Lucas Elbert
lucas.elbert@polytechnique.edu

1 EM and KMeans

1.1 Derivation and explanation

We consider a gaussian mixture model with the notation as given in the exercise statement with parameters $\Theta = \{p_k, \mu_k, D_k | k = 1...K\}$. I'll write random variables in capital letters and our datapoints in lowercase, however I'll use $p(x_i) = p(X_i = x_i)$ for the ease of notation. The corresponding EM-algorithm to find approximate the parameters can be written as:

1. Initial guess for $Theta_0$
2. Repeat until convergence:

   E-step: $q_i(k) := p_{\Theta_t}(Z_i = k|x_i)$

   M-step: $\Theta_{t+1} := \underset{\Theta}{\text{argmax}}\ F(\Theta, q)$ where

$$F(\Theta, q) = \sum_{1 \le i \le n} \sum_{1 \le j \le K} q_i(j) \log(p_\Theta(x_i, Z_i = j))$$

To derive formulas for Θ_{t+1} we search for critical points of F.

w.r.t. p_k

We add the constraint $\sum_{j=1}^K p_j = 1$, build the Lagrangian $L(\Theta, \lambda) = F(\Theta) + \lambda(1 - \sum_{j=1}^K p_j)$ and derive

$$\begin{aligned}\frac{\partial L}{\partial p_k} &= \frac{\partial F(\Theta)}{\partial p_k} - \lambda \\ &= \sum_i \sum_j q_i(j) \frac{1}{p_\Theta(x_i|Z_i = j)p_j} \frac{\partial p_\Theta(x_i|Z_i = j)p_j}{\partial p_k} - \lambda \\ &= \sum_i q_i(k) \frac{1}{p_k} - \lambda\end{aligned}$$

And therefore setting this term to zero yields

$$p_k \propto \sum_i q_i(k)$$

This solution is valid as by definition all $q_i(k)$s are non negative and the scaling makes it a probability distribution.

w.r.t. μ_k

$$\begin{aligned}\frac{\partial F(\Theta)}{\partial \mu_k} &= \sum_i \sum_j q_i(j) \frac{\partial \log(p_\Theta(x_i, Z_i = j))}{\partial \mu_k} \\ &= \sum_i q_i(k) \frac{\partial \log \mathcal{N}(x_i; \mu_k, D_k)p_k)}{\partial \mu_k}\end{aligned}$$

a short side calculation shows $\frac{\partial \log \mathcal{N}(x;\mu,D)}{\partial \mu} = (x - \mu)^\top D^{-1}$ and so we continue

$$= \sum_i q_i(k)(x_i - \mu_k)^\top D_k^{-1}$$

Setting this term to zero yields in

$$\mu_k = \frac{\sum_i q_i(k) x_i}{\sum_i q_i(k)}$$

w.r.t. D_k^{-1}
As the covariance matrices are diagonal we should theoretically only set the derivatives of these diagonal elements to zero. However as the notation is a little easier we'll first derive with respect to the whole matrix and see in the end how to deal with the fact that D has to be diagonal.

$$\begin{aligned}\frac{\partial F(\Theta)}{\partial D_k^{-1}} &= \sum_i \sum_j q_i(j) \frac{\partial \log(p_\Theta(x_i, Z_i = j))}{\partial D_k^{-1}} \\ &= \sum_i q_i(k) \frac{\partial \log(\mathcal{N}(x_i; \mu_k, D_k))}{\partial D_k^{-1}}\end{aligned}$$

Another side calculation shows $\frac{\partial \log(\mathcal{N}(x_i;\mu_k,D_k))}{\partial D_k^{-1}} = \frac{1}{2}(D - (x-\mu)(x-\mu)^\top)$ so we get

$$= \sum_i q_i(k) \frac{1}{2}(D_k - (x_i - \mu_k)(x_i - \mu_k)^\top)$$

The diagonal elements of that matrix give us $\frac{\partial F(\Theta)}{\partial \sigma_{kl}^{-1}}$ (σ_{kl} are the diagonal entries of D_k). Setting the derivative w.r.t σ_{kl}^{-1} to zero is equivalent to setting it to zero w.r.t. σ_{kl} as we can assume $\sigma_{kl} \neq 0$. Now setting the whole term to zero leads to

$$D_k = \frac{\sum_i q_i(k)(x_i - \mu_k)(x_i - \mu_k)^\top}{\sum_i q_i(k)}$$

To keep our covariance matrix diagonal only the diagonal elements need to be updated to this D_k. This can equivalently but more efficiently also be calculated by only requiring the derivatives of the diagonal elements to be zero which leads us to

$$\sigma_{kl} = \frac{\sum_i q_i(k)(x_{i_l} - \mu_{k_l})^2}{\sum_i q_i(k)}$$

We now have formulas for p_k, μ_k, D_k and all that is left is to compute all the $q_i(k)$ which can easily calculated by the formulas for conditional probabilities and the law of total probability:

$$q_i(k) = p_\Theta(Z_i = k | x_i) = \frac{p_\Theta(x_i | Z_i = k) p_k}{\sum_j p(x_i, Z_i = j)} = \frac{\mathcal{N}(x_i; \mu_k; D_k) p_k}{\sum_j \mathcal{N}(x_i; \mu_j, D_j) p_j}$$

This completes the algorithm.
The formulas also make sense in an intuitive sense as the $q_i(k)$s give us probabilities that sample x_i belongs to cluster k and the updates for the mean and covariance matrix are just weighted averages over the samples where the weighting follows the $q_i(k)'s$. The updates for p_k is just proportional to the sum of probabilities of the samples to belong to cluster k. We can view this as kind of a soft-assignment version of the KMeans algorithm with assumed normal distributions.

1.2 Advantages

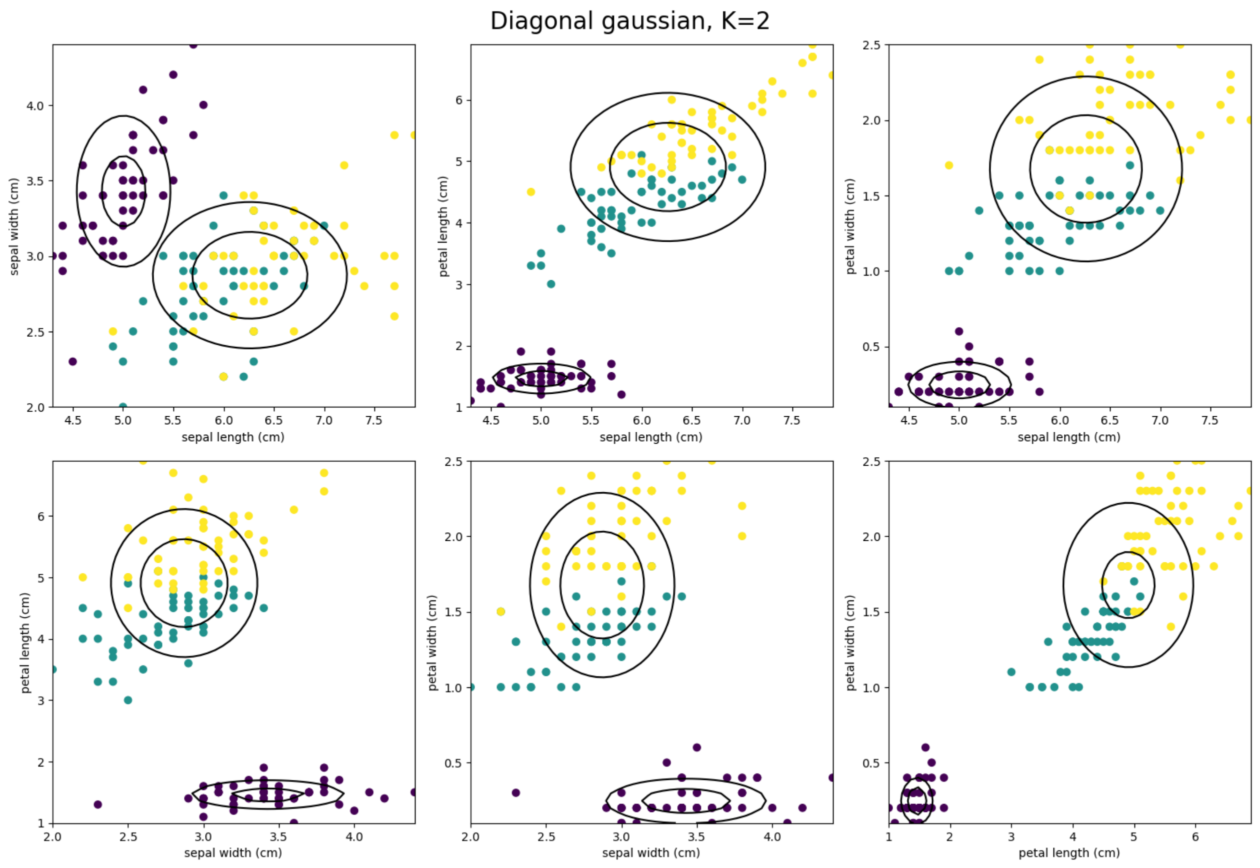
Whilst the diagonal GMM is less general than the full GMM we can state two mayor advantages:

- Less parameters. The number of parameters only grows linear with the dimension whereas for the full GMM it grows quadratically
- Easier computations. In the update (M-step) of the covariance matrices for a full GMM we need to compute the inverse matrices which is very costly. In the diagonal GMM this computation is trivial which gives a runtime advantage.

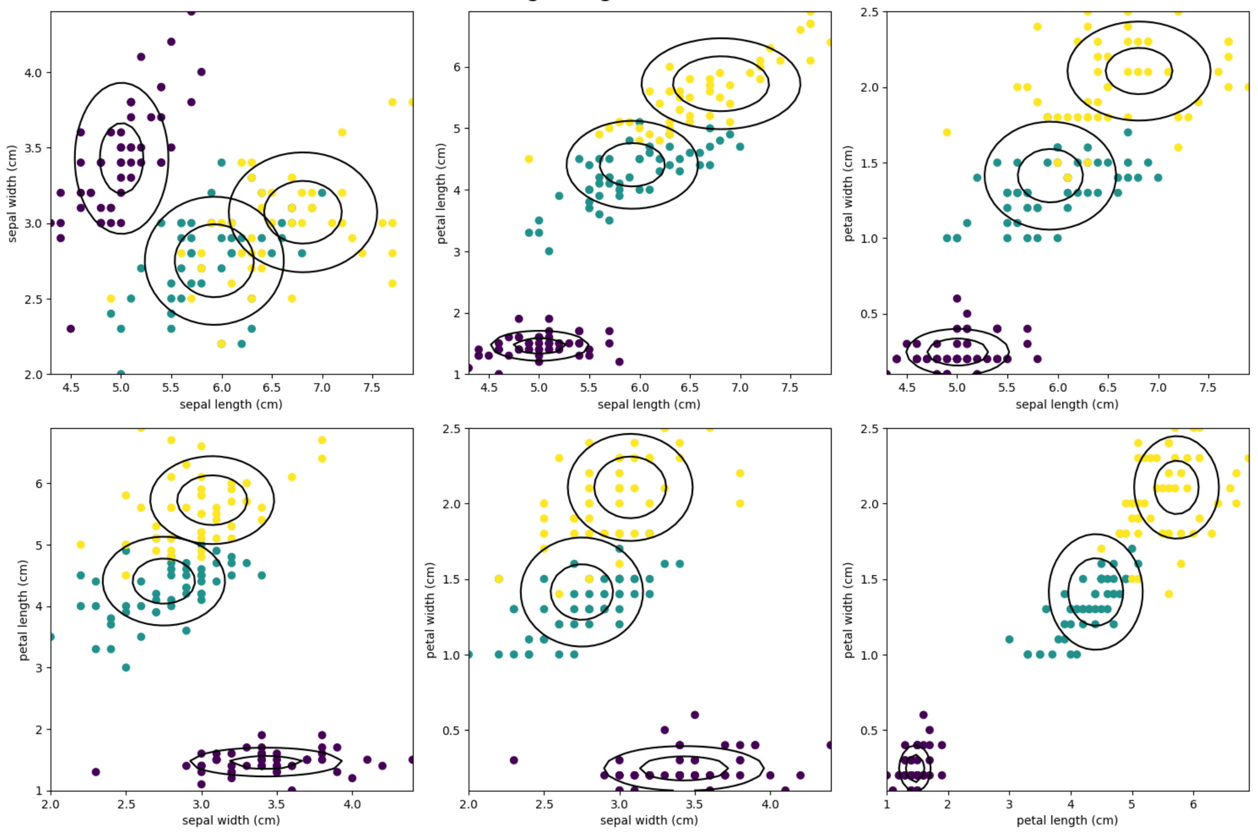
1.3 Comparison - IRIS application

We apply the diagonal GMM, the full GMM and a KMeans algorithm to the IRIS dataset each with different values for K. The IRIS data is four-dimensional, for each of our models we plot a projection of the data and learned distributions onto all pairs of axes giving six plots for each model. In the pictures the colors of the data points indicate the ground truth labels. The learned gaussian distributions are depicted by ellipses that show contour levels of their respective densitiy functions. The three different models (diag GMM, full GMM, KMeans) mostly converge to very similar means. But the predictions that one would make from these models would still differ significantly as the assumed distributions are different and in the case of KMeans we would base a prediction only on the distances to the means. The overall best fit is obtained by the full GMM with $K = 3$.

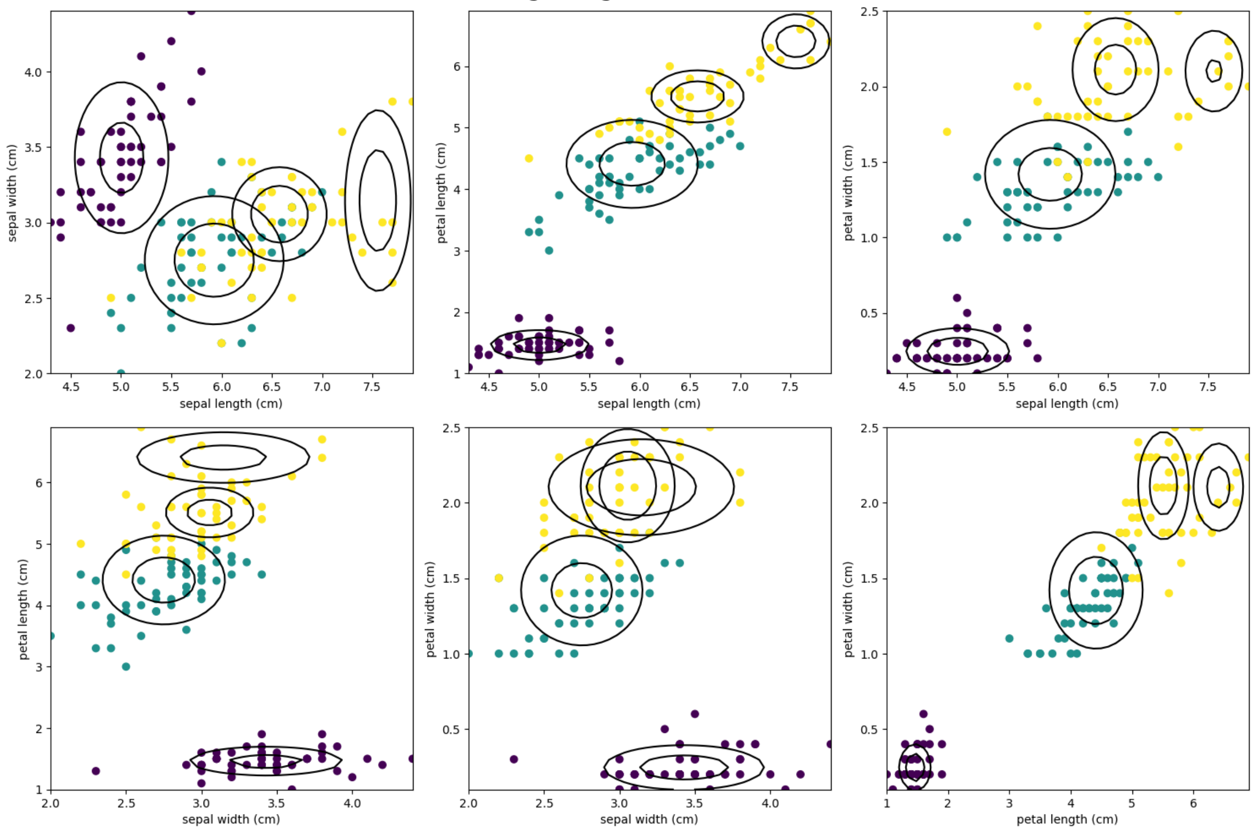
Diagonal GMM



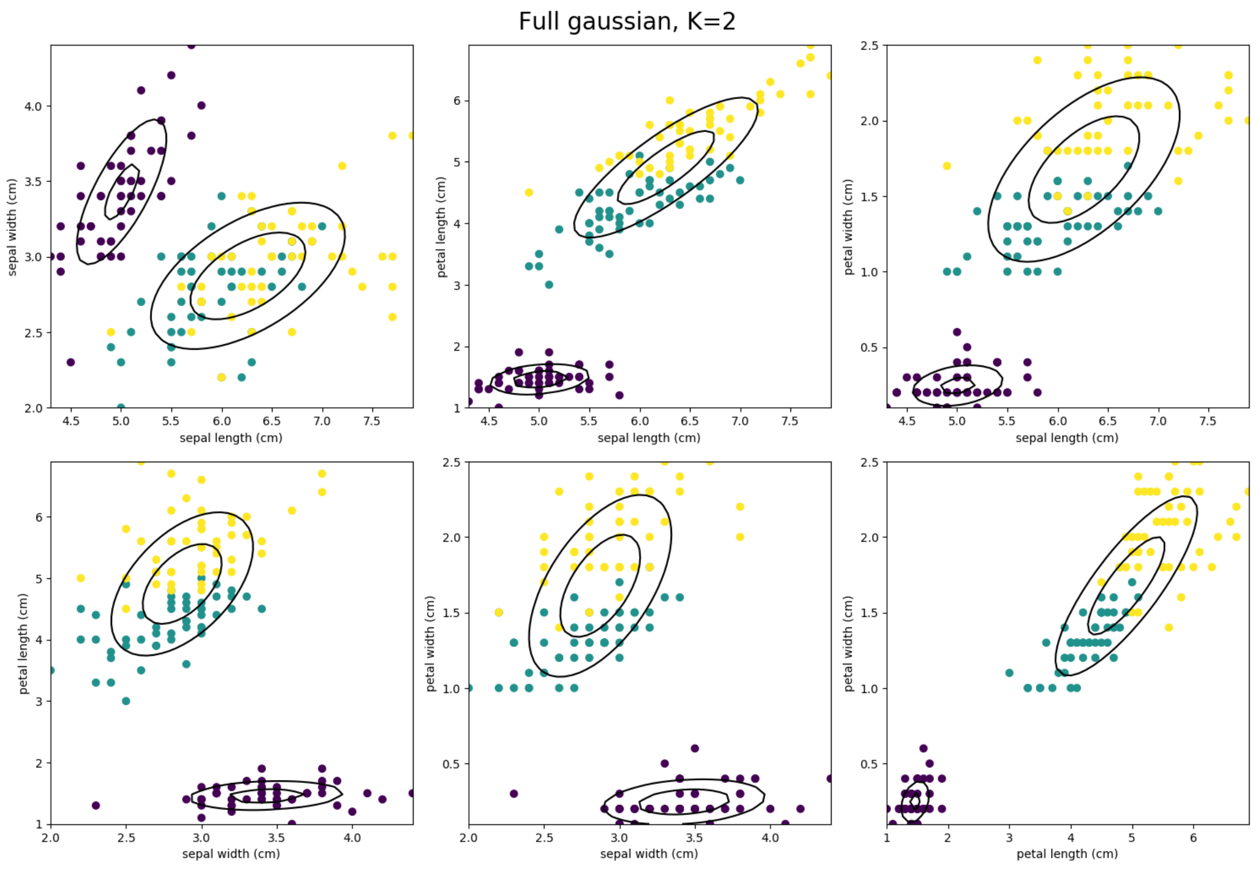
Diagonal gaussian, K=3

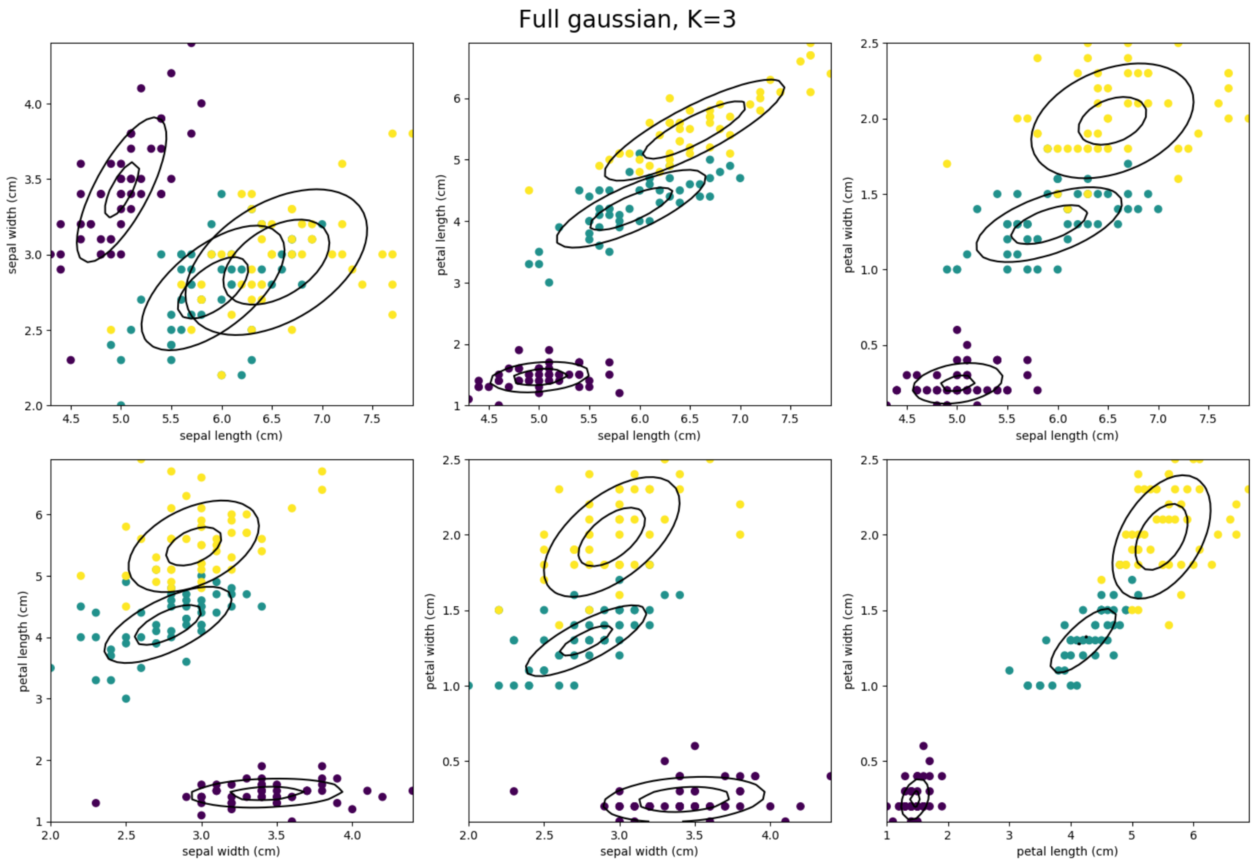


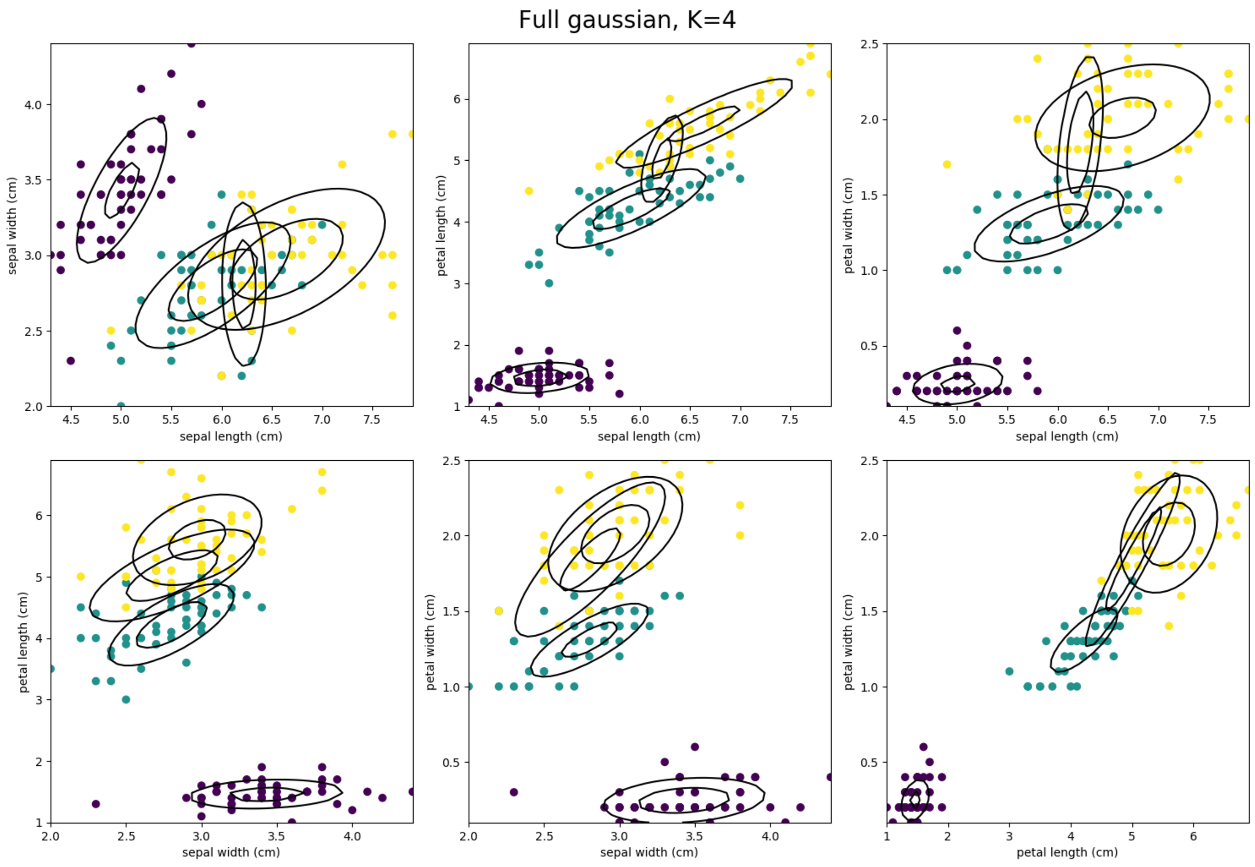
Diagonal gaussian, K=4



Full GMM

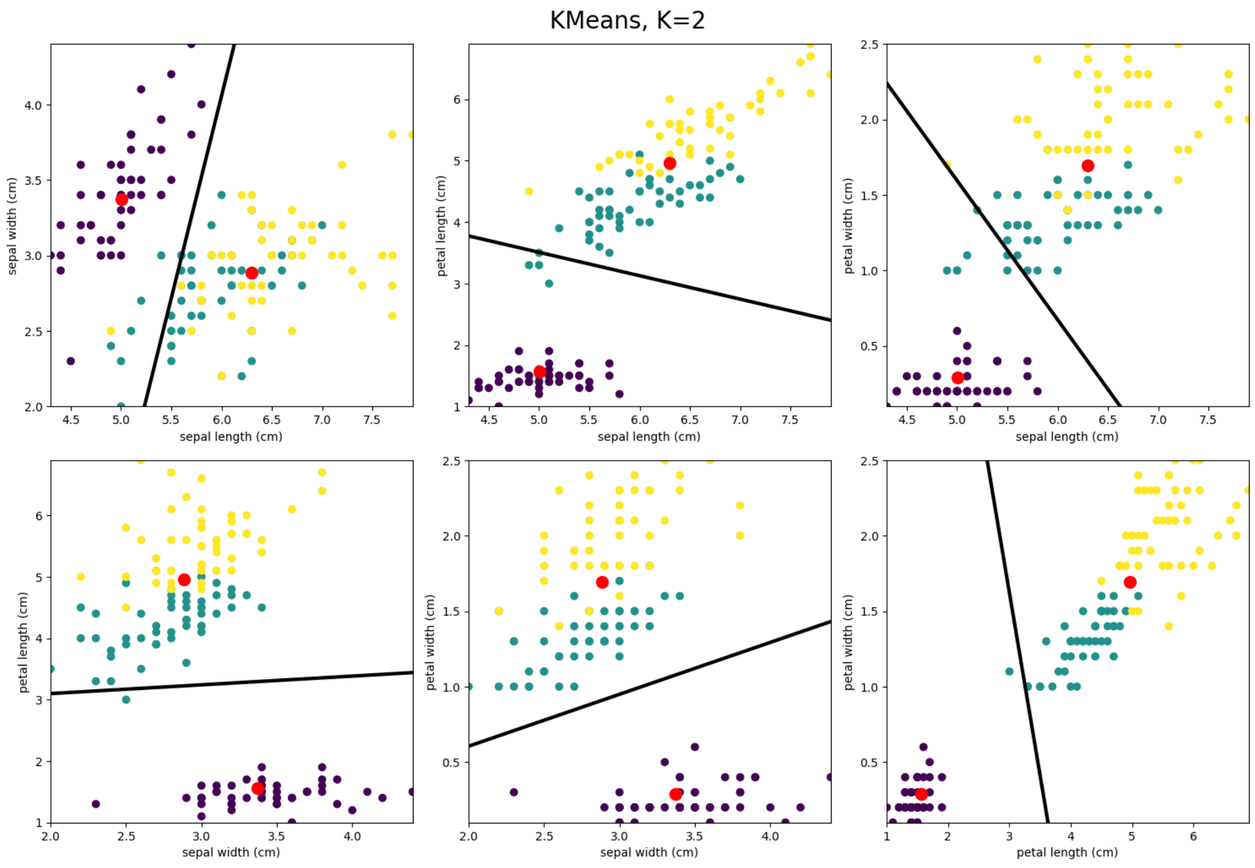

Full gaussian, K=2
sepal width (cm)
sepal length (cm)
petal length (cm)
petal width (cm)
sepal width (cm)
petal length (cm)


Full gaussian, K=3
sepal width (cm)
sepal length (cm)
petal length (cm)
petal width (cm)
sepal width (cm)
petal length (cm)

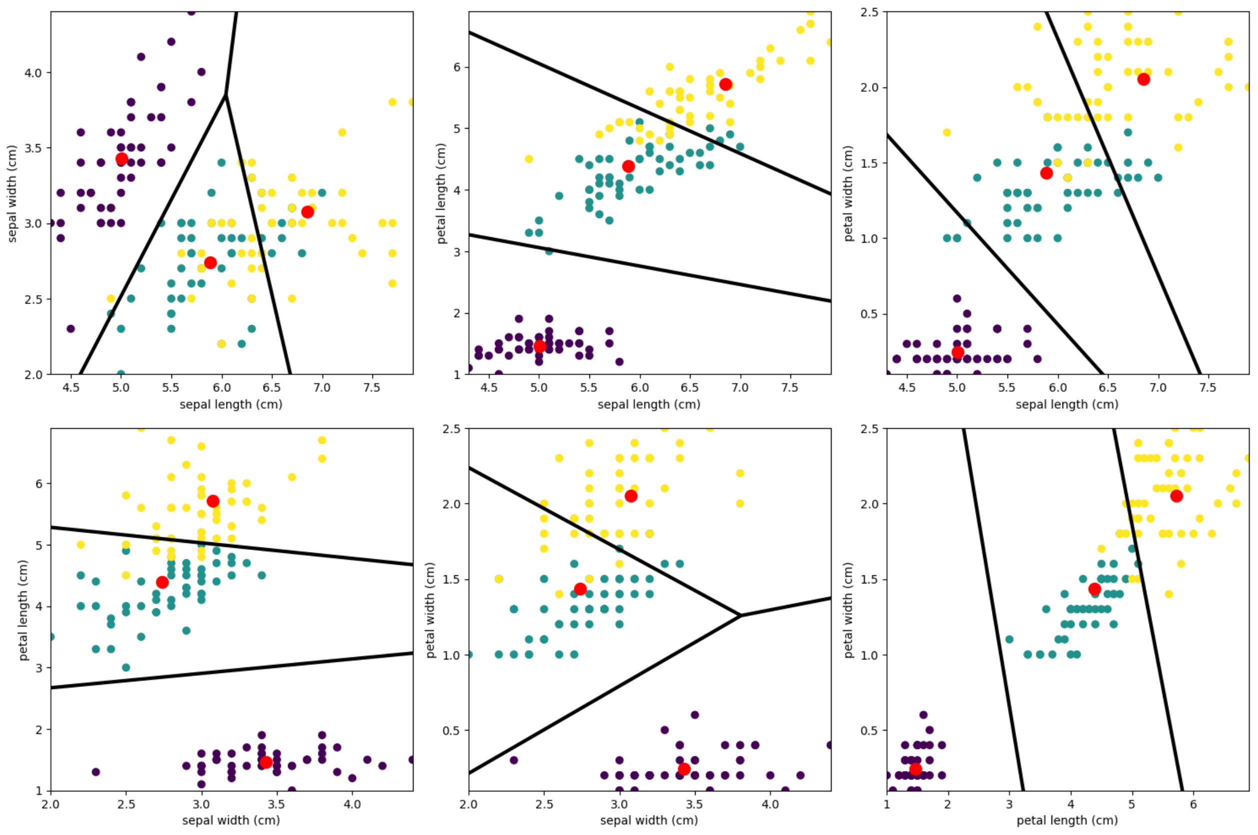


KMeans

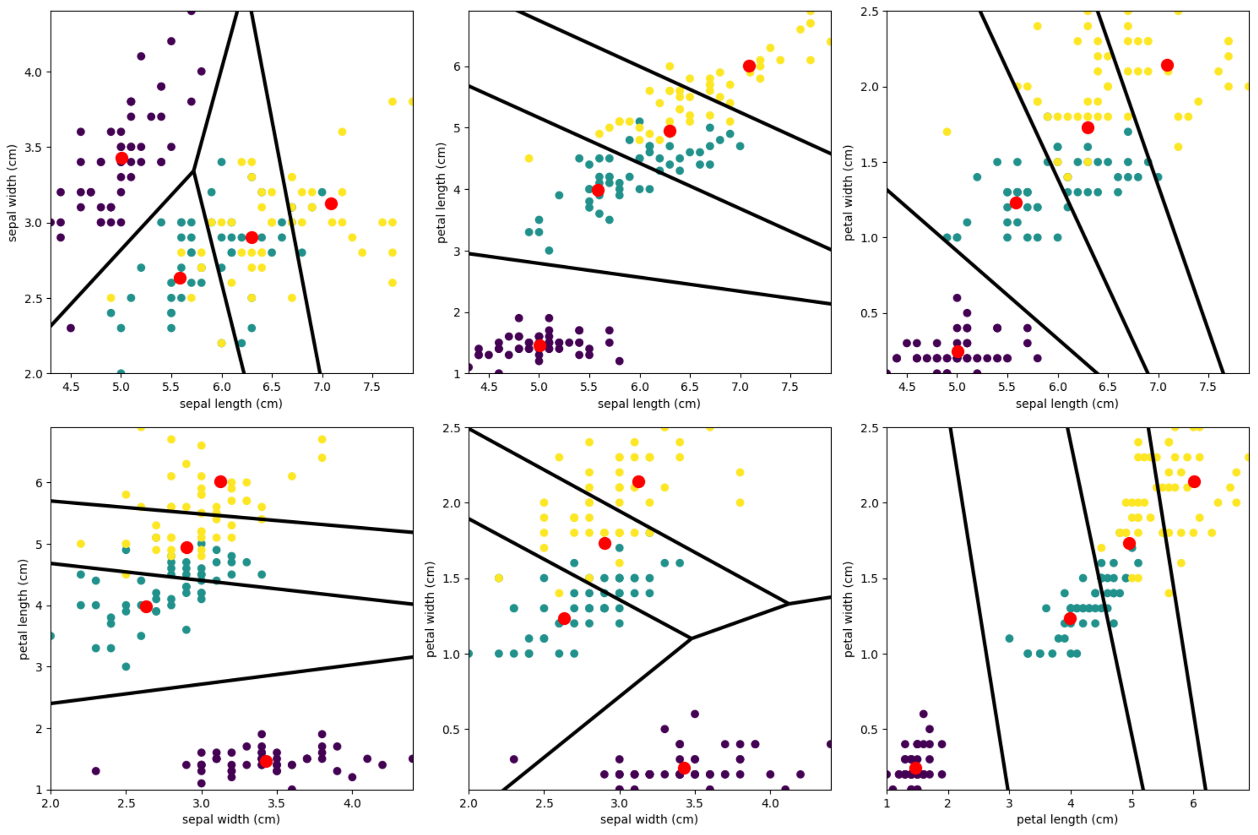
The found mean values are depicted by red dots and the lines show the partition of the space according to the distance to the means (Voronoi cells). Note that the lines indeed intersect the imagined connections of the means orthogonally! It is only the scaling of the axes that skews the angles.



KMeans, K=3

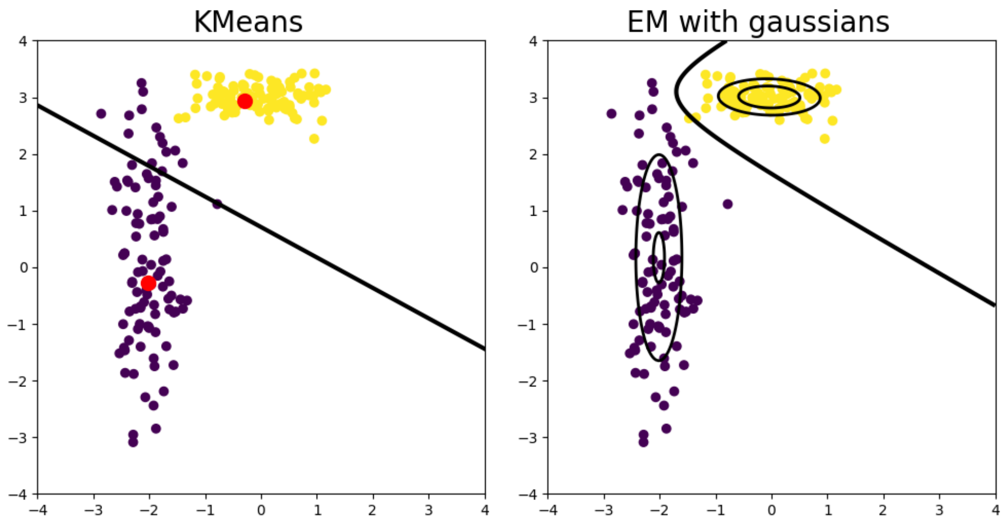


KMeans, K=4

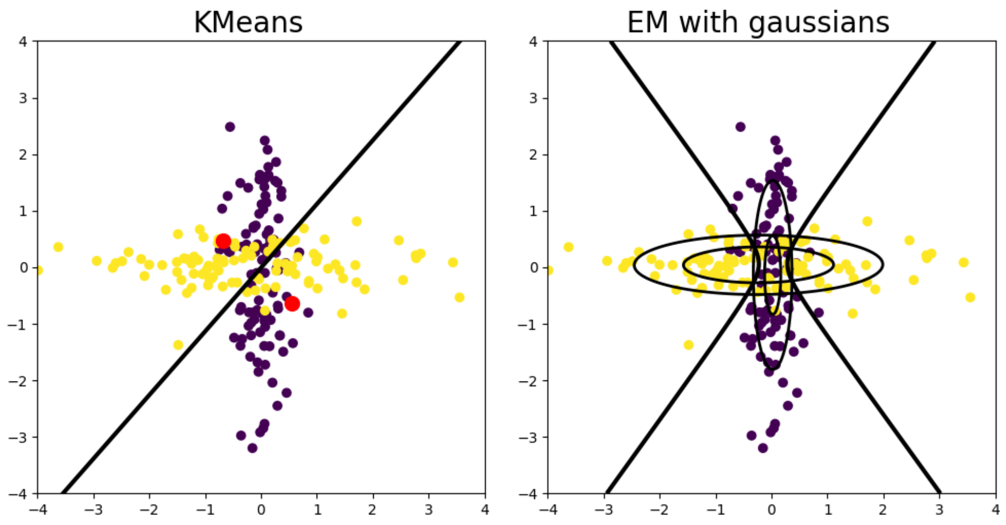


1.4 EM outperforms KMeans

A mayor drawback of the KMeans algorithm comes into play when trying to predict from it. The space will just be separated based on the distance to the means which does not account for mixture models of distributions of different ’size’. Even though KMeans might find the correct means it might have poor prediction quality.



Another even more striking example can be seen when samples are drawn from different distributions with the same mean. KMeans is unable to separate the data but with EM-algorithm and a fit of gaussian distributions we can create a meaningful separation.



2 Graphs, algorithms and Ising

2.1 Chain implementation

I implement the sum-product algorithm for an undirected chain using the formulas for forward and backward messages as derived in the lecture:

$$\mu_{j\to j+1}(x_{j+1}) = \sum_{x_j} \psi_j(x_j)\psi_{j,j+1}(x_j, x_{j+1})\mu_{j-1\to j}(x_j) \tag{1}$$

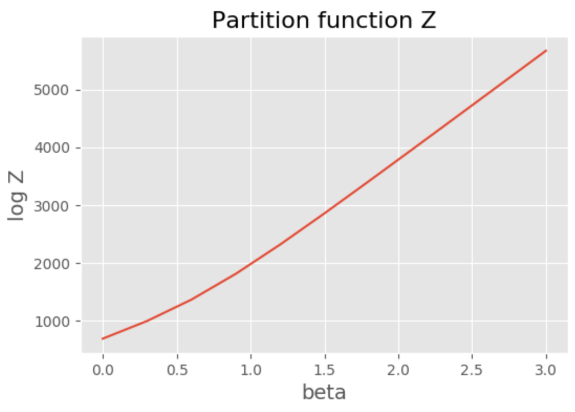
$$\mu_{j\to j-1}(x_{j-1}) = \sum_{x_j} \psi_j(x_j)\psi_{j-1,j}(x_{j-1}, x_j)\mu_{j+1\to j}(x_j) \tag{2}$$

My implementation assumes that we deal with n discrete random variables on a finite set of size K. In view of the input representation $(\psi_i, \psi_i j)$ I implemented two versions.

- The first one is quite general and takes two functions $\text{psi}_1, \text{psi}_2$ as inputs that are callable as in the mathematical formulas where $\text{psi}_1(i, x_i) = \psi_i(x_i)$ and $\text{psi}_2(i, j, x_i, x_j) = \psi_{i,j}(x_i, x_j)$.
- The second version is a wrapper of the first and represents the functions as matrices/tensors. The functions ψ_i are represented as a $n \times K$-matrix and the functions $\psi_{i,i+1}$ as a $n \times K \times K$-Tensor.

Both representations are a little more general than what we need for the Ising model and one could make it more space-efficient when only considering the Ising model but I found it more satisfying this way.

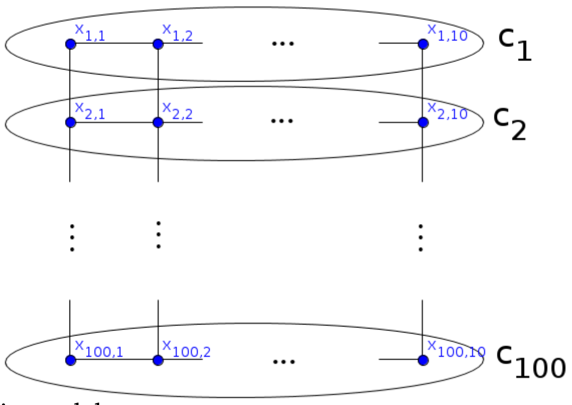
2.2 Exact partition function



We consider the Ising model given in the exercise and want to calculate the exact partition function $Z(\alpha, \beta)$. The nodes $x_{i,j}$ are numbered according to their grid position. The Ising model is a graphical model with maximum clique potentials $\psi_{x,x'} = \exp(\beta\mathbb{1}_{x=x'})$. To be able to use the algorithm from the first part we group the nodes of the model by rows to form clusters $c_i = \{x_{i,j} | 1 \leq j \leq 10\}$. On a cluster level the graph is just a chain of length 100. To be equivalent to the original Ising model each cluster is now allowed to take $2^{10} = 1024$ different values and we use cluster level potentials $\hat{\psi}_c, \hat{\psi}_{c_i,c_{i+1}}$. The $\hat{\psi}_i$ are just the product of the cluster internal potentials (horizontal edges) and $\hat{\psi}_{c,c'}$ are the products of the cluster connecting edges (horizontal edges) between cluster c_i and c_{i+1}. An easy calculation shows that indeed those formulations are equivalent.

$$\hat{\psi}_i(c_i) = \exp(\beta \sum_{j=1}^{9} \mathbb{1}(x_{i,j} = x_{i,j+1}))$$

$$\hat{\psi}_{i,i+1}(c_i, c_{i+1}) = \exp(\beta \sum_{j=1}^{9} \mathbb{1}(x_{i,j} = x_{i+1,j}))$$



We can then compute $Z = \sum_{c_i} p(c_i)$ with the chain model.