# Machine Learning I – Exercise Sheet 1

Jan Zwank

March 19, 2021

## 1 Maximum Likelihood

We generate 50 samples of two dimensional data following a Gaussian distribution by applying `multivariate_normal` from the `numpy.random` library. This function takes the mean, covariance matrix and number of samples as input and returns a `np.ndarray` with the desired properties.

There data generated in this process, using

$$\mu = \begin{pmatrix} 2 \\ -2 \end{pmatrix} \qquad c = \begin{pmatrix} 0.9 & 0.2 \\ 0.2 & 0.3 \end{pmatrix}$$

is depicted in Figure 1.

The maximum likelihood estimator for the mean is given by

$$\mu_{ML} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

where $\{x_i\}_{i=0}^{N}$ are samples following a Gaussian distribution, considered as column vectors.

We can compute the maximum likelihood estimator the the covariance matrix via

$$c_{ML} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_{ML}) \cdot (x_i - \mu_{ML})^T$$

This yields

$$\mu_{ML} = \begin{pmatrix} 1.94630497 \\ -2.24439076 \end{pmatrix} \qquad c_{ML} = \begin{pmatrix} 1.03057726 & 0.3168775 \\ 0.3168775 & 0.35350999 \end{pmatrix}$$

This corresponds to errors of

$$e_\mu = \begin{pmatrix} 0.05369503 \\ 0.24439076 \end{pmatrix} \qquad \|e_\mu\| = 0.22$$

$$e_c = \begin{pmatrix} -0.13057726 & -0.1168775 \\ -0.1168775 & -0.05350999 \end{pmatrix} \qquad |||e_c||| = 0.11$$

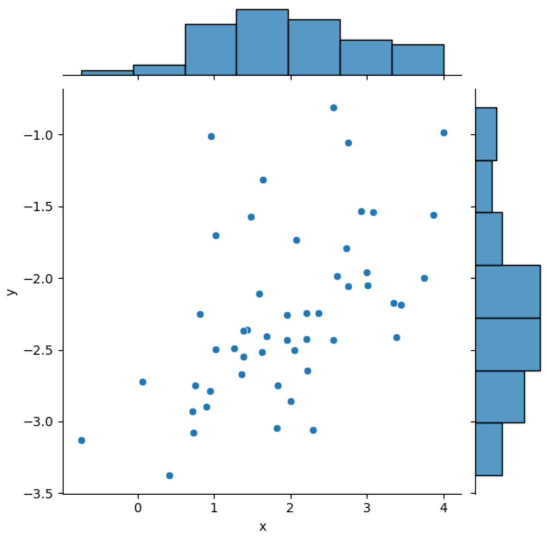

Fig. 1: 50 Samples following Gaussian distribution

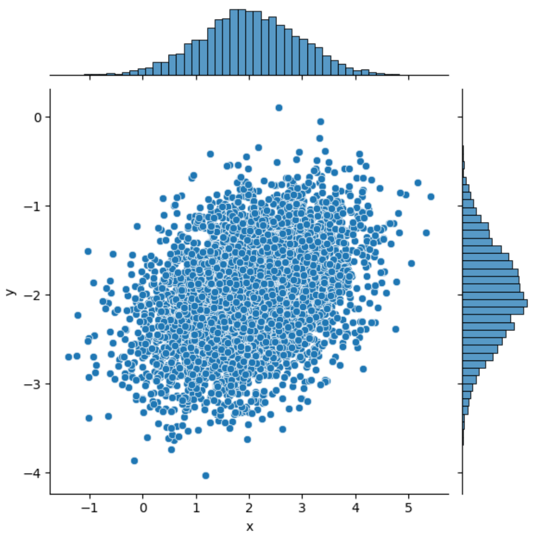

Fig. 2: 500 Samples following Gaussian distribution

Using the same method with 500 samples yields

$$\mu_{ML} = \begin{pmatrix} 2.00219803 \\ -2.00860215 \end{pmatrix}$$

$$c_{ML} = \begin{pmatrix} 0.8910831 & 0.19733638 \\ 0.19733638 & 0.30275286 \end{pmatrix}$$

corresponding to the following errors:

$$e_\mu = \begin{pmatrix} -0.00219803 \\ 0.00860215 \end{pmatrix} \qquad \|e_\mu\| = 0.020$$

$$e_c = \begin{pmatrix} -0.13057726 & -0.1168775 \\ -0.1168775 & -0.05350999 \end{pmatrix} \qquad |||e_c||| = 0.016$$

It is clear to see that the accuracy of the maximum likelihood estimator increases if more data is provided. This is due to the fact that Maximum likelihood estimators are asymptotically unbiased, meaning

$$\lim_{N\to\infty} E[\Theta_{ML,N}] = \Theta$$

For the second part, two Datasets $D_1$ and $D_2$ were constructed to follow the distribution of a Gaussian mixture composed of three components, each of which had the same weight. This was achieved by adding one sample at a time. First the component of the new sample was randomly chosen (uniform distribution) and then the analogous method to the first part was used to create a single sample of the corresponding Gaussian distribution. The means, which were used here were the following

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \mu_2 = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix} \qquad \mu_1 = \begin{pmatrix} 3 \\ 3 \\ 4 \end{pmatrix}$$

and using a common covariance matrix

$$c = 0.8 \cdot I_3$$

For each of the classes, we use the same methods as before to find maximum likelihood estimates for the corresponding mean and covariance. Since it is known that the components share a common covariance matrix, the average of the estimated covariance matrices was computed. This yields:

$$\mu_{1,ML} = \begin{pmatrix} -0.01408578 \\ -0.0366634 \\ -0.05517594 \end{pmatrix} \quad \mu_{2,ML} = \begin{pmatrix} 1.06444317 \\ 2.04763129 \\ 1.9935278 \end{pmatrix} \quad \mu_{3,ML} = \begin{pmatrix} 2.98143391 \\ 3.03341008 \\ 4.00861147 \end{pmatrix}$$

and

$$c_{ML} = \begin{pmatrix} 0.82923873 & -0.01134525 & 0.00393332 \\ -0.01134525 & 0.77432891 & -0.03037824 \\ 0.00393332 & -0.03037824 & 0.82263878 \end{pmatrix}$$

To classify the points in $D_2$ we compute the distance to the estimated means and assign the sample to the class corresponding to the smallest such distance.

This classification assigns 84.5% of the test data to the correct class.

The same procedure is applied with a changed covariance matrix of

$$c = \begin{pmatrix} 0.8 & 0.2 & 0.1 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.2 & 0.8 \end{pmatrix}$$

The estimated means are

$$\mu_{1,ML} = \begin{pmatrix} -0.08198677 \\ -0.00574208 \\ 0.02214036 \end{pmatrix} \quad \mu_{2,ML} = \begin{pmatrix} 0.952525 \\ 1.94216761 \\ 1.98512829 \end{pmatrix} \quad \mu_{3,ML} = \begin{pmatrix} 3.01247125 \\ 2.96911757 \\ 3.92329761 \end{pmatrix}$$

and the estimated covariance

$$c_{ML} = \begin{pmatrix} 0.7987 & 0.21611717 & 0.13492815 \\ 0.21611717 & 0.82112518 & 0.22281418 \\ 0.13492815 & 0.22281418 & 0.80311355 \end{pmatrix}$$

The accuracy is now 80.9%. This decrease in accuracy is due to the fact, that since the covariance matrix is no longer a multiple of the identity, the euclidean distance between the samples and the estimated means is a worse estimator.

## 2 Expectation Maximization

We begin by generating 600 samples of a Gaussian mixture with means

$$\mu_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \mu_2 = \begin{pmatrix} 3 \\ 3 \end{pmatrix} \mu_3 = \begin{pmatrix} 2 \\ 6 \end{pmatrix}$$

with covariances $c_1 = 0.1 \cdot I, c_2 = 0.2 \cdot I, c_3 = 0.3 \cdot I$ and weights $P_1 = 0.4, P_2 = 0.4$ and $P_3 = 0.2$ in essentially the same way as in Exercise 1.

To estimate the parameters of this Gaussian mixture we use `GuassianMixture` from the scikit-learn library [1], which provides a expectation maximization estimator for Gaussian mixtures.

We use the following parameters to set the initial parameters for this model:

```
parameters_dict={
0:{"n_components":3,
"covariance_type":"spherical"},
1:{"n_components":3,
"covariance_type":"spherical",
"means_init":[[0,2],[5,2],[5,5]],
"precisions_init":[1/0.15,1/0.27,1/0.4],
"weights_init":[1/3]*3},
2:{"n_components":2,
"covariance_type":"spherical",
"means_init":[[1.6,1.4],[1.4,1.6]],
"precisions_init":[1/0.2,1/0.4],
"weights_init":[0.5]*2}
}
```

Recall that precision is the inverse of covariance.

This results in the following classifications:

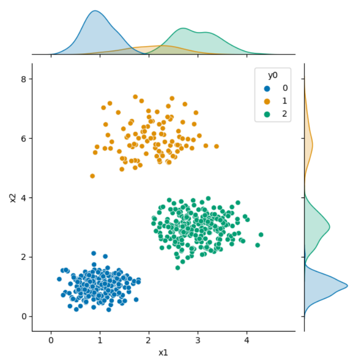

Fig. 3: using set of parameters 0

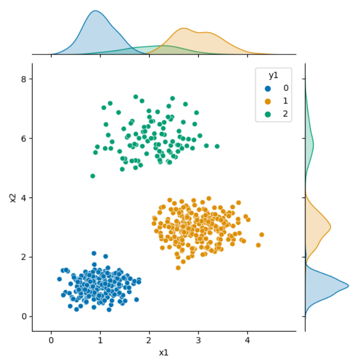

Fig. 4: using set of parameters 1

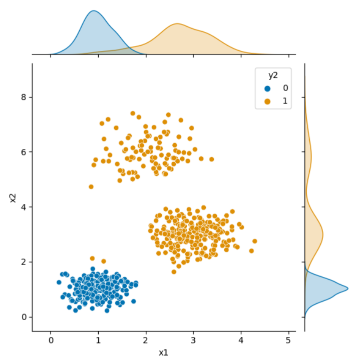

Fig. 5: using set of parameters 2

It becomes clear from these images that the number of components is a very important parameter when classifying Gaussian mixtures. In Figure 5 the class with label 1 clearly consists of two smaller clusters and the resulting density is quite different from that of a regular Gaussian distribution.

## 3 Clustering

We create a dataset $D$ consisting of 1000 2-dimensional points, consisting of 4 equally sized groups. Each group contains vectors that stem from Gaussain distributions with means:

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad \mu_2 = \begin{pmatrix} 10 \\ 0 \end{pmatrix} \qquad \mu_3 = \begin{pmatrix} 0 \\ 6 \end{pmatrix} \qquad \mu_4 = \begin{pmatrix} 9 \\ 8 \end{pmatrix}$$

respectively and covariance matrices:

$$c_1 = I \qquad c_2 = \begin{pmatrix} 1 & 0.2 \\ 0.2 & 1.5 \end{pmatrix} \qquad c_3 = \begin{pmatrix} 1 & 0.4 \\ 0.4 & 1.1 \end{pmatrix} \qquad c_4 = \begin{pmatrix} 0.3 & 0.2 \\ 0.2 & 0.5 \end{pmatrix}.$$

This is done by essentially the same process as in the previous two Exercises.

We apply the `Kmeans` algorithm from the scikit-learn library in the following cases:

1. We set $C = 4$ and randomly initialize the parameters. This yields the following estimated means:

$$\mu_{1,KM} = \begin{pmatrix} 9.996861 \\ 0.037494 \end{pmatrix} \qquad \mu_{2,KM} = \begin{pmatrix} 0.056832 \\ 6.054396 \end{pmatrix}$$
$$\mu_{3,KM} = \begin{pmatrix} 9.030599 \\ 8.002927 \end{pmatrix} \qquad \mu_{4,KM} = \begin{pmatrix} 0.000978 \\ 0.032436 \end{pmatrix}.$$

   Up to relabeling this matches well with the original means of the Gaussian distributions. This is also depicted in Figure 6.

2. If we repeat the same process with $C = 3$, we get the following means:

$$\mu_{1,KM} = \begin{pmatrix} 9.996861 \\ 0.037494 \end{pmatrix} \qquad \mu_{2,KM} = \begin{pmatrix} 0.028905 \\ 3.043416 \end{pmatrix} \qquad \mu_{3,KM} = \begin{pmatrix} 9.030599 \\ 8.002927 \end{pmatrix}.$$

   Here, $\mu_{1,KM}$ matches well with the original $\mu_2$ and $\mu_{3,KM}$ matches well with the original $\mu_4$, however $\mu_{2,KM}$ is close to the average of $\mu_1$ and $\mu_3$. This is the case because the reduced number of clusters, the original classes 1 and 3 are combined to one larger cluster. The situation is depicted in Figure 7.

3. We repeat the process with $C = 5$. The resulting means are

$$\mu_{1,KM} = \begin{pmatrix} 9.996861 \\ 0.037494 \end{pmatrix} \qquad \mu_{2,KM} = \begin{pmatrix} 0.063343 \\ 6.068601 \end{pmatrix} \qquad \mu_{3,KM} = \begin{pmatrix} 9.030599 \\ 8.002927 \end{pmatrix}$$
$$\mu_{4,KM} = \begin{pmatrix} 0.596993 \\ 0.504864 \end{pmatrix} \qquad \mu_{5,KM} = \begin{pmatrix} -0.565815 \\ -0.388171 \end{pmatrix}.$$

Here, $\mu_{1,KM}$ matches well with $\mu_2$, $\mu_{2,KM}$ matches well with $\mu_3$, $\mu_{3,KM}$ matches well with $\mu_4$, $\mu_{4,KM}$ and $\mu_{5,KM}$ avererage to approximately $\mu_1$. This is due to class 1 being split in multiple clusters in order to realize 5 clusters. The situation is depicted in Figure 8

4. We now set $C = 4$ again, but now manually set the initial cluster representatives to

$$\Theta_1 = \begin{pmatrix}-2\\-2\end{pmatrix} \qquad \Theta_2 = \begin{pmatrix}-2.1\\-2.1\end{pmatrix} \qquad \Theta_3 \qquad = \begin{pmatrix}-2\\-2.2\end{pmatrix} \qquad \Theta_4 = \begin{pmatrix}-2.1\\-2.2\end{pmatrix}$$

   This yields the following means:

$$\mu_{1,KM} = \begin{pmatrix}0.028905\\3.043416\end{pmatrix} \qquad \mu_{2,KM} = \begin{pmatrix}9.441874\\8.417418\end{pmatrix}$$
$$\mu_{3,KM} = \begin{pmatrix}9.996861\\0.037494\end{pmatrix} \qquad \mu_{4,KM} = \begin{pmatrix}8.632277\\7.601492\end{pmatrix}.$$

   This not only erroneously classifies classes 1 and 3 as one big cluster, but also simultaneously splits class 4 into two. This is a result of the initial cluster representatives all being very close together. The situation is depicted in Figure 9.

5. Lastly we change $Theta_4$ to $(30, 30)^T$. This results in the following means:

$$\mu_{1,KM} = \begin{pmatrix}0.028905\\3.043416\end{pmatrix} \qquad \mu_{2,KM} = \begin{pmatrix}9.441874\\8.417418\end{pmatrix}$$
$$\mu_{3,KM} = \begin{pmatrix}9.996861\\0.037494\end{pmatrix} \qquad \mu_{4,KM} = \begin{pmatrix}8.632277\\7.601492\end{pmatrix}.$$

   which is the exact same result as in part 4. Adding a single outlier to the cluster representatives doesn't improve the classification. The situation is depicted in Figure 10

## References

[1] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

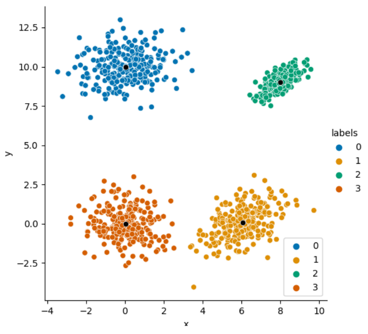

Fig. 6: C=4, randomly initialized

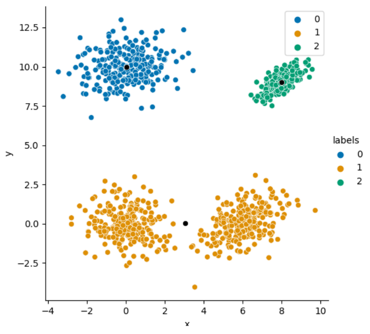

Fig. 7: C=3, randomly initialized

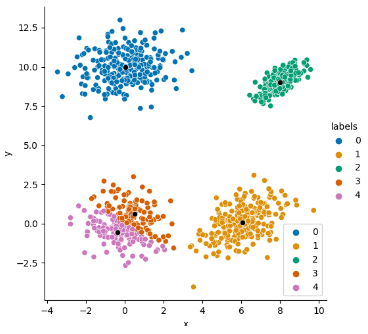

Fig. 8: C=5, randomly initialized

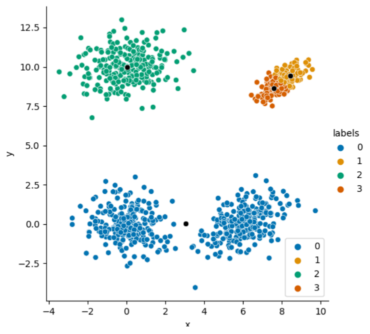

Fig. 9: C=4, manually initialized

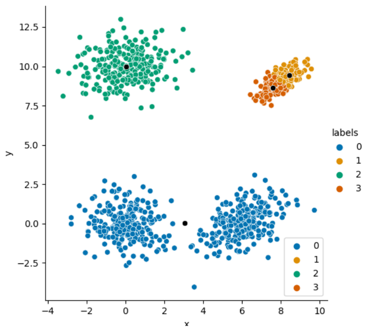

Fig. 10: C=4, manually initialized with outlier