Spectral Clustering

Christos Psychalas
Contact: chrispsyc@yahoo.com

16 January, 2024

Abstract

In this assignment we implement the spectral clustering algorithm, although not from scratch, and continue with its application to the known handwritten digit dataset, MNIST. The basic steps of the algorithm are presented and followed using scikit-learn's modules. Then we perform an exploratory analysis on the MNIST images, to assess the algorithm. The main point is to measure some metrics to assess the clustering quality, but also confirm their validity based on visual results. More specifically, we measure two clustering, intrinsic, metrics, the Calinski-Harabasz and the silhouette score. The final results yield $\sim 70\%$ accuracy on the test set. Even though the clustering is an unsupervised learning process, we leverage the labels provided in the dataset to validate our metrics. For our surprise we find out that the higher scores do not correspond to better clustering results. We conclude that the Calinski-Harabasz is a metric that can be used, but for our case, flipped, that is the less the score the better the clustering.

1 Spectral Clustering

The spectral clustering algorithm consists of the steps of spectral embedding (dimensionality reduction) and the application of a simple clustering algorithm, such as KMeans. There are many dimensionality reduction methods, but not many of them take into consideration the possible manifold of the data. As presented by the authors, [1], the spectral embedding, or the Laplacian Eigenmaps, as they call it, give us a solution that takes into account the intrinsic geometry of the data, with low computational cost (sparse eigenvalue problem). The main algorithm is as follows:

1. Construct the graph of the data (either using, ϵ-neighborhoods, n- nearest neighbors or their combination).
2. Choose weights (W) for the graph's connections (either $1-0$, if connected or not, or use the Radial Basis Function kernel).
3. Calculate the laplacian $L = D - W$, where D is the degree matrix ($D = \sum_j W_{ij}$).
4. Compute the eigenvectors and eigenvalues of the generalized problem: $L\boldsymbol{y} = \lambda D\boldsymbol{y}$.

The generalized solution is equivalent to the eigen-decomposition of the normalized laplacian (random walk) [2]. The solutions $\boldsymbol{y}$ are ordered according to their eigenvalues, in ascending order. The first eigenvalue has zero value and corresponds to the $\mathbf{1}$ eigenvector. The data to the new space do not need to be projected onto our eigenvectors, rather the obtained eigenvectors are the projected data, so $\boldsymbol{y}$ is an $n \times m$ matrix, where n is the number of samples and m the projected dimensions ($m > 1$). The trivial solution $\mathbf{1}$ is neglected as it turns all the graph's connections to one, but this is the case assuming our graph is connected. For the case of V connected graph components, we will have V trivial solutions, that need discarding.

So far, this was the embedding algorithm, but the spectral clustering has one last step. That is use a clustering algorithm such as KMeans and perform clustering on the acquired - embedded data from the spectral embedding. For the whole process we use the scikit-learn's *kneighbors_graph* in order to compute the adjacency matrix and then we either use it as such or apply RBF's weights to form the affinity matrix. Then we solve the generalized problem described above using scipy's *linalg* module. Finally, we apply *KMeans* from scikit-learn's modules to obtain the clusters.

2 MNIST Application

The MNIST dataset (http://yann.lecun.com/exdb/mnist/) is a handwritten digit image dataset including all digits from 0 to 9. They are gray-scale images of size 28×28. The original dataset contains $60,000$ training and $10,000$ test samples. In this exploratory analysis we apply the spectral clustering algorithm in order to test whether it is possible to find clusters that correspond exactly to each of the ten digits, as they are highly entangled. We won't try to find the best solution to the problem, as we could simply add more than ten clusters and then merge them visually, rather we take the number of clusters as fixed (10 clusters) and observe the results obtained using different parameters. Generally, the clustering is used for unsupervised learning, that is we don't know the labels of our data and try to find groups of them, so we can later create the labels. This analysis could be helpful, if it yields good results of course, to create a pipeline using spectral clustering in order to label automatically each new image intended for the dataset.

First of all, we start by sub-sampling the dataset so we can have some visual results and of course reduce the runtimes. We keep 150 samples from each digit image, from which the 50 (per digit) images are used as test samples. Totally we end up with 1000 training samples and 500 test samples. The number of features per image is 784, so our data look like this: (n_samples, n_features) = (1500, 784). We now proceed with the visualization process. In this part we want to lower the dimension of the data so we can have a 2D representation. For this MNIST application, as we already know the number of clusters - classes, we search for the best visually result. In other cases we would like a representation that would be able to give us an insight on the data's structure - possible classes. We will try two different algorithms, the spectral embedding, as implemented above and the t-SNE (scaling data to 255). In the table below we present the parameter search for the two algorithms.

Table 1: Spectral Embedding - t-SNE parameter search

Embedding	Neighbors	Perplexity	Heat Kernel (τ)
Spectral	$[4, 8, 10, 15, 20, 40]$	-	-
Spectral	$[4, 8, 10, 15, 20, 40]$	-	$[10, 20, 40, 100]$
t-SNE	-	$[5, 10, 20, 30, 40, 50]$	-

Given the search above we end up with the following best cases for each algorithm, as presented in the figure below. As can be seen t-SNE (for $perplexity = 50$) gives a better visualization (the classes are more spread) than the spectral embedding (for $neighbors = 15$, no heat kernel applied). So we keep the t-SNE algorithm to visualize our clustering results later on. All plots for the parameters searched above can be found in: https://github.com/Xritsos/spectral-clustering.git/MNIST/visualization_plots.

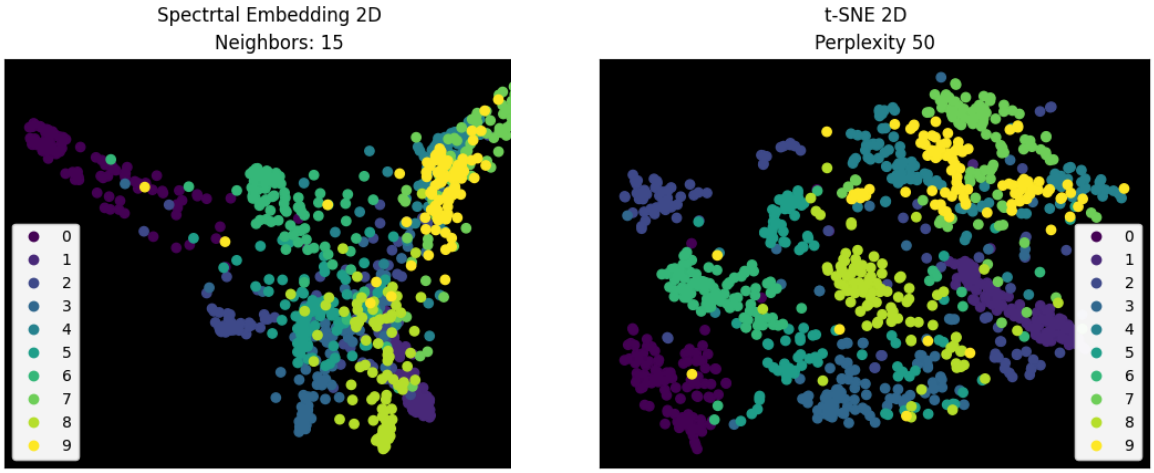


Figure 1: 2D visualization of the MNIST dataset

Next, we proceed with the spectral clustering of the data projected to the 2D space as shown above. These are the steps followed:

- Drop data to 2D space (using t-SNE)
- Apply spectral embedding to the 2D data (increase data's dimensions)
- Use a KMeans clustering algorithm to assign cluster labels on data
- Get clustering scores (Calinski-Harabasz, Silhouette)
- Take 4 cases: max-min Calinski score and max-min Silhouette score
- For each of these cases measure the accuracy on the test set

The idea here is to try and find a mapping of one (or both) of the clustering scores mentioned, to a perfect clustering, that is each cluster contains only samples of one digit, or at least each cluster has one dominant digit. We begin by inserting our 2D data in the spectral embedding algorithm in order to increase their dimensions, so they become as separated as possible. Now we have to scale our data in order to feed them in the KMeans algorithm. Prior to this we split our data to train - test. We should mention that for the process of embedding to 2D space and then during the spectral embedding to higher dimensions train and test data must be together. It is for the process of clustering that we need to split them. Then we check visually the clustering results for each of the four cases and try to map the clusters obtained to a digit (KMeans labels do not correspond to digits). Finally, we calculate the accuracy score for each case. One could say that it is obvious that the best scores would yield the best clustering. As we will see that is not the case. In the table below we can see the search space for the spectral embedding step (data to higher dimensions).

Table 2: Spectral Embedding

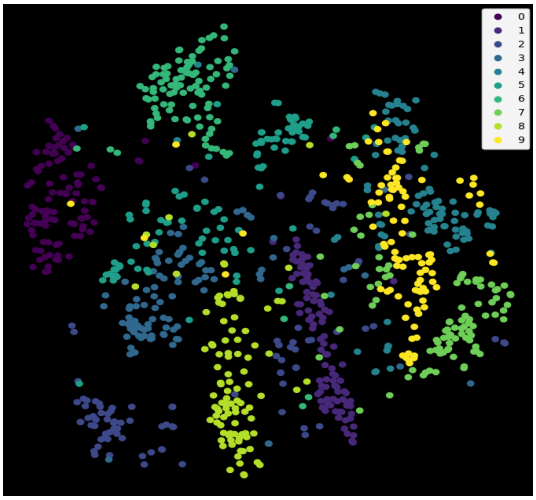
Neighbors	Dimensions	Heat Kernel (τ)
$[8, 10, 15, 20, 40, 50]$	$[2, 3, 5, 8, 10, 15, 20]$	-
$[8, 10, 15, 20, 40, 50]$	$[2, 3, 5, 8, 10, 15, 20]$	$[5, 10, 20, 40, 100]$

The scores for all of the above combinations can be found in https://github.com/Xritsos/spectral-clustering.git/MNIST/heat_off.csv (and heat_on.csv). Now we only present the min and max scores and their parameters in the table below.

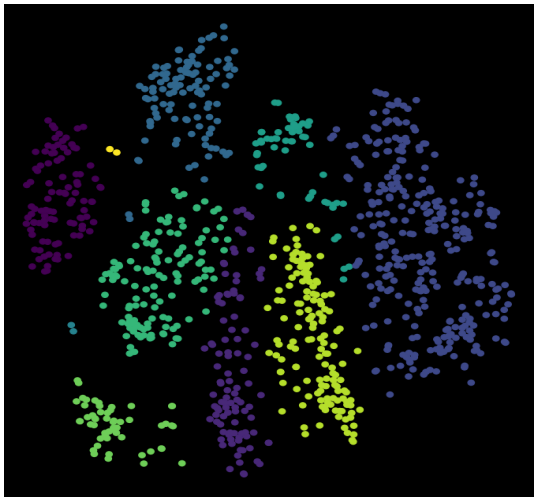
Table 3: Minimum and Maximum Clustering Scores

Score	Value	Neighbors	Dimensions	Heat Kernel (τ)
Calinski (Max)	$83,149$	15	2	5
Calinski (Min)	142	15	20	100
Silhouette (Max)	0.849	8	2	5
Silhouette (Min)	0.354	15	20	-

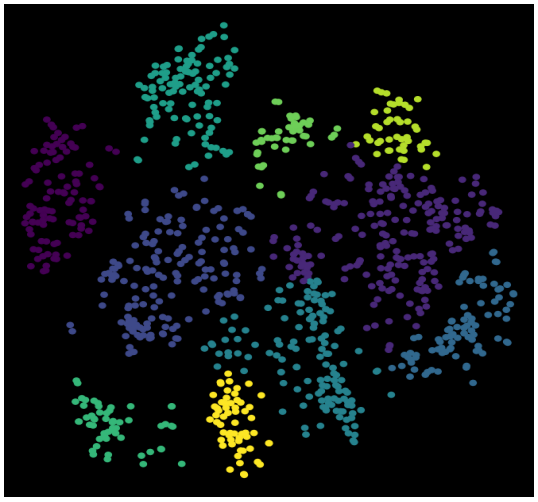
Below we see the clustering results for the cases presented (for the training set).



2D t-SNE original labels

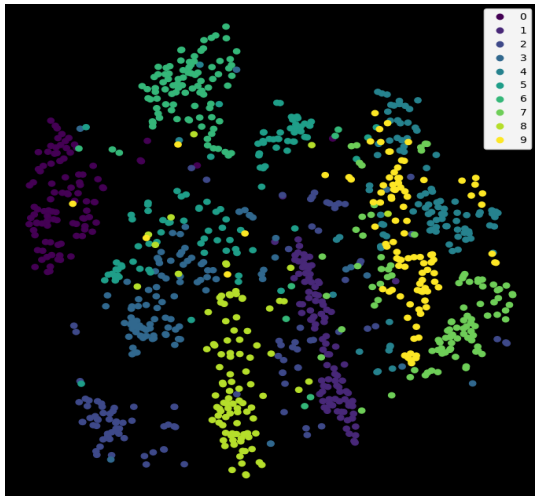


Clustering Result for Max Calinski Score

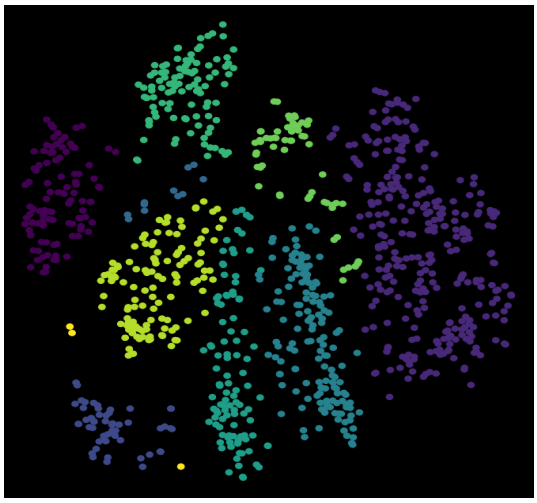


Clustering Result for Min Calinski Score

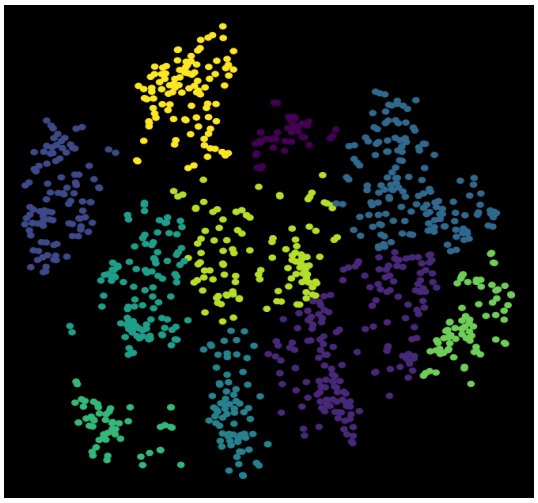
Figure 2: Calinski Scores Comparison



2D t-SNE original labels



Clustering Result for Max Silhouette Score



Clustering Result for Min Silhouette Score

Figure 3: Silhouette Scores Comparison

For the Max Calinski case we can see that there isn't an exact digit - cluster mapping. The clusters 9 and 4 have only a couple of samples, while the digits "4", "7" and "9" are all in one cluster. In this case all we can do is map cluster 9 to digit "6", cluster 4 to digit "2" and then only choose one of the entangled digits "4", "7" and "9" and assign the remaining two to the one left. So, we only keep digit "4" and merge the "7s" and "9s" into "4s", in order to be able to compute the accuracy score. On the other hand for the Min Calinski case we are able to map each digit to one exactly cluster. For the Silhouette scores we come across the same problem, there aren't clusters to match each digit. Specifically for the max Silhouette score we get the clusters 3 and 9 with only a couple of samples and we assign to them the digits "5" and "2" respectively. The digits "4", "7" and "9" are again entangled leading us to treat the problem as we did for the Calinski score, merge "7s" and "9s" with "4s". The case of Min Silhouette score is quite similar. Again the number "9" is mixed with "4s" and we can only merge it and lose one digit. Additionally, the cluster 8 contains a mixture of many different digits and based on the plot we decide to assign it to digit "5". Below there is a table with all accuracies for the cases studied along with the digits we cannot predict.

Table 4: Accuracy Scores

Metric Case	Accuracy	Digit(s) Excluded
Calinski (Max)	82.4%	"7s" and "9s"
Calinski (Min)	69.4%	None
Silhouette (Max)	77.8%	"7s" and "9s"
Silhouette (Min)	72.8%	"9s"

3 Conclusions

Concluding, we examined if there is a correlation between each of the clustering scores to the actual clustering mapping to the digits. Although one would expect to get the best mapping with the highest scores, as seen that's not the case. Specifically, the Calinski-Harabasz score which is supposed to yield higher values for better clustering, in our case it was the lowest score that yielded better results. Also, we notice that the digits "4", "7" and, especially, "9" are highly mixed.

4 References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

[2] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

A Appendix

A.1 Runtimes

Table 5: Runtimes

Process	Samples	Features	Runtime
t-SNE	1500	784	1 *sec.*
Spectral Clustering	1000	-	3.5 *sec.*
Spectral Embedding	1500	784	1.2 *sec.*

Processor: Intel® Core™ i7-8550U, 4 GHz (turbo mode)
RAM: 16 GB
Operating System: Ubuntu 23.10