Lab Course Machine Learning
— Problem Set 1—

Alberto Rodriguez (376116)

May 17, 2020

Part 1: Application

Assignment 1 Principal component analysis

In this exercise, the usps dataset will be analyzed by using the PCA method. The dataset consists of 2007 16x16 pixel images displaying numbers between 0 and 9. More formally, we have a data matrix $\mathbf{X} \in \Re^{d \times n}$ with $d = 256$ features (every pixel value) and $n = 2007$ image samples.

At first, the principal component analysis algorithm will be applied to the original dataset. Since we have $d = 256$ features, we expect 256 eigenvalues and its correspondent eigenvectors. We also expect the resluting eigenvalues to ordered from highest to lowest.

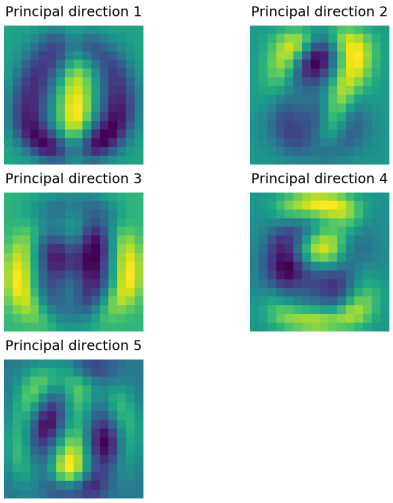


Figure 1: First 5 principal directions

Figure (1) shows the first five principal directions (eigenvectors) after performing the eigenvalue decomposition of the dataset covariance matrix $\mathbf{C}$. We can recognize that the principal directions, specially the fourth one which seems to be a three, appear to represent some structure numbers too. This can be explained by the fact that the principal directions are a linear combination of the original dimensions.

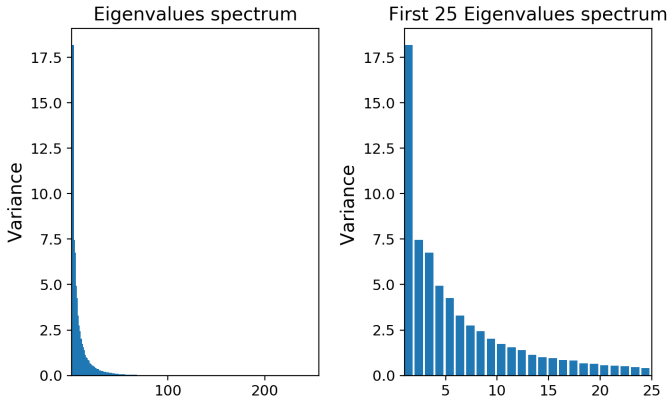
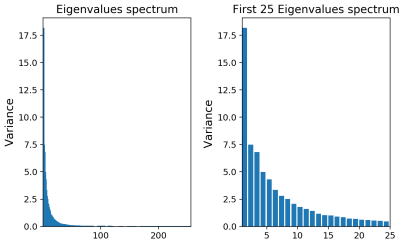


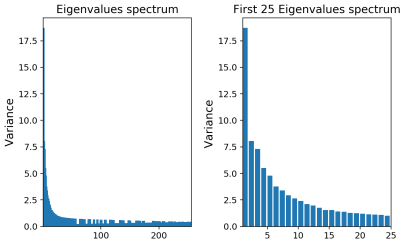
Figure 2: Principal values of the original dataset

Figure (2) presents the principal components values (eigenvalues). At first, we can recognize that most of the variance is concentarted in circa the first 25 principal components. Since the images are not noisy, there is a strong decay so that $\lambda_i \approx 0$ for $i > 25$ .For dimensionality reduction, one could project the entire dataset into a e.g 5 dimensional space and still preserve most of the variance . Additionally, one can denoise images with a similar procedure which we will later discuss.

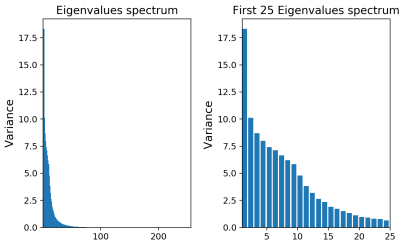
Next, three scenarios to produce noisy images have been considered. In the first one, we consider low gaussian noise with $\mathcal{N}\left(\mu = 0, \sigma^2 = 0.2\right)$ such that the resulting images are moderately noisy. In the second one, we consider high gaussian noise with $\mathcal{N}\left(\mu = 0, \sigma^2 = 0.7\right)$. Lastly, we consider extreme gaussian noise with $\mathcal{N}\left(\mu = 0, \sigma^2 = 7.7\right)$ for only five images such that they are not recognizible.



(a) Low noise for all dataset



(b) High noise for all dataset



(c) Extreme noise (outliers) for 5 images

Figure 3: Principal values of the noisy datasets

Figure (3) shows the principal values of all noisy datasets. Figure (3a) indicates that when low noise levels is applied to the dataset, the spectrum slightly changes in comparison to the original dataset spectrum (2). The spectrum seems to be almost identical for the first 25 principal values. The effect of the low noise is reflected in the higher principal values spectrum $\lambda > 50$ where they still carry some variance. Figure (3b) reinforces this observation and shows that applying more noise leads to a higher "contamination" of all principal values. We will use this trend when denoising the images by using only the first m principal values.

The goal of the last scenario, when generating 5 outliers by applying extreme noise, is to explore how sensible PCA is towards outliers. In this case, the outlier contamination ratio (i.e. percentage of outliers in the data set) is 0.25%. Figure (3c) indicates that in this case the noise does not affect all principal values of the spectrum, but only the main ones i.e the first 25. This is an important fact since the the noise affects the principal values that we will use to denoise images. In comparison, the other two noise levels did not greatly affect the first 25 principal values. Consequently we suggest that *PCA can be very sensitive towards outliers and one should remove them from the dataset before applying this technique.*

Next, the images will be denoised by reconstruction from projections on the m largest principal components. In order to achieve the best possbile results, an optimal value for m has to be choosen. We consider two ways of making this decision. At first. one can choose m using the following contraint:

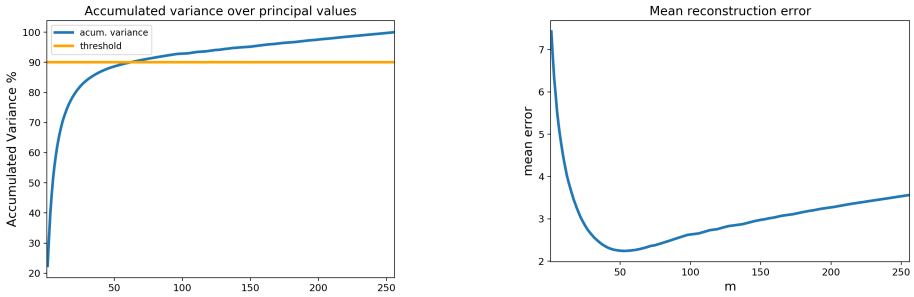
$$\frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \cdot 100 = 90\%. \tag{1}$$

Equation (1) represents the condition of mantaining 90% of the data variance in the lower m dimensional subspace.

A second method to select an optimal m is to minimize the total reconstruction error

$$\underset{m \in [0,d]}{\arg\min} \left(\frac{\| \sum_{i=1}^{n} \mathbf{x} - \widetilde{\mathbf{x}}(m) \|}{n} \right) \tag{2}$$

where $\widetilde{\mathbf{x}}(m)$ is the reconstruction of its correspondent data point $\mathbf{x}_{noisy}$ which depends on the choice of m and $\mathbf{x}$ is the original "correct" data point. Note that we can only use this method when the "correct" dataset is given which is the case in this academic example. Nervetheless, in most real life applications one would get a noisy dataset without the perfect denoised dataset and therefore, we could only use the first method.

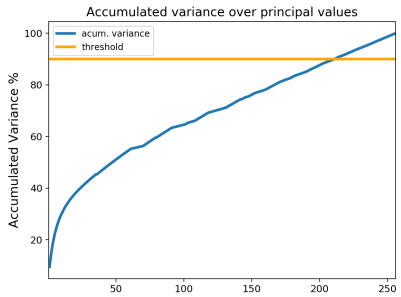


(a) Accumulated variance

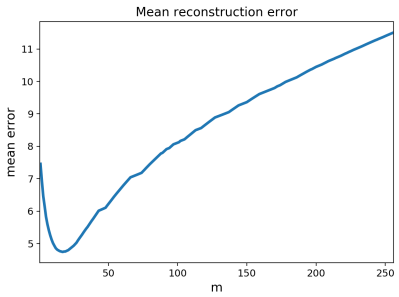
(b) Mean reconstruction error

Figure 4: Methods to select m for the low noisy dataset

Figure (5a) shows that if we want to mantain 90% of the variance (low noisy dataset), we should select the first $m = 60$ principal values. Figure (4b) indicates that the optimal value is $m = 50$. In this case, we would have been near the optimal result by choosing the threshold of 90%. One could conclude that choosing a 90% threshold in all scenarios is the optimal way of selecting m. Unfortunatelly this is not the case.



(a) Accumulated variance high noise

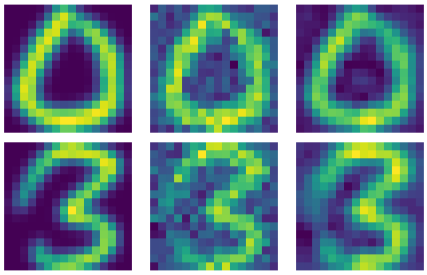


(b) Mean reconstruction error

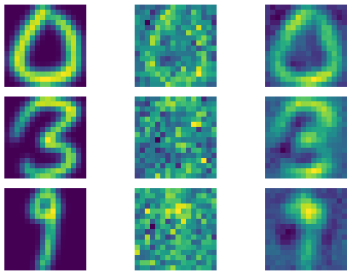
Figure 5: Methods to select m for the high noisy dataset

Figure (5) shows that if we followed the same 90% threshold criterium for the high noisy dataset, we would end up with a much higher reconstruction error by choosing $m = 210$. In this case, figure (5b) indicates that $m = 20$ is the optimal choice.

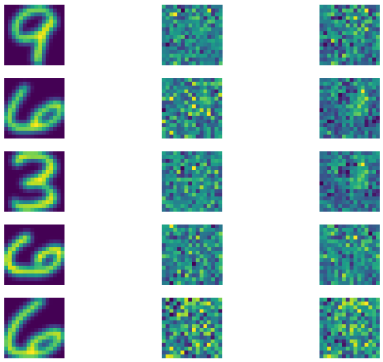
We suggest that the choice of the number of components m used to reconstruct/denoise the data takes into account how noisy the data is. The noise level of the data can be estimated from the principal values spectrum. According to our results, if there is low noise, one can select a greater threshold then if the data has high noise levels.



(a) Low noise, m=50



(b) High noise, m=17



(c) Extreme noise (outliers), m=5

Figure 6: Denoised data, order: original (left), noisy(middle), denoised(right)

Figure (6) shows the different obtained results after denoising the images. We observe from figure (6a) that PCA is able to reduce the noise from the least noisy dataset. Furtheremore, the denoising process has very good results for the high noisy dataset from figure (6b). The noisy images are hard to recognize. Still, the algorithm is able to reduce most of the noise and reconstruct the image in a way such that we can read the digits from it. Last but not least, PCA is not able as expected to denoise the outliers even if we choosed a small value $m = 5$ of components to reconstruct the image.

Assignment 2 Outlier detection and AUC

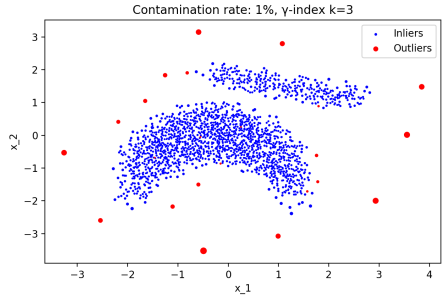
Outliers are extreme values that deviate from other observations on data , they may indicate a variability in a measurement or experimental errors. For this reason, outlier detection plays a major role in several domains as in PCA where the previous exercise showed how sensible this method towards outleirs can be. In this exercise, we consider the positive class of the banana dataset as "inliers" to which we add outliers generated from a random process. Each data point has $d = 2$ features and the inlier dataset has $n = 2376$ samples. We consider two different method of detecting outliers. Both methods rely on the calculation of a score value for every data point. The higher the score value of a data point is, the more possible it is that this data point is an outlier. In the first method, the calculation of the outlier score γ is based on the average distance to its k-th nearest neighbors

$$\gamma\left(\mathbf{x}\right) = \frac{1}{k}\sum_{i=1}^{k} \|\mathbf{x} - z_j\left(\mathbf{x}\right)\|. \tag{3}$$

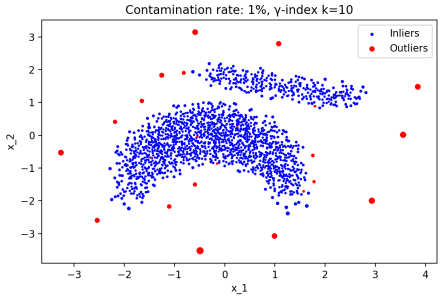
In the second method, the outlier score corresponds to the distance to the data mean $\widetilde{\mathbf{x}}$

$$m(\mathbf{x}) = \|\mathbf{x} - \widetilde{\mathbf{x}}\|. \tag{4}$$

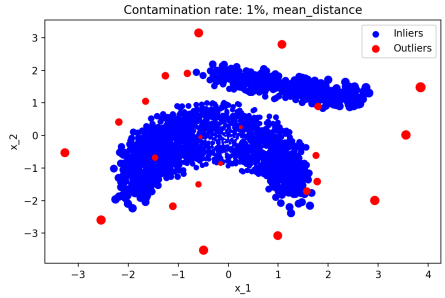
Three different scenarios regarding the contamination rate (i.e. percentage of outliers in the data set, relative to the positive class) have been investigated: 1%, 10%, 50% and 100%. The outliers are uniformly randomly sampled with $\mathbf{x}_{out} \sim \left[\mathcal{U}\left(-4,4\right), \mathcal{U}\left(-4,4\right)\right]^T$.



(a) γ index, k=3



(b) γ index, k=10



(c) mean distance

Figure 7: Contamination rate: 1%, data points scaled by its outlier score

Figure (7) presents the data points, both inliers and outliers, scaled by its correspondent outlier score for three different configurations. In all cases, one can recognize as expected that the outliers have a higher score then the inliers. Specially, the outliers that are not far from the inliers dataset, get the highest score values. We can also observe in all cases that outliers lying within the inliers dataset, have almost the same score values as the inliers. Therefore, we expect that these outliers will probably not be classified as outliers when applying the threshold later on and will lead to a not perfect classification. Figure (7c) shows that in comparison to the γ scores values (Fig.7a and Fig.7b), the m (mean distance) scores have a higher value.

Next, the data points will be classified as inliers or outliers based on their outlier score value. Since this outlier score value can be computed in three different ways ($\gamma, k = 3$, $\gamma, k = 10$ and m), we want to compare all three methods. One way of doing this is by comparing the AUC (area under the curve of ROC) values. In the case that all outliers and correctly classified, the AUC value is 1. In this case, we define a positive case as a correclty classified outlier and a negative case as a correctly classified inlier (not outlier) point. Consequently, false positives are inliers that are incorrectly classified as outliers and false negatives are outliers that are incorrectly classified as outliers. In order to classify a data point as an inlier or outlier based on its outlier score, a threshold ϵ has to be set so that

$$\begin{aligned} \mathbf{x} \in \Omega_{inlier} \forall \gamma(\mathbf{x}) < \epsilon \\ \mathbf{x} \in \Omega_{outlier} \forall \gamma(\mathbf{x}) \geq \epsilon \end{aligned} \tag{5}$$

Since the score value range can vary depending on the configuration, an aqudistant finite "grid" ϵ for the thresholds has to be set. The first option we considered was to set $\epsilon = \mathbf{y}_{pred}$, so that $\epsilon \in \Re^n$. This approach seems to have one main limitation when having a large ammount of samples since e.g in our case one would need to compute 2376 iterations over the for loop to calculate one AUC value. Furtheremore, we are computing for statistical reasons each contamination rate/method combination 100 times. We believe that this approach is computationally too expensive.

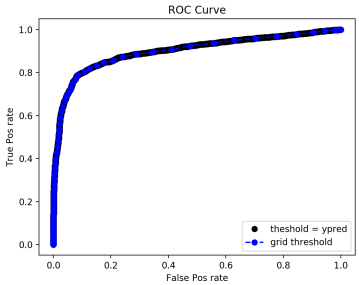
Consider the range $[y_{min,pred}, y_{max,pred}]$ of the outlier score values $\mathbf{y}_{pred}$. We can generate all threshold values as

$$\epsilon = [y_{min,pred}, y_{min,pred} + h, y_{min,pred} + 2h, ..., y_{max,pred}]^T \tag{6}$$

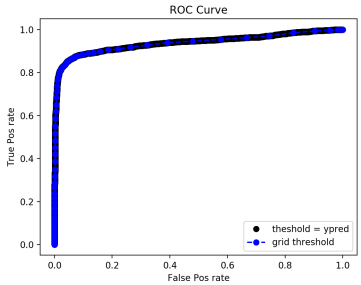
where

$$h = \frac{y_{max,pred} - y_{min,pred}}{N_{grid}}. \tag{7}$$

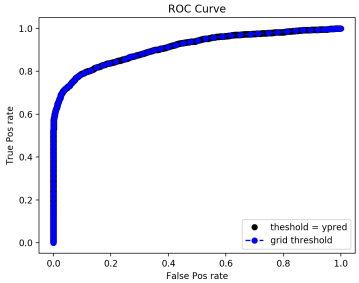
. In other words, we are generating an aquidistant threshold grid with step size h which depends on the number of grid points N_{grid}. Indeed, this method is not perfect. We recognized that if the intervall $[y_{min,pred} ... y_{max,pred}]^T$ is very large, then one would select a higher value N_{grid} and consequently, we have the same problem as with the first method, a large ammount of loops to perform for the calculation of each AUC value.



(a) γ, k=3



(b) γ, k=10



(c) mean distance

Figure 8: ROC curves using a 100% contamination rate and both threshold methods

In order to confirm that we were not making important errors by taking the threshold grid approach with $N_{grid} = 100$ points, we compared the ROC curves and its correspondent

AUC values for both threshold methods. Figure (8) shows that both approaches are aquivalent in this case. Furtheremore, the relative error regarding the AUC values of both curves, is about 0.01% in all cases. For this reason, we decided to compute the AUC values using the grid threshold approach.

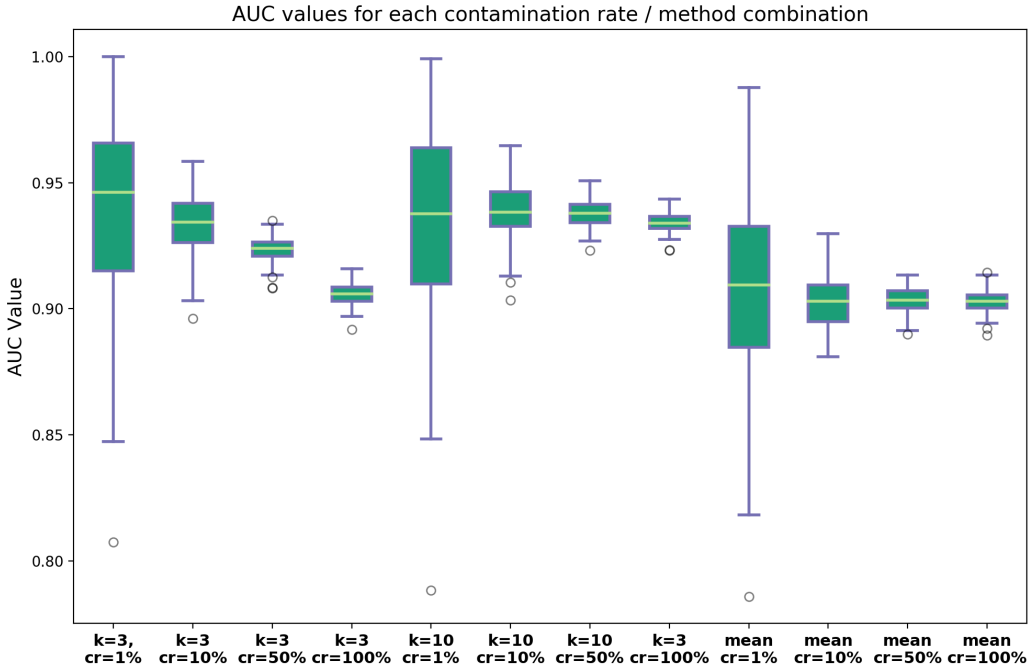


Figure 9: Principal values of the original dataset

Figure (9) shows the distribution of the 100 AUC values for each contamination rate / method combination. . The ideal AUC value is 1. At first, we can observe that the γ method delivers better results then the *meandistance* method, since $AUC_{\gamma} > AUC_{mean-distance}$. Interestingly, we can see that for higher contamination rates, the mean AUC values of the $\gamma-$ method using its $k = 3$ nearest neighbours fall from $AUC = 0.95$ to 0.90 while the AUC values using the same method but with $k = 10$ neighbours remain stable at $AUC = 0.94$.

To sum up, the results from this exercise suggest that based on the AUC values, the best method to detect outliers in the positive class of the banana dataset, is the $\gamma-$method with $k = 10$.

Assignment 3 LLE on three different datasets

The first dataset we investigated is the *flatroll*. This dataset consists of 1000 one two dimensional samples. By using the Locally Linear Embedding technique, we wish to visualize the nonlinear structure of the data in a lower dimensional space.

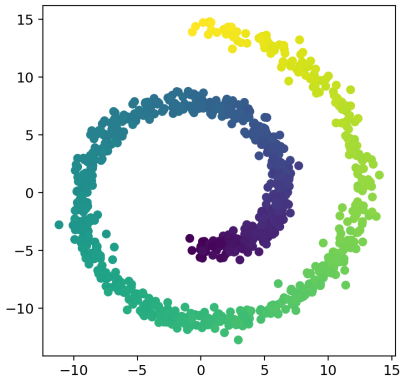
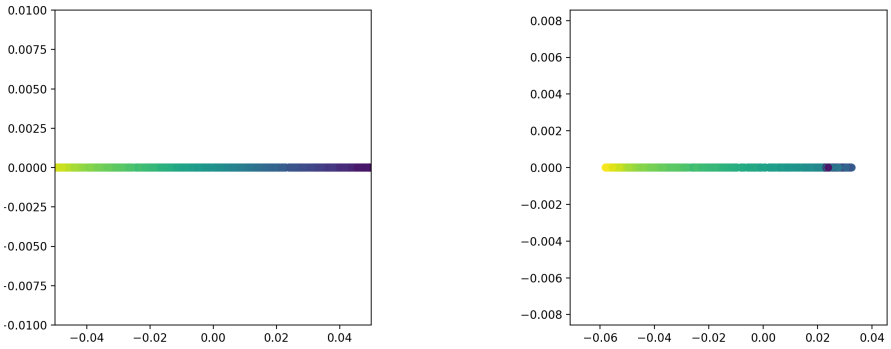


Figure 10: Flatroll 2D dataset

Figure (10) illustrates the structure of the *flatroll* dataset. Since it is a 2 dimensional dataset, we wish to reduce it to a one dimensional dataset. We expect to recognize the dark blue datapoints on one edge of the 1D axis and the yellow data points on the other edge of the 1D axis. We also expect to see a smooth transition from the dark blue data points to the yellow data points. This expectations correspond to the idea of "unfolding" the flatroll.

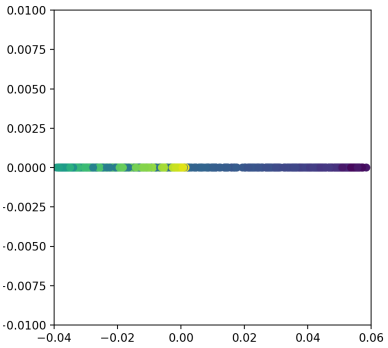


(a) LLE with k=14

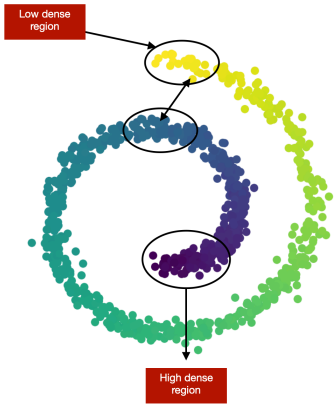
(b) LLE with $\epsilon = 1.7$

Figure 11: LLE on the flatroll dataset

We set a tolereance of $1e-14$ and computed the LLE for different k (nearest neighbours) values. Figure(12a) highlights the positive results when using the $k = 14$ nearest neighbours. We observe that the flatroll is unfolded as we expected. Furtheremore, the successful 1D representation of the data mantains for $k \in [10, 19]$. Nevertheless and as expected, for larger k values the unfolding is not succesfull. We obtained similar results when using the ϵ-ball neighbours method with $\epsilon = 1.7$. In this case (see Fig.11b) the unfolding does not perfectly work in the "blue dark" area of the dataset.



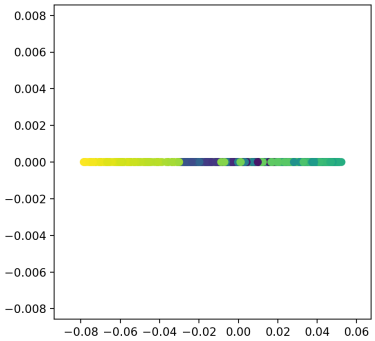
(a) LLE with k=25



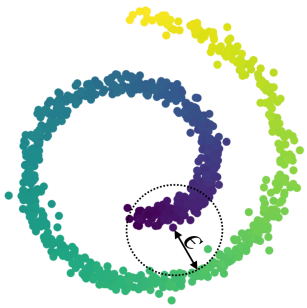
(b)

Figure 12: knn error visualisation

We investigated the case when using $k = 25$ neighbours. Figure (**??**) shows that in this case the unfold is incorrect. We see that the yellow data points are mapped next to the blue datapoints. When using *knn* (k-nearest neighbours) the data densitiy around one data point plays a major role when performing the algorithm. For a single data point, LLE tries to preserve the local distances to its k nearest neighbours. In other words, if some yellow points are some of the nearest neighbours of the blue data points, LLE will indeed try to pack them together when reducing the dimensions. We believe two factors influence this event. At first, the high $k = 25$ value makes it more possible for blue points to be neighbours of the yellow poitns. Secondly, a low dense region (see Fig.12b) makes this even more possible.



(a) LLE with $\epsilon = 2.5$



(b)

Figure 13: $\epsilon-$ball error visualisation

We investigated the case when using a larger ball radius $\epsilon = 2.5$. The main problem of using a large radius is the danger of grouping, for example, points of the blue dark group with points of the green group as illustrated in figure (13b). We believe that this is the main source of problems in this case. Furtheremore, having wide spread points-i.e., a low dense region will prevent being able to choose a small radius.

Next, we will perform LLE on the three dimensional *swissroll* dataset which consists of 400 samples.

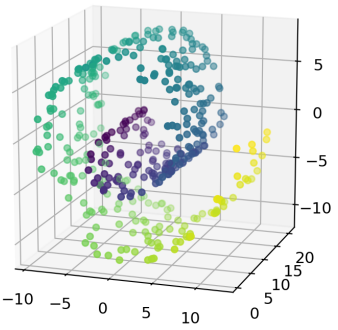
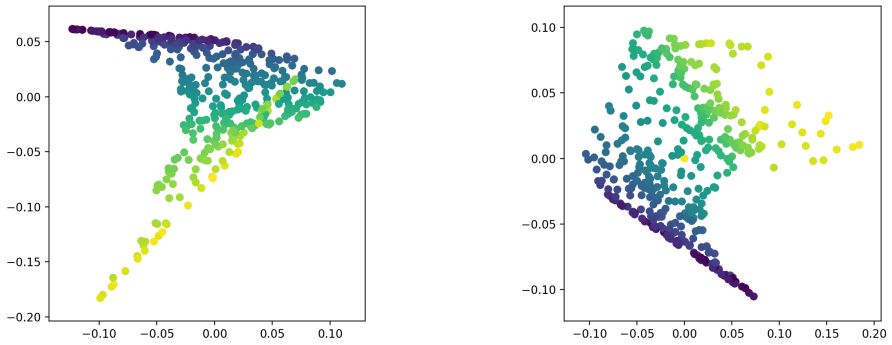


Figure 14: Swissroll 3D dataset

Figure (14) shows the structure of the data. Again, we wish to unfold the swissroll into a two dimensional space and preserve its structure.



(a) LLE with $k = 6$

(b) LLE with $\epsilon = 4.4$

Figure 15: LLE on the Swissroll dataset

We noticed that LLE was extremely sensitive towards k and ϵ changes. Figure (15) shows the results after finding an "optimal" value for $k = 6$ and $eps = 4.4$. In this case the unfolding is not perfect although both methods are still capable of showing the basic data structure.

We believe that the main reason for the parameters sensibility and not optimal unfolding is the low sample number of the dataset (400 samples). Consequently, low data density areas are probably present within the dataset and negatively affects LLE.

Last but not least, we will perform LLE on the three dimensional $fishbowl$ dataset which consists of 2000 samples.

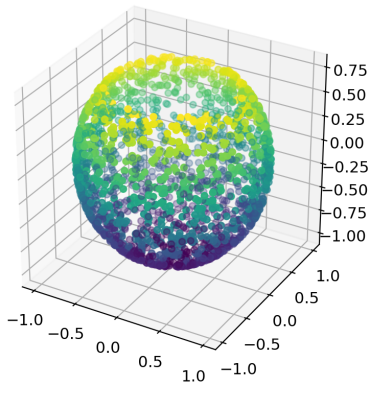
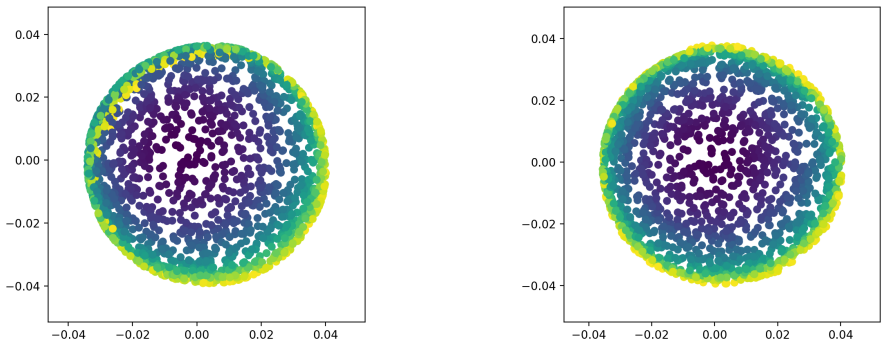


Figure 16: Fishbowl 3D dataset

Figure (16) shows the $fishbowl$ dataset sphere structure. We expect that the LLE unfolds the $fishbowl$ into a 2 dimensional circle where we should observe the yellow data points on the circumference of the circle and the dark blue data points on the center of it.



(a) LLE with $k = 50$

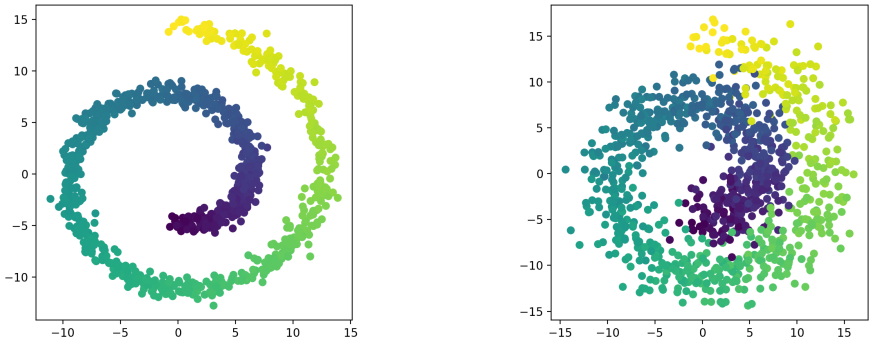
(b) LLE with $\epsilon = 0.2$

Figure 17: LLE on the Fishbowl dataset

Figure (17) shows the succesfull unfolding. When using knn, LLE appeared to be much more robust then in the $swissroll$ dataset. Therefore, a slight change of k did not lead to a much different result. Because of the much denser dataset (3000 samples) we could choose a much higher k value. The unfolding using the $\epsilon - ball$ with $\epsilon = 0.2$ was also successful. As expected, most yellow data points lie on the circumference of the circle and there is a "smooth" color transition towards tje dark blue points in the center.

Assignment 4 Noise effect on LLE

We will investigate how a noisy dataset affects the LLE performance. In this example, we will add gaussian noise to the two dimensional *flatroll* dataset. We will investigate low gaussian noise with a 0.2 variance and high gaussian noise with a 0.8 variance. We will perform LLE by using the *knn* algorithm to select the neighbours and use an optimal and a large non optimal k value respectively. Furtheremore, we will plot the neighborhood graph and the resulting embedding.

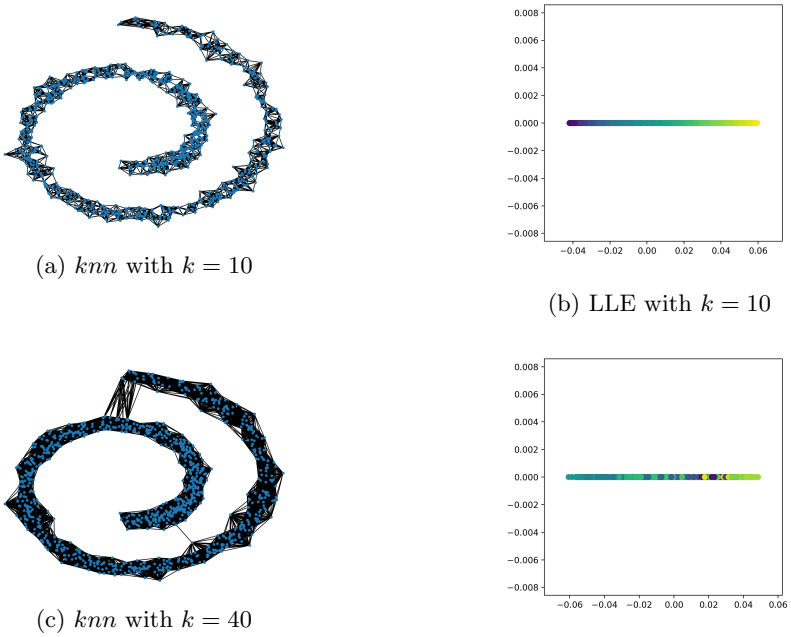


(a) Low noisy flatroll dataset

(b) High noisy flatroll dataset

Figure 18: Noisy datasets

Figure (20) shows both noisy datasets. We can observe that the low noisy dataset preserves the flatroll data structure. Thus, we can predict that the unfolding will be optimal for this dataset. On the other side, the high noisy dataset preserves much less this structure and different regions that were far from each other before applying high noise come very close to each other after adding high noise e.g. the yellow and blue regions . Consequently, we expect a complicated unfolding for this dataset.



(a) *knn* with $k = 10$

(b) LLE with $k = 10$

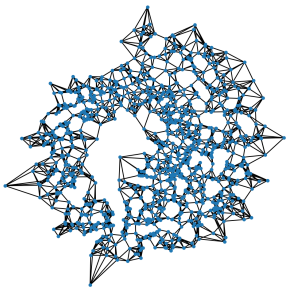
(c) *knn* with $k = 40$

(d) LLE with $k = 10$

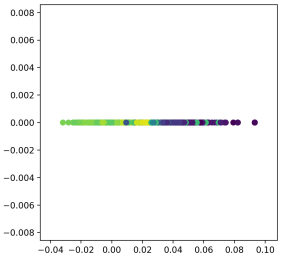
Figure 19: Low noisy dataset results

As expected, the unfolding of the low noisy dataset is optimal. We found out that using $k = 10$ delivered good results (see Fig.19b). We analyzed the results for a larger $k = 40$

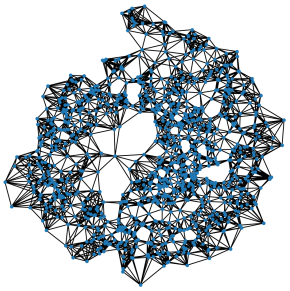
value. Due to the reasons that were already explained in assignment 3, LLE attempts to mantain the distance between some yellow and blue data points (see Fig.19c) which causes the non optimal unfolding (see Fig.20).



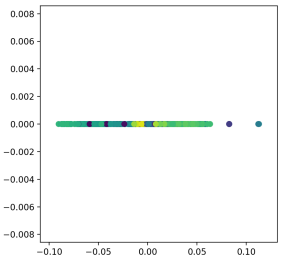
(a) *knn* with $k = 6$



(b) LLE with $k = 6$



(c) *knn* with $k = 10$



(d) LLE with $k = 10$

Figure 20: Low noisy dataset results

As expected, the unfolding of the high noisy dataset is not optimal. We found out that using $k = 6$ delivered the best possible results (see Fig.20b). We analyzed the results for a larger $k = 10$ value. This resulted in even more suboptimal results (see Fig.20d).