Machine Learning - Homework 9 Report

Lukas Johannes Ruettgers (2023403372)

December 6, 2023

1 Introduction

Unsupervised Learning collects learning tasks where no labeled data is available. *Clustering* refers to classifying such unlabeled data into an unknown number of clusters. If one has more domain knowledge about the distribution of the data, one might integrate this domain knowledge into a parametric model of probability density functions such as a *Gaussian Mixture Model*. However, the hard cases of unsupervised learning do not offer solid hypotheses in advance. In such cases, model-free approaches are more suitable, that regard only the observed data and usually associate resemblance with proximity in the observation space.

1.1 K-Means Clustering

One of the earliest methods is *K-Means Clustering*, which assumes a fixed number of clusters k in advance and tries to find the k means that most likely represent the assumed k clusters. Given an estimate of the k means position in the feature space, the algorithm iteratively assigns each point to the cluster with the closest mean and updates the means afterwards to the new distribution they generated. However, this method will in general only yield reasonable results for convex clusters. In the usual case of Euclidean distance metrics, this method is only suitable if the hidden clusters distribute their data as hyperspheres.

1.2 Spectral Clustering

To overcome this restriction and generalize to more complex distributions, a more mature *NJW algorithm* developed by Ng et al. regards the data as a graph, which connects each node based on their distance measured in some distance metric. While the criterion in K-Means Clustering for belonging to a cluster depended only on the distance to its mean, this graph representation also considers the connectivity between close data points. In other words, it enables to decentralize the criterion for cluster membership to the local coupling strength between data points. The resulting weight matrix – also referred to as *affinity matrix* – A is then normalized by the sum of its rows. Note that summing along the columns would yield the same result, since the distance metric is symmetrical and consequently is the affinity matrix too. Afterwards, the orthonormal eigenvector basis corresponding to the k largest eigenvectors is computed. Unfortunately, k needs to be fixed here in advance aswell. This will effectively project all vectors in the normalized affinity matrix to the k-dimensional eigenspace that covers the highest fraction of variance. Finally, a classical K-Means Clustering is conducted on this eigenspace and the clustering results are directly transferred to the original vectors.

2 Experiment Objective

This experiment examines the clustering quality of the NJW algorithm on the *MNIST Digits* dataset, which contains 60,000 grayscale 28×28 images of handdrawn digits. In the last experiment, we observed that the first 45 principal components of the MNIST Digits feature vectors express 80% of the entire dataset's variance. Similarly, the NJW algorithm considers only to the k-dimensional eigenspace of the normalized affinity matrix for clustering and neglects the variance that is not expressible in this low-dimensional representation. For that reason, I hypothesize that the performance of the NJW algorithm should not exhibit notable differences when applied only to the first 45 principal components of the MNIST Digits dataset. Over and above, I want to observe the clustering results on digit 3, because the last experiment surprised me with the proximity between the UMAP embeddings of digit 3 and 5. I hope that the analysis of the clustering results unravels some properties of the distribution of the hand-drawn 3 digits that improve my

understanding why the representations of 3 and 5 were so close despite their significant differences in the upper part of their body.

To assess the clustering results, I will also avail to UMAP since it proved to yield high-quality two-dimensional representations that separate the true 10 clusters. I will use the parameters $N_{\text{Neighbors}} = 15, \text{MINDIST} = 0.1, \text{SPREAD} = 1.0$ and a Minkowski distance metric with $p = 2$, because these hyperparameters yielded one of the best results in the previous experiment.

2.1 Implementation Details

2.1.1 NJW Algorithm

I chose to implement the NJW algorithm by myself using functionality provided by `numpy`. Since computing an affinity matrix for 60,000 flattened images turned out to be infeasible for the computing resources of my laptop, I decided to randomly sample 1,000 images from the dataset to proceed with. After these 1,000 images were clustered, for each cluster I computed the sample which was the closest to the mean in the eigenspace of the normalized affinity matrix as a representative mean. Then, I assigned each of the remaining 59,000 images to the cluster whose representative mean was closest to the flattened image in the original feature space. Indeed, this rather simple approach exhibits the exact drawbacks of the K-Means clustering which the NJW algorithm tried to overcome. To stick to the decentralized connectivity approach of spectral clustering, one could also iterate over all 1,000 samples and assign each of the remaining 59,000 images to the cluster to which its closest sample point belongs. However, I sticked with the simple mean-based approach since it would allow to compare the clustering result on the samples and the clustering result on the entire dataset. If the clustering result on the smaller dataset would have a better quality than the clustering result on the entire dataset, this would showcase the superiority of spectral clustering over the primitive K-Means approach. To get a feeling how large this difference might be, I deliberately sticked with the simple mean-based approach to cluster the remaining 59,000 images.

Since the original pixel values were sparse in many outer regions of the image and mostly contained values close to 0 (black), it was crucial to ensure the numerical stability because small rounding errors in the close-to-zero regime could either flip the sign of values or in general aggravate the accuracy of arithmetic operations. Furthermore, a treatment of this case was necessary to ensure valid input for the square root or logarithmic function. To that end, I chose an absolute zero tolerance `ZERO_TOL`. Before performing aforementioned critical arithmetic operations, I clipped small values undercutting this threshold off and set them equal to `ZERO_TOL` or equal to 0 according to the current mathematical operation.

In their paper, Ng et al. propose to compute the weights for the affinity matrix as $a_{ij} = e^{-\frac{|x_i - y_i|_2^2}{2\sigma^2}}$, where σ is a constant value across all dimensions. However, as mentioned above, the raw fixel features exhibited highly different scales. Because the K-Means algorithm with the Euclidean norm weighs distances equally along each dimension, this would attribute an unbalanced weight to dimensions with very high and small fluctuations. For that reason, I computed the standard deviation for each dimension and normalized the distance along each dimension before computing the Euclidean distance. To avoid that small noise in the outer pixel value features was weighted too high, I again capped off the standard deviation at `ZERO_TOL`. Because the distances still took very large values due to the high dimensionality of the dataset, I further normalized the term in the exponent by the square root of dimensions N_{DIM}.

2.1.2 K-Means Clustering

In contrast to the original K-Means clustering method, I did not initialize the means randomly. Instead, I adopted a suggestion from the lecture and iteratively selected the next mean as the point that had the largest average distance of the previous means. While this worked perfectly fine for uniform data, the eigenspace of the normalized affinity matrix contained a lot of outliers. Effectively choosing them as initial mean values while the first mean was selected randomly mostly resulted in all points being assigned to this first mean and only a few other outliers being assigned to the outlier means. For that reason, I revised my mean initialization strategy and decided to initialize them purely random, even though this was shown to yield bad results in general. Of course, I assured that no mean was randomly selected multiple times. To reduce the probability of outliers, I replaced the random means by the middle point between them and the mean of all samples. Then, the K-Means Clustering was performed until either a maximum number of iterations was reached or a threshold that indicated convergence was undercut. I observed that the algorithm sometimes

was unstable and reached the maximum number of iterations `MAX_ITERATIONS` while still exhibiting strong fluctuation in the clustering results. To ensure the algorithm still yielded good results in such a case, I gave the algorithm at most another `MAX_ITERATIONS` iterations to reach a region where the total difference in the means was not larger than twice the minimum total difference observed during the entire number of iterations. Even though it did not occur in the final experiment, I also covered the edge case where a cluster happened to be empty. In such a case, I randomly set a point as the new mean, given that its previous cluster has at least two elements.

2.1.3 Model selection score

Since the above methods assume a fixed k, I created a score metric to balance the model expressivity with the clustering quality. In principle, the score is quite similar to the *Bayesian Information Criterion* $\text{BIC} = \log(n) * k - 2 * \log(L)$, where n is the number of samples and L the likelihood of the model. Since the notion of likelihood was difficult for me to implement, I used one of the clustering evaluation metrics from the lecture to replace the likelihood. Specifically, I computed the ratio S between the average intra-cluster and inter-cluster distance. I then changed the score to $\log(\log(n)) * k + 2 * \log(S)$. Since a smaller S is better, I reverted the sign in front of the log term. Since n was quite large, I decided to wrap another log term around it to achieve a good balance. The smaller the score, the better. I tested the score on a dataset of three uniform distributions and it achieved the best score of 0.59 for $k = 3$, while the scores were 1.87 and 3.01 for $k = 2$ and $k = 4$ respectively.

3 Results

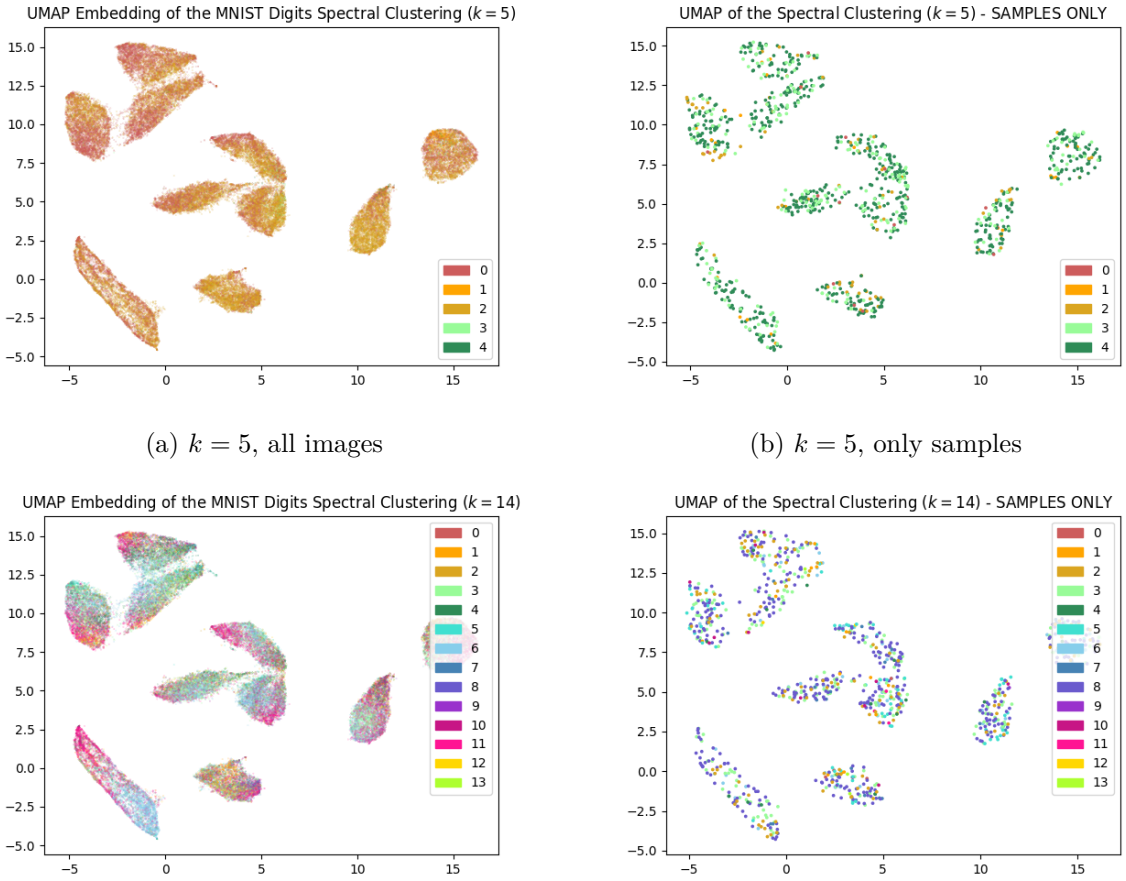
3.1 Spectral Clustering on raw image features

While the spectral clustering achieves well results on the uniform mixture dataset used for testing, using the raw image pixels as feature vectors leads to unsatisfactory clusterings. While the sizes of the clusters are balanced at the beginning, they eventually converge to one large cluster, while the other clusters seem to converge to outlier regions. Moreover, the clustering results for the samples in the eigenspace of the normalized affinity matrix (e.g. Fig. 1b) and the clustering of the entire image (e.g. Fig. 1a) dataset in the original feature vector space yield quite different results. As hypothesized in the experiment introduction, this might most likely be due to the different notion of distance that is used in the both clusterings. Even though we also use the K-Means Clustering algorithm in the eigenspace of the normalized affinity matrix, the transformation to the eigenspace will apply the decentralized notion of connectivity introduced in the first section, while the assignment to the closest mean in the raw feature vector space follows the centralized mean-based distance notion.

On the other side, the algorithm does not find meaningful results for larger k. Consequently, the clustering score monotonically increases. These unsatisfactory results could be due to the fact that the eigenspace of the affinity matrix only captures k dimensions, which is very small compared to the total dimension of the matrix. Though I did not conduct any experiments on the variance that is explained by this subspace, the results from the last experiment might suggest that 5 to 14 dimensions might only be able to represent a quite small fraction of the total variance, which is why we observe these bad results where only a few patterns can be derived from. However, Fig. 1c depicts that although the overall clusterings do not provide any global consistency, they are locally consistent.

3.2 Spectral Clustering on first principal components

We observe a significant improved of the clustering results when we provide the first 45 principal components as input to the NJW algorithm (Fig. 2). While the clusterings can still not capture the true separation of the data, they improve their global consistency, as Fig. 2a depicts. This disproves my hypothesis in the experiment introduction that the clustering result will be invariant to the reduction to the first principal components. I assume that the construction of the normalized affinity matrix must be highly influenced by this reduction and the eigenvectors of the matrix are able to express a way larger fraction of the total variance. Because the clustering results is still far from perfect, the ratio between the intra-cluster and inter-cluster average distance is still not large enough to improve the score for larger k. For this reason, $k = 5$ still receives the best score. However, since the score is hardcoded by myself based on heuristics, this score does not truly reflect the goodness of the clustering.



(a) $k = 5$, all images

(b) $k = 5$, only samples

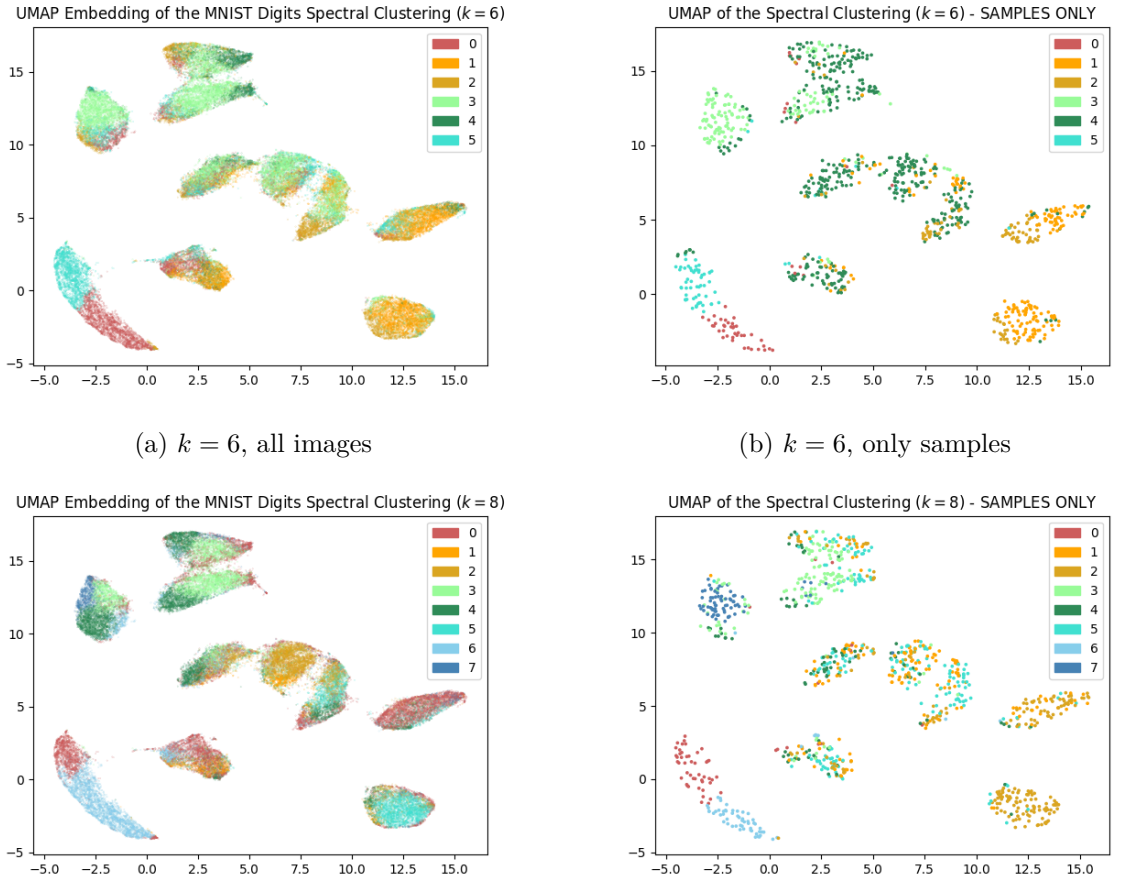
(c) $k = 14$, all images

(d) $k = 14$, only samples

Figure 1: Clusterings of the raw image pixel feature vectors for $k = 5, 14$ are depicted. On the left side, the clusterings extended to the entire image dataset are depicted, while the clusterings on the right side are restricted to the sampled dataset for which the actual computation of the NJW algorithm was performed. The clusterings are depicted in a UMAP embedding with the aforementioned hyperparameters.

3.3 Spectral Clustering on one digit

If we limit ourselves to the images that represent the digit 3, $k = 2$ yields the best results, which are depicted in Fig. 3. In this figure, two representative clusterings with representative points are depicted. As we observe in Figures 3c and 3b, the shape of the different handdrawn digits differs strongly. While the digit in Fig. 3c is poorly drawn, the angles in Fig. 3b are crisp and the shape nearly perfect. In this case, the clusterings truly obtained some meaningful separation of the data, even though the boundary is not fixed.



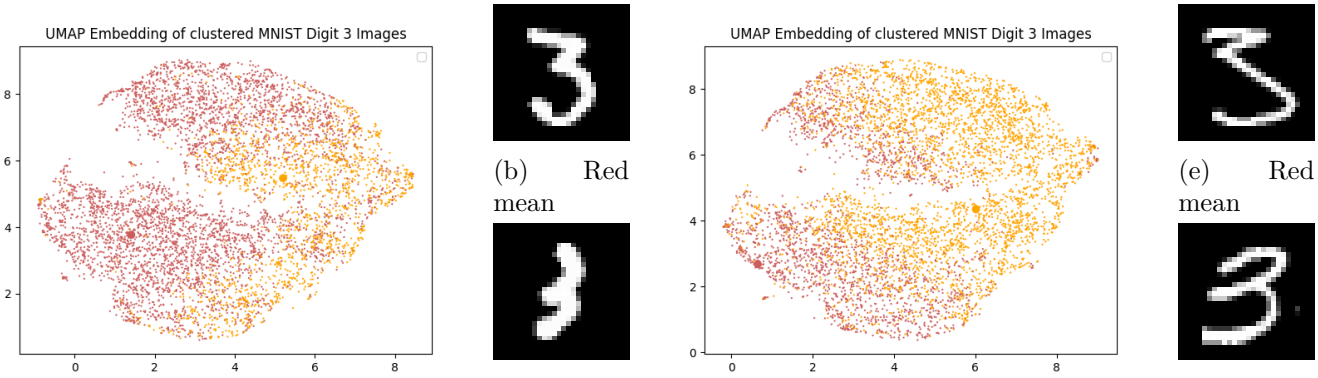
(a) $k = 6$, all images

(b) $k = 6$, only samples

(c) $k = 8$, all images

(d) $k = 8$, only samples

Figure 2: Clusterings of the first 45 principal components of the images for $k = 6, 8$ are depicted. On the left side, the clusterings extended to the entire image dataset are depicted, while the clusterings on the right side are restricted to the sampled dataset for which the actual computation of the NJW algorithm was performed. The clusterings are depicted in a UMAP embedding with the aforementioned hyperparameters.



(a) Clustering

(b) Red mean

(c) Orange mean

(d) Clustering

(e) Red mean

(f) Orange mean

Figure 3: Clustering of MNIST Digit 3 Images into 2 clusters. For each cluster, the representative means were calculated as the sample points that were the closest to the true mean in the eigenspace of the normalized affinity matrix. These representatives are highlighted by larger points and their image depicted on the right side.

4 Conclusion

All in all, the clustering results of the NJW algorithm depend strongly on how compressed the data represents the entire variance. The eigenvalues of the affinity matrix do not seem to always represent the largest possible fraction of variance, which might be why restricting on the first PCs yield better results.