Assignment 6: Probabilistic ML

Marcus Hansen

December 2024

1. Principal Component Analysis (PCA) of Genomes (40 Points)

...

Q1 Say we ran PCA on the binary matrix X above. What would be the dimension of the returned vectors? [2 Points]

W are unsure of if the returned vectors are, either 1. the dimension of the principal components or 2. the dimension of the projected data samples. So we will try to answer both.

For 1. From the lectures we know that the principal components are the eigenvectors of the (empirical) covariance matrix of the data X. The matrix is square since it is a covariance matrix and its eigenvectors will have dimension $\max(\dim(X))$. In our case $\dim(X) = (N, R)$ and $R > N$, hence the eigenvectors and principle components are of dimension R, i.e. the number of features which is 10101.

For 2. the projected data samples will be of dimension k, which is a number we choose, when deciding the number of principal components to keep. Hence we will have a $N \times k$ output when transforming all N data samples.

Q2 Examine the first two principal components of X. These components contain significant information about our dataset. Create a scatter plot with each of the 995 rows of X projected onto the first two principal components. In other words:

- **The horizontal axis should be v1.**
- **The vertical axis should be v2.**
- **Each individual should be projected onto the subspace spanned by v1 and v2.**

Your plot must use a different color for each population and include a legend. [8 Points] See figure 1.

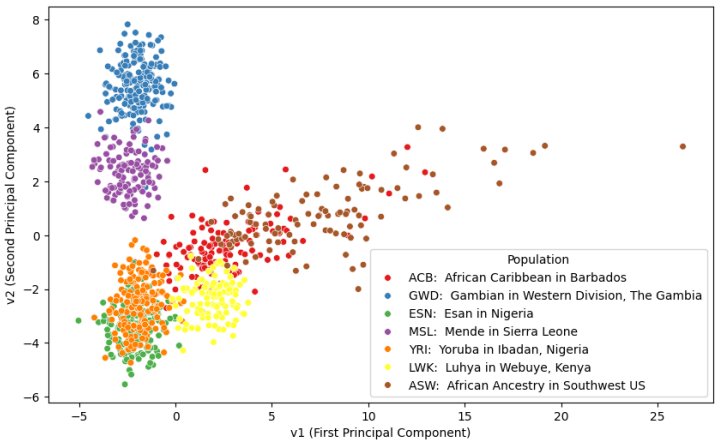


Figure 1: Scatter plot of transformed data over the first two principal components.

Q3 In two sentences, list one or two basic facts about the plot created in part (b). Can you interpret the first two principal components? What aspects of the data do the first two principal components capture? Hint: Think about history and geography. [10 Points]

One or two facts in two sentences: Most data points on the left in figure 1 are from people living in Africa, the ones spread out to the right are from people living in central parts of America. Gambia and Sierra leone are both countries located in the north west of Africa and the points corresponding to these people are all located in the top part of the plot, whereas orange and green points are both representing points of people living in Nigeria which is a country south east of Gambia.

What aspects of the data we think the first two principal components capture: It seems that the first principal component captures changes in genes for Africans after emigration to America. The second principal component seems to capture differences in north or west-, large positive values and south or east-, negative values, African genes.

Q4 Examine the third principal component of X. Create another scatter plot with each individual projected onto the subspace spanned by the first and third principal components. After plotting, play with different labeling schemes (with labels derived from the meta-data) to explain the clusters you observe. Your plot must include a legend. [6 Points]

See figure 2. In the notebook there is also a plot including both population and gender in different colors of edges and centers, but it is a bit messy and therefore not included in the report.

Q5 What information does the third principal component capture? Write a one-sentence answer. [8 Points]

The plot of the first and the third principal component, see figure 2, reveal that the third principal component seems to capture quite a lot of information about the gender of the sample since there are two clusters that can be separated quite good with a line perpendicular to the third principal

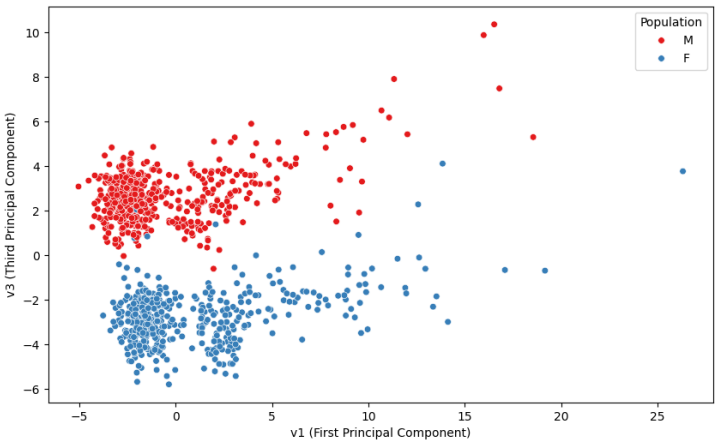


Figure 2: Scatter plot of transformed data over the first (horizontal axis) and third (vertical axis) principal component.

component axis.

Q6 Inspect the third principal component. Plot the nucleobase index versus the absolute value of the corresponding value of the third principal component in that index. (The x-axis of your plot should go from 1 to 10,101—you're literally just plotting the 10,101 values in the third principal component.) What do you notice? What's a possible explanation? Hint: Think about chromosomes. [6 Points]

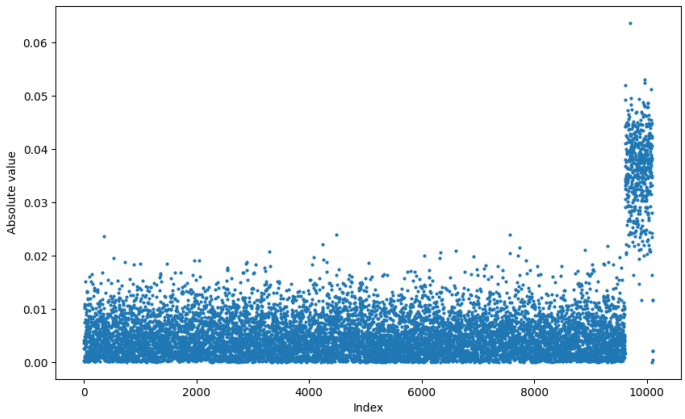
In figure 3a we see that the magnitude of the first $\approx$ 9600 elements of the third principal component is quite small (looks a bit like noise), but for the remaining elements we see a sudden jump to a much higher mean. This jump in higher values can be seen better in figure 3c (although there it is not the absolute values, only values). One possible explanation for this is that the the data is ordered in chromosome pairs so that the last pair of chromosomes, which is the one that differs between women and men comes last in the dataset, hence it captures the parts of the genes that matter when distinguishing between male and female.

To elaborate a bit on this, we know from the lectures that when we transform data sample x to the transformed point $\tilde{x}$ in the new space. We take the dot product between the i'th principal component u_i and x to calculate the i'th element in the transformed point $\tilde{x}_i$, projecting x onto the direction u_i:

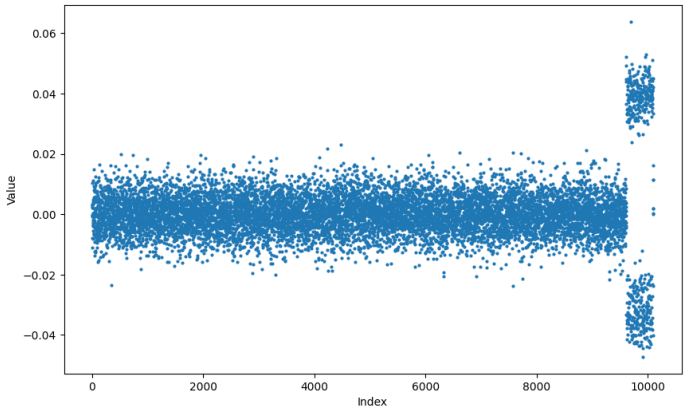
$$\tilde{x}_i = u_i \cdot x$$

I.e. to calculate the values $\tilde{x}_3$ plotted on the vertical axis in figure 2, we take the dot product of each data sample x and the third principal component u_3.

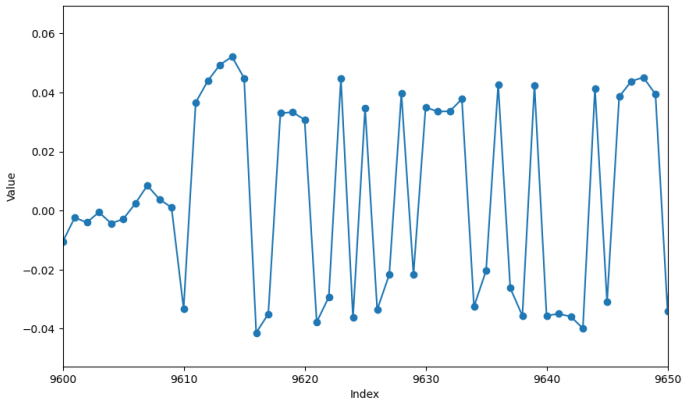
$$\tilde{x}_3 = u_3 \cdot x$$



(a)



(b)



(c)

Figure 3: Scatter plot of the absolute values (a) and values (b) of the elements in the third principal component. (c) is figure (b) zoomed in on index $j \in [9600, 9650]$.

Because of this dot product, element j in a given data sample x will be weighted by element j in u_3 in the weighted sum that defines $\tilde{x}_3$. Hence it is evident from figure 3a that the elements in x for $j > 9600$ will be the ones deciding whether $\tilde{x}_3$ is likely to be positive (male) or negative (female).

2. Clustering with k-Means and EM (40 Points)

This exercise requires you to implement the k-means and Expectation-Maximization (EM) algorithms for clustering a synthetic dataset. You are provided with a partially implemented Python file `E2.ipynb`, which includes:

- Data generation: A synthetic 2D dataset with Gaussian clusters.
- Plotting utilities: A function to visualize the dataset and clustering results at different iterations.
- Class definitions: Definitions for:
  - `KMeansCustom` (for k-means clustering)
  - `EMGMM` (for Expectation-Maximization)

Your task is to fill in the missing functions in the provided class definitions to complete the implementation of the k-means and EM algorithms. You will then use these implementations to analyze clustering performance and compare the results.

Q1. Data Visualization Run the provided `E2.ipynb` file to generate and visualize the synthetic 2D dataset.

Verify that the ground-truth cluster labels and means are plotted correctly.

The Ground-truth cluster labels and means were plotted correctly.

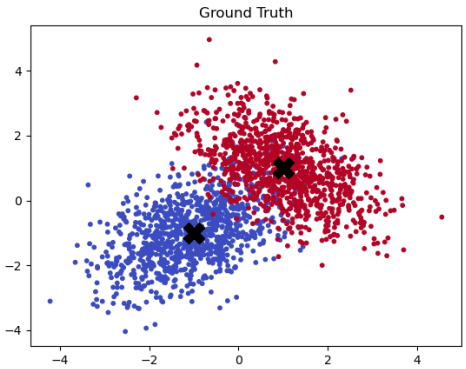


Figure 4: Ground-truth cluster labels of data.

Q2. Implement k-Means Clustering (10 Points) Complete the implementation of the following functions in the `KMeansCustom` class.

See the Jupyter Notebook for the implementation. In Figure 5, the results of the K-means clustering for different numbers of iterations are displayed. It is clear that the algorithm visually converges after approximately 5 iterations. K-means assigns hard labels meaning that each point is either class 1 or class 2.

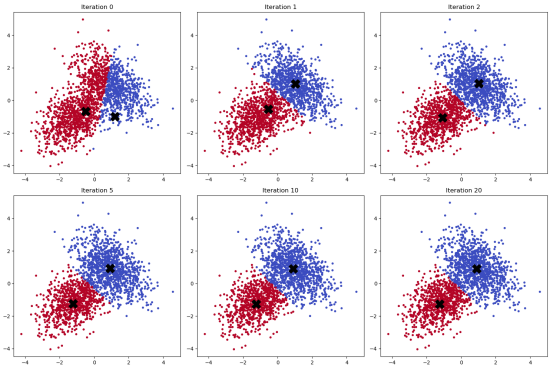


Figure 5: K-means clustering after 0, 1, 2, 5, 10, and 20 iterations.

Q3. Implement EM Algorithm for GMM (10 Points) Complete the implementation of the following functions in the `EMGMM` class.

Run the EM algorithm for 20 iterations and visualize the clustering progress at iterations 0, 1, 2, 5, 10, and 20. For each visualization:

See the Jupyter Notebook for the implementation. In Figure 6, the results of the EMGMM clustering for different numbers of iterations are displayed. We observe that the algorithm produces a soft clustering, where labels are assigned based on the probability of how certain we are. The results are quite similar to K-means, but with EMGMM, we can quantify the certainty of the clustering, especially for points in overlapping regions that are more shaded out.

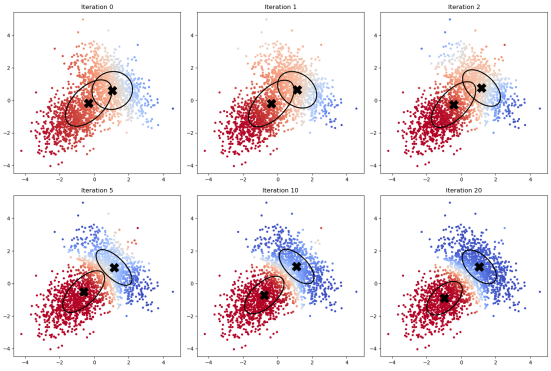


Figure 6: EMGMM clustering after 0, 1, 2, 5, 10, and 20 iterations.

Q4. Compare and Analyze (10 Points) Compute the Adjusted Rand Index (ARI) for the final cluster assignments from both k-means and EM. Which algorithm performs better? Why?

For this question, we calculated the Adjusted Rand Index (ARI) for the final cluster assignment after 20 iterations of both the K-means and EM algorithms. The results are shown in Figure 1. We can see that the EM algorithm performance is slightly better than K-means. This is because the data is generated as two Gaussian distributions with different means and covariances. K-means is more effective when the data has a spherical shape, as it clusters based on that assumption. However, our data has different variances and is not spherical. The EM algorithm for GMM performs better as it models the data as Gaussians with varying variances, which aligns better with the characteristics of our dataset. Additionally, since our two clusters have some overlap, the probabilistic approach of GMM is more suitable than K-means, which assigns hard labels.

Method	**Adjusted Rand Index (ARI)**
K-means	0.702
EM	0.771

Table 1: Comparison of clustering methods based on Adjusted Rand Index (ARI).

How does k-means handle overlapping clusters compared to EM?

As previously discussed, K-means does not handle overlapping clusters well because it assigns hard labels to data points, meaning each point can belong to only one cluster. This limitation can lead to the misclassification of points in regions where clusters overlap. In contrast, the EM algorithm for GMM uses a soft clustering approach, where each point is assigned a probability of belonging to each cluster. This method is more effective for overlapping clusters, as points can belong to multiple clusters, and the certainty of each cluster assignment can be quantified. The EM algorithm achieves this by modeling clusters as Gaussians and calculating posterior probabilities for each point.

How do the assumptions of Gaussian distributions in EM affect its performance?

This assumption can, of course, impact performance if the data does not follow a Gaussian distribution. If the data follows a different type of distribution, the EM algorithm may struggle to produce accurate clustering results.

Discuss the visual differences in cluster assignments at intermediate iterations for both algorithms.

The first clear visual difference in cluster assignments is that the K-means algorithm assigns hard labels. This means that at each iteration, a straight line divides the two clusters. For these two clusters, K-means had a relatively easy time finding the cluster centers.

In contrast, the EM-GMM assigns soft labels, and at the beginning of the algorithm, we observe significant overlap between the Gaussians, creating fuzzy regions where points belong to both

clusters. Additionally, it takes a bit more time for the EM-GMM to converge compared to K-means. However, the transition between the two clusters in the overlapping regions is smoother in the EM-GMM algorithm due to its probabilistic approach.

Q5. The Effect of Overestimating Clusters (5 Points) What happens when the number of clusters k is overestimated (e.g., $k > 2$)? Simulate and plot the results.

To simulate this, we ran the two algorithms with different numbers of clusters, using k between 2 and 10. The results are shown in Figure 7. We observe that the EM-GMM algorithm performs better than K-means; however, when the number of clusters is overestimated, the score drops significantly for both of the algorithms.

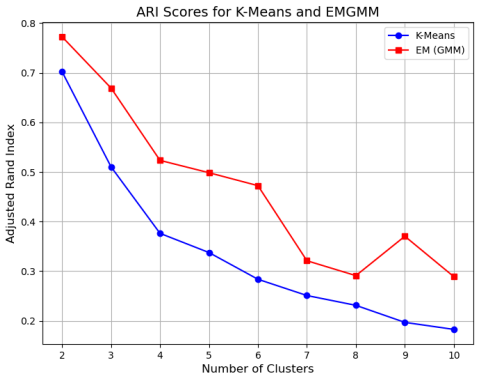


Figure 7: ARI score for K-means and EMGMM with different number of clusters.

Q6. Non-Circular Clusters (5 Points) Generate a synthetic dataset with two non-circular clusters using Scikit-learn's `make_moons` function.

The synthetic dataset is generated in the jupyter notebook. The two moons can be seen in figure 8.

Apply k-means and EM to this dataset. Visualize the clustering results for $k = 2$ and $k = 4$.

In Figure 9, we see the K-means algorithm applied to the synthetic moon dataset, assuming two clusters, and in Figure 10, assuming four clusters. The algorithm fails to capture the two distinct clusters.

In Figure 11, we see the EM-GMM algorithm applied to the synthetic moon dataset, assuming two clusters, and in Figure 12, assuming four clusters. The algorithm also fails to capture the two distinct clusters.

Discuss the performance of both algorithms on non-circular clusters. Which algorithm performs better, and why?

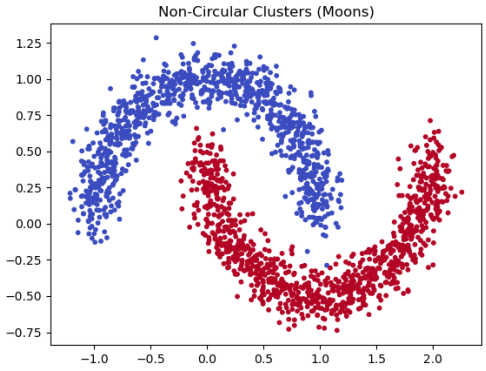


Figure 8: Plot of the synthetic dataset.

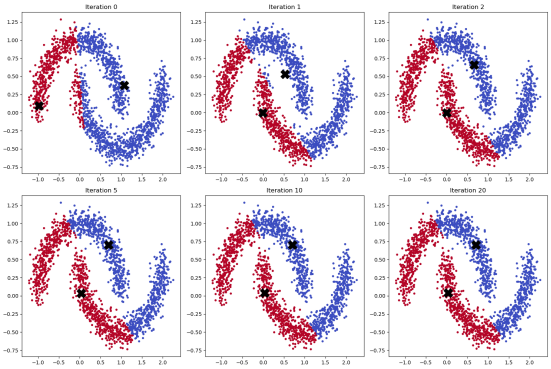


Figure 9: K-means with k=2 applied on the synthetic moon dataset.

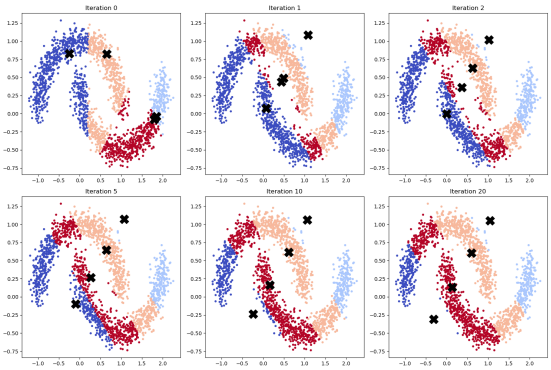


Figure 10: K-means with k=4 applied on the synthetic moon dataset.

As we observed in the previous question, neither algorithm successfully captured the two clusters in the synthetic dataset. With k=2, K-means split the two clusters, correctly classifying half of the points while misclassifying the other half. When k=4, the algorithm instead produced four

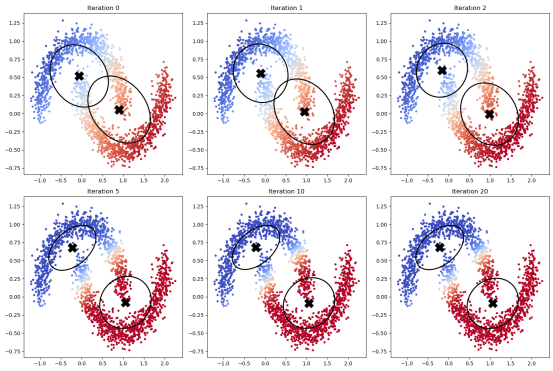


Figure 11: EM-GMM with k=2 applied on the synthetic moon dataset.

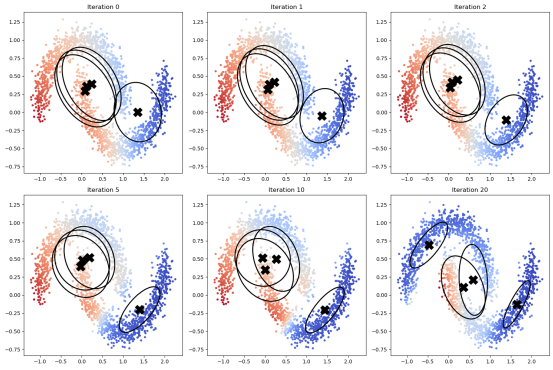


Figure 12: EM-GMM with k=4 applied on the synthetic moon dataset.

clusters. This highlights that K-means is not well-suited for this type of data, as it assumes clusters are spherical and assigns points based only on their Euclidean distance to the center. It does not adapt well to curved shapes, such as those in this dataset. EM-GMM performs slightly better than K-means, as it models clusters as ellipses, allowing it to capture more of the curved structure. It's still not an optimal choice for clustering this type of data but if the covariance matrix is more complex we can capture more of the clusters than K-means.

(Optional) Can you suggest a method to fix this?

There are several methods we could use to address this problem. One approach is to apply a kernel function to the data, transforming it into a space where it is better suited for K-means or EM-GMM clustering. Another option is to use a different type of clustering algorithm, such as spectral clustering, which uses an adjacency graph to define closeness within a cluster. This method is more effective at capturing the structure of non-spherical or curved clusters.

Figure 13: The original image

Figure 14: Clustering with different number of clusters

Exercise 3

Q1. Load and Visualize the Image

The original image is shown in figure 13

Q2. Apply k-Means Clustering for Image Segmentation (15 Points)

The result from the clustering is shown in figure 14

Q3. Analyze the Results

1. **Effect of Increasing k:**
   As the number of clusters k increases, the segmented image becomes more detailed, capturing finer color variations. However, this also leads to smaller, fragmented clusters. For instance, in the 8-cluster image in Figure 14, the background is split into multiple clusters, and the mango exhibits subtle variations in color that are treated as distinct clusters. Similarly, in the 4-cluster image, the mango and parts of the parrot are represented using a limited number of colors.

2. **Trade-offs Between Small and Large k:**

   - **Small k:** A smaller number of clusters simplifies the image, making it easier to distinguish broader regions such as the background versus the foreground. This can be useful when only high-level segmentation (e.g, object vs. background) is required. However, fine details and subtle variations are lost.
   - **Large k:** A larger k captures finer details and color variations, improving segmentation quality for complex images. However, this comes at the cost of increased computational load and may introduce unnecessary fragmentation of regions that should logically be grouped together (e.g, parts of the background in the *8-cluster image*).

   Conclusion: Choosing the optimal k depends on the task. For broad segmentation (e.g., background removal), a smaller k is sufficient. Conversely, for tasks requiring finer distinctions, a larger k is preferred.