Exercise 3
Clustering
Advanced Practical Course In Machine Learning

Alon Netser
31160253-6

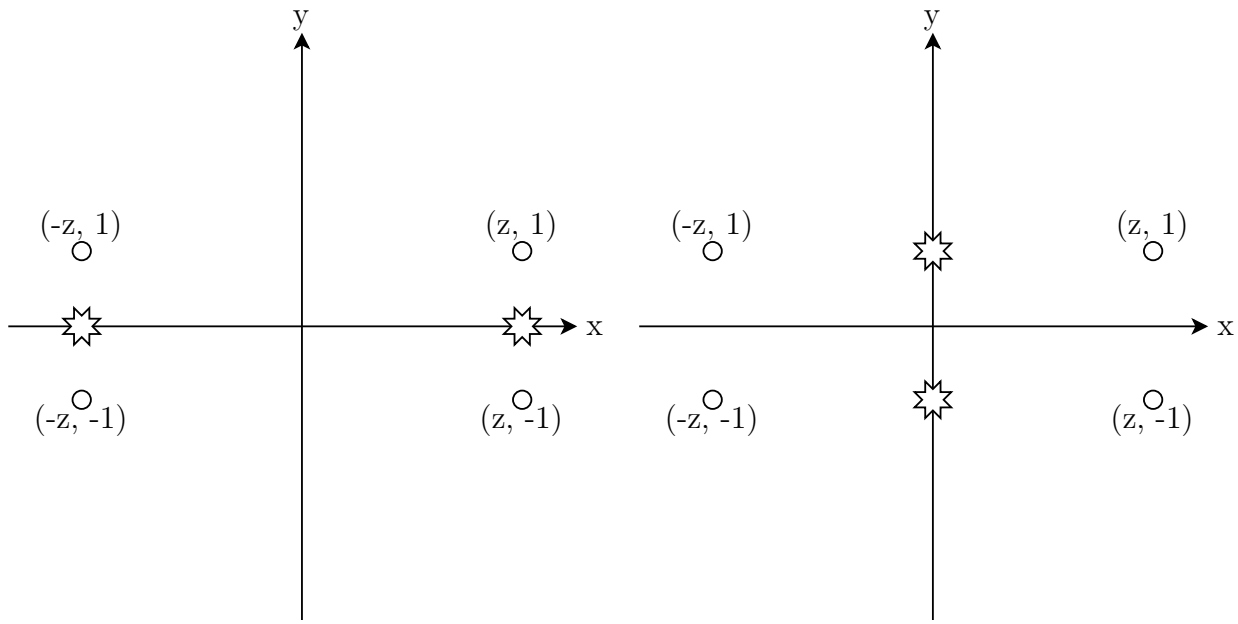
December 19, 2019

1 Theoretical Questions

1.1 Convergence of k-means

We'll prove that k-means converges in a finite number of steps. The halting condition of the algorithm is when the clusters remains the same from one iteration to the next (and therefore also centroids themselves and the cost function, since they are determined by the clusters - the centroids are the mean of points in their cluster and the cost is the average distance squared of each data-point to its assigned centroid). Note that if the clusters changed then the cost function decreased (the algorithm never moves from one clustering to another one that gives a higher cost). This is true since the centroids themselves minimize the cost function (when the assigning of points to clusters is fixed) - they are the center of mass of there points. Furthermore, if in the next step some data point is moved from one cluster to the other, it means that the distance to its new centroid is lower, and therefore the overall cost function is lower. This gives us the conclusion that if the algorithm did not stop (i.e. the clusters changed from the previous iteration to the current one) then the cost function decreased.

Note that the number of possible clusters is at most k^n, where k is the number of clusters and n is the number of points. This is a large but finite number. Assume by contradiction that the algorithm does not halt at all. This means that at some point it repeats some clustering of the points, and between these two occurrences other clustering were chosen. This means the the cost of this repeated clustering is lower than the cost of this clustering in the previous iteration it appeared, but because it's the same clustering it's definitely not possible (equivalent to a number being strictly lower than itself) - a contradiction.



,

(a) Optimal centroids

(b) Sub-optimal centroids

1.2 Sub-optimality of k-means

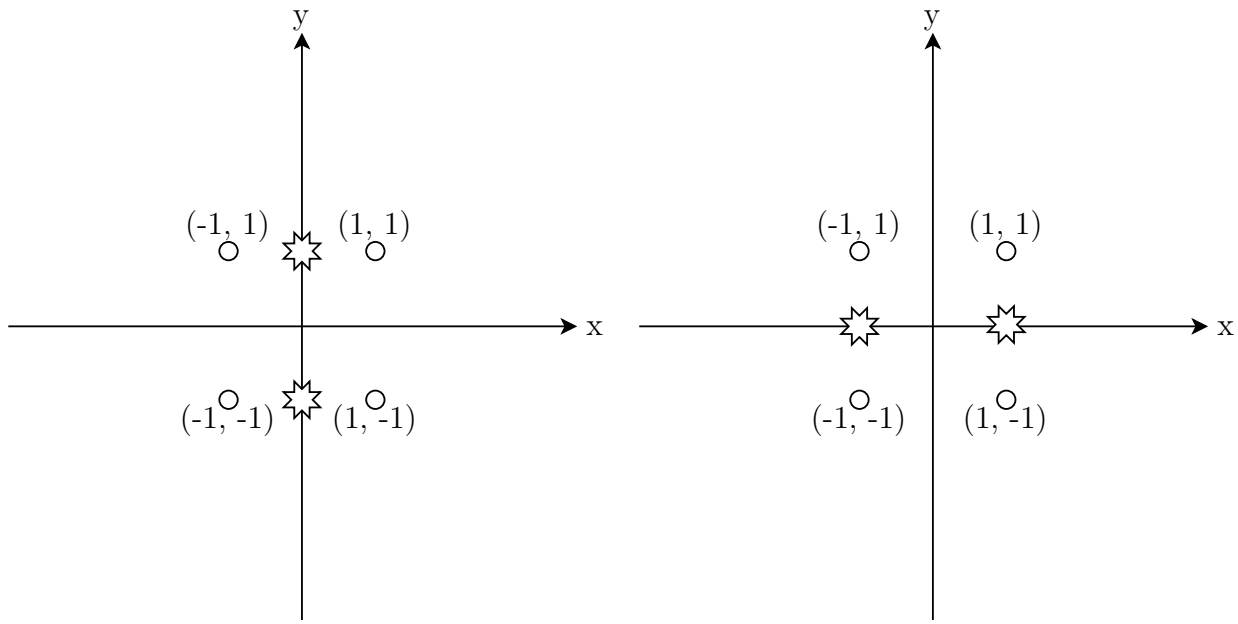
1.2.1 Converge to a suboptimal solution

We'll show an example where a bad initialization of k-means will result in a sub-optimal solution, which is "as worse as we like" - for any number $\alpha \in \mathbb{R}$ we can have the sub-optimal solution cost to be α times larger (a.k.a. worse) than the optimal solution cost.

So the setting is the following - we choose some value $z \in \mathbb{R}$ such that $z >> 1$, and we look at 4 points in $\mathbb{R}^2$, which are $(\pm z, \pm 1)$. In Figure 1a we have the optimal solution, which has a cost of 4 (sum of 4 times 1, which is the distance of a point from its centroid). In Figure 1b we have the bad initialization - if the k-means algorithm (with $k = 2$) is initialized with these centroids, it will converge immediately to that solution. This is because the points $(z, 1)$ and $(-z, 1)$ will be assigned with the centroid $(0, 1)$, and it's their mean (so it will also be the centroid defined by the points assigned to it). The same goes for the points $(z, -1)$ and $(-z, -1)$ with the centroid $(0, -1)$. Note that the cost function in this sub-optimal solution is $4 \cdot z^2$ (4 times the squared distance between a point and its centroid).

1.2.2 More than one optimal clustering solution

A small tweak to the previous example gives an example of a dataset where that has more than one optimal solution. The dataset is the 2-dimensional unit-cube vertices - $(\pm 1, \pm 1)$. One optimal solution is shown in Figure 2a, where the centroids are $(0, \pm 1)$, and the other optimal solution is shown in Figure 2b where the centroids are $(\pm 1, 0)$. Both solutions have a cost of 4.



(a) One optimal solution

(b) Another optimal solution

,

1.3 Initializations of k-means

1.3.1 Expected number of trials before we have a good initialization event

We want to calculate the probability of a single "good event" (i.e. an event where each initial centroid was chosen from a different cluster). Note that there are $k!$ possibilities for the order of the choosing - we can choose the first point from cluster 2, the second from cluster 1 and the last from cluster 3, or another one of the possible orders. For each possible order of the choosing, the probability for actually choosing each point form its cluster is the product of the probabilities, since the samples are independent. Therefore the probability of each possible order is $\prod_{j=1}^{k} \alpha_{i_j}$ where $i_1, i_2, \ldots, i_k$ are the order of the k clusters. Anyway, it's equal to $\prod_{i=1}^{k} \alpha_i$. Overall, the probability to sample each initial centroid from a different cluster is the sum of this probabilities over all possible ordering, which is

$$p = k! \cdot \prod_{i=1}^{k} \alpha_i$$

Now, since this is the probability of a success in a geometric random variable, the expected number of trials until success is 1 divided by that probability, meaning it's

$$\frac{1}{k! \cdot \prod_{i=1}^{k} \alpha_i}$$

1.3.2 Lower bound on this number

We want to bound $\frac{1}{k! \cdot \prod_{i=1}^{k} \alpha_i}$ from below, which is the same as bounding the denominator $k! \cdot \prod_{i=1}^{k} \alpha_i$ from above. So we want to find the maximum value for $k! \cdot \prod_{i=1}^{k} \alpha_i$ under the

constraints that $\sum_{i=1}^{k} \alpha_i = 1$. We can use Lagrange multipliers, and get that

$$\frac{\partial}{\partial \alpha_j}\left(\prod_{i=1}^{k} \alpha_i + \lambda \cdot \left(1 - \sum_{i=1}^{k} \alpha_i\right)\right) = 0$$
$$\Downarrow$$
$$\frac{1}{\alpha_j}\prod_{i=1}^{k} \alpha_k - \lambda = 0$$
$$\Downarrow$$
$$\prod_{\substack{1 \le i \le k \\ i \ne j}} \alpha_i = \lambda$$

This is true for every $j \in \{1, \ldots, k\}$ so we get that for all $\ell, m \in \{1, \ldots, k\}$ such that $\ell \ne m$ it holds that

$$\prod_{\substack{1 \le i \le k \\ i \ne \ell}} \alpha_i = \prod_{\substack{1 \le i \le k \\ i \ne m}} \alpha_i$$

and from this we get that all α_i must be equal to each other, and since they sum to 1 it means that for all $i \in \{1, \ldots, k\}$ we have that $\alpha_i \frac{1}{k}$.

Combining this with the result we got from the previous question, we get that the expected number of trial until success is at least

$$\frac{1}{k! \cdot \prod_{i=1}^{k} \frac{1}{k}} = \frac{1}{k! \cdot k^{-k}} = \frac{k^k}{k!}$$

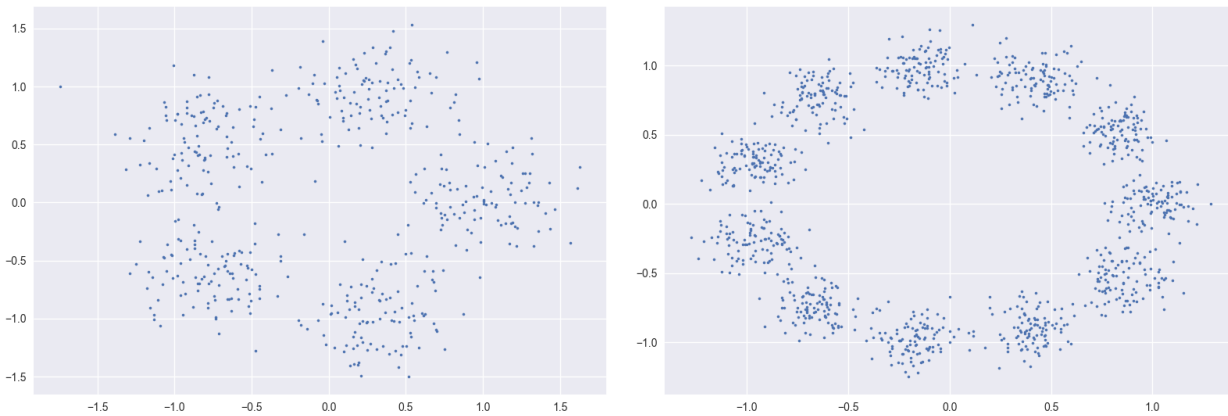
If we want to get another look on this term, we can use Stirling formula $k! \approx \sqrt{2\pi k} \cdot \left(\frac{k}{e}\right)^k$ (meaning that as k goes to infinity the limit between these two terms is 1). So this lower bound is approximately $\frac{e^k}{\sqrt{2\pi k}}$. which is exponentially in k.

1.3.3 So how does k-means still work?

The previous results indeed seems surprising, since in practice we can't do an exponentially number of trial every time we try to cluster data using k-means, and this is in our best scenario, when the clusters are evenly distributed (if the probability to sample from some clusters is greater than others the expected number of trial is larger).

However, we can argue that this theoretical result does not reflect to practice since most of the times we don't use a value of k which is large, and for a value of $k = 10$ it's about 2755 trials (and many times we use $k < 10$.

Furthermore, the result is true where we sample the initial centroids uniformly from the data point. In practice there are better ways to initialize the centroids, such as the one we learned in class (k-means++).



(a) 5 Gaussians

(b) 11 Gaussians

,

Figure 3: The generated gaussians datasets.

2 Practical Exercise

In this section I'll go over the different datasets and discuss the results. This will include comparison between k-means and spectral clustering, comparison between different parameters of the spectral clustering, plotting the similarity graphs that were used in the spectral clustering algorithm, methods for choosing k, etc.

2.1 Gaussians dataset

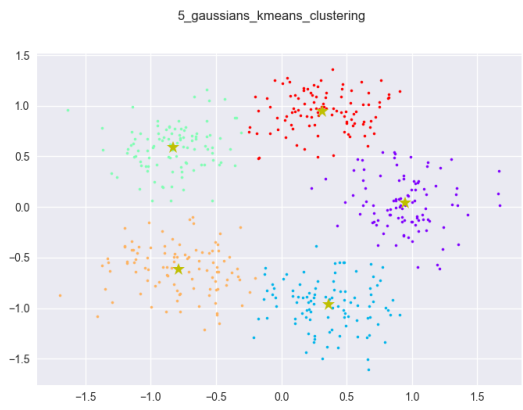
I created my own synthetic dataset in $\mathbb{R}^2$ to examine the results of the clustering, and show the methods to for choosing k. This dataset contains N gaussians (where N is a tunable number). The centers are on the unit circle, in an equal distance between each other. The standard deviation and number of points per cluster are also tuneable. Figure 3 shows this dataset for two values of N - 5 and 11.

As expected, k-means works very good on these datasets. In fact, this is the reason why this dataset was created - in order to examine the k-means algorithm where it should work good (since on the other synthetic dataset it does not work well). Figure 4 shows the clusters obtained using k-means on the datasets of 5 and 11 gaussians.

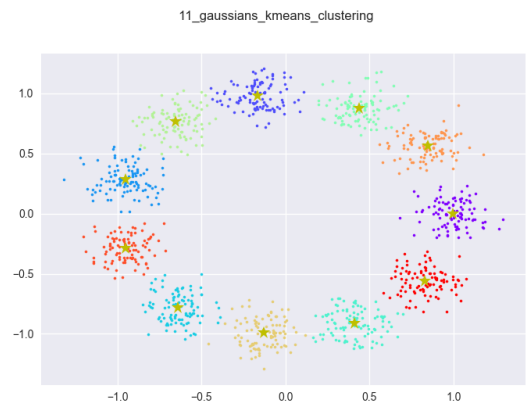
Using spectral clustering on this dataset worked also pretty well. Figure 5 shows the histogram and the cumulative histogram of the distances between every pair of data points in the 5 gaussians dataset. Note that the distance from a point to itself is removed from the histogram, and the distance between every pair is taken only once. This is done by looking at the upper triangular of the symmetrical distances matrix (excluding the main diagonal).

Different parameters were chosen for each of the two kernels (gaussian kernel and m-nearest-neighbors). Figure 6 shows the results of the spectral clustering using the 0.25% and 1% percentiles, showing clearly that 0.25% is too small and cluster outliers in their own clusters. Using 1% percentile works good. Figures 7 and 8 show the similarity graph of these two σ values, showing that the lower one does not capture similarity between data points properly.

This dataset can be used to demonstrate the methods for choosing k. Figure 9 shows



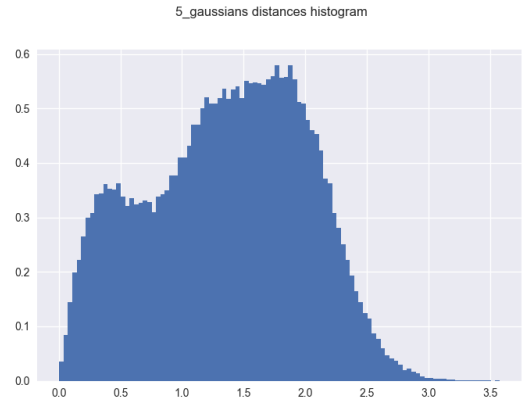
(a) 5 Gaussians k-means clustering



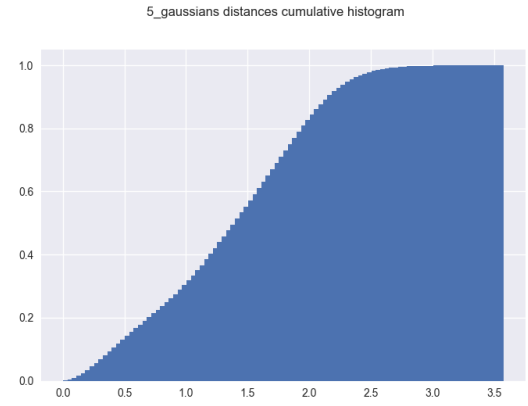
(b) 11 Gaussians k-means clustering

,

Figure 4: K-means clustering on the gaussians datasets.
Note that clustering the 11 gaussians required multiple tries (about 5-6) until the correct result was achieved. This is because the random initialization must put each centroid in different gaussian, and it's not always the case.



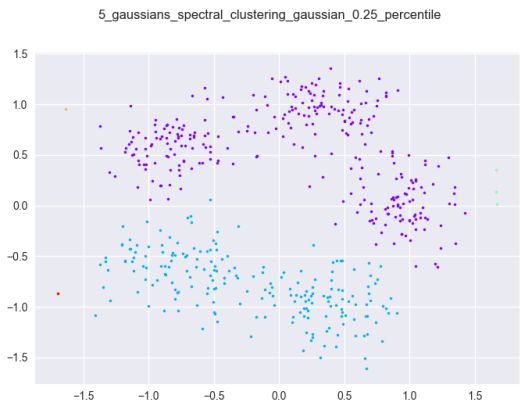
(a) Pairwise distances histogram



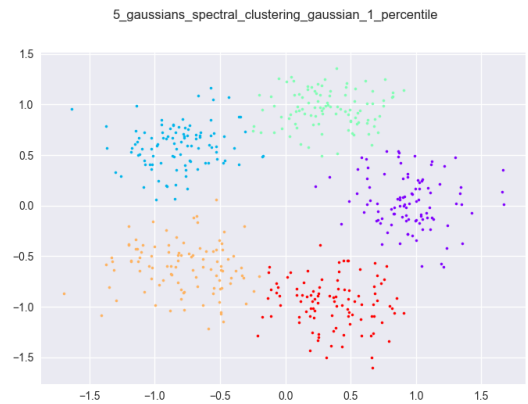
(b) Pairwise distances cumulative histogram

,

Figure 5: Pairwise distances in the 5 gaussians dataset. Note that there are two peaks, corresponding to distances between pairs of points within the same cluster, and pairs of points in other clusters. The separating value is between 0.5 and 1, and this should be a good choice for σ in the gaussian-kernel.

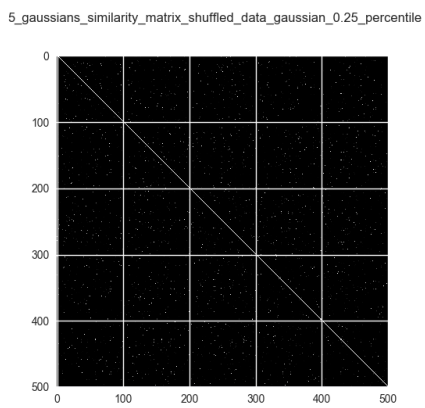


(a) Spectral clustering using gaussian kernel and σ equals to the 0.25% percentile of the distances.

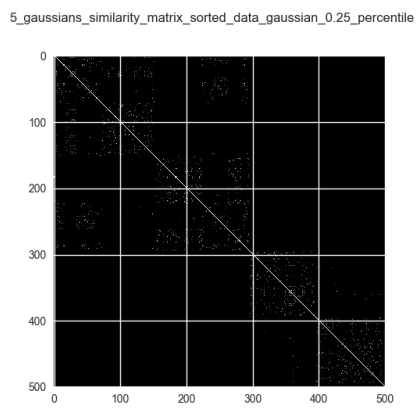


(b) Spectral clustering using gaussian kernel and σ equals to the 1% percentile of the distances.

Figure 6: Spectral clustering of the 5 gaussians using gaussian kernel with different values for σ.

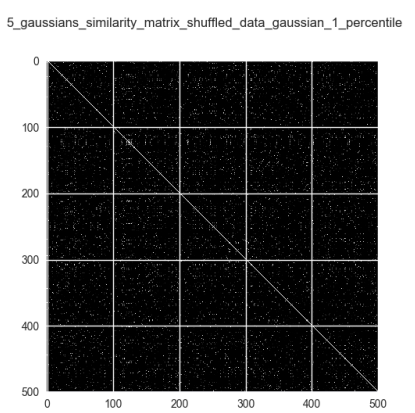


(a) Shuffled similarity matrix

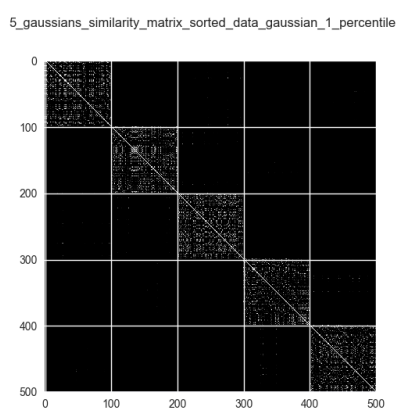


(b) Sorted similarity matrix

Figure 7: Shuffled and sorted similarity matrices using gaussian kernel with σ equals to the 0.25% percentile of the distances.

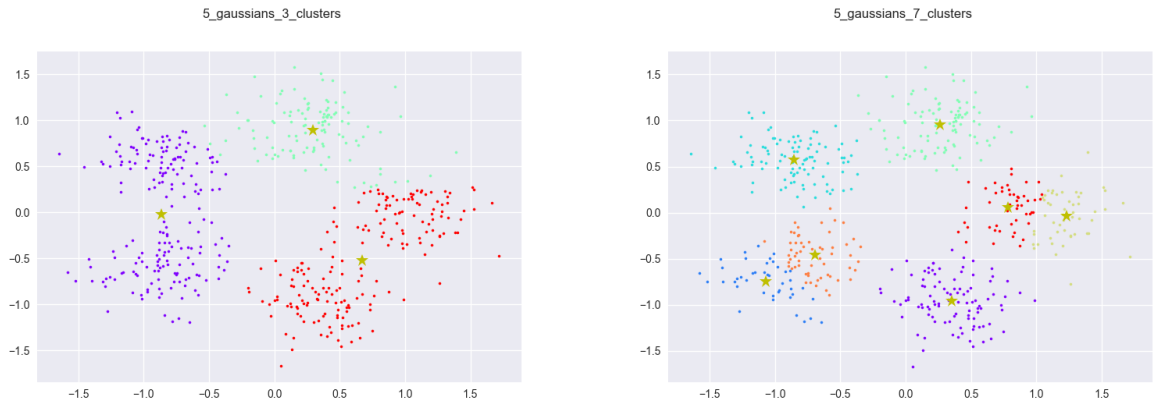


(a) Shuffled similarity matrix



(b) Sorted similarity matrix

Figure 8: Shuffled and sorted similarity matrices using gaussian kernel with σ equals to the 1% percentile of the distances.



,

(a) K-means using $k = 3$

(b) K-means using $k = 7$

Figure 9: Running K-means with lower and larger number of clusters.

clustering of the 5 gaussians datasets into smaller and larger number of clusters, using k-means. Figure 10 shows the cost of the clustering for many different k's, showing the 'elbow' effect. Note that running k-means for each k was tried multiple times with different random initialization, in order to pick the best clustering for that specific k. One can see clearly that 5 is the best k to use in this dataset, since it's exactly the 'elbow' point.

We can also use the silhouette scores in order to find the best k - we'll look for the k which gives the highest silhouette score. Figure 11 shows the silhouette scores while clustering the 11 gaussians dataset using k-means (each k was tried 50 times and the try that reached the highest score was picked). Clearly, 11 has the highest silhouette score and therefore it's the best k.

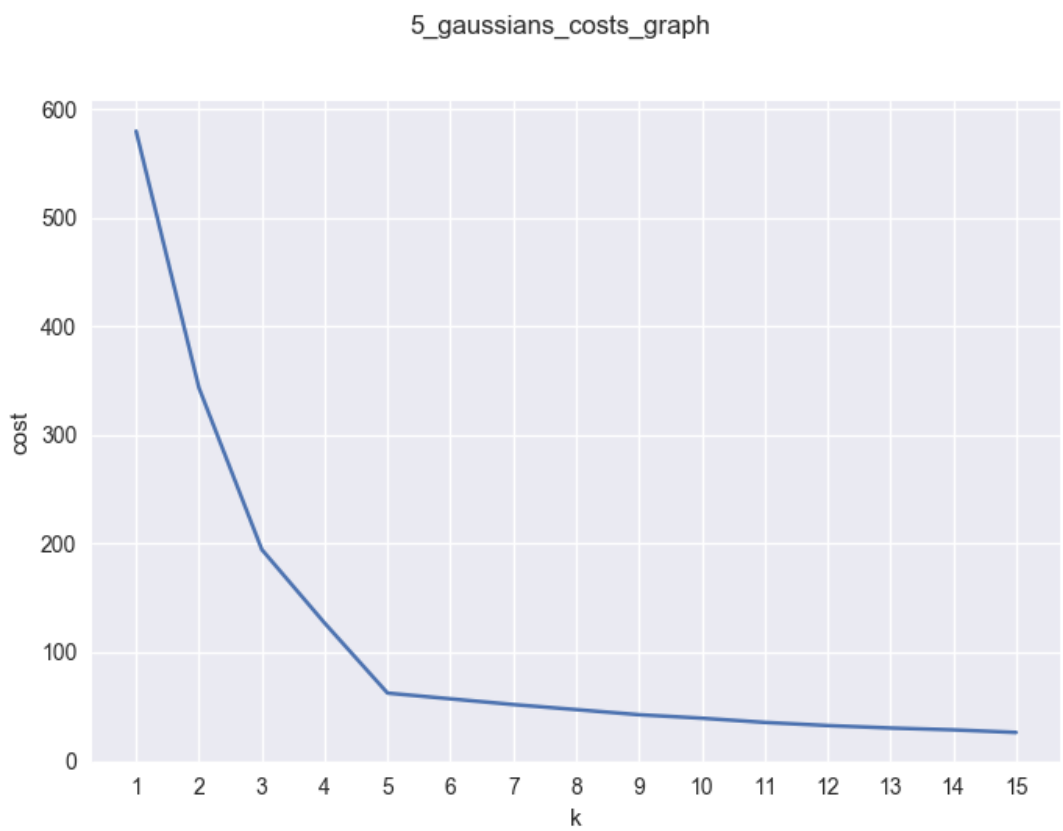
We can also use the eigengap method - plot the eigenvalues of the Laplacian matrix in an ascending order, and look for a 'jump' in the value. The lower eigenvalues correspond to actual clusters, and the higher one don't. Figure 12 shows eigenvalues of the Laplacian matrix used in the spectral clustering algorithm on the 11 gaussians dataset, using gaussian kernel with σ equals to the 0.1% percentile of the pairwise distances. One can see the jump in the value of the 12-th eigenvalue, indicating that 11 is the k to choose.

2.2 Circles dataset

Figure 13 shows the results of k-means clustering v.s. spectral clustering (using 11-nearest-neighbors similarity function). Clearly, k-means fails to capture the actual clusters, while the spectral clustering works good. Figure 14 shows the similarity graph while using 11-nearest-neighbors similarity function in spectral clustering this dataset. Figure 15 shows how too small and too large values of m in the m-nearest-neighbors similarity function affect the spectral clustering.

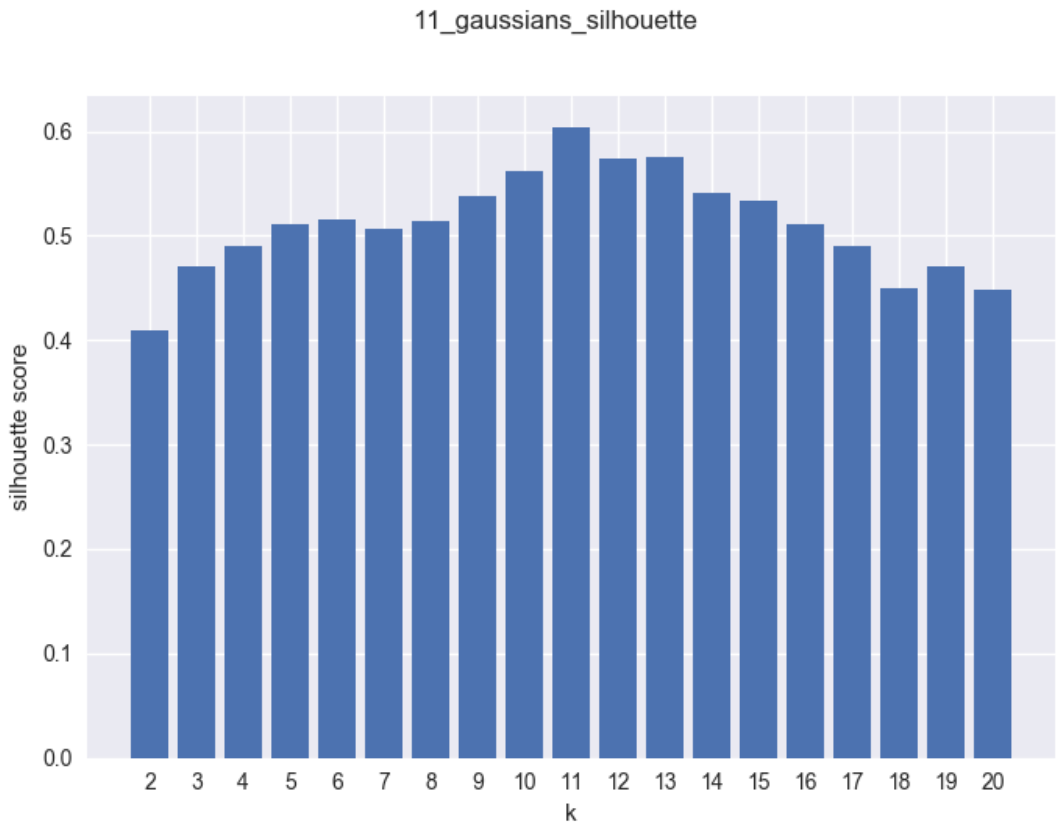
2.3 APML letters dataset

Figure 16 shows the results of k-means clustering v.s. spectral clustering (using 15-nearest-neighbors similarity function). Clearly, k-means fails to capture the actual clusters, while



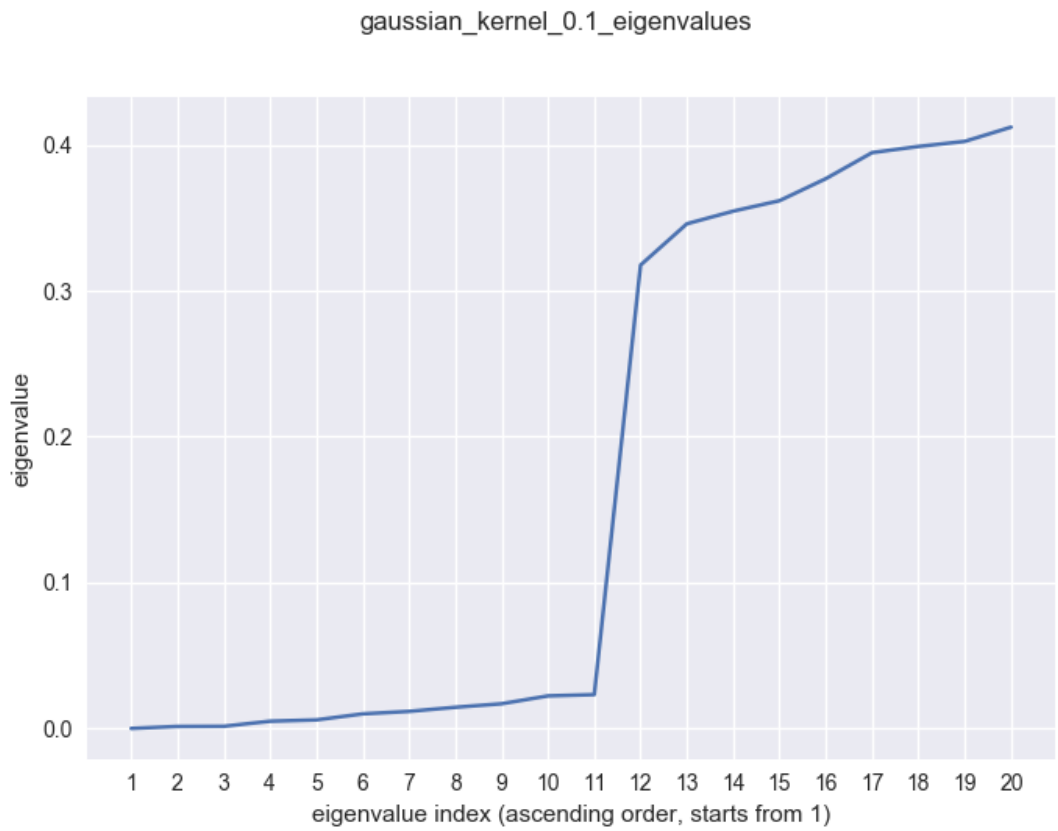
,

Figure 10: Cost as a function of k.



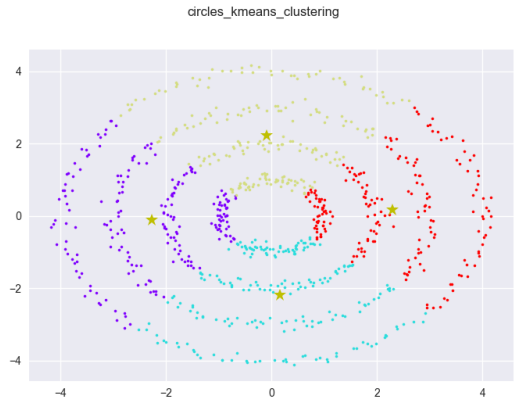
,

Figure 11: Silhouette scores as a function of k, on the 11 gaussians dataset.

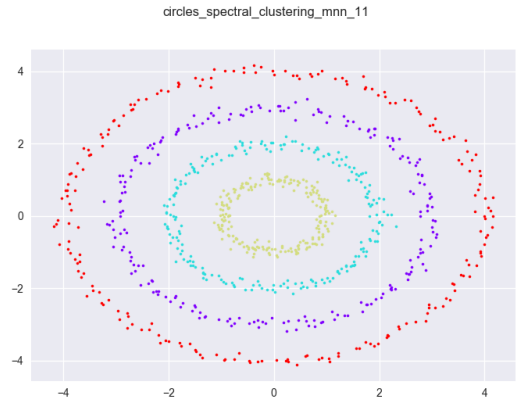


,

Figure 12: Eigenvalues in an ascending order, on the 11 gaussians dataset.



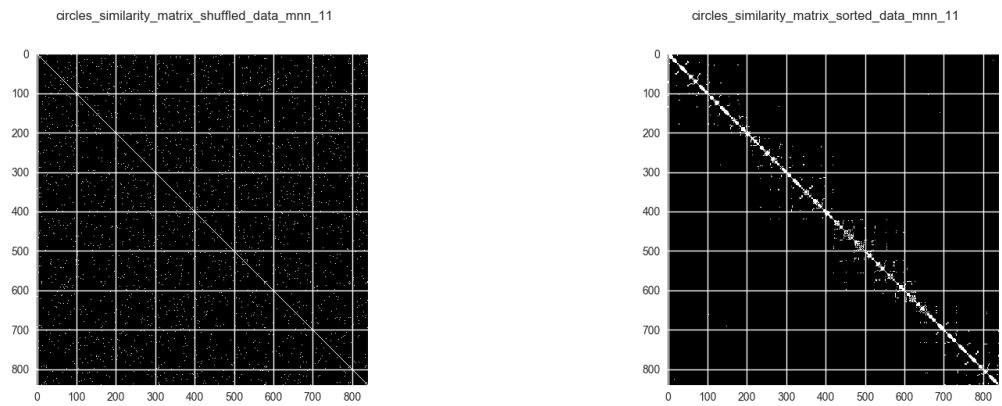
(a) K-means clustering.



(b) Spectral clustering, using 11-nearest-neighbors similarity function.

,

Figure 13: Comparison between k-means and spectral clustering. Clearly, k-means fails to capture the actual clusters, while the spectral clustering works good.

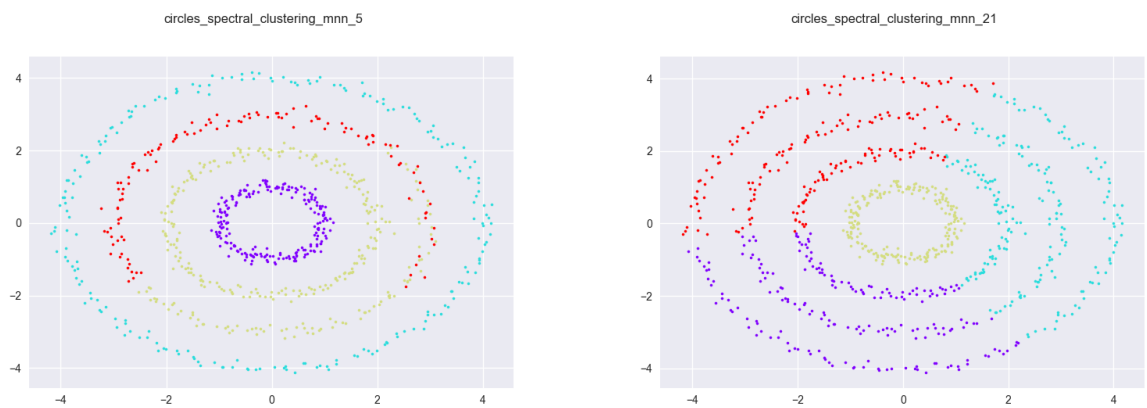


(a) Shuffled similarity matrix

(b) Sorted similarity matrix

,

Figure 14: Shuffled and sorted similarity matrices using 11-nearest-neighbors similarity function. One can see the four squares on the main diagonal, corresponding to the four circles being connected components.

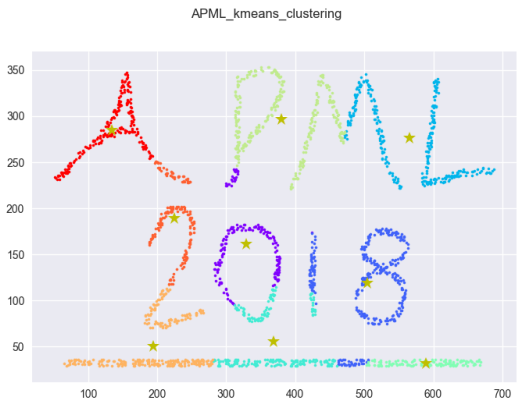


(a) Using 5-nearest-neighbors similarity function.

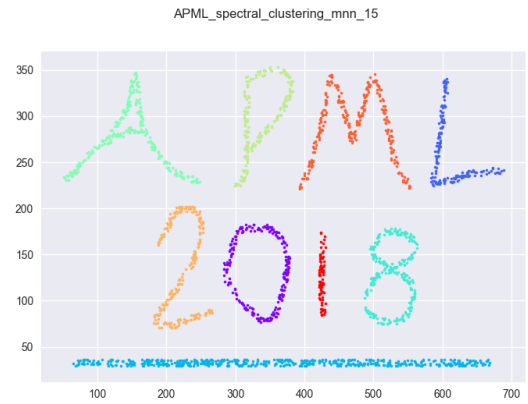
(b) Using 21-nearest-neighbors similarity function.

,

Figure 15: How too small and too large values of m in the m-nearest-neighbors similarity function affect the spectral clustering.



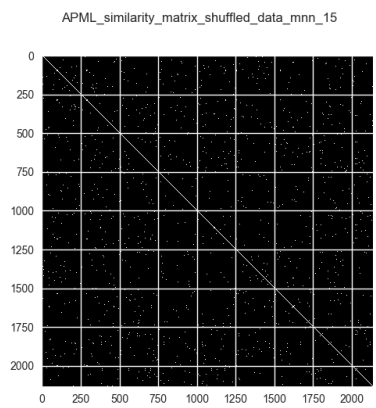
(a) K-means clustering.



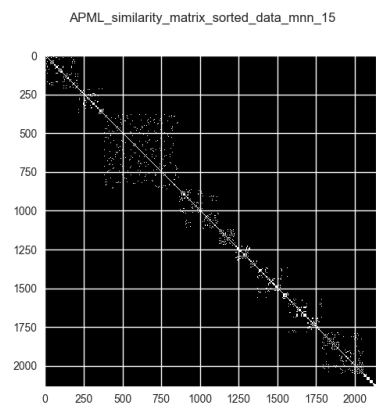
(b) Spectral clustering, using 15-nearest-neighbors similarity function.

,

Figure 16: Comparison between k-means and spectral clustering. Clearly, k-means fails to capture the actual clusters, while the spectral clustering works good.



(a) Shuffled similarity matrix



(b) Sorted similarity matrix

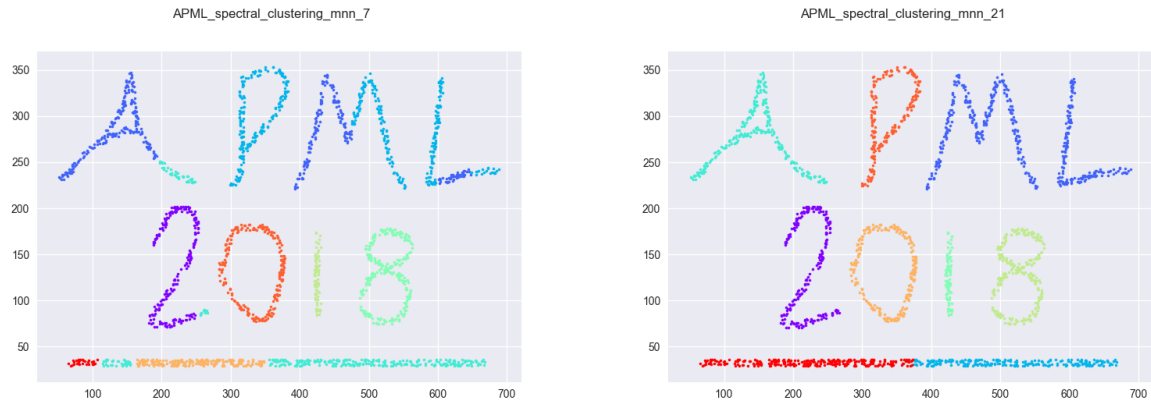
,

Figure 17: Shuffled and sorted similarity matrices using 15-nearest-neighbors similarity function. One can see the nine squares on the main diagonal, corresponding to the nine connected components.

the spectral clustering works good. Figure 17 shows the similarity graph while using 15-nearest-neighbors similarity function in spectral clustering this dataset. Figure 18 shows how too small and too large values of m in the m-nearest-neighbors similarity function affect the spectral clustering.

2.4 Biological data

I tried to understand the biological data and how to cluster it. First of all, I tried to cluster the data using k-means, and plot the cost as a function of k in order to find the 'elbow' and pick the best k for this data (each k was tried 10 times and the one that got the lowest cost was chosen). Figure 19 shows this graph, and it seems that the elbow is at $k = 4$ (one can argue it's in $k = 3$ or $k = 5$, because it's not the best elbow I saw, but in my opinion $k = 4$ seems the best). After all, trying to cluster the data into 4 clusters using k-means gives clusters of sizes $2175, 446, 720, 3360$ which seems reasonable (at least it did not cluster

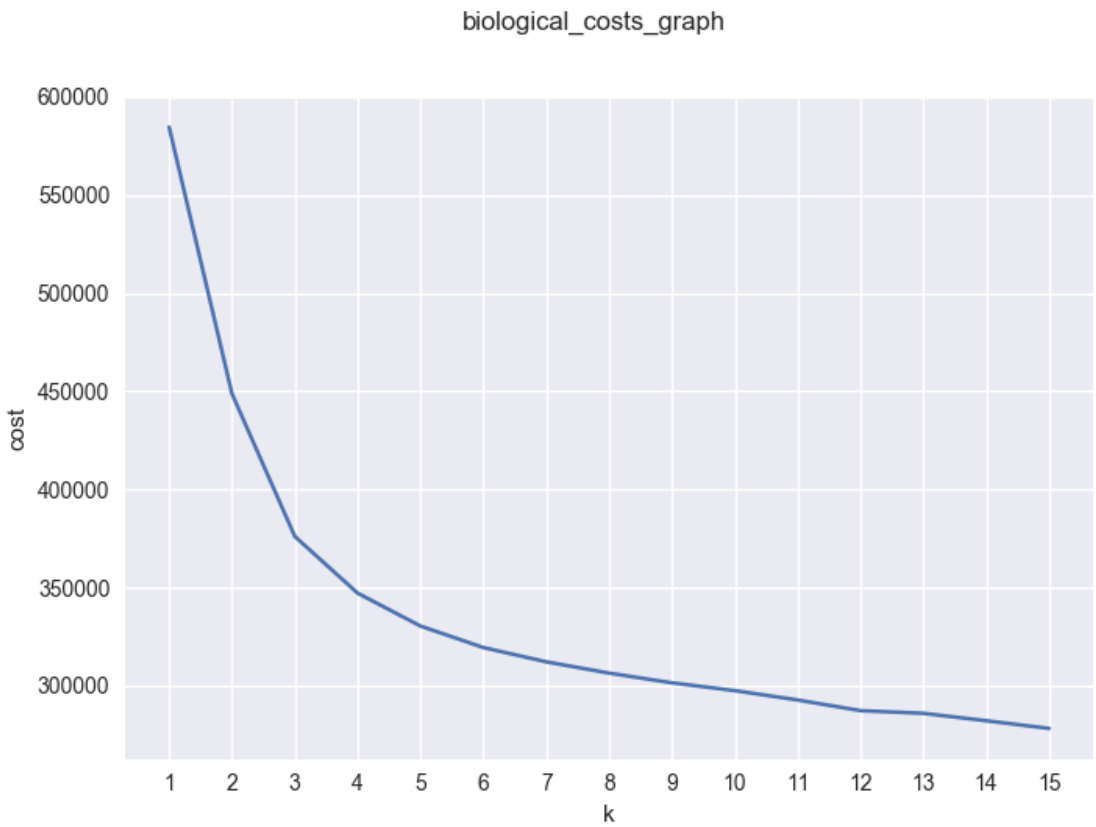


(a) Using 7-nearest-neighbors similarity function.

(b) Using 21-nearest-neighbors similarity function.

,

Figure 18: How too small and too large values of m in the m-nearest-neighbors similarity function affect the spectral clustering.

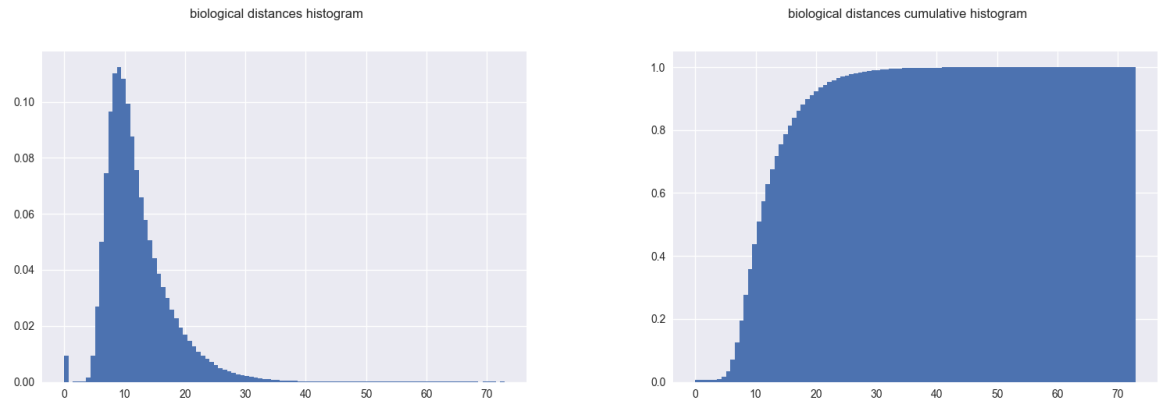


,

Figure 19: The cost as a function of k. It seems that the elbow is at $k = 4$

outliers into singletons clusters).

Now I wanted to try the spectral clustering on this data. Trying m-nearest-neighbors as a similarity function (with different values for m between 7 and 21) and plotting the eigenvalues showed a graph with only one eigenvalue close to 0, and from the second eigenvalue there was a jump. This means that the clusters are not well separated into connected components

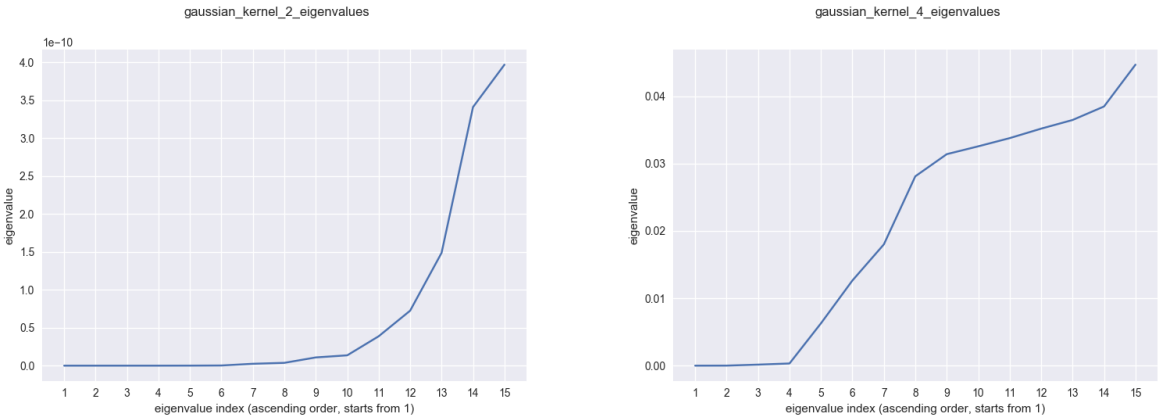


(a) Pairwise distances histogram

(b) Pairwise distances cumulative histogram

,

Figure 20: Pairwise distances in the biological microarray data.



(a) $\sigma = 2$. It seems that the eigengap is at $k = 8$, but clustering using this σ does nor work well (put outliers as clusters.

(b) $\sigma = 4$. It seems that the eigengap is at $k = 4$, but clustering using this σ does nor work well (put outliers as clusters.

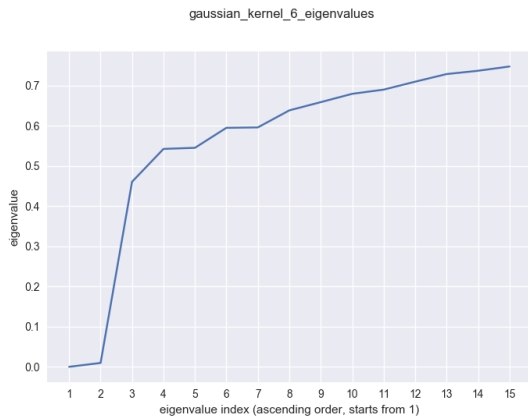
,

Figure 21: Eigenvalues - trying to find the 'eigengap'.

in the similarity graph.

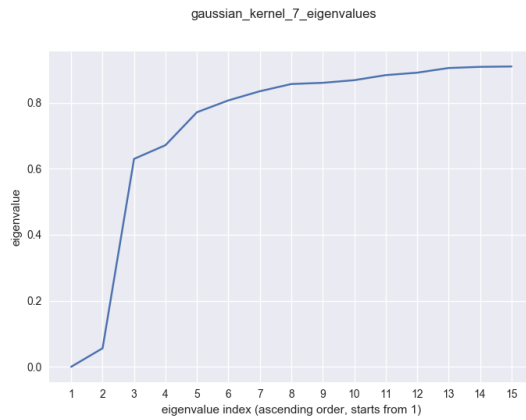
After realizing that m-nearest-neighbors as a similarity function does not work well for this data, I tried to use the gaussian kernel. First I looked at the pairwise distances histogram and found that most of the distances are above 4-5. In terms of percentiles, 4% of the pairwise distances are below 4. Figure 20 shows the histogram of the pairwise distances of this data. Then I ran spectral clustering while using different values of σ, from 2 until 8 with jumps of 0.5. Each time I checked how many samples are in each cluster, to verify that the clustering did not cluster outliers are singletons. For $\sigma = 2$ until $\sigma = 5$ this was indeed the case (clusters of 1-3 data points, which seems wrong). I also plotted the similarity graphs, in order to see if the clustering make sense in terms of different clusters being connected components. Figure 21 shows the eigengap while using $\sigma = 2$ and $\sigma = 4$, but clustering using this parameter (trying $k = 2, \ldots, 8$ clusters) gives clusters containing single (or very few) outliers points. Figure 22 shows the eigengap while using $\sigma = 6$ and $\sigma = 7$, which shows that clustering into 2 clusters seems reasonable.

To summarize, despite the fact that the eigengap seems okay (although the eigengap



(a) $\sigma = 6$. It seems that the eigengap is at $k = 2$.

,



(b) $\sigma = 7$. It seems that the eigengap is at $k = 2$.

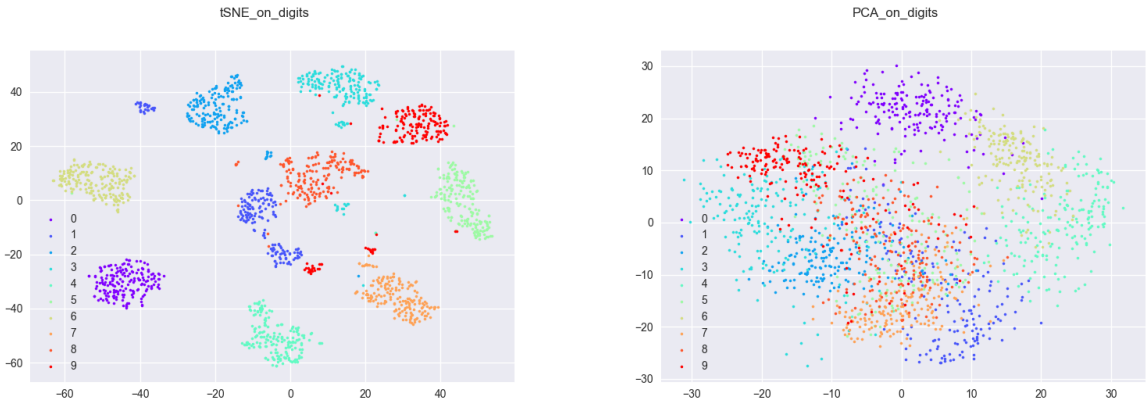
Figure 22: Eigenvalues - trying to find the 'eigengap'.

occur at different k's when using different σ's), clustering using these values does not give good results. As mentioned previously, most of the times it clusters outliers to their own clusters. So overall it seems that k-means works better on this dataset (or maybe we just haven't found the right similarity function and its parameter, but I guarantee I tried a lot :)).

2.5 Projecting high dimensional data into a plane - tSNE v.s. PCA

Figure 23 shows the embedding of the digits dataset (MNIST of size 8×8) into $\mathbb{R}^2$ using tSNE versus PCA. Clearly, tSNE capture the local structure of the data much better than PCA.

In order to further investigate the differences between tSNA and PCA, I created my own synthetic dataset as follows. I created k multivariate gaussians in $\mathbb{R}^d$ with a standard deviation of std (in each dimension), each containing n points. I used values of $k = 8, d = 32, std = 0.2, n = 128$. Figure 24 shows the embedding of this dataset into $\mathbb{R}^2$ using tSNE versus PCA. Clearly, tSNE capture the local structure of the data much better than PCA.

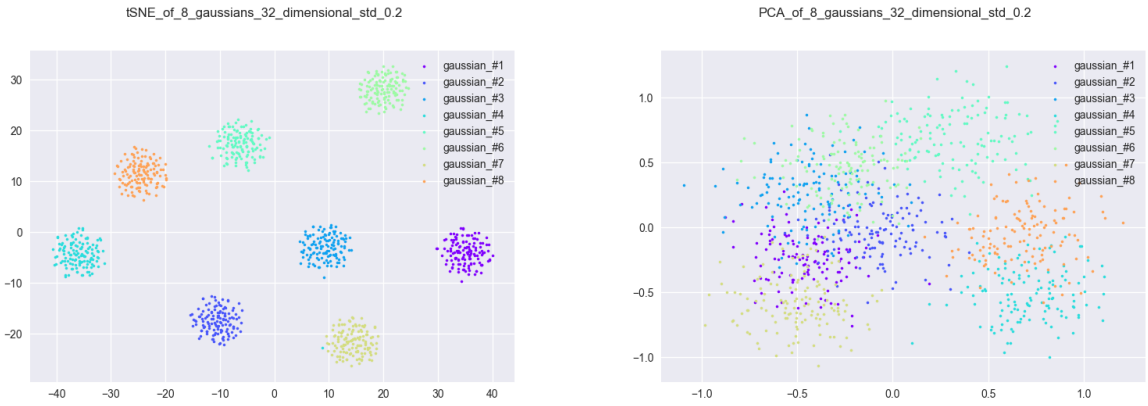


(a) tSNE

(b) PCA

,

Figure 23: Embedding of the digits dataset into $\mathbb{R}^2$ using tSNE versus PCA. Clearly, tSNE capture the local structure of the data much better than PCA.



(a) tSNE

(b) PCA

,

Figure 24: Embedding of the multivariate gaussians dataset into $\mathbb{R}^2$ using tSNE versus PCA. Clearly, tSNE capture the local structure of the data much better than PCA.