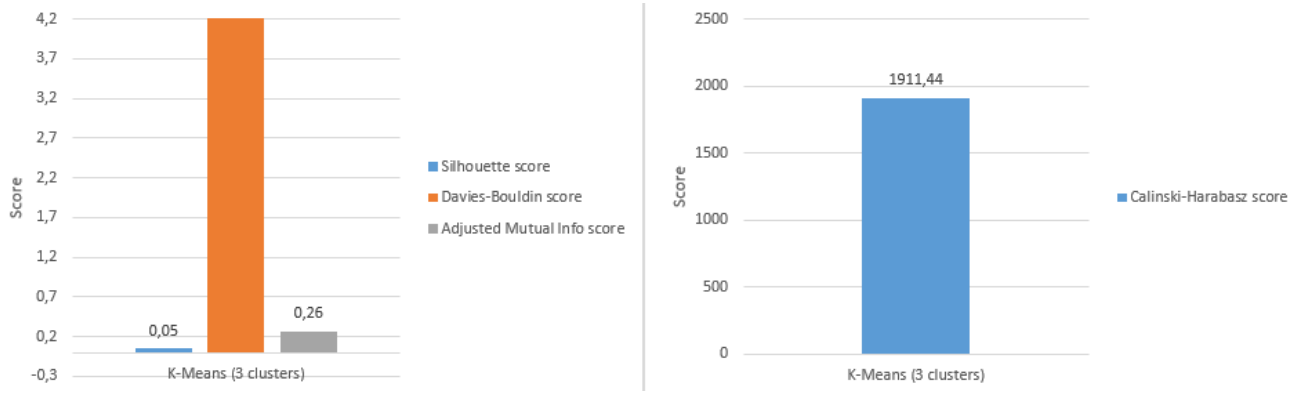
1) Silhouette Coefficient: Number 1 denotes non-overlapping clusters.
2) Calinski-Harabasz score: Higher value means dense and well separated clusters.
3) Davies-Bouldin score: Close to 0 means cluster similarity.
4) Adjusted Mutual Info score: Measures the agreement of the two clusterings.

Picture 2.1: Images from each class and the reconstructed ones by the autoencoder.



***Comments*:** In the above picture we can see that even with 3 epochs the CNN Autoencoder tries to imitate each image at a fairly good point.

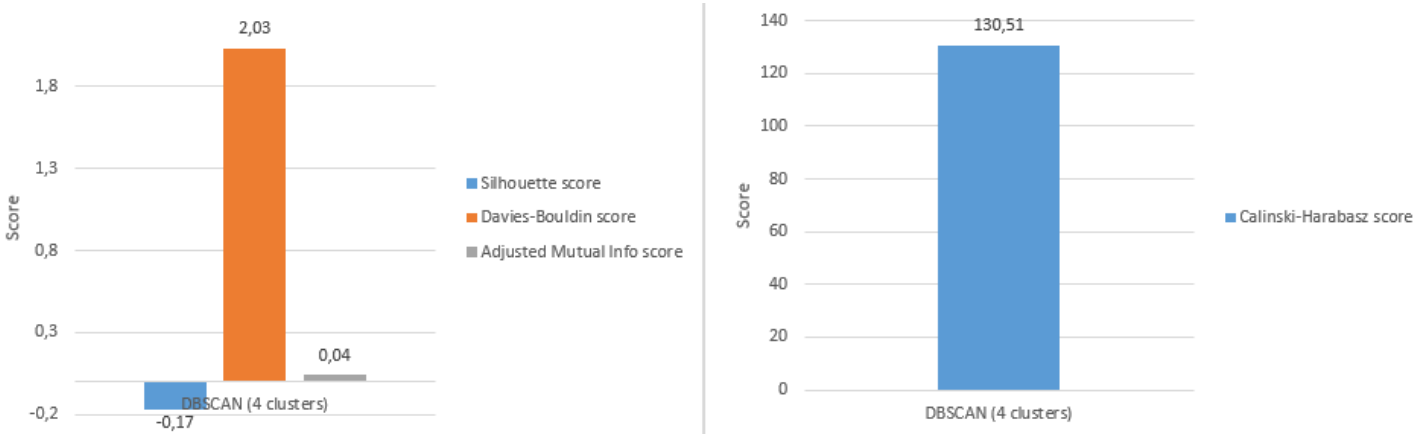
Figure 2.1: Bar chart comparison of clustering metrics for K-Means (3) - Normalized



***Comments*:** The above bar chart shows Silhouette score is low and far from 1, while Calinski-Harabasz score is quite high. Davies-Bouldin Score is not close to 0 and AMI has a value of 0,26. Here we have conflicting scores but they were the best ones. So we chose 3 number of clusters for K-Means.

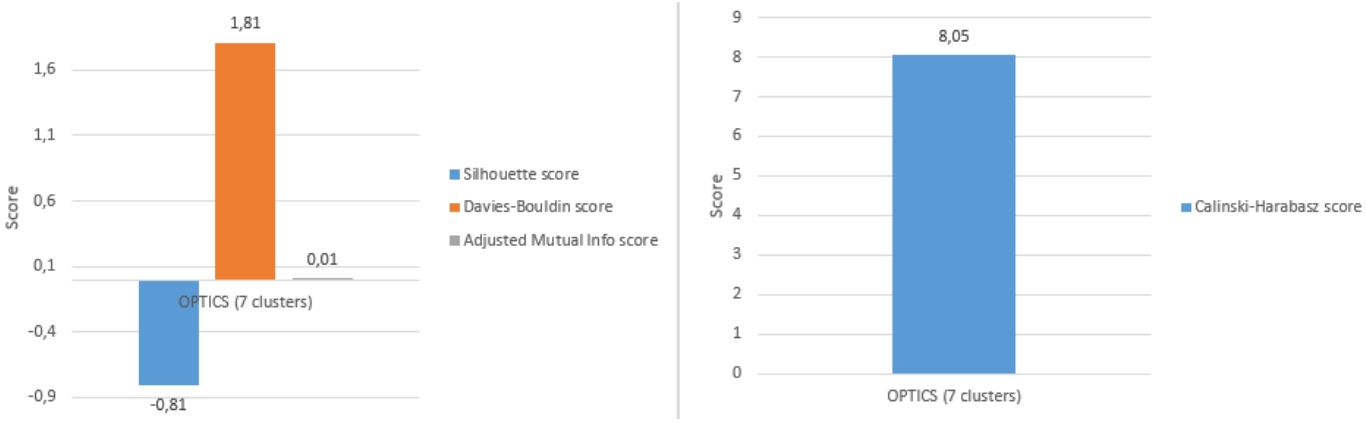
For both OPTICS and DBSCAN, we run a loop in order to find the best parameters that give the best possible clustering scores. For OPTICS those are eps:0.5, min_samples:7 and for DBSCAN are eps:1, min_samples:9.

Figure 2.2: Bar chart comparison of clustering metrics for DBSCAN (4) - Normalized



***Comments*:** Again, the calculated scores were far from surprising and the values we got had contradictions. The fact that Silhouette Coefficient, which is a very popular one, has negative values shows that DBSCAN isn't good for our dataset. DBSCANs best hyperparameters revealed that the optimum number is 4 clusters.

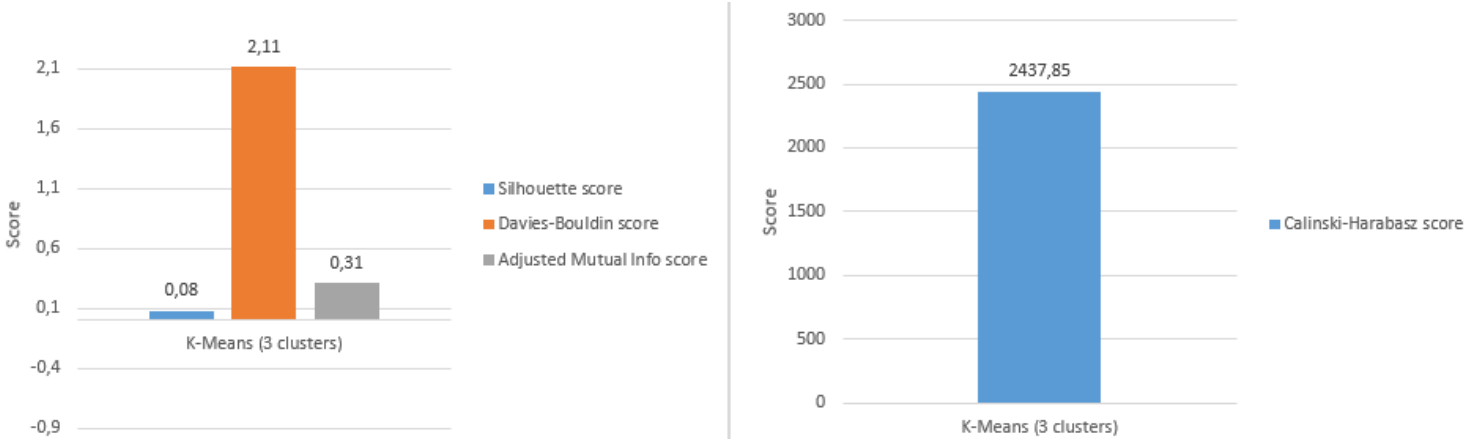
Figure 2.3: Bar chart comparison of clustering metrics for OPTICS (7) - Normalized



***Comments*:** Nothing changed with OPTICS algorithm and its scores. As we can see, Silhouette score tends to go even lower, while Calinski-Harabasz score went all the way down to 8,05. Davies-Bouldin score is also far from 0. OPTICS went with 7 clusters as optimum number.

Our experiment continues with the same test, but now we won't normalize our pixel values. So, the results with values produced by the encoders part are the following:

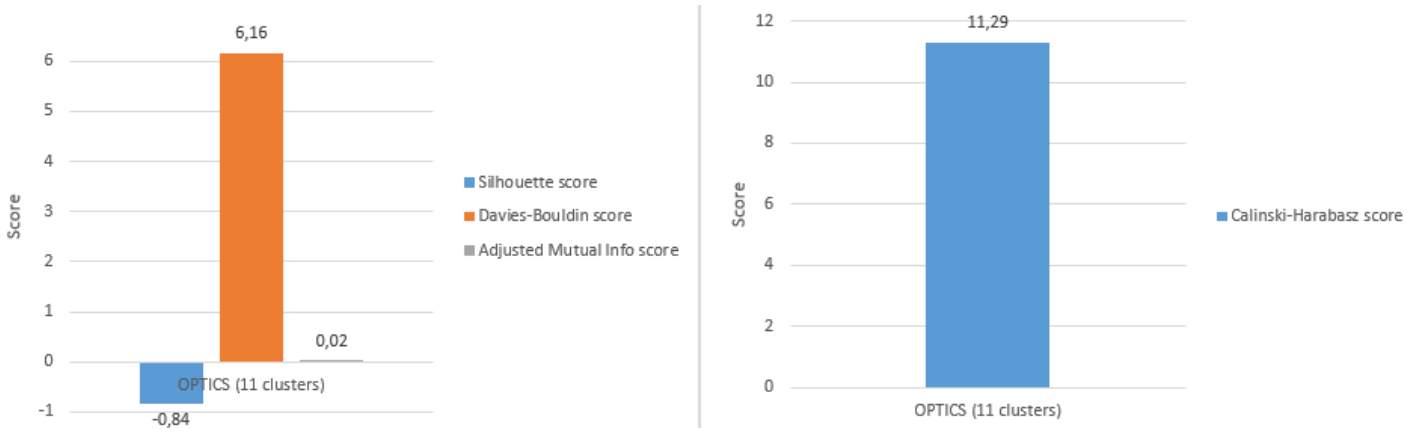
Figure 2.4: Bar chart comparison of clustering metrics for K-Means (3) - NonNorm



***Comments*:** The above bar chart shows a fairly high value for Calinski-Harabasz score, but as expected from the previous experiment the other ones are not ideal. Again K-Means showed that the optimal number of clusters are 3.

Again, for OPTICS the best parameters that give the best possible clustering scores are eps:0.5, min_samples:7. For DBSCAN none of the metrics could be calculated, except for Adjusted Mutual Info score.

Figure 2.5: Bar chart comparison of clustering metrics for OPTICS (11) - NonNorm



***Comments*:** Without a doubt we can say that none of the scores really gives us a good-looking clustering. All of them are not close to the numbers we want and this time OPTICS number of clusters were 11.

3 Conclusions

The main task was to cluster images and identify them as one of many clusters and to perform cluster analysis on fashion MNIST dataset using unsupervised learning. The model's effectiveness was measured by metrics calculated by sklearn function.

In conclusion, our best performing model was implemented with K-Means without the pixel values normalized. The model chose that the optimum number of clusters is 3. With that being said, we have to note that neither of the metric results were good. This can be attributed mainly to the CNN architecture that was chosen.