Assignment 5

Computational Intelligence

Team Members		
Last name	First name	Matriculation Number
Merdes	Malte	01331649
Riedel	Stefan	01330219

1 Background

2 Classification / Clustering

2.1 Scenario 1: 2 dimensional feature

2.1.1 EM algorithm

1. First we fitted the given data set (fig. 1) with a Gaussian-Mixture-Model (GMM), applying the Expectation-Maximization (EM) algorithm for iterative parameter optimization. Depending on the initialization of the distribution parameters for our three Gaussian components, we observe different fitting performance, as seen in figures 2 and 3. We run the EM algorithm 100 times and return the overall maximum likelihood and the initial means of the best run (used to recreate the according plots).

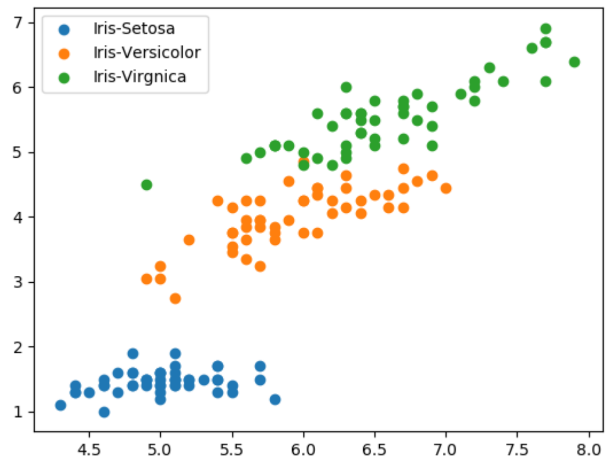

Figure 1: The 2D iris flower data set.

In case of the good fit, we inspected 4 misclassified data points, as seen in figure 8.
Class-dependent observation yields:
Class 0: 0 misclassifications
Class 1: 1 misclassifications
Class 2: 3 misclassifications

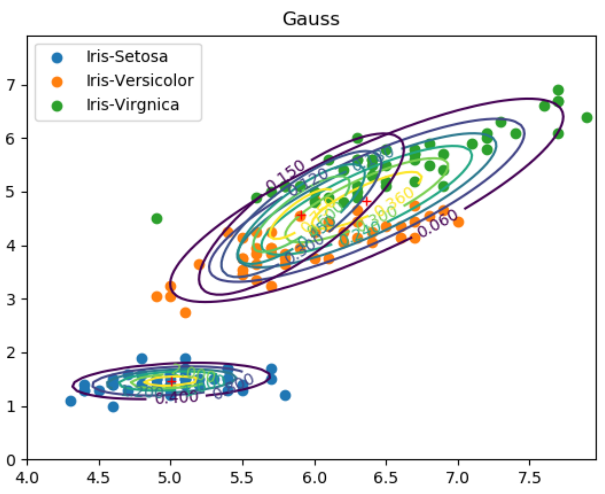

Figure 2: For bad initialization values of the mean parameters, we observe bad fitting after EM.

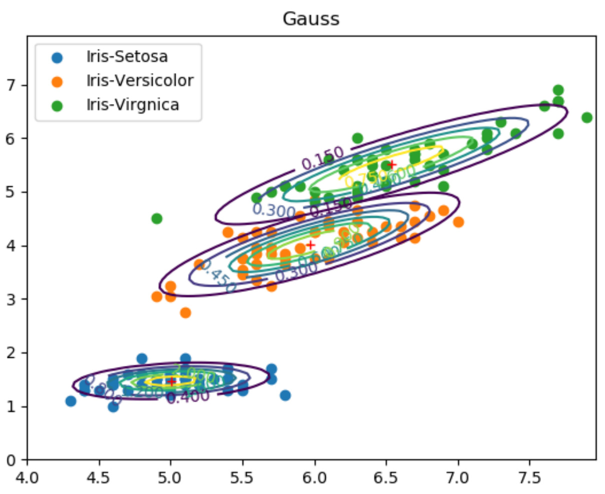

Figure 3: For good initialization values of the mean parameters, we observe good fitting after EM.

2. For lower number of components (K=2, seen in fig 4) we can observe under fitting, but under the circumstances quite a good classification (neglecting that 2 Gaussian component are insufficient for 3 classes): the nearby "Iris-Versicolor" and "Iris-Virgnica" data are grouped as one Gaussian and separated from the second distribution around "Iris-Setosa". Figure 5 with K=5 shows over fitting.

In term of the initialization of the Gaussian mixture model we used a uniformed random distribution for the 3 means and a identity matrices for the covariances and equal weights alphas (1/K).
Choosing the covariance matrices randomly can result in non invertable, singular matrices which causes problems in the computation of the algorithm. Also certain initial random means lead to ill-conditioned covariance matrices.

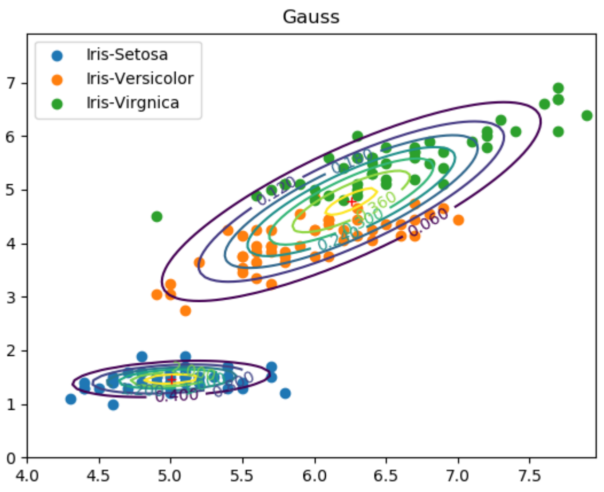

Figure 4: Classified data with 2 Gaussian components.

3. As seen in fig 6 the log-likelihood is maximized and converges. In case of a optimal convergence the log likelihood already reaches its maximum at around 10 iterations.
Figure 7 shows a non-optimal case, first reaching a saddle point and then converges later at around 20 iterations.

4. By finding the maximum value of the soft classification array $(r^n{}_k)$ for each data point we achieved the classification, as seen in fig 8. To produce comparable plots we reassigned the Gaussian components with the correct labels.

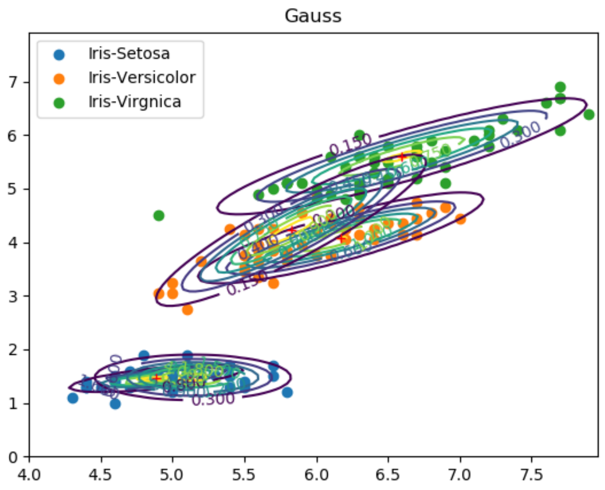

Figure 5: Classified data with 5 Gaussian components.

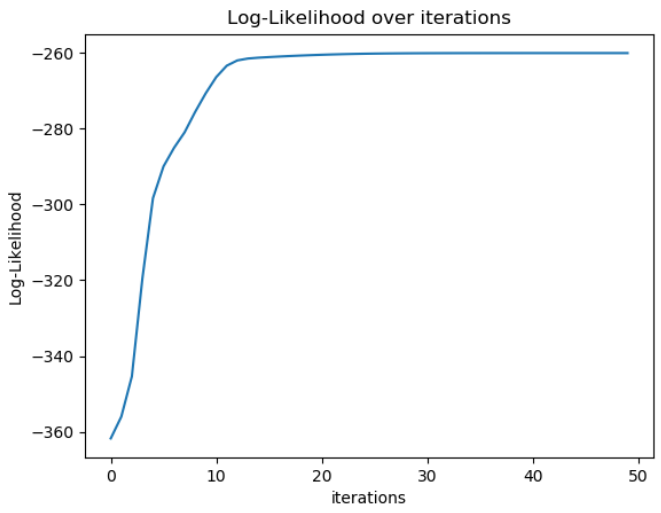

Figure 6: Log-likelihood over iterations.(optimal)

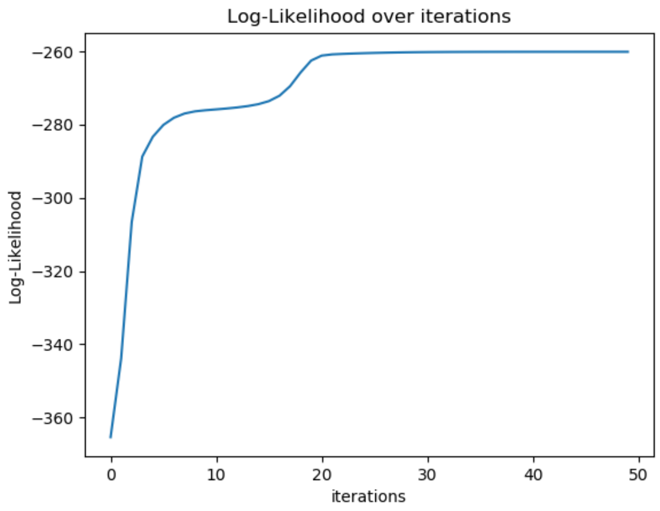

Figure 7: Log-likelihood over iterations.(non-optimal)

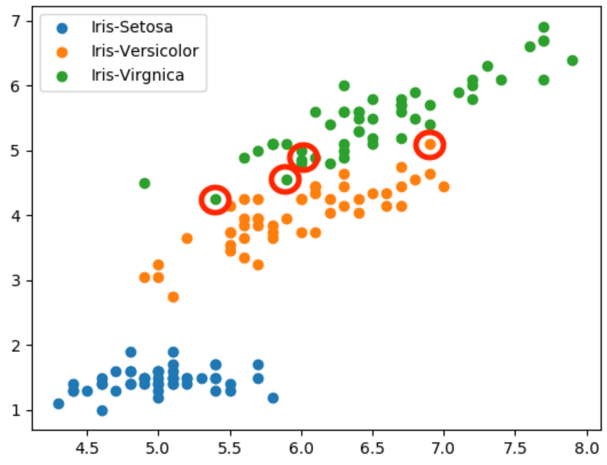

Figure 8: Classified data with 3 Gaussian components. (4 misclassified data points, marked in red)

2.1.2 K-means algorithm

1. In the following task we cluster the data with a k-means algorithm. We run the KM algorithm 100 times and return the overall minimum cumulative distance and the initial means of the best run (used to recreate the according plots). As seen in fig. 9 the k-means algorithm fits k=3 cluster centers quite accurate to the given data. (15 misclassified data points)

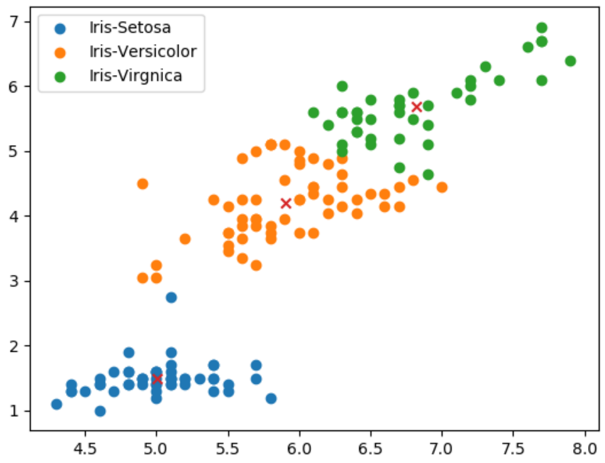

Figure 9: K-means with k=3 cluster centers and 50 iterations. (centers marked in red)

2. K-means with k=2 cluster centers results in Iris-Versicolor and Iris-Virginica to be clustered together (fig 10). Five cluster centers yields wrong clustering (fig 11).

3. The following plot shows the cumulative distance of the data points to their assigned means over 50 iterations (fig. 12).

4. In case of the k-means, we perform hard classification in the E-Step. The result is as seen in fig. 9.

5. In case of the 3-means, we inspected 15 misclassified data points, as seen in figure 9.
Class-dependent observation yields:
Class 0: 1 misclassifications
Class 1: 12 misclassifications
Class 2: 2 misclassifications

6. In general, k-means yields linear decision boundaries due to the euclidean distance measures of the algorithm.

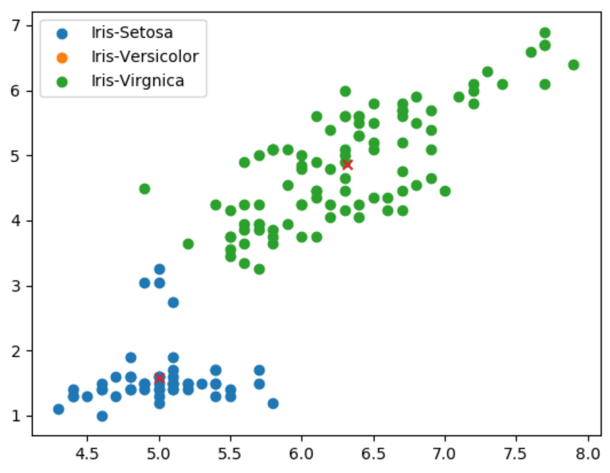

Figure 10: K-means with k=2 cluster centers and 50 iterations. (centers marked in red)

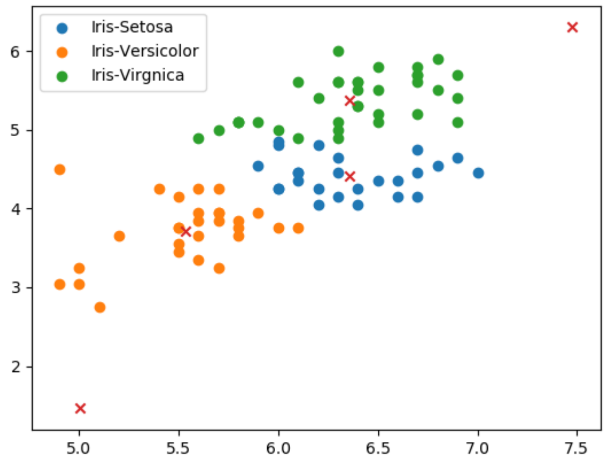

Figure 11: K-means with k=5 cluster centers and 50 iterations. (centers marked in red)

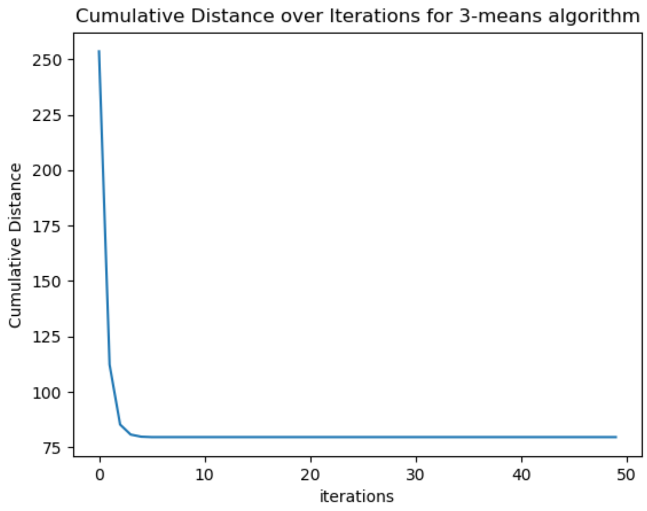

Figure 12: Cumulative distance over iterations.

2.1.3 Bonus task: Changing 2D-features

When selecting different features (sepal length and sepal width) two classes become harder to separate. Thus, we observe a higher rate of misclassification with the 3-means algorithm. (28 misclassified data points)

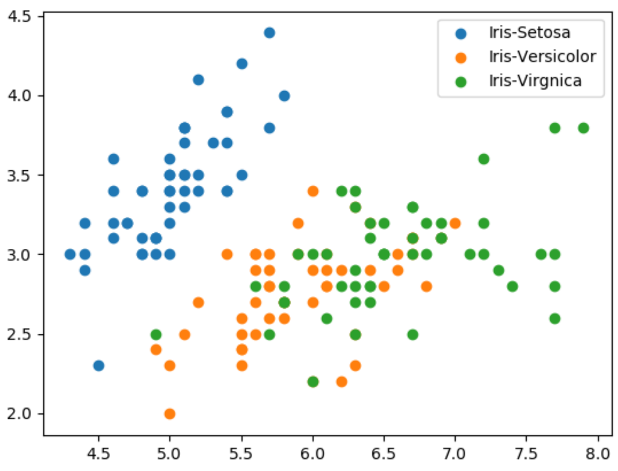

Figure 13: Sepal length and sepal width as the 2 features.

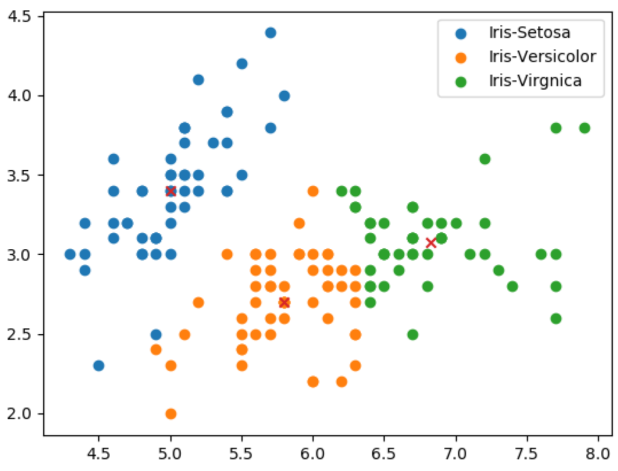

Figure 14: Classified data with Sepal length and sepal width as the 2 features (3-means algorithm).

However, if we select the petal length and width as our 2 features, we get easily separable clusters of data. (2 misclassified data points) (fig. 16)

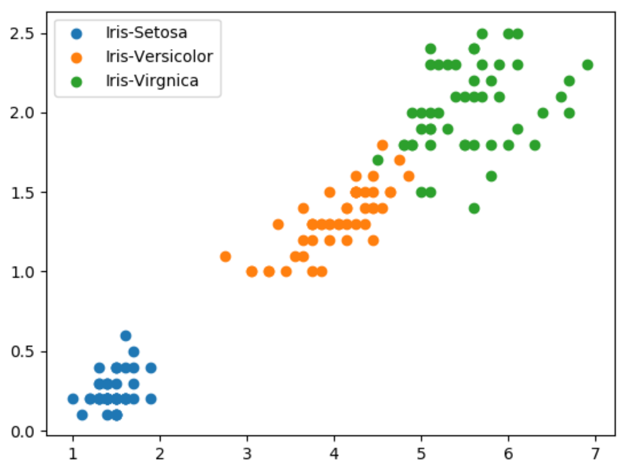

Figure 15: Classified data with Petal length and petal width as the 2 features (3-means clustering).

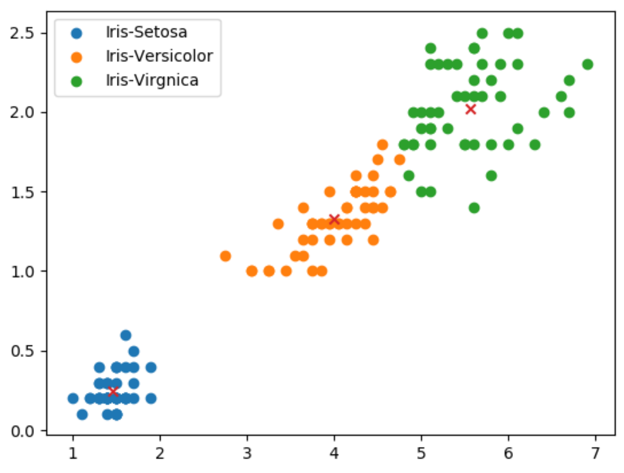

Figure 16: Classified data with Petal length and petal width as the 2 features (3-means clustering).

2.2 Scenario 2: 4 dimensional feature

2.2.1 EM algorithm

1. Now we fitted the full 4D data set (fig. 17) with a Gaussian-Mixture-Model (GMM), first applying the Expectation-Maximization (EM) algorithm for iterative parameter optimization.

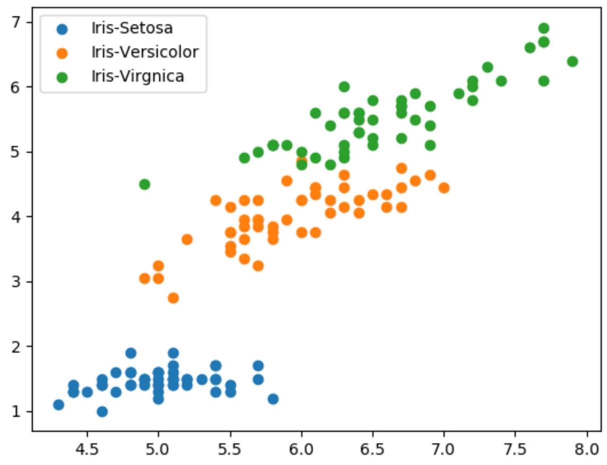

Figure 17: Plot of 2 components of the 4D iris flower data set.

As the best fit, we inspected 4 misclassified data points, as seen in figure 22. This is the exact **same performance as in the 2D case**. However, we observe that now it is always the class 2 that is misclassified. In general, we evaluated the smallest possible number of misclassifications by running the EM algorithm 100 times.
Class-dependent observation yields:
Class 0: 0 misclassifications
Class 1: 0 misclassifications
Class 2: 4 misclassifications

2. In case of 4D data, for lower number of components (K=2, seen in fig 19) we observe that all points become classified as ”Iris Setosa” which means 100 wrong classifications. For K=5 components (fig. 20 we find the same classificatons. In general, the 2D plots can't capture the 4D classifications and components in a clear and descriptive way.

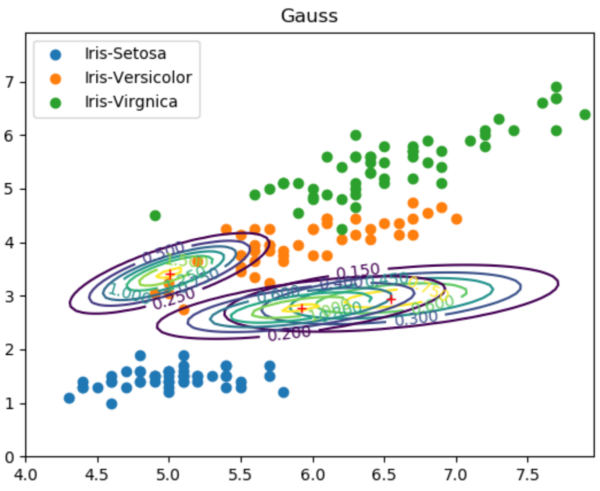

Figure 18: 2D visualization of the 4D fitted Gauss components.

In term of the initialization we expanded our mean and covariance matrices and choose suitable initial values for the 2 added dimensions.

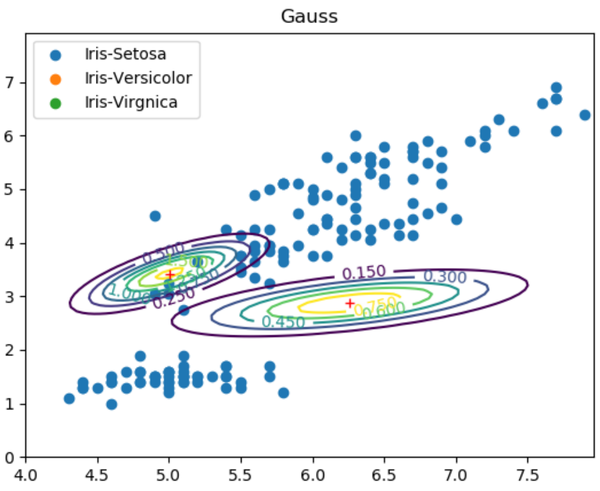

Figure 19: Classified 4D data with 2 Gaussian components.

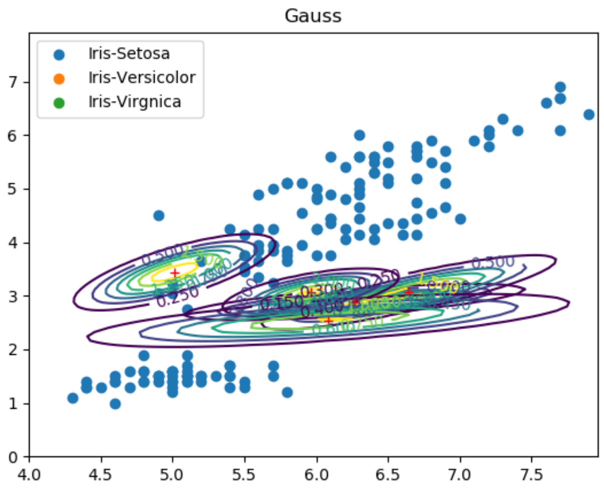

Figure 20: Classified 4D data with 5 Gaussian components.

3. As seen in fig 21 the log-likelihood is maximized and converges. **For 4D data it reaches a higher value then in the 2D case**.

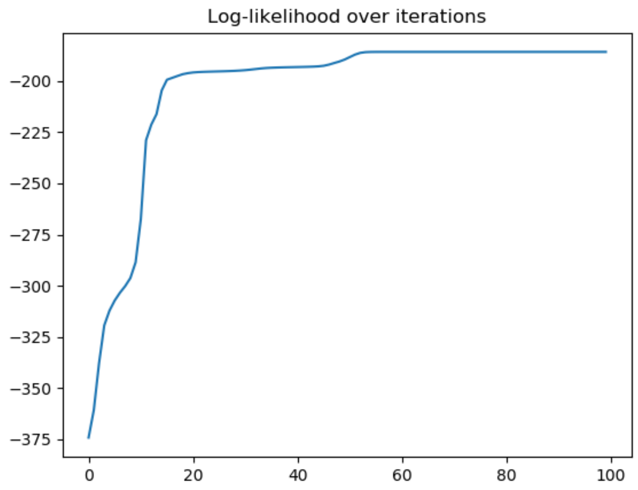

Figure 21: Log-likelihood over iterations for 4D data, 3 components.

4. By finding the maximum value of the soft classification array $(r^n{}_k)$ for each data point we achieved the classification, as seen in fig 22. To produce comparable plots we reassigned the Gaussian components with the correct labels.

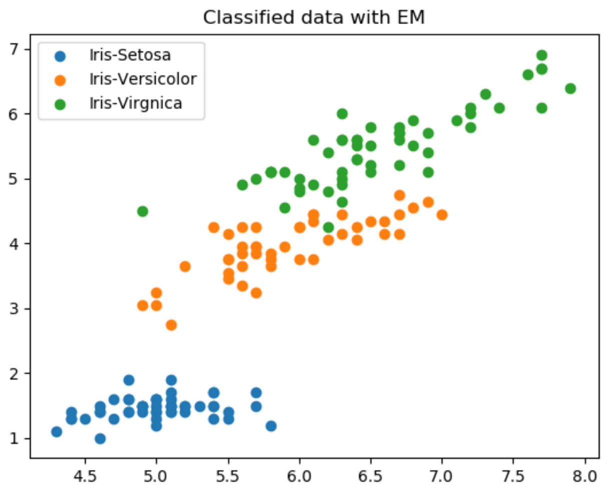

Figure 22: 2D plot of classified 4D data with 3 Gaussian components. (4 misclassified data points)

2.2.2 K-means algorithm

2.2.3 Confining the structure of the covariance matrices to diagonal matrices

We confined the covariance matrices to diagonal matrices by setting the off-diagonal elements to zero within the EM-iterations. Thus, we end up with the variances in the main coordinate axes. Graphically, this means gauss components without "rotation" as seen in fig. 23. We expected and found that the performance decreases slightly, as the model can fit the means quite well, but can't adjust the data perfectly due to the lacking rotation in the gauss components. (6 misclassified points compared to 4 classified points with full covariance matrices).

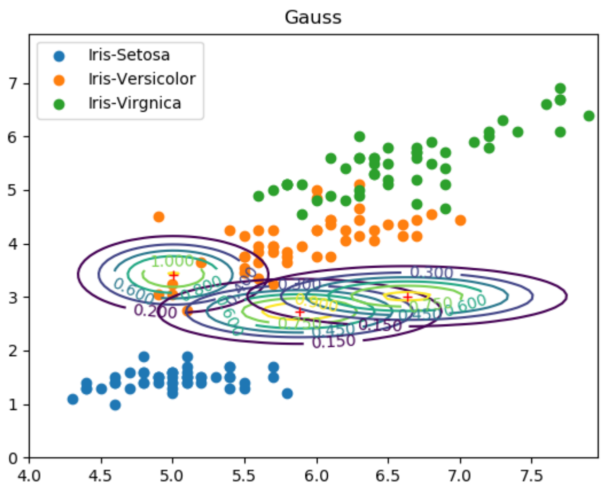

Figure 23: 2D plot of fitted Gauss components (confined covariance matrices).

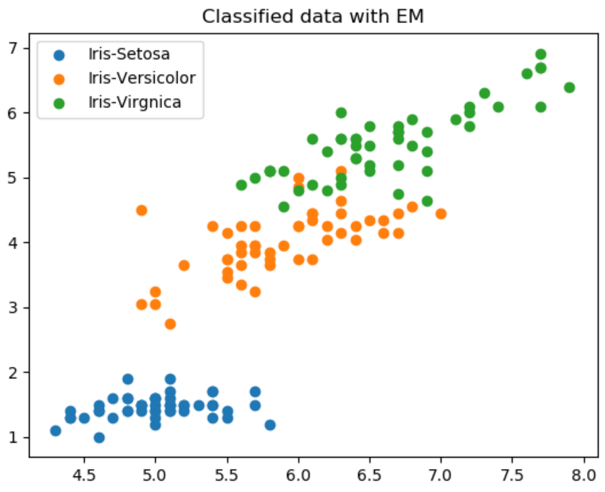

Figure 24: 2D plot of classified 4D data with diagonally forced covariance matrices of the 3 Gaussian components. (6 misclassified data points)

2.3 Scenario 3: Processing the data with PCA

2.3.1 Variance explained with PCA

Projecting the 4D data points onto the 2 principal components by means of the PCA algorithm, we can explain up to 97,7% of the total variance. This is computed by dividing the sum of the two largest eigenvalues of the covariance matrix by the total sum of all eigenvalues.

2.3.2 Performance comparison with unprocessed 2D and 4D data

Figure 25 shows the transformed data after the principal component analysis.

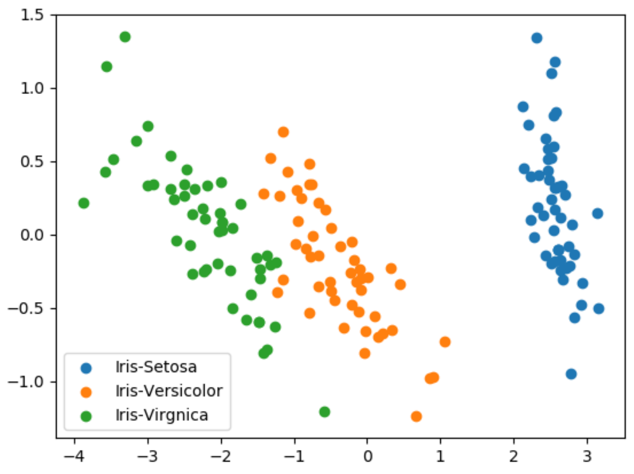

Figure 25: Transforming the 4D data into 2D data by PCA.

EM: For the EM algorithm we find a slight improvement of the performance. We now get 3 misclassified data points, in comparison to 4 misclassified points for both the unprocessed 2D and 4D data scenarios (fig 26). The maximum likelihood is -285 which is actually smaller then in the other scenarios. It is plotted over the iterations in fig. 27.

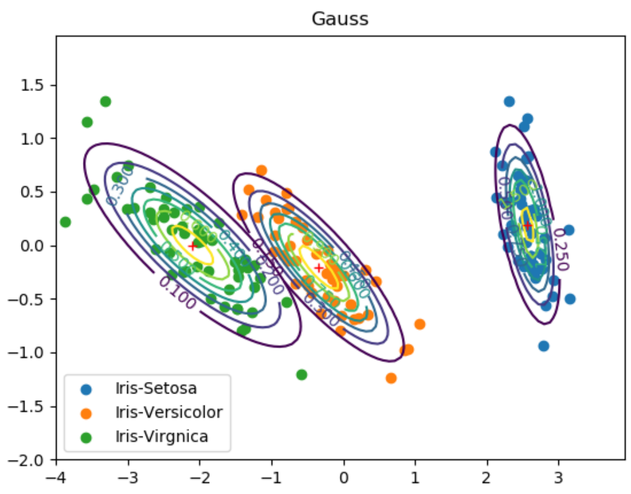

Figure 26: Classified data after PCA (EM, 3 gauss components).

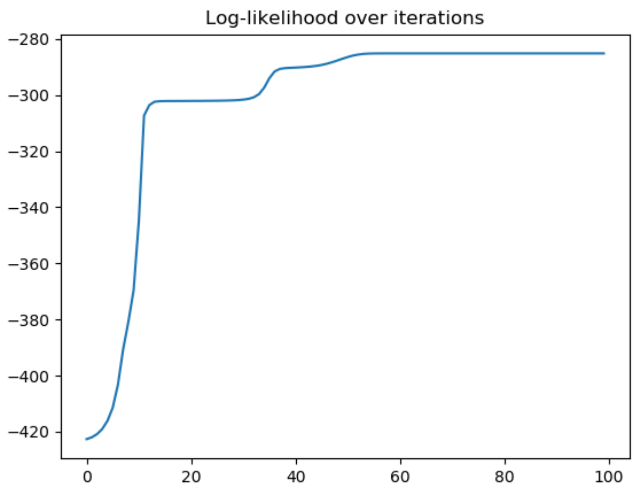

Figure 27: Log-Likelihood over iterations with PCA.

k-means: For the k-means algorithm we can achieve the same performance as in the full 4D data case (as in scenario 2). This proves that the information was well conserved in the 2D principal component data. The cumulative distance is plotted over the iterations in fig 29.

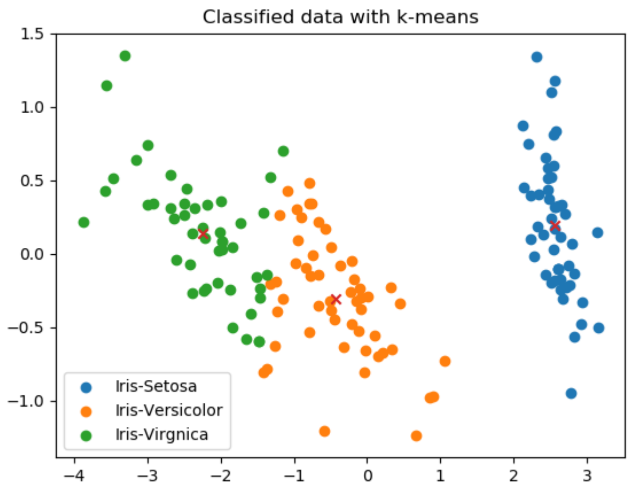

Figure 28: Classified data after PCA (3-means).

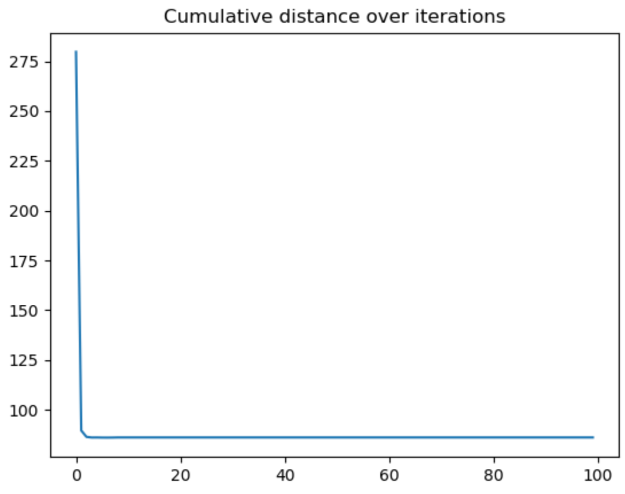

Figure 29: Cumulative distance over iterations with PCA.

2.3.3 Bonus task: PCA with whitening

The principal component analysis can be extended to achieve complete whitening of the dimension-reduced data by dividing the transformed (rotated) data points by the roots of the eigenvalues. In addition to the decorrelation of the data (diagonal covariance matrix) we now achieve unit variances, which means that the covariance matrix of our data is now the identity matrix.
Numerically, this is observed as neglectable off-diagonal entries in Python. We get the following PCA whitened covariance matrix of the data:

```
[[ 1.00000000e+00 -7.48841705e-17]
[-7.48841705e-17 1.00000000e+00]]
```

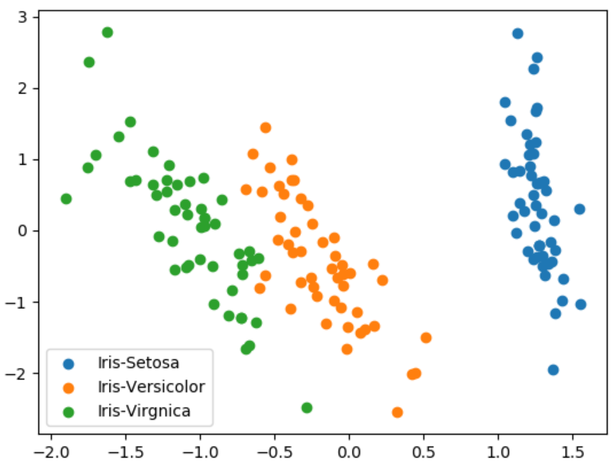

Figure 30: Whitened data with identity covariance matrix.

In terms of the initialization, we now have to choose mean values that are centered around the coordinate origin in order to enable convergence of the algorithm. For a well-fitted, convergent case we observe 3 misclassified data points (fig. 31).

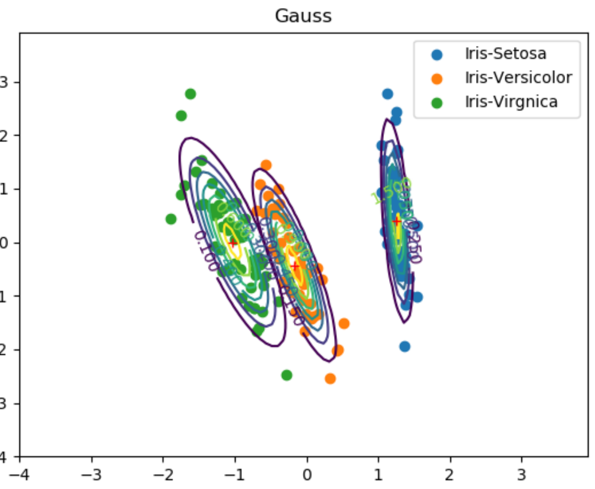

Figure 31: Classified data with PCA whitening as pre-proccesing.

3 Samples from a Gaussian Mixture Model

3.1 Function that draws N samples from GMM

We implemented the function by evaluating the *multivariate_normal*-function in python. To achieve the weighting of the components we use the *choice*-function.

3.2 Plot of GMM Samples

In figure 32 we plotted the drawn samples of 5 component gaussian mixture model with equally weighted components. The covariance matrices were set as unit matrices, the means distributed by a random function that was scaled to make the single components visible.

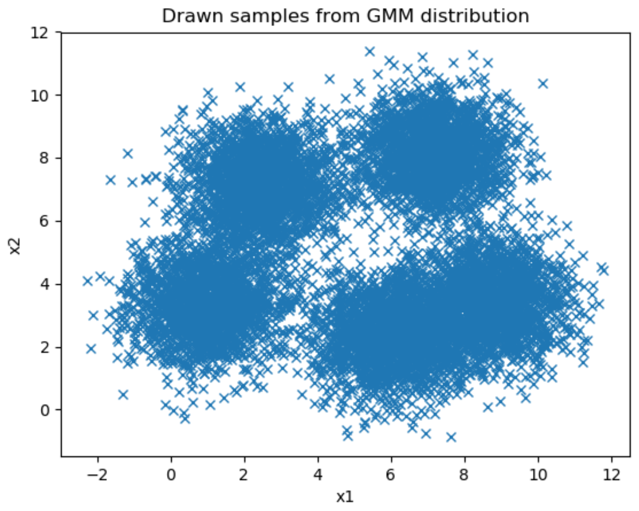

Figure 32: Samples from a GMM with K=5 components, equal weights.

Figure 33 shows samples drawn from an unequally weighted gaussian mixture model.

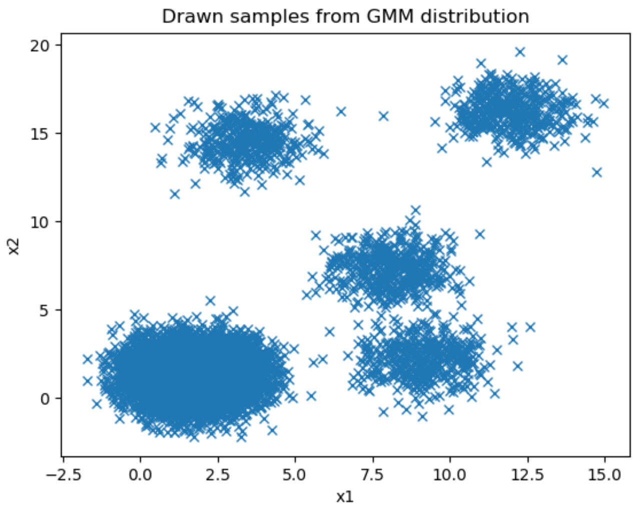

Figure 33: Samples from a GMM with K=5 components, unequal weights.