APML - Clustering

Rhea Chowers, 204150643

December 19, 2019

Question 1

To prove this we'll show that at each iteration the cost function's value does not increase and therefore converges (since the cost function is greater or equal to zero, and we are working on computers with limited accuracy). Denote the centroids at the t iteration as μ_i^t:

$$cost = \sum_{i=1}^{k} \sum_{x \in C_i^t} ||x - \mu_i^t||^2$$

After this iteration we update $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$. Notice that μ_i which minimizes $\sum_{x \in C_i^t} ||x - \mu_i^t||^2$ is the center of mass which is $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$. Therefore for each cluster:

$$\sum_{i=1}^{k} \sum_{x \in C_i^t} ||x - \mu_i^{t+1}||^2 \leq \sum_{i=1}^{k} \sum_{x \in C_i^t} ||x - \mu_i^t||^2$$

Now we move x's between the cluster i and j iff $||x - \mu_j|| < ||x - \mu_i||$. Therefore:

$$\sum_{i=1}^{k} \sum_{x \in C_i^{t+1}} ||x - \mu_i^{t+1}||^2 < \sum_{i=1}^{k} \sum_{x \in C_i^t} ||x - \mu_i^{t+1}||^2$$

plugging everything together we get that:

$$cost_{t+1} = \sum_{i=1}^{k} \sum_{x \in C_i^{t+1}} ||x - \mu_i^{t+1}||^2 < \sum_{i=1}^{k} \sum_{x \in C_i^t} ||x - \mu_i^{t+1}||^2 \leq \sum_{i=1}^{k} \sum_{x \in C_i^t} ||x - \mu_i^t||^2 = cost_t$$

And we get the required.

Question 2

1. Observe the rectangle around the origin $(a, 1), (a, -1), (-a, 1), (-a, -1)$. Let $k = 2$. If we initialize the 2 centroids at $(0, \pm 1)$ we get that the two

clusters are $(a,1),(-a,1)$ and $(-a,-1),(a,-1)$ - the uppper points and lower points. The value of the cost function is $c_{k-means} = 4a^2$. Notice that k-means won't move any point to the other cluster since the distance of any point from the other cluster's centroid is $\sqrt{2^2+a^2}$ while from its own centroid is $a < \sqrt{2^2+a^2}$. For any $a > 1$ we get that the optimal centroids are at $(1,0)$ and $(-1,0)$ where the cost is $c_{opt} = 4$. Therefore for any $a > 1$ we get $c_{k-means} = 4a^2 > 4 = c_{opt}$.

2. An example of a set of points with a number optimal solutions is the set of the unit square in 2d - $(1,1),(1,-1),(-1,1),(-1,-1)$ when clustered into 2 clustered - each neighboring pair can be clustered together with the same total cost: $c = 4$.

Question 3

1. The probability of sampling a specific good initialization is $p = \prod_{i=1}^{k} \alpha_i$ - since we want to sample exactly one point from each cluster at the i'th sampling. Taking into consideration that there are $k!$ ways to order $\alpha_1, ..., \alpha_k$ we get that the probability for a any good event is $p = k! \prod_{i=1}^{k} \alpha_i$. Therefore the expected number of trials is $EX = \frac{1}{p} = \frac{1}{k! \prod_{i=1}^{k} \alpha_i}$

2. We want to find the maximum of $\prod_{i=1}^{k} \alpha_i$ subject to $\sum_{i=1}^{k} \alpha_i = 1$. We can use lagrange multipliers and see that

$$\frac{d}{d\alpha_j}\left(\prod_{i=1}^{k}\alpha_i + \lambda\left(1 - \sum_{i=1}^{k}\alpha_i\right)\right) = 0 \Rightarrow \frac{1}{\alpha_j}\prod_{i=1}^{k}\alpha_k - \lambda = 0 \Rightarrow \lambda = \prod_{i\neq j}\alpha_i$$

Since this is true for every α_j we get that $\forall m \neq n : \prod_{i\neq m} \alpha_i = \prod_{i\neq n} \alpha_i$ giving us the result that all α_i must be equal and therefore $\sum_{i=1}^{k} \alpha_i = k\alpha_i = 1 \Rightarrow \alpha = \frac{1}{k}$ - meaning the clusters are evenly distributed. Plugging that into the result from (1) we get that:

$$EX = \frac{1}{k! \prod_{i=1}^{k} \alpha_i} = \frac{1}{k! \left(1/k\right)^k} = \frac{k^k}{k!}$$

Using Sterling's approximation which is a good approximation even for low values of $k!$:

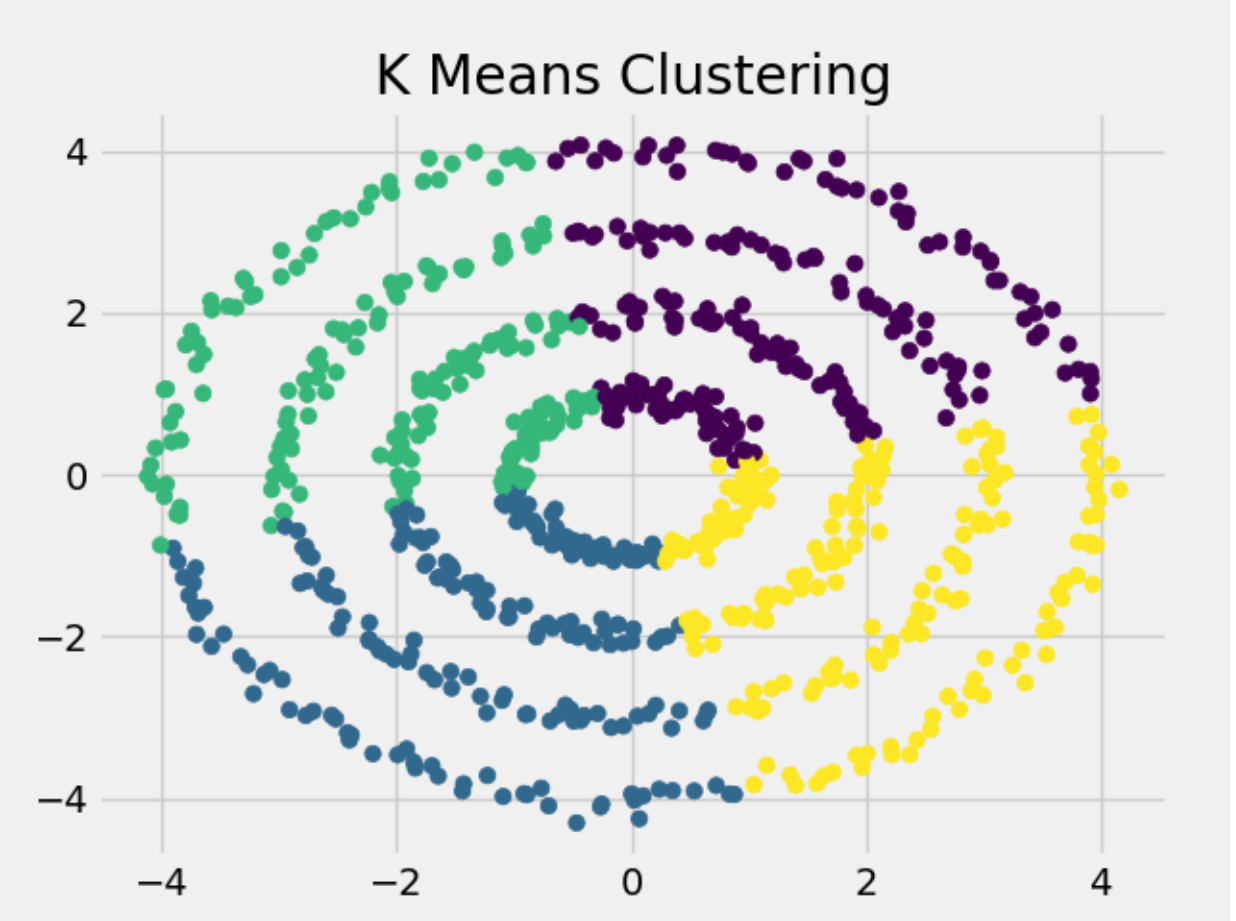
$$\frac{k^k}{k!} \approx \frac{k^k}{\sqrt{2\pi k}\left(\frac{k}{e}\right)^k} = \frac{e^k}{\sqrt{2\pi k}}$$

Meaning that in the best case the expected number of initializations is exponential in k.

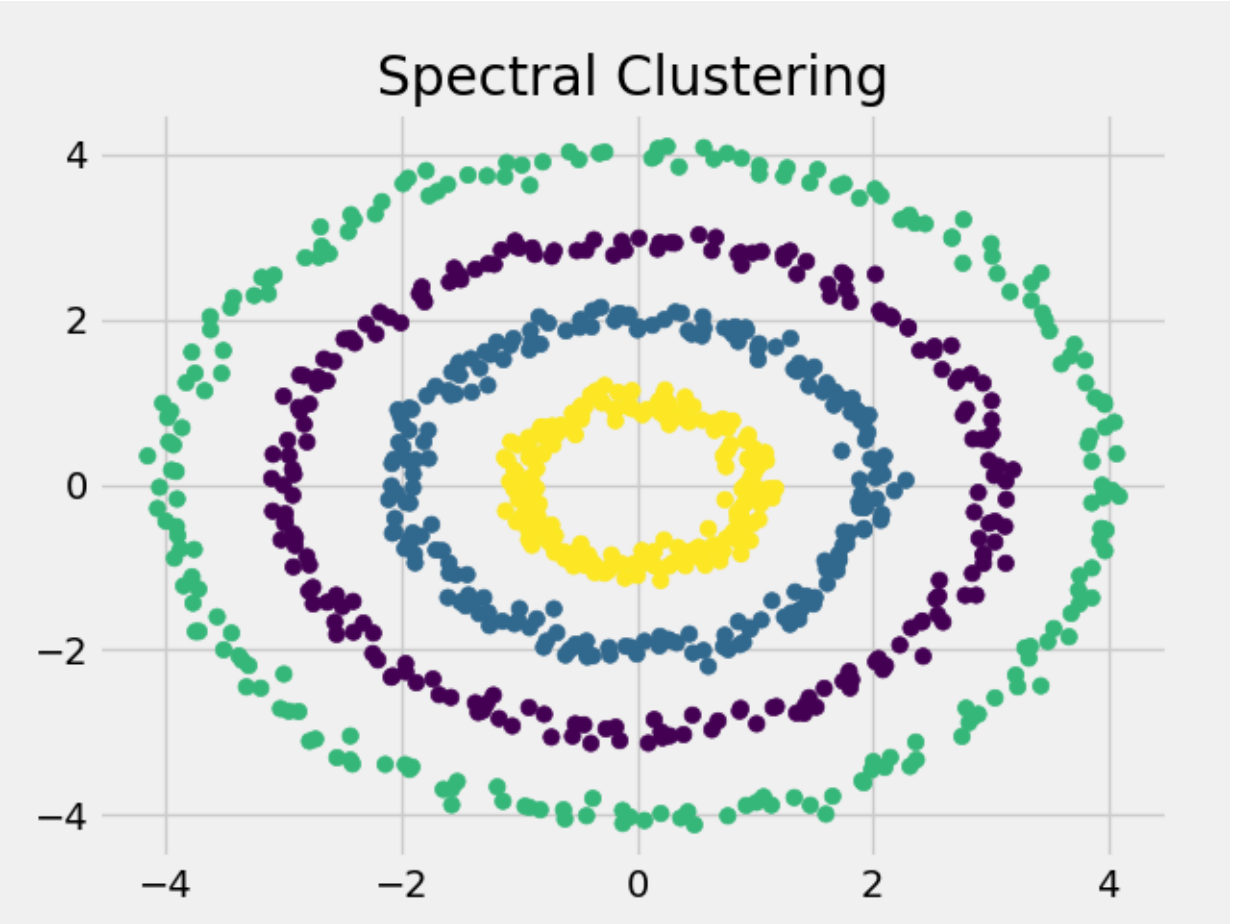
3. From complexity theory, we know that things that grow exponentially are not feasible. But, in most cases our k isn't very large, and lies in the **low** single digit range. Even for $k = 10$ we would get $EX \approx 2700$ which isn't an infeasible number of initialization iterations.

K means vs Spectral Clustering

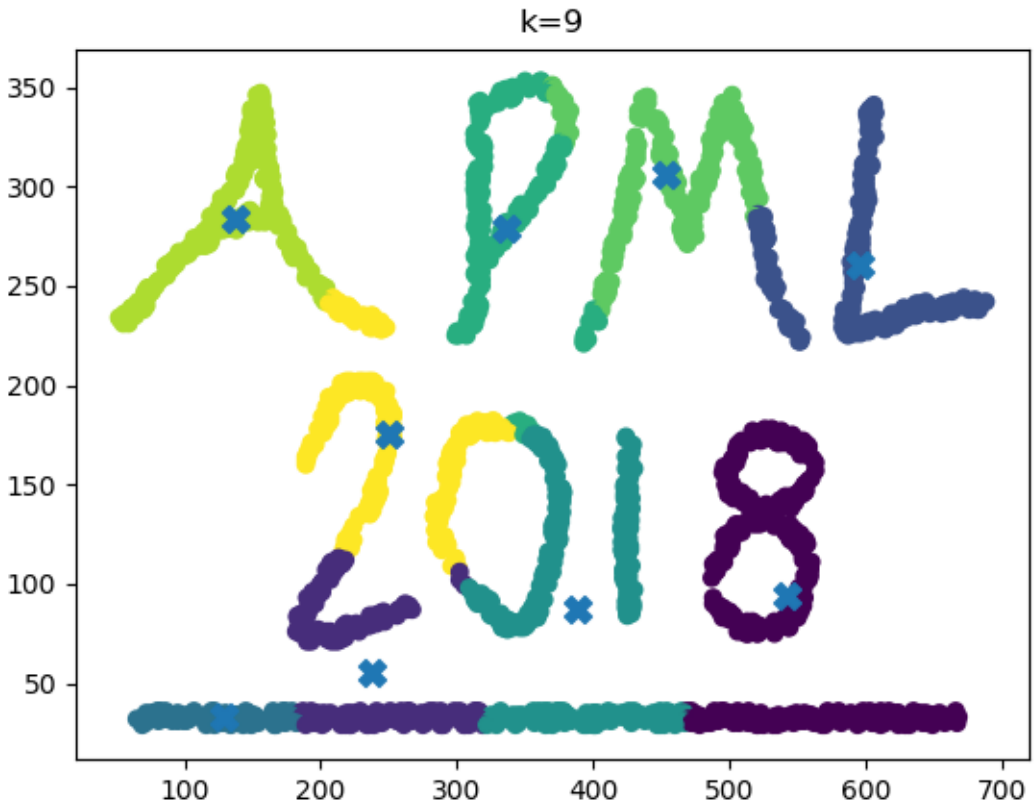
K means preforms the best when we have clusters which are seperable by hyperplanes due to it being based on euclidian distances between the points. Therefore we expect it to fail on tasks such as the circle clustering, as can be seen in the following figure:

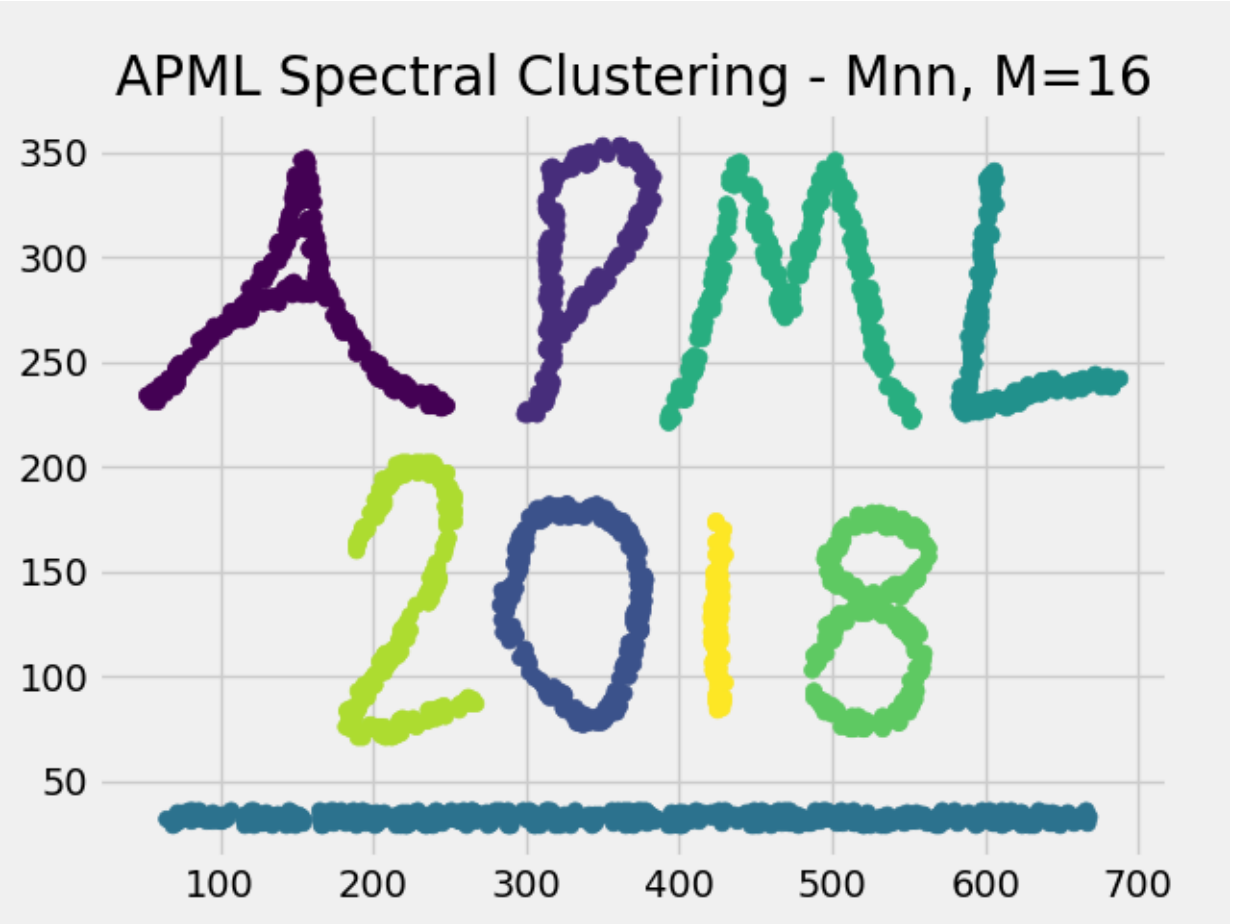


On the other hand, spectral clustering using a gaussian kernel gives better results since this method uncovers the "hidden" (in a euclidian point of view) connected components in the graph:



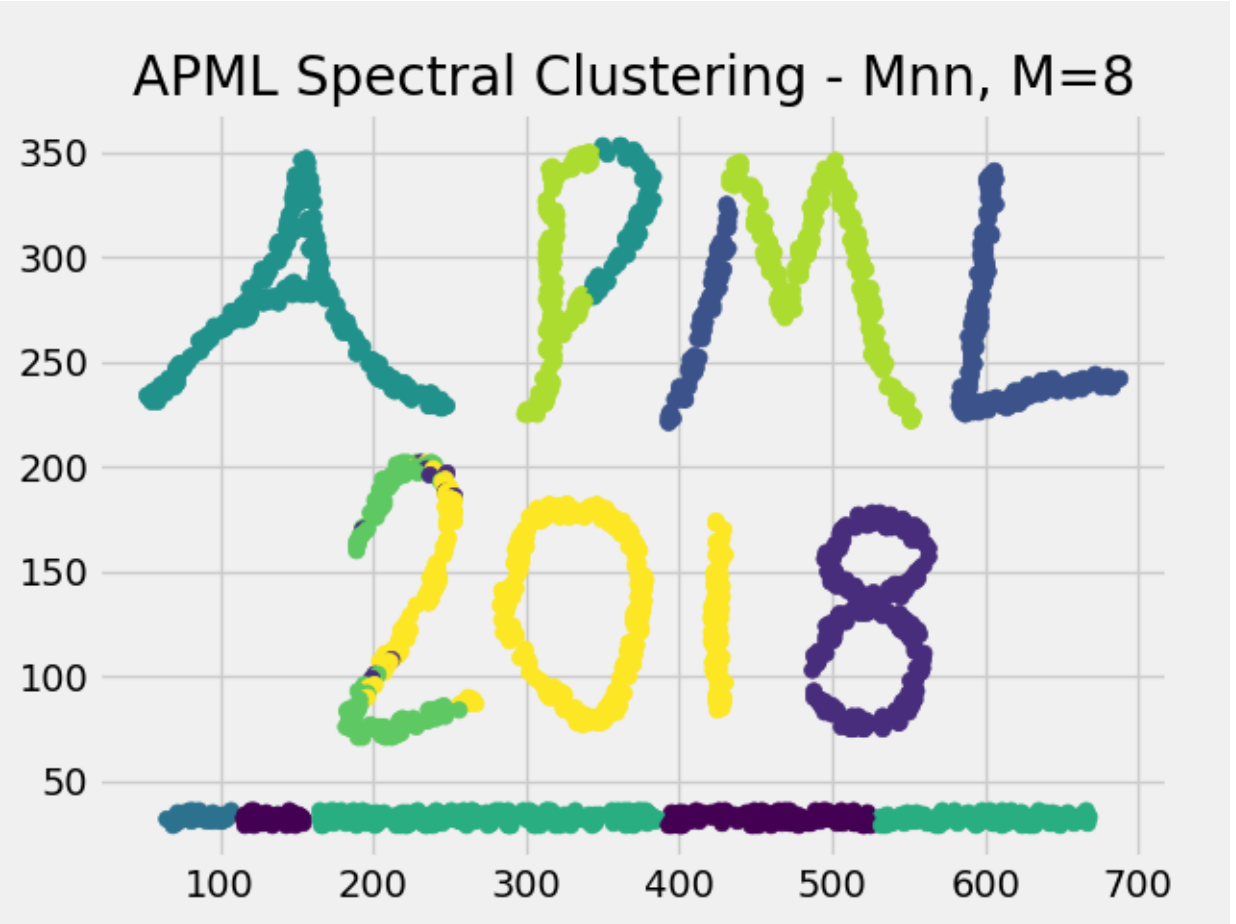
We can see that this is the case for the APML data as well - the first figure displays the kmeans algorithm's result and the second displays the spectral clustering using a mnn kernel:

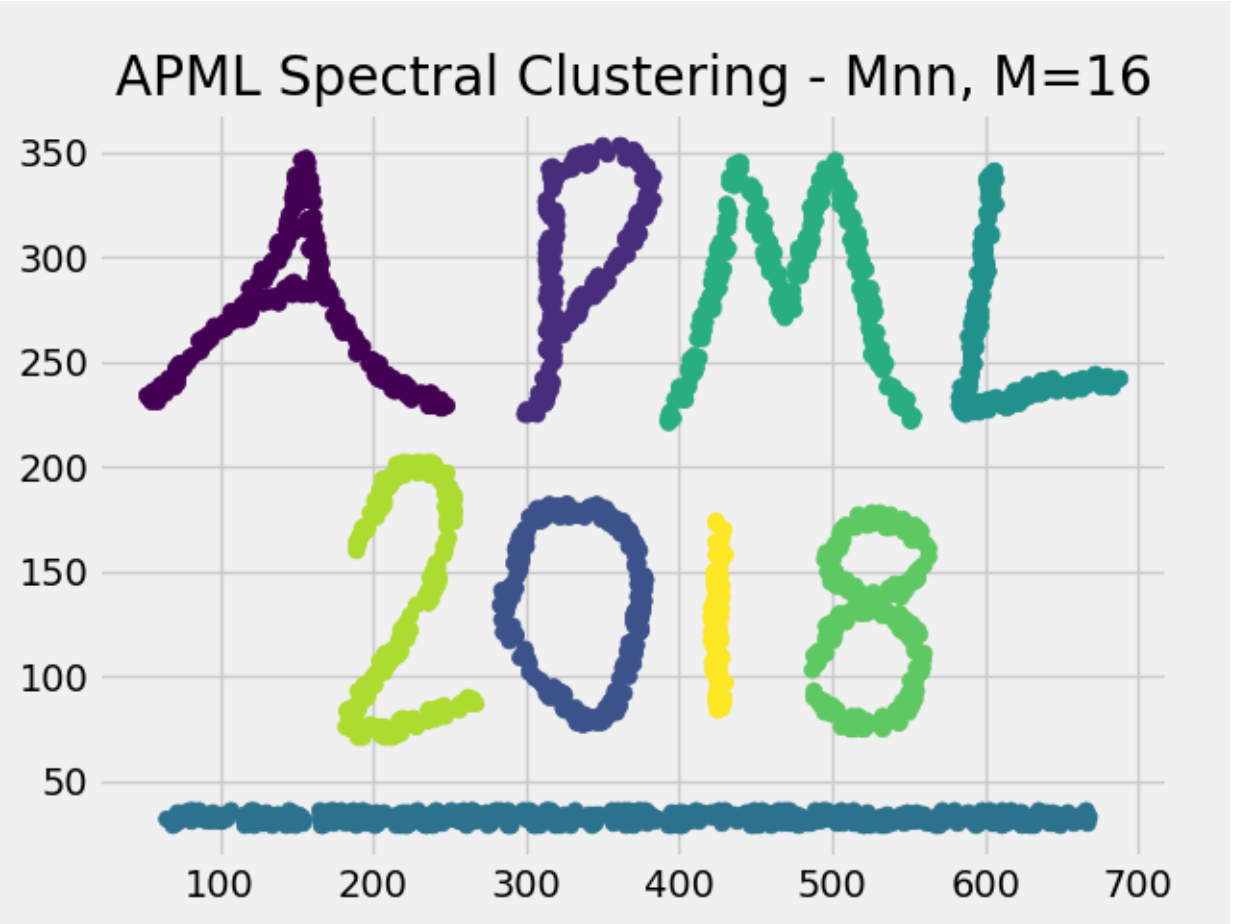

k=9
350
300
250
200
150
100
50
100
200
300
400
500
600
700



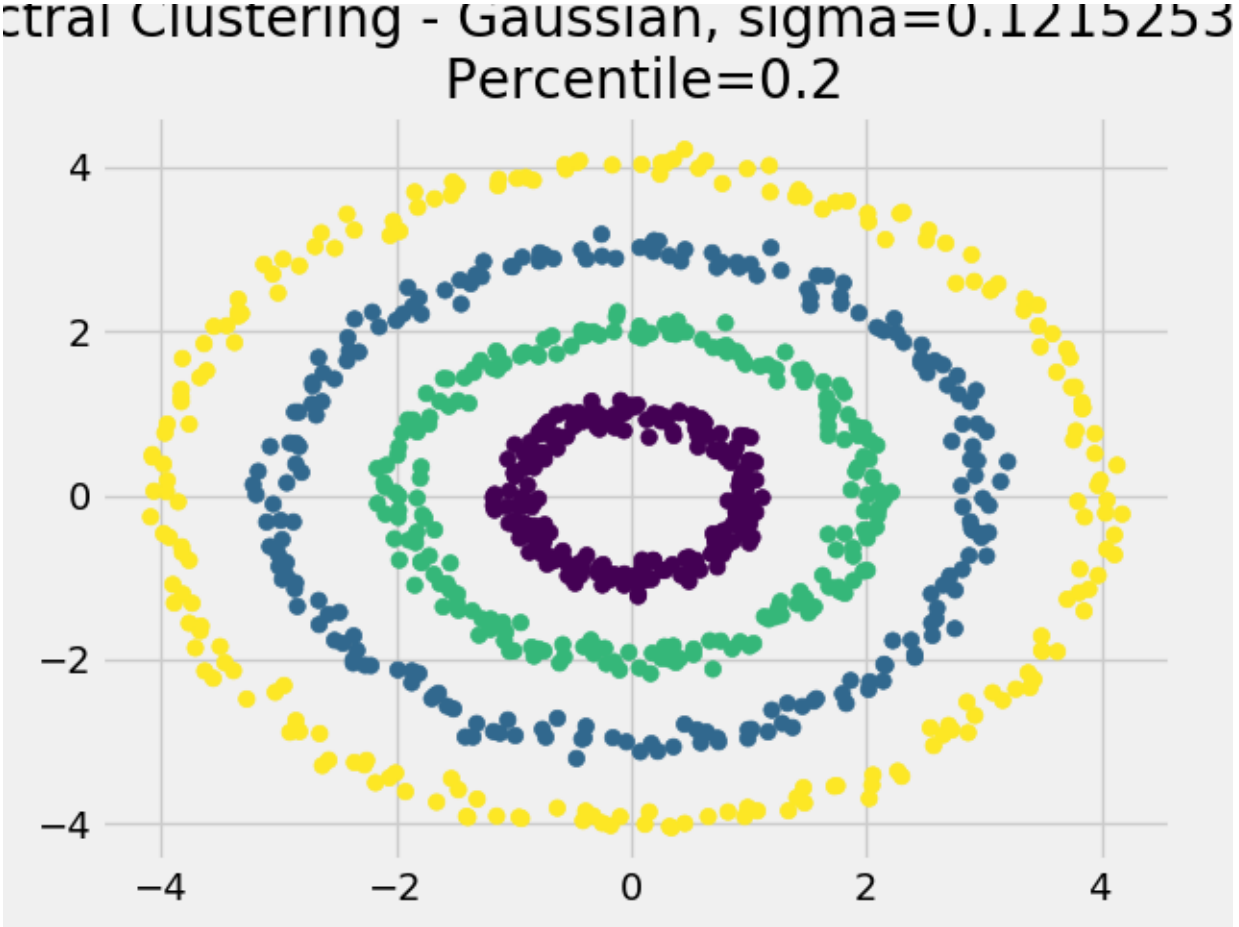
Parameters of Spectral Clustering

The parameters of spectral clustering - m for the mnn and σ for the gaussian kernel - define connectivity between the samples. While mnn is straightforward, σ can be thought of as a 'radius' around each point where samples that fall in this radius are considered neighbors and the rest are not (this is of course done in a smooth manner and not discretely). Therefore, choosing this parameters depends on the specific set of points we are looking at. For example, in the circles dataset we don't want to choose a large σ - although the distances between two opposite points in each circle is $2r$, the distance between two points in different circles can be r. Therefore we need to choose σ s.t. only points in the same circle will be considered neighbors, and rely on the transitivity of connectiveness for clustering entire circles together.
For the mnn case, we'll show see how this affects us in practice. The following figures cluster the APML data using different neighbor parameters:

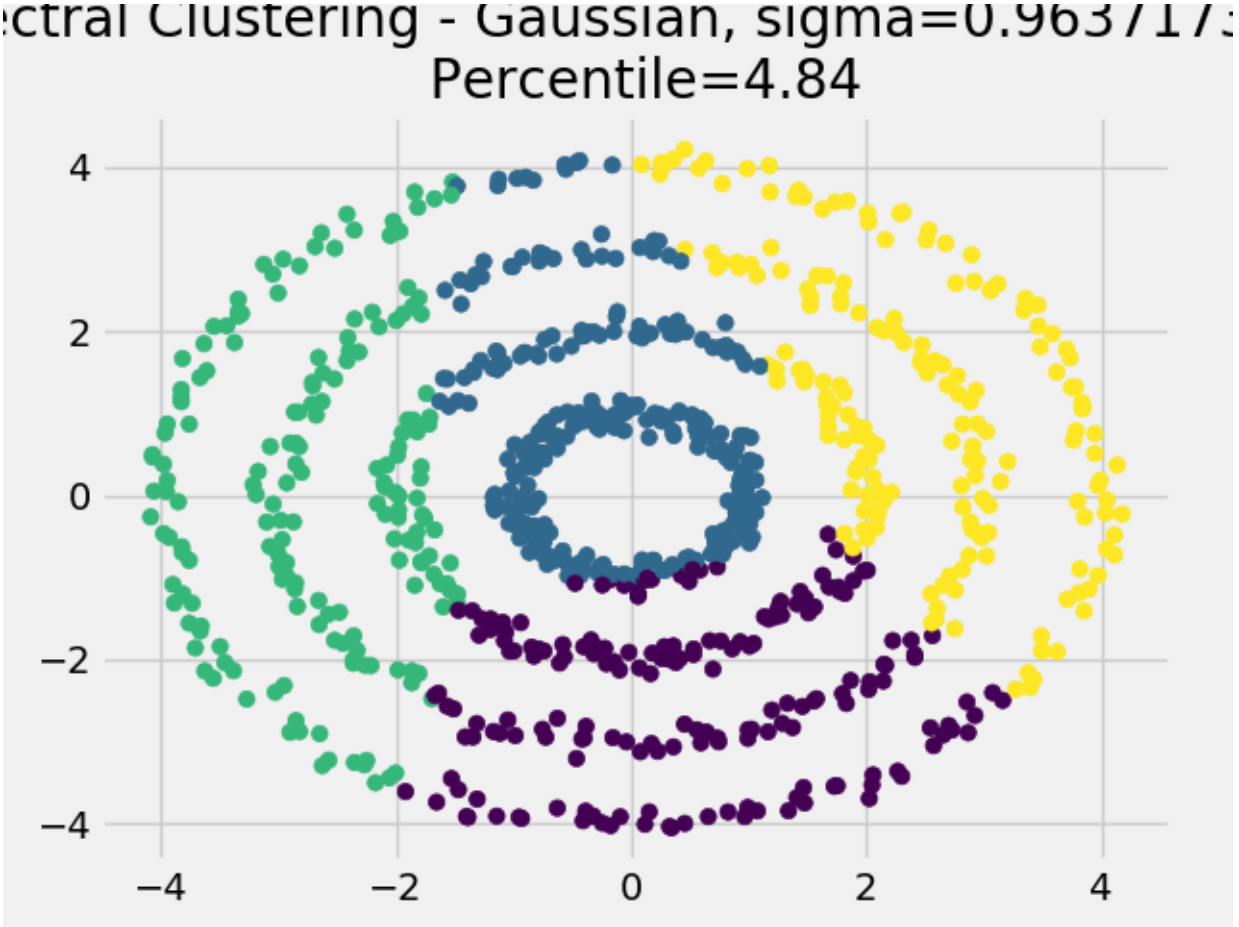

APML Spectral Clustering - Mnn, M=8
350
300
250
200
150
100
50
100
200
300
400
500
600
700



And for the gaussian kernel, we calculated σ according to different percentiles. We can see that for low values we get excellent clustering:

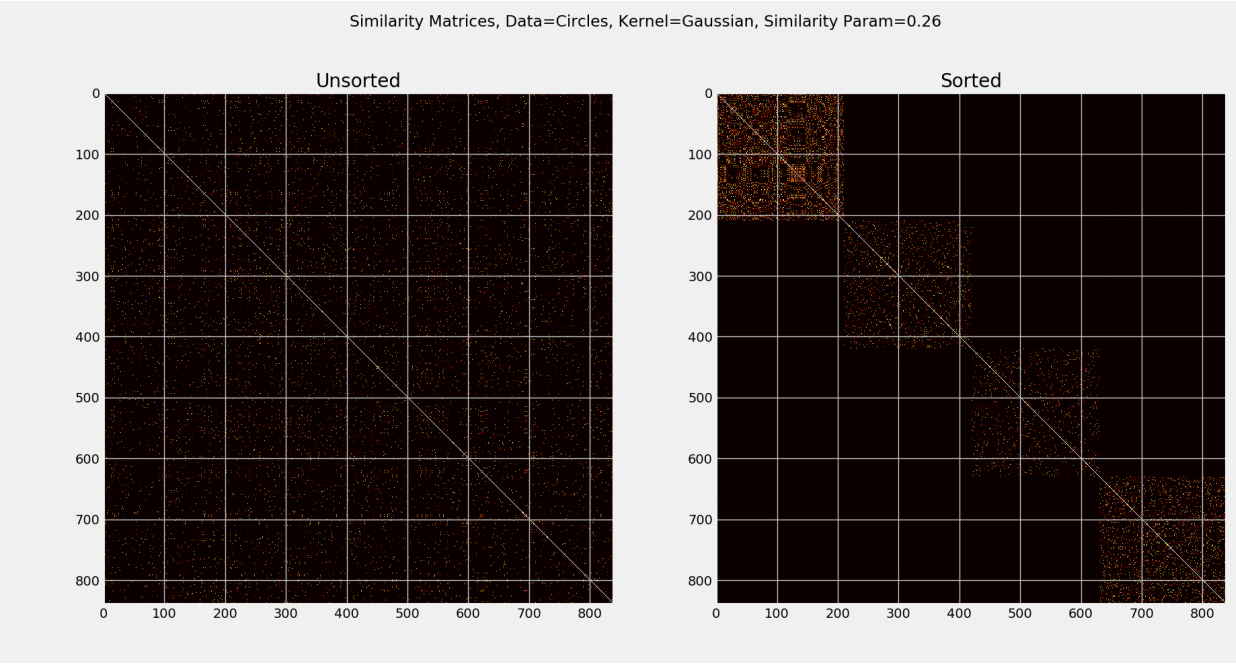


But for higher values of σ we get that the definition of neighbor starts to include points from different circles, and we get a result more similar to a regular k-means:



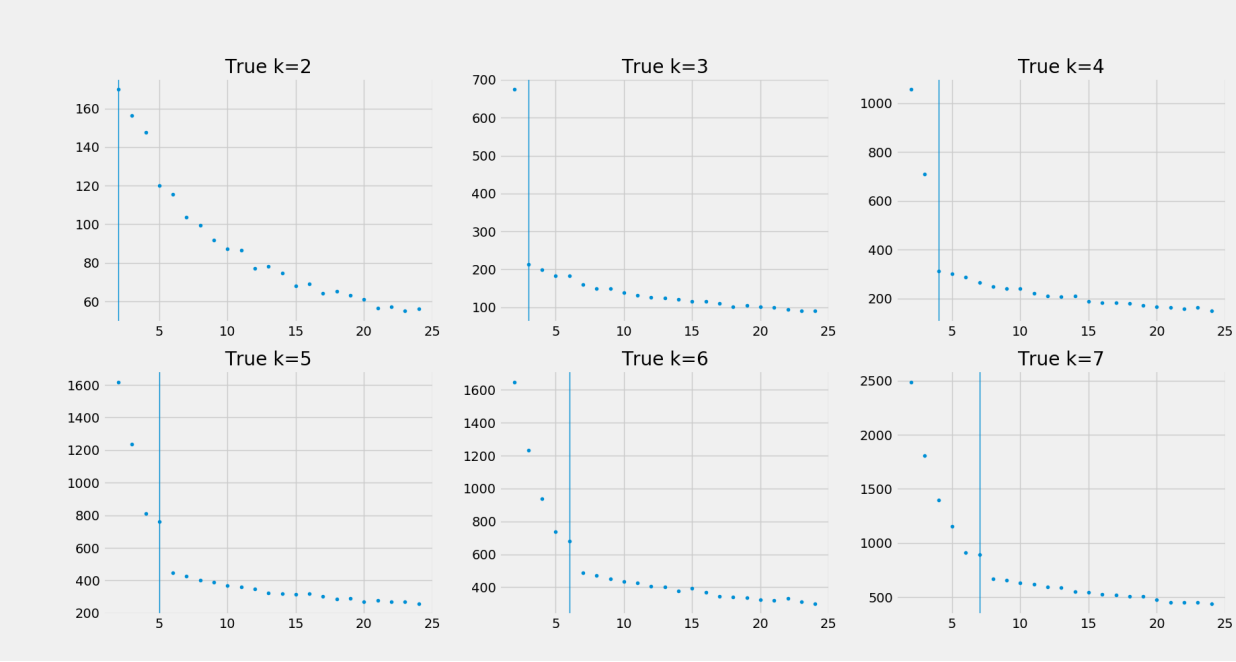
Plotting the Similarity Graph

In the following figure we can see the similarity matrices depicted for the original similarity matrix (a gaussian kernel of the distances matrix for shuffled data) and after it has been sorted according to the clustering returned by the spectral clustering algorithm. In this case I used the true value of k, and divided 4 circles around the origin into 4 clusters. We can clearly see that the clusters form square blocks in the similarity matrix, each block the size of the points in the cluster. This way we can easily understand why the eigenvectors of the laplacian are indicators of cluster members.



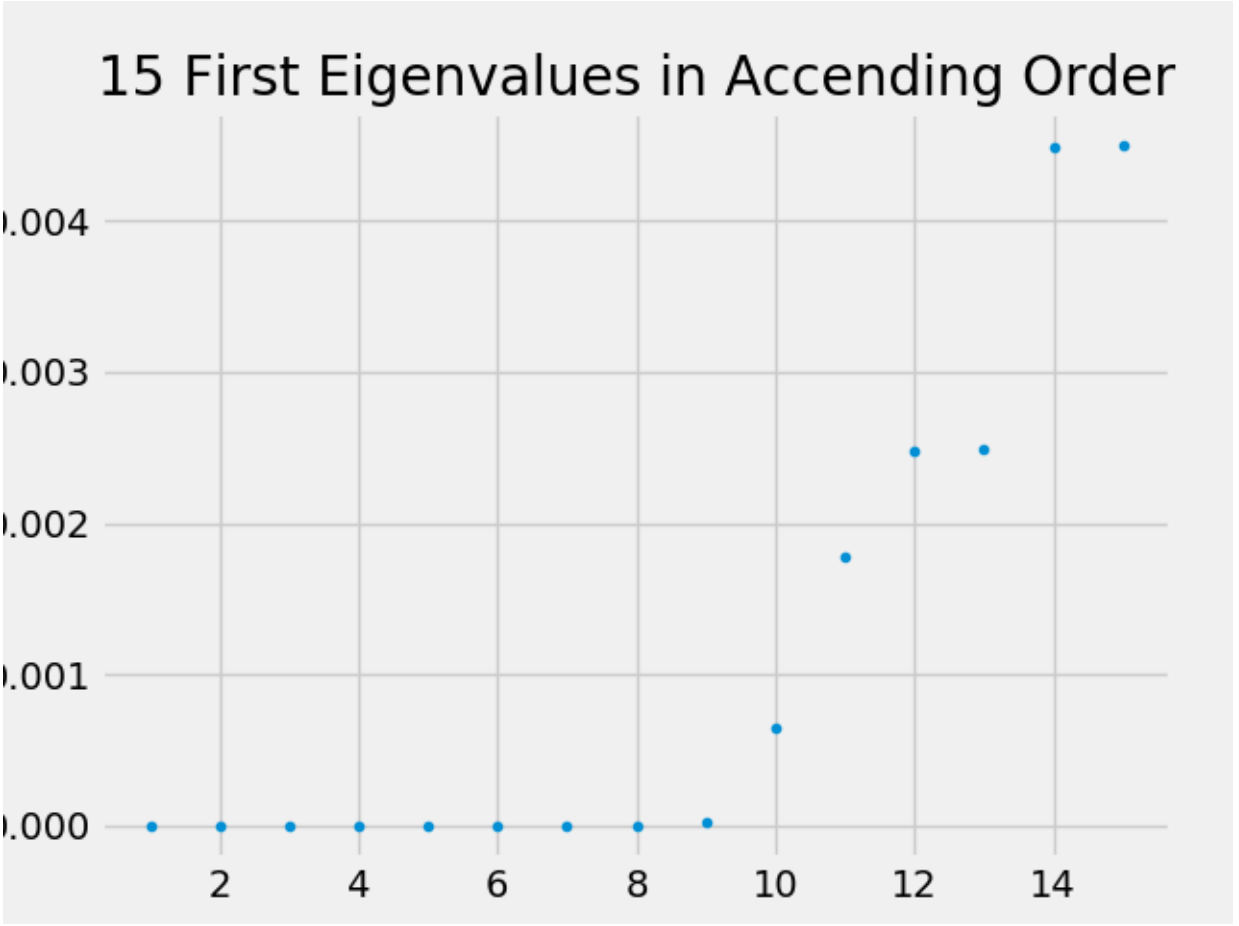
Choosing K

In the following figure we can see a demonstration of the effectiveness of the "Elbow" method. I created a synthetic data set - for each k, I generated k normally distributed set of points in 2d (of a random size between 100 and 200) with the same variance and different mean. For each of the "true" k's I calculated the cost of k-means for a wide range of values of k. In the following plot the y-axis represents the cost and the x-axis the value of k. The real value of k is represented by a vertical line. We can see that the vertical line intersects with the elbow of each plot, therefore demonstrating its effectiveness - if we were to choose the "elbow" k we would be very close (± 1 relative to the true value, depending on how we define the exact point of the elbow) if not exactly on the right value of k.



Eigengap

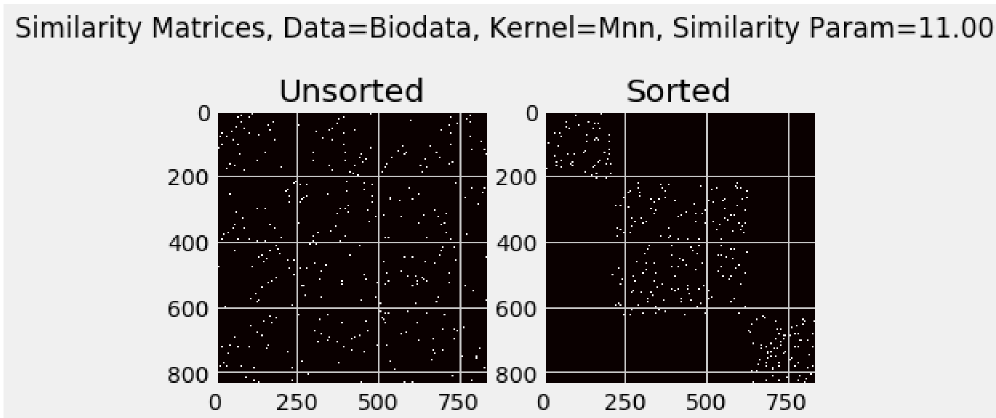
In the following figure we can see the first 15 eigenvalues of the normalized laplacian in accending order for the APML data:

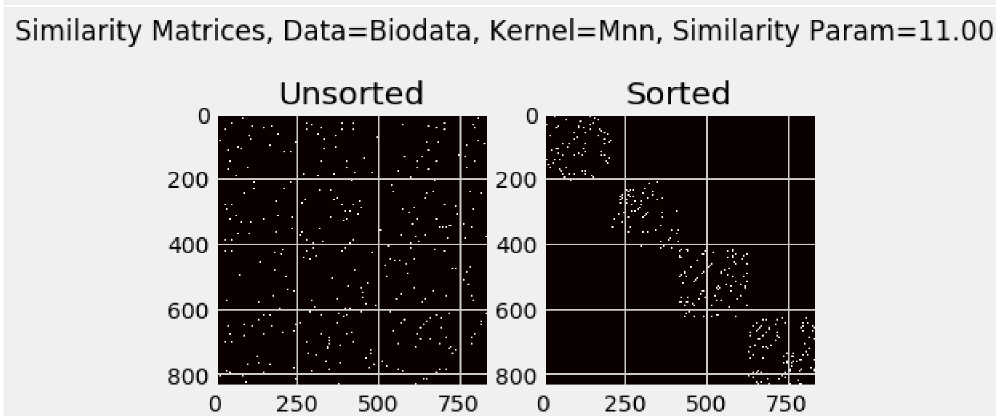


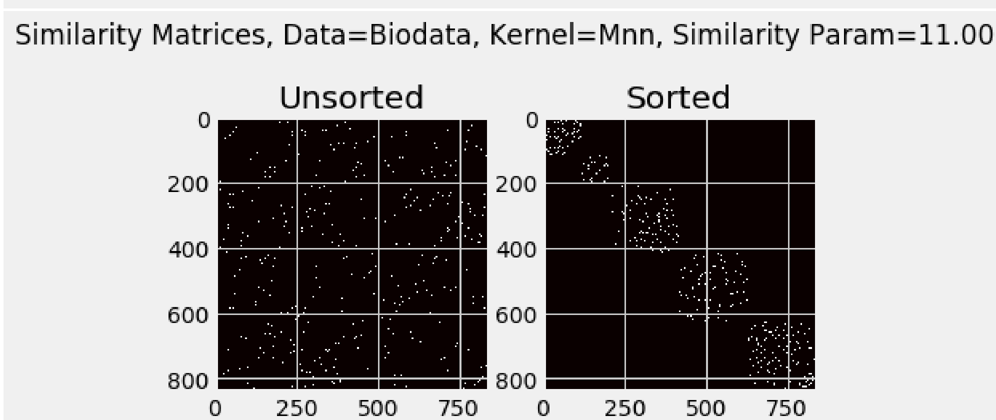
As clearly shown in the figure, the first 9 eigenvalues are very close to zero, and the 10'th eigenvalue takes a first 'jump' in value, and creates the first eigengap. We know that the correct clustering for the data is 9 clusters and this supports this claim (since each cluster can be derived from some eigenvector composed of 1's and 0's of the laplacian with eigenvalue 0, indicating which samples are in the cluster).

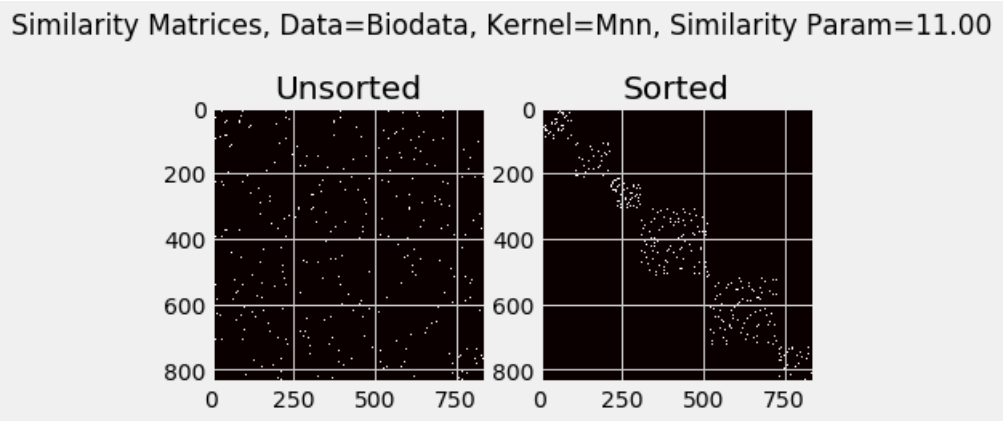
Biological Clustering

Now we can put what we have learned to real use. First of all, just by observing the similarity charts we get a sense of the correct clustering, we can see that clustering with $k = 4$ creates four equally sized clusters, while using $k = 3, 5, 6$ returns an unbalanced clustering. If we knew that the data was uniformly distributed (which we don't) we could choose $k = 4$ just by these graphs (the number of clusters in each image is 3,4,5,6 accordingly):

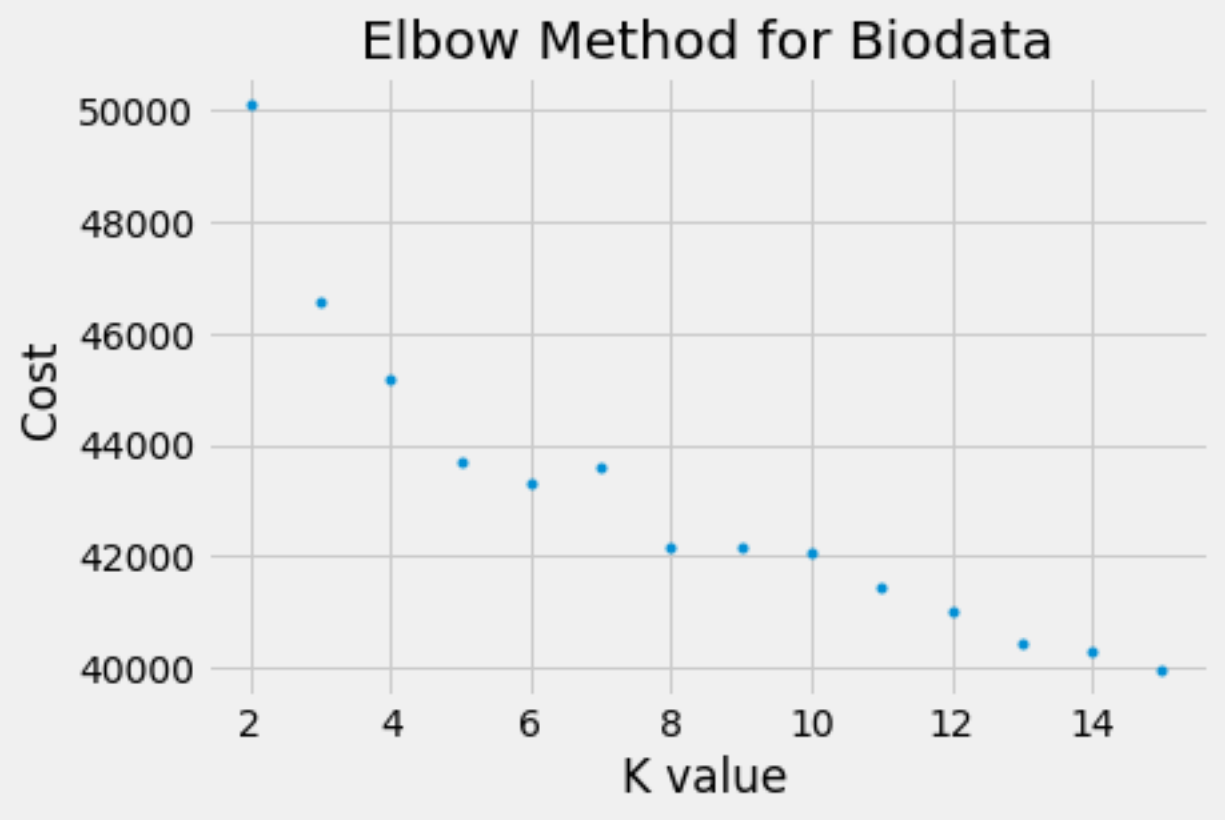

Similarity Matrices, Data=Biodata, Kernel=Mnn, Similarity Param=11.00
Unsorted
Sorted
0
200
400
600
800
0
250
500
750


Similarity Matrices, Data=Biodata, Kernel=Mnn, Similarity Param=11.00
Unsorted
Sorted
0
200
400
600
800
0
250
500
750

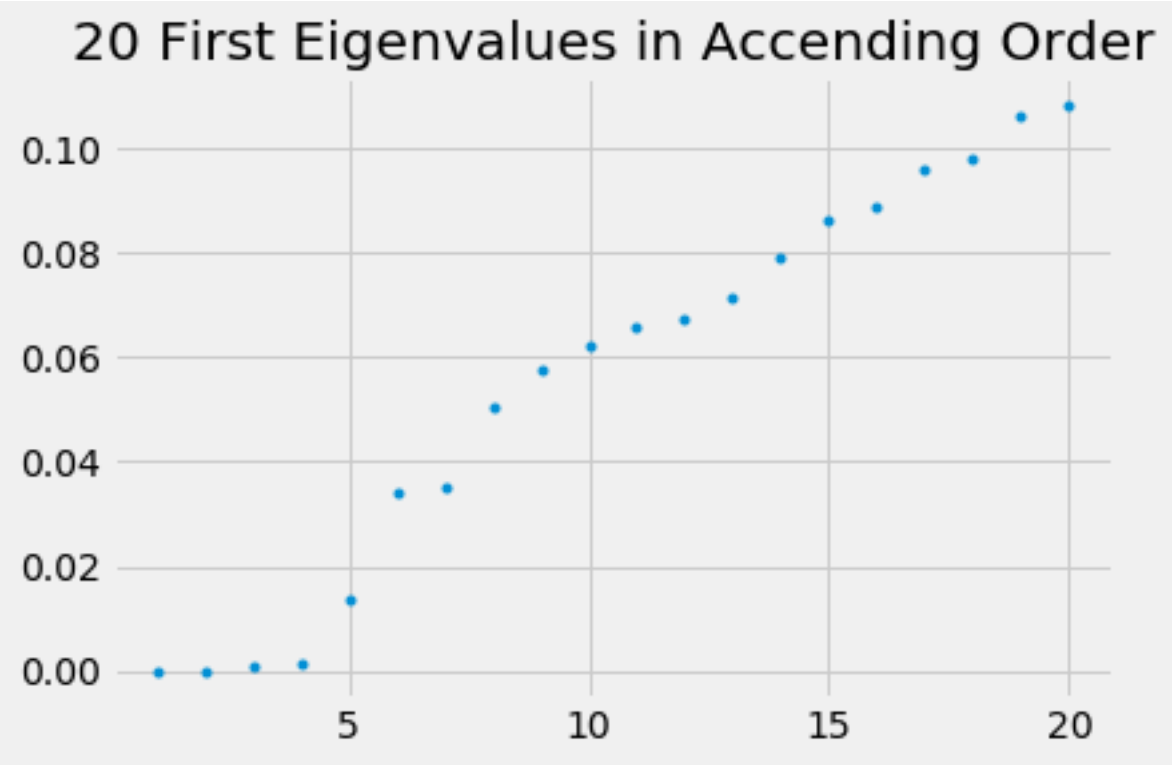

Similarity Matrices, Data=Biodata, Kernel=Mnn, Similarity Param=11.00
Unsorted
Sorted
0
200
400
600
800
0
250
500
750



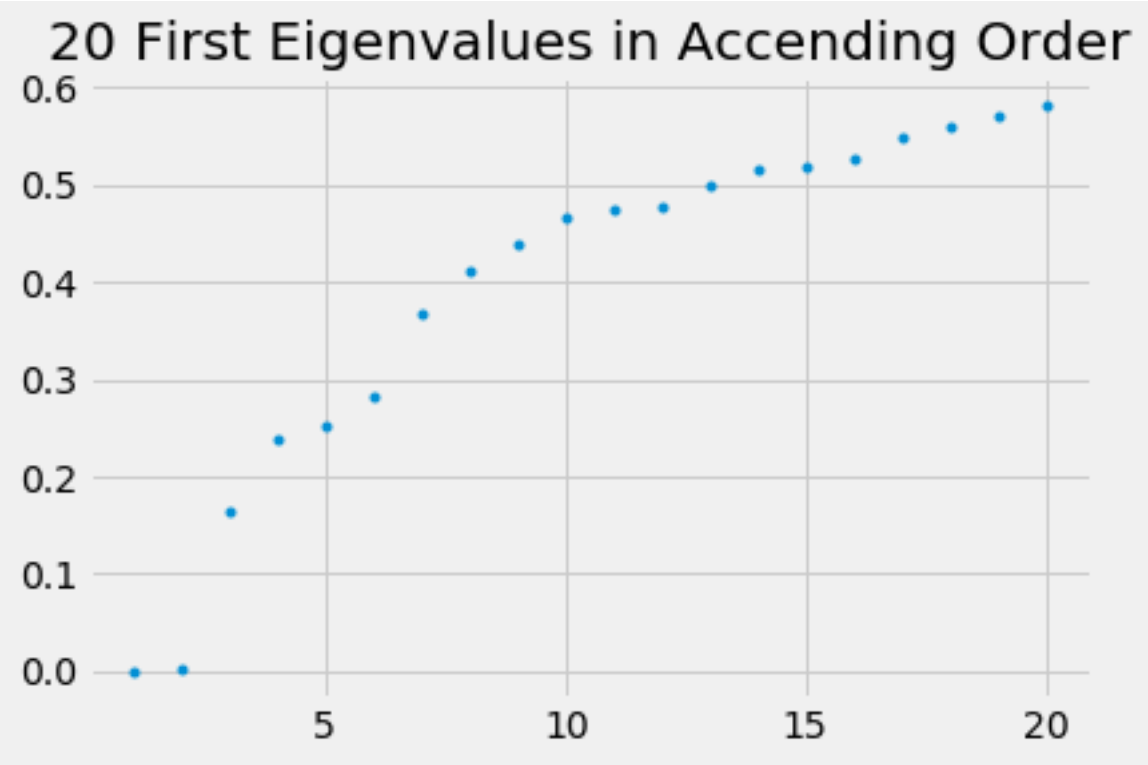
Next we can look at the elbow and eigengap methods. Using the elbow method we can see it is around 4-5, strengthening our observation from before:



Next we would like to calculate the eigengap. I tried this for a couple of values of σ. For example, for $\sigma \approx 4$ we get that the eigengap is also around $k \approx 4$ as before.



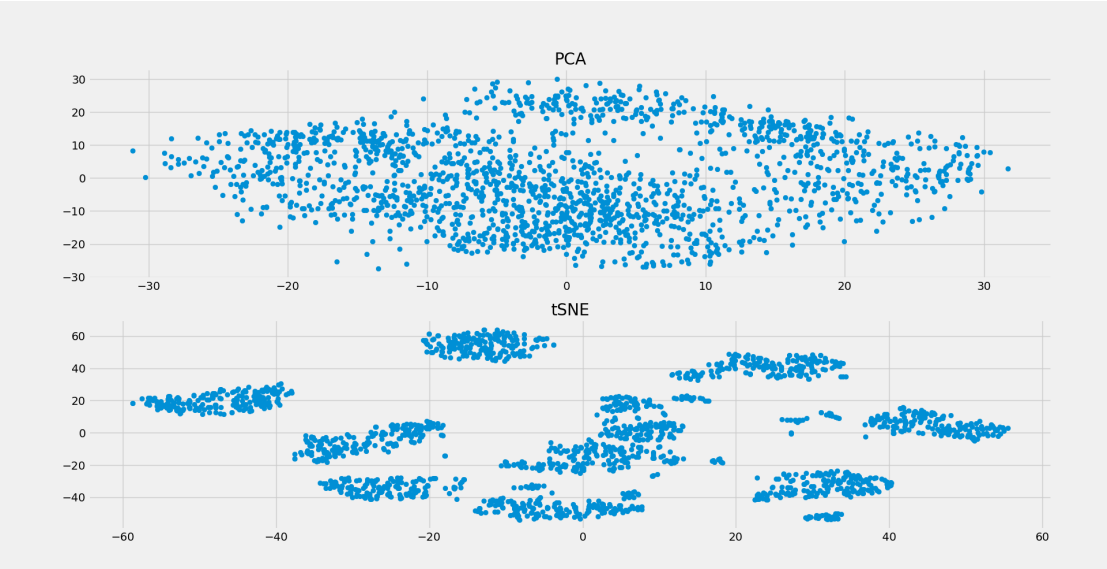
But for different values of σ such as $\sigma \approx 5$ we can see that the eigengap is at $k = 2$:



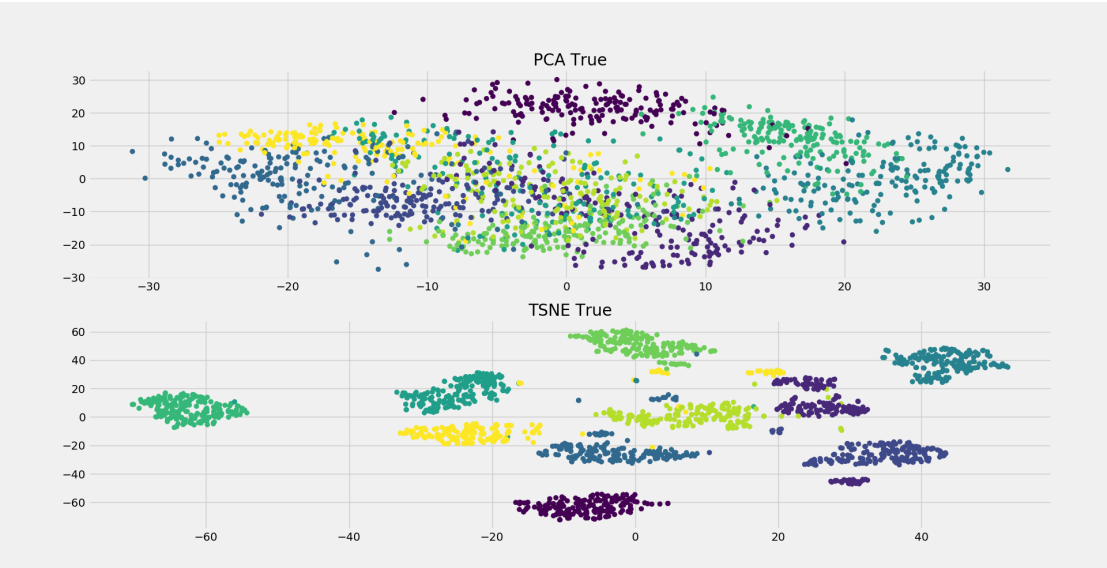
We conclude by understanding that first of all the data is clusterable - for $k = 3, 4, 5, 6$ we can see meaningful clusters, and for $k = 4, 5$ we get justification from the elbow and eigengap method. On the other hand, we can see that the eigengap changes when applying a different σ. This leaves us with questions - should we take the k that spectral clustering (eigengap) and the elbow method (kmeans) agree on? Should we continue testing various values of σ? This is an entire field of study so we will leave this as an open question.

tSNE

In the following figure we can see the dimensional reductions of the 8x8 MNIST database to 2-d. Just from looking at the image we can see that tSNE returns highly seperable clusters compared to the PCA dimensionality reduction, which returns an almost uniform cluster.



But this isn't enough. Obviously, k-means will achieve a relatively good result in seperating the tSNE clusters. But now we must wonder if these seperable clusters are actually the different digits in the MNIST dataset or not. To do this, the next figure shows the two dimensionality reductions with the true data labels. As clearly seen in the image, the clusters created by the tSNE algorithm are almost identical to the true labeling, meaning points with the same label are clustered together. In PCA we also see a trend - points with the same label are mostly in the same region, but due to the low seperability of the reduction these tend to overlap.



Now we can apply the k-means algorithm to show that it would have managed

clustering using the tSNE reduction, and less succesfully with the PCA reduction.