University of Macedonia

Applied Computer Science

Assignment on Clustering

Konstantinos Pasvantis

December 22, 2022

Contents

List of Figures

1 Introduction

In this assignment we will talk about clustering with different methods. We will make use of an usupervised machine learning model that helps to deal with the problem of big amount of data in a given dataset. The model we are talking about is this of Convolutional Neural Network autoencoders. The general idea behind this simple, yet helpful model, is to compress the data of a given dataset into chunks of code that are still interpretable by the machine. We will talk in more detail about this part in a later section.

The main purpose of this assignment though, is to test different clustering methods in the known dataset of fashion-mnist. The dataset consists of 60.000 train images and 10.000 images that can be separated to 10 different classes. We will first cluster the images using the chunks of code for every image the model of autoencoder generates Then we will compare the results when we don't make use of the model.

More speciffically, after we load the dataset and split it to train, validation and test data, we will define the architecture of our convolutional neural network. We will then select only the part of the CNN that compresses the images (more details about this one will be given later) to cluster the dataset. We will then use the original values of images normalised in [0,1] for clustering and we will compare the clustering results from both of the techniques. The way we will compare them is by using four different metric scores to calculate the effectiveness of each clustering mehtod, namely silhouette score, calinski harabasz score, davies boulding score and v measure score. The clustering methods we will use in this assignment are, the old-time classic, k-means, spectral clustering, and agglomerative clustering.

We have to denote that, evenif we know the number of classes our dataset has, we will not use 10 clusters in order to cluster the images. After we compare the best scores for each number of cluster, we will use the one that produces the best results in this case. The way that we will comapre metric scores will become clear in the next section of the assignment. Also, it is normal for the results to not be produced the same way they were produced the first time, so, in this assignment we will report the results of our first time.

2 Definitions and Methods

In this part we will talk more about the methods we will use to produce our results .

2.1 CNN autoencoder

The CNN autoencoder is nothing more that a convolutional neural network that encodes the data into smaller chunks of code (we already refered to this), and then using these chunks of code, reconstructs the images, as close as it can be to the original. The first part of compressing the images is made by the encoder part, and the reconstruction of the images using the chunks of code is made by the decoder part.

The encoder, the first layers of the convolutional neural network, decreases the dimension of images using the same methods we introduced in the convolutional neural network assignment. They scan every image pixel by pixel in order to decrease the dimensionality of our dataset, so that the images can be better manipulated from our system. They search for values of each image that can be skipped in order to keep only the important values, the values that, when used, are easier to classify the specific image. The image that stores the encoder can't be interpreted easily by humans, but are more than enough for the decoder to reproduce the initial image.

The decoder now, which is the last layers of the neural network, generates the initial images based on the values of the compressed images. The way it does this is to read every value stored in the compressed image, and then increasing the dimensionality in each layer, step by step, until it reaches the original size, every time comparing the original image to the reconstructed image, in a way that it minimizes the loss.

2.2 Clustering Algorithms

As we said, we will make use of three different clustering methods, k-means, spectral clustering and agglomerative clustering.

First of all we will talk about **k-Means**. K-Means uses as fixed number of clusters to begin with, in order to group similar data instances of a dataset together. By the time a fixed number of clusters is given (we will refer to this number as k), this algorithm selects randomly k points or centers, not necessarily from our dataset, in order to assign each instance of our dataset in a single center, its closest one. Once all of the instances are assigned in a center, our clusters are formed. Now the algorithm calculates the total variance within each cluster, in order to place the cluster in a better spot. This spot is obviously where the total variance within each cluster is minimized. Once new centroids are placed, the algorithm assigns again each instance of the dataset in its closest centroid. The above iterations are repeated until the variance of each cluster, hence the clusters, are the same for two continuous iterations.

Now, **spectral clustering**, is a little more trickier than k-means. It uses k-means, but this time it is not clustering the instances themselves, but the features of each instance, that are calculated using the eigenvectors of the Laplacian Matrix that is generated from the similarity graph between all datapoints of the dataset. With that being said, the simililarity graph is created, simply using k-nearest neighbors graph. Each instance of a dataset is connected to some of their neirest neihgobrs. Once this graph is created, the Laplacian matrix is calculated from this graph, and now the clustering method computes the eigenvecotrs of this matrix to create a feature vector for each object. Finally, if we want to cluster the dataset into k different clusters, we use k-means in every feature created, by making use only the first k eigenvectors we calculated.

Last but not least, **agglomerative clustering** is a clustering algorithm that works by merging all data points step by step until they all belong in a single cluster. It may sounds a little bit confusing in the begginging but it isn't. At first, every data point is a single cluster. Next, using a distance metric, it searches for the two datapoints that are closest with each other, and merging them into a single cluster. Then, it calculates all of the distances between clutsers, and merges these with the lowest distance into a single cluster. The tricky part is that, when a cluster with more than one instances is determined, its distance from the other clusters is calculated usually (in this assignment also), by calculating the minimum distance between each instance of the cluster with each other instance of the other clusters. This process is repeated until all data points are merged into one cluster. When this cluster containing all datapoints is formed, we can tell the algorithm to use a specific number of clusters, to cut off the big cluster containing all datapoints, into the number of clusters we specified.

2.3 Metric Scores

The metric scores we used in order to find which number of clusters fits better in our dataset are silhouette score, Calinski - Harabasz score, Davies Boulding Index score and v measure score.

Every metric score is used in order to obtain information on how well the clutsering method we used work. We do not explain the maths behind each score and how they are calculated, but we give a brief explanation about when a value of a single metric score is better than another.

- **Silhouette score** is working best with linearly separable data, while working with very high dimensions of datapoints may cause severe errors. The value of Silhouette score is between -1 and 1, with 1 indicating the perfect clustering.
- **Calinski - Harabasz score** measures how similar an object is to the cluster it is assigned to, compared to other clusters formed. Here the value of the Calinski - Harabasz score does not have any limits. The higher the value, the better the clustering.
- **Davies Boulding score** calculates the similarity of each cluster with the cluster that it is most similar to it. Apparently the lower the value for this metric score is, the less similar a cluster is with its most similar cluster, hence, we have better partitioning and better clustering.
- **V measure score** is another name for Normalised Mutual Information. It measures the average of homogenity and completeness. Homogeninty measures how much the instances of each cluster are similar to each other, and completeness measures how much similar instances are put together in the same cluster from the algorithm. It can take values ranged from 0 to 1, with 1 indicating that the clustering is both complete and homogenious.

3 Results produced by k-Means

3.1 Encoded Images

First of all we have to construct the arcitecture of the autoencoder. We build a convolutional neural network and then we extract only the part of the encoder, the part that compress the images into a code chunk. Here we can see the original images of the dataset we are using with their restored image below them.

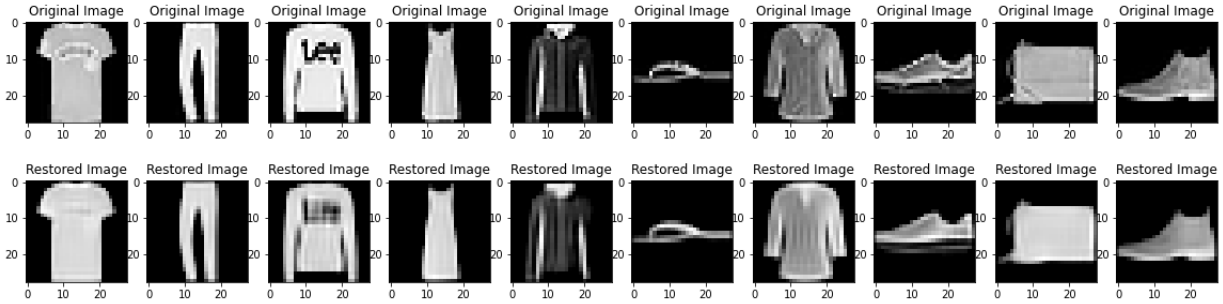

Figure 1: Original and restored images provided from the autoencoder

First of all we will begin with the code chunks the autoencoder produces to cluster the original images. To choose the right number of clusters we will use for k-means we conduct a small research to give us a hint. We calculate all four metric scores for clustering for all possible values from 3 to 10 clusters. The values of the metric scores can be obtained in the the excel file. The fact that Calinski- Harabasz index score was in a different scale than the other metrics was a problem in terms of making a graph, so we used a min-max scaler function in order to create a dataframe with values between 0 and 1. Using these values we created the below graph:

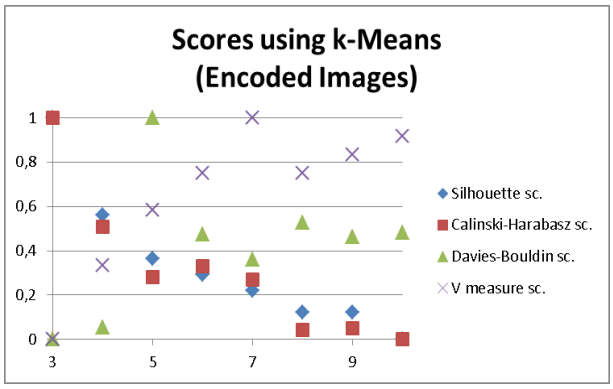

Figure 2: Metric scores using k-means on the encoded images

To pick the best number of clusters that fits our data we must take into account all four possible metric scores. For this clustering method, we can see that the best DCalinski- Harabasz value is obtained when we used 3 clusters, Silhouette sc. is also the best value, like Davies-Bouldin score, and while V measure sc. takes it best value for 8 clusters, the other scores are relatively low when compared with this of 3 clusters. So, for this model we choose 3 clusters, and when using this model to cluster our encoded images we can see that we have decent results for clustering.

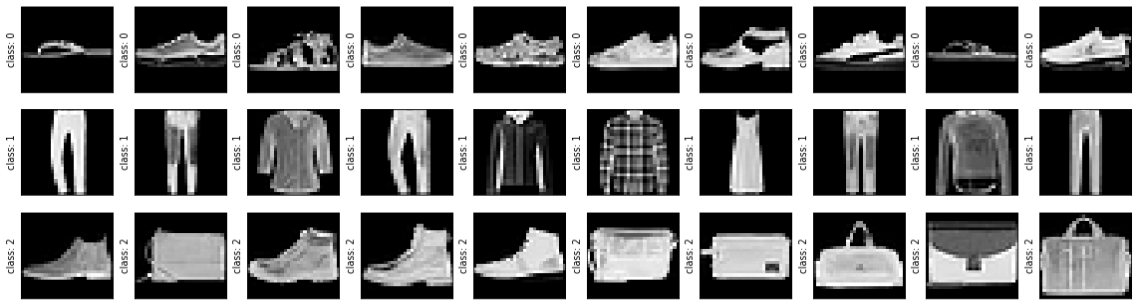

Figure 3: Clustering results using k-means on the encoded images

3.2 Original Images Normalised

We once again present the distribution of our four metric scores using the graph: Here we can see that

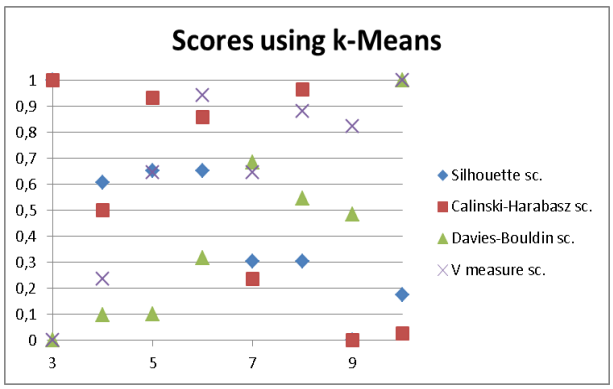

Figure 4: Metric scores using k-means on the original images

the best results are also produced when we use 3 clusters, with Silhouette and Calinski - Harabasz scores taking the highest values, and Davies - Bouldin score taking the lowest value when compared to another number of clusters. V measure score is taking the lowest value, when we wanted for this to be higher than

the others, but when compared the other metric scores we can see clearly that we will pick 3 clusters. With this number of clusters we use k- means and we are taking the below results:

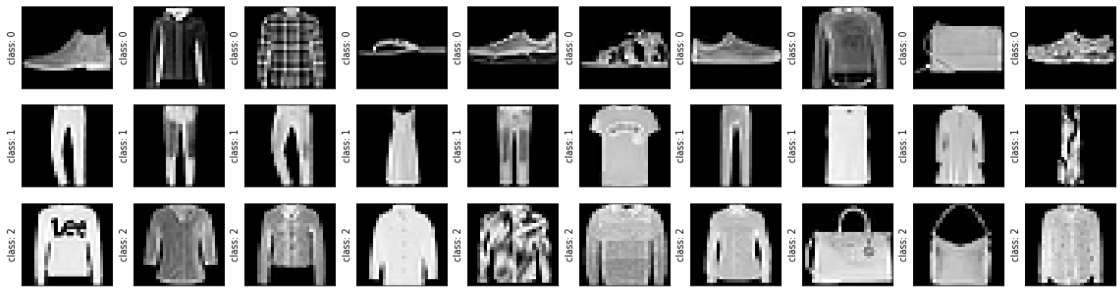

Figure 5: Clustering results using k-means on the original images

As we can see k- means worked pretty well for both cases. We used the same number of clusters for them and the results were fine. We can notice here that, when we use the encoded images provided from the autoencoder, the results for 3 and 4 clusters are clearly better, but when we raise the number of clusters, the choice for this number seems to be more difficult.

4 Results produced by Spectral clustering

4.1 Encoded Images

Here, when we are comparing the metric scores for this clustering method we can notice that this method is far more worse than k-means using encoded images. Silhouette score is negative for every number of clutsers we are using, V measure score is below 0,07, and Calinski- Harabasz score has values that are below 1. If we had to choose a ceratin number of clusters for this method, we would pick 6 or 7, but even with this number of clusters, the method seems to be generally bad.

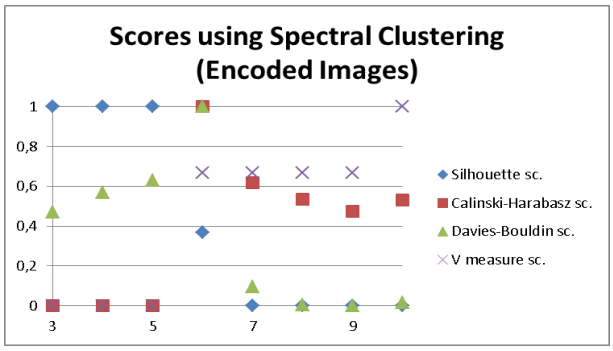

Figure 6: Metric scores using spectral clustering on the encoded images

So with that being said, we use 7 clusters for this model and the clustering results are shown in the next page. We can clearly see that the images aren't clustered in a proper way , and we can see that some classes have no more than 4 pictures.

4.2 Original Images Normalised

Apparently, when we tried to use spectral clustering for original images, the scripts was stuck for a decent amount of time, even when we tried only 3 clusters. Here we can see the difference that dimensionality reduction makes when we are speaking for high dimensions dataframes.

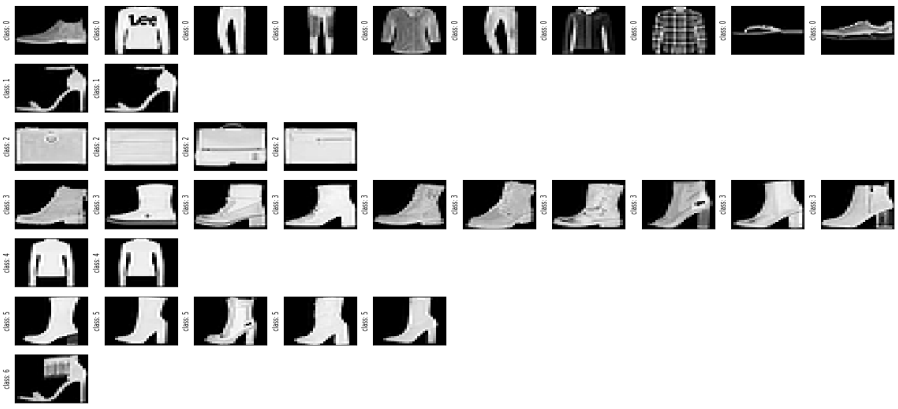

Figure 7: Clustering results using spectral clustering on the encoded images

5 Results produced by Agglomerative clustering

5.1 Encoded Images

Finally, we get to the last clustering method we tried, the agglomerative clustering. Once again we can see that the best option for this method is to use 3 clusters, while Silhouette, Calinski-Harabasz, and Davies -Bouldin scores are better than these of any other number of clusters, and V measure score is not that bad, even if it is the lowest of all v measure scores for this method. So, with 3 clusters the clustering

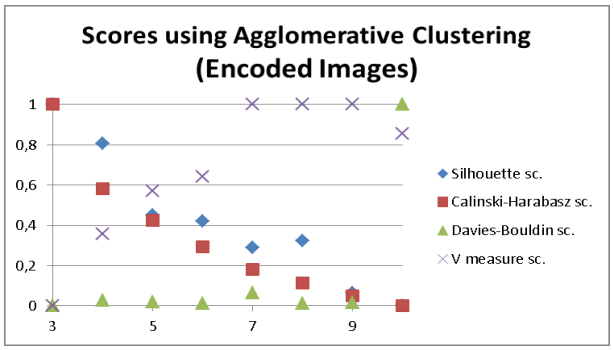

Figure 8: Metric scores using agglomerative clustering on the encoded images

looks something like this :

5.2 Original Images Normalised

This time, the clustering method gave results even if we used the original images of the dataset. Once again the best number of clusters is 3. Despite the fact that Silhouette score has better value than this of the clustering when we used the encoded images, all the other metric scores are better.
Generally speaking, this model is worse than the previous one, for all possible numbers of clusters we used.But for 3 clusters it seems to work like the rest clustering techniques.

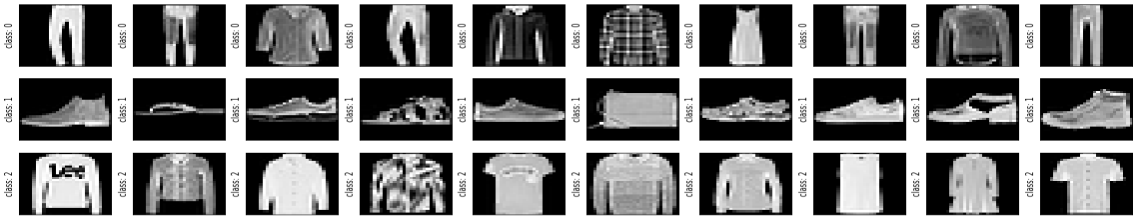

Figure 9: Clustering results using agglomerative clustering on the encoded images

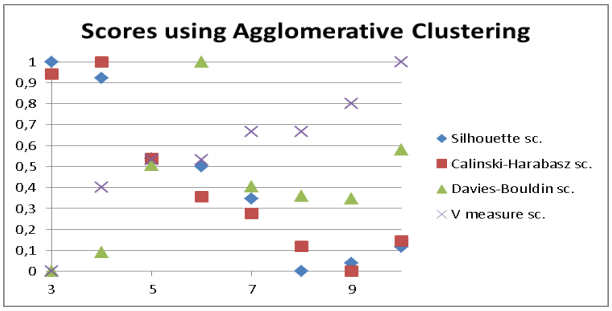

Figure 10: Metric scores using agglomerative clustering on the original images

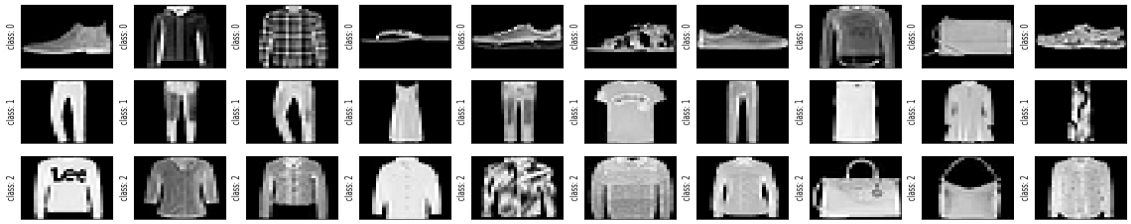

Figure 11: Clustering results using agglomerative clustering on the encoded images

6 Conclusion

In this assignment we tested three different clustering methods, k-means, spectral clustering and agglomerative clustering. All of these methods produced some remarkable results when used for both the images produced by the encoder and the original images of the fashion-mnist dataset, with an exception of spectral clustering, the only algorithm that could not cluster the original images.

To summamrize, when we used k-means, the results were equally fine for both original and encoded images. The best results came from the encoded images, giving us the general idea that dimensionality reduction can help a lot when we have to work with images. When we used spectral clutsering, the resuls can not even compare with these produced by k-means or agglomerative clustering. Agglomerative clustering on the other hand, seem to be even better than k-means.

If we want to compare the usage of the models in a much more general talk, we can say by far that spectral clustering is the worst between the 3 models, and the best model to clutser the images, both original and encoded, is this of agglomerative clustering.