Homework 2

1 Classification: K-means, and the EM algorithm

1.1 Estimators computation

We consider a mixture model, with K components. The data datapoints X_i, $i = 1, ..., n$ are such that: $P(Z_i = k) = p_k$ and $X_i|Z_i = k \sim \mathcal{N}(\mu_k, D_k)$, a multivariate Gaussian distribution with mean μ_k, and diagonal covariance matrix D_k. Let denote $\theta = (\pi, \mu, D)$ the parameter to optimize.

1. At the t-th iteration, we calculate first $p_\theta(z_i = k|x_i)$ (we denote θ_t by θ for simplification). We have by the Bayes formula:

$$\begin{aligned} p_\theta(z_i = k|x_i) &= \frac{p_\theta(x_i|z_i = k)p_\theta(z_i = k)}{\sum_j p_\theta(x_i|z_i = j)p_\theta(z_i = j)} \\ &= \frac{p_k \mathcal{N}(x_i|\mu_k, D_k)}{\sum_j p_j \mathcal{N}(x_i|\mu_j, D_j)} \\ &= \tau_i^k \end{aligned}$$

After that, we calculate the complete likelihood:

$$\begin{aligned} l_{c,t} = log(p_\theta(x, z)) &= \sum_{i=1}^{n} log(p_\theta(x_i, z_i)) \\ &= \sum_{i=1}^{n} log(p_\theta(z_i)) + \sum_{i=1}^{n} log(p_\theta(x_i|z_i)) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} z_i^k log(p_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} z_i^k log(\mathcal{N}(x_i|\mu_k, D_k)) \end{aligned}$$

where $z_i^k = 1$ if $z_i = k$, 0 otherwise.

- The E_Step:

$$\begin{aligned} \mathbb{E}_{Z|X}(l_{c,t}) &= \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{E}_{Z|X}(z_i^k) log(p_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{E}_{Z|X}(z_i^k) log(\mathcal{N}(x_i|\mu_k, D_k)) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} p_\theta(z_i = k|x_i) log(p_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} p_\theta(z_i = k|x_i) log(\mathcal{N}(x_i|\mu_k, D_k)) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_i^k log(p_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} \tau_i^k log(\mathcal{N}(x_i|\mu_k, D_k)) \end{aligned}$$

- The M_Step: We maximize $\mathbb{E}_{Z|X}(l_{c,t})$ with respect to $\theta_t = (\pi_t, \mu_t, D_t)$)
  As the sum is seperated into 2 terms independent along the variables. We can first maximize with respect to π_t :

$$\begin{cases} \max \sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k log(p_k) \\ s.t \quad \sum_{k=1}^{K} p_k = 1 \end{cases}$$

  We use her the dual problem which imply: $\forall k \in \{1, ...K\}, \quad p_{k,t+1} = \frac{1}{n} \sum_{i=1}^{n} \tau_i^k$.

  In fact, we have the dual problem:

$$L(\pi, \lambda) = -\sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k log(p_k) + \lambda(\sum_k p_k - 1)$$

  where $\lambda \in \mathbb{R}$. Since $\tau_i^k \geq 0$ for all (i, k) in {1,...n}*{1,...K}, we have a convex optimization problem. It is trivial that there exist π such that $\pi > 0$ and $\sum_k p_k = 1$, then by Slater's constraint qualification, the problem had strong duality propriety.Thus, we have:

$$\min_{\pi} -\sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k log(p_k) = \max_{\lambda} \min_{\pi} L(\pi, \lambda)$$

  As $L(\pi, \lambda)$ is convex with respect to π, it suffices to take derivatives with respect to p_k to zero to find the minimum. This yiels:

$$\frac{\partial L}{\partial p_k} = -\sum_{i=1}^{n} \frac{\tau_i^k}{p_k} + \lambda_1 = 0 \quad \forall k \in \{1, ..., K\}$$

  Since we have $\sum_{k=1}^{K} p_k = 1$ and $\sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k = n$ we get :

  $\forall k \in \{1, ...K\}, \quad p_{k,t+1} = \frac{1}{n} \sum_{i=1}^{n} \tau_i^k$.

  After that we maximize $\sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k log(\mathcal{N}(x_i|\mu_k, D_k))$.We have:

$$\sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k log(\mathcal{N}(x_i|\mu_k, D_k)) = \sum_{i=1}^{n} \sum_{k=1}^{K} \tau_i^k [log(\frac{1}{(2\pi)^{K/2}}) + log(\frac{1}{|D_{k,t}|^{1/2}}) - \frac{1}{2}(x_i - \mu_{k,t})^T D_{k,t}^{-1}(x_i - \mu_{k,t})]$$

  It's a convex problem without constraint, then we can use the gradient directly with respect to μ_t and D_t. we have done the same calculus in the homework 1. We obtain :

$$\mu_{k,t+1} = \frac{\sum_i \tau_i^k x_i}{\sum_i \tau_i^k} \quad and \quad D_{k,t+1} = diag(\frac{\sum_i \tau_i^k (x_i - \mu_{k,t})(x_i - \mu_{k,t})^T}{\sum_i \tau_i^k})$$

where $diag(A)$ is a diagonal matrix such that the elements in the diagonal are the elements from the diagonal of A. Finally, we have:

-$\forall k \in \{1, ...K\}, \quad p_{k,t+1} = \frac{1}{n} \sum_{i=1}^{n} \tau_i^k$.

-$\forall k \in \{1, ...K\}, \quad \mu_{k,t+1} = \frac{\sum_i \tau_i^k xi}{\sum_i \tau_i^k}$.

-$\forall k \in \{1, ...K\}, \quad D_{k,t+1} = diag\left(\frac{\sum_i \tau_i^k (xi - \mu_{k,t+1})(xi - \mu_{k,t+1})^T}{\sum_i \tau_i^k}\right)$

1.2 Advantages of diagonal model

(a) In the diagonal case, the contour axes are oriented along the coordinate axes. The shape of clusters are ellipsoids along both axes whereas in Kmeans for example the shape is circlar

(b) With Full covariance method we have arbitrary covariance matrixes for each class with $O(Kd^2)$ parameters, whereas in the Diagonal case, clusters are axis-aligned ellipsoids with $O(Kd)$ parameters

(c) The Full covariance fits data best but it is costly in high-dimensional feature spaces though and we need to consider then doing a PCA or LDA, However, the Diagonal case in a good compromise between quality and model size

1.3 Implementation

1.3.1 Diagonal Gaussian Mixture Model

The unique difference in the implementation is that at each iteration **t** and for each cluster **k**, the covariance matrix D_{kt} is diagonal. Please refer to the Jupyter notebook for detailed implementation.

1.3.2 Comparaison on Iris Dataset

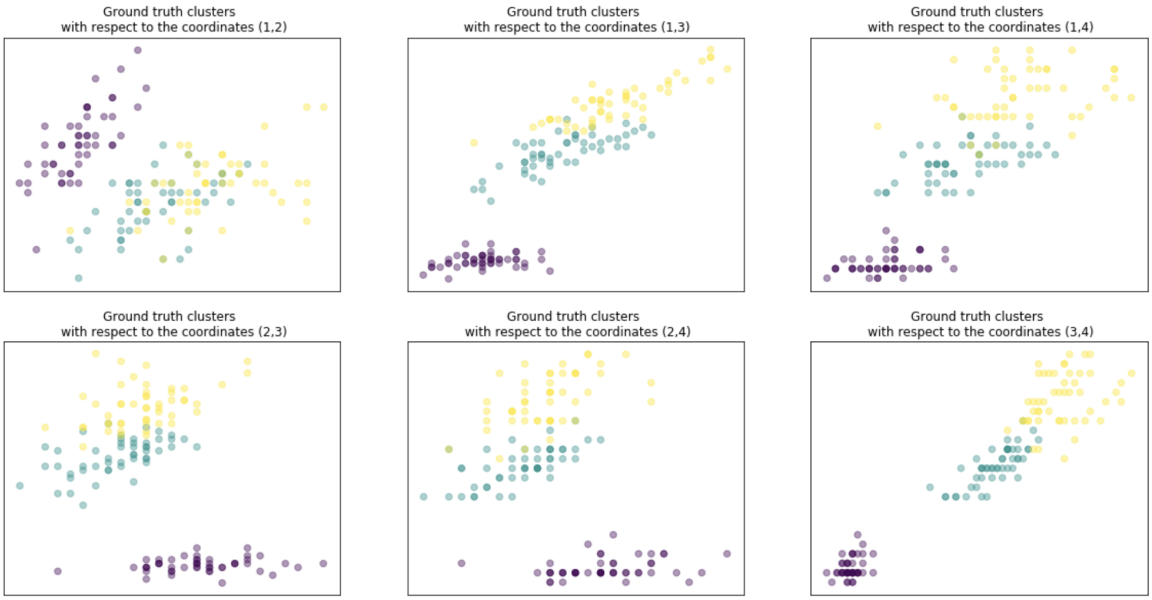

Figure 1: Ground truth on Iris dataset clusters

Case K=2

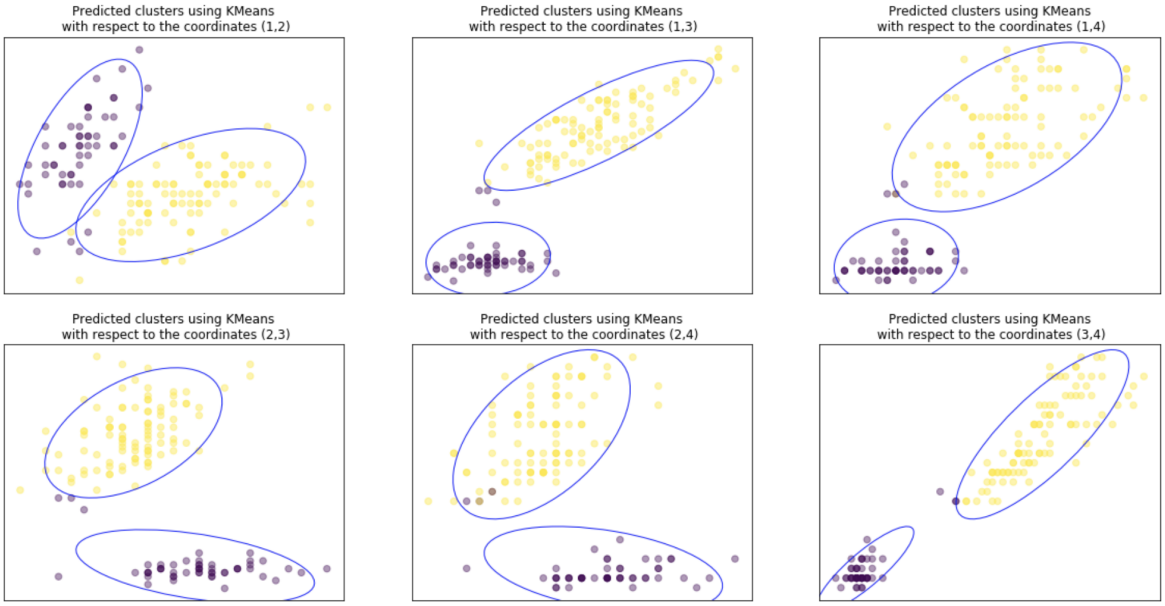

Figure 2: Kmeans Clusters (K=2)

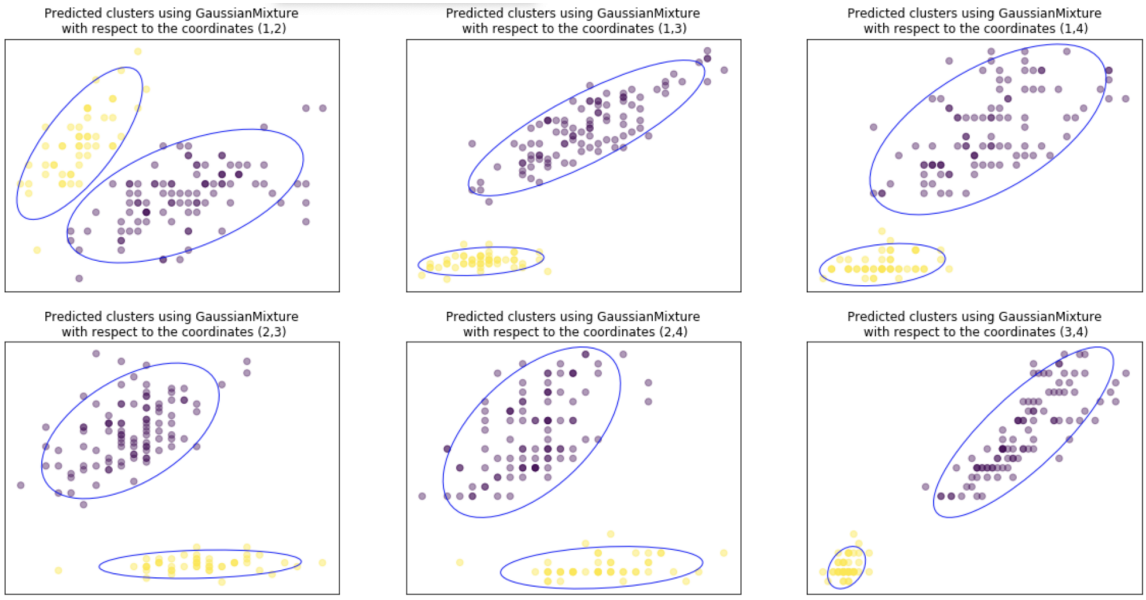

Figure 3: Gaussian mixture model Clusters (K=2)

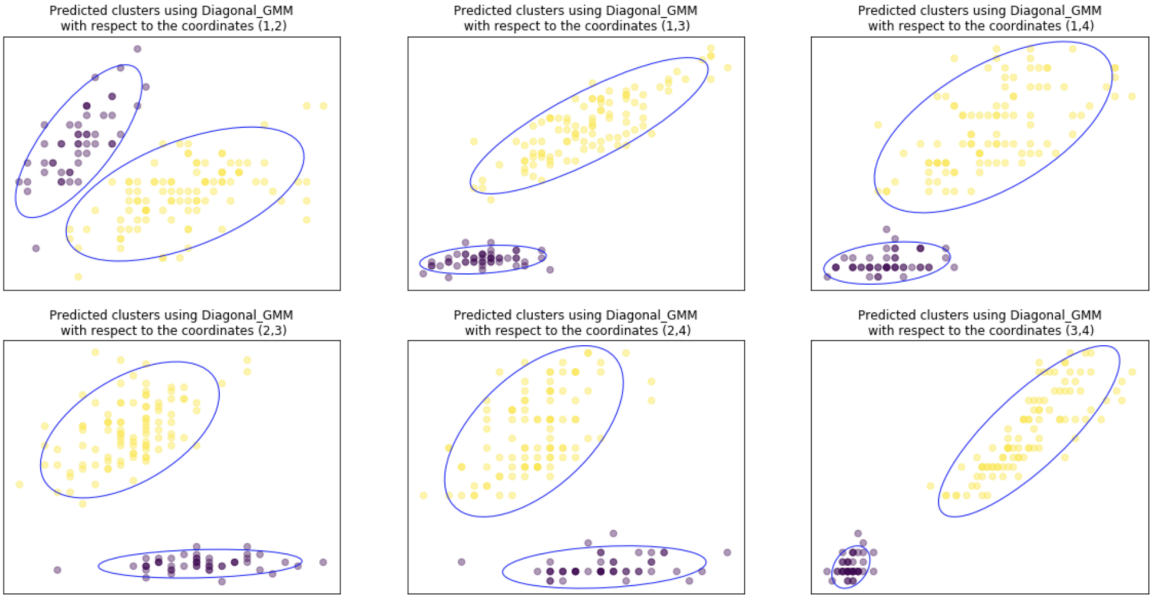

Figure 4: Diagonal Gaussian mixture model Clusters (K=2)

For $K = 2$, the given data is easily separable, in particular when using a suitable pair of dimension as in the figures in the bottom-right. Note that the dimension pair used for clustering significantly affects the results. All three models have comparable clustering results, GMM and DGMM are slightly outperforming Kmeans.

Case K=3

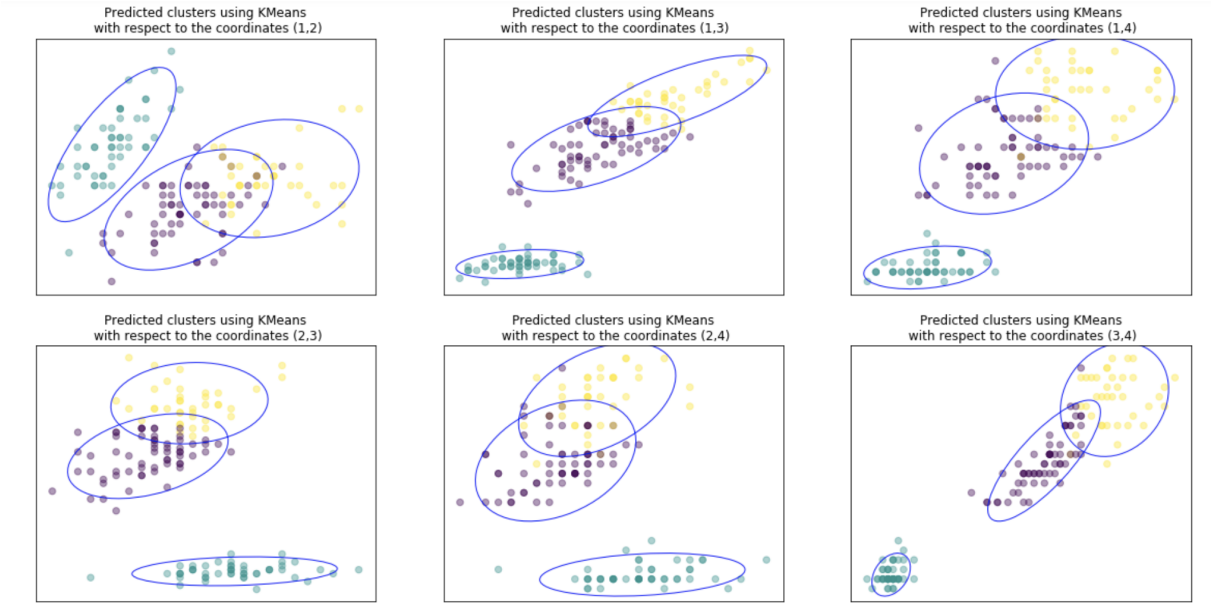

Figure 5: Kmeans Clusters (K=3)

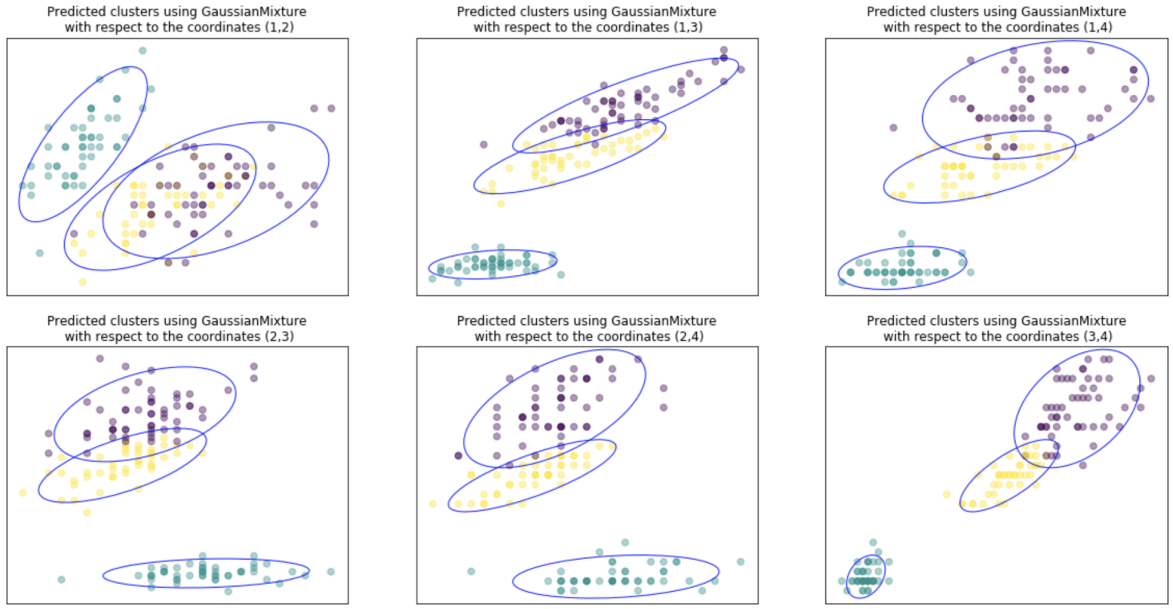

Figure 6: Gaussian mixture model Clusters (K=3)

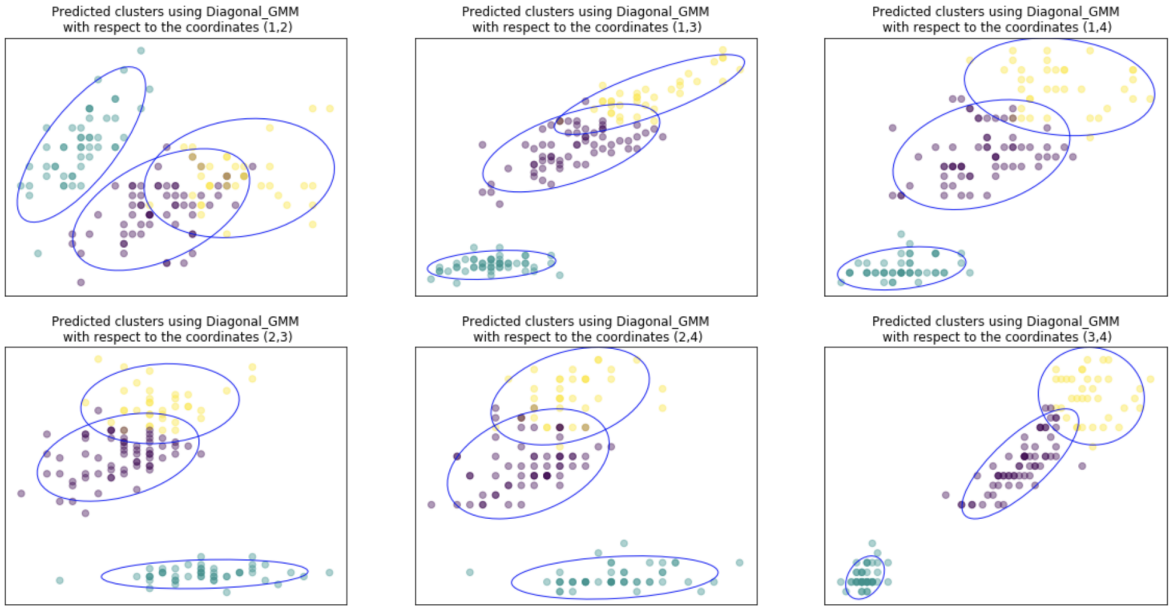

Figure 7: Diagonal Gaussian mixture model Clusters (K=3)

For $K = 3$, the data is not easily separable. Clustering results depend a great deal on the chosen dimension. Note that clustering results are slightly different for the two Gaussian mixture models.

Case K=4

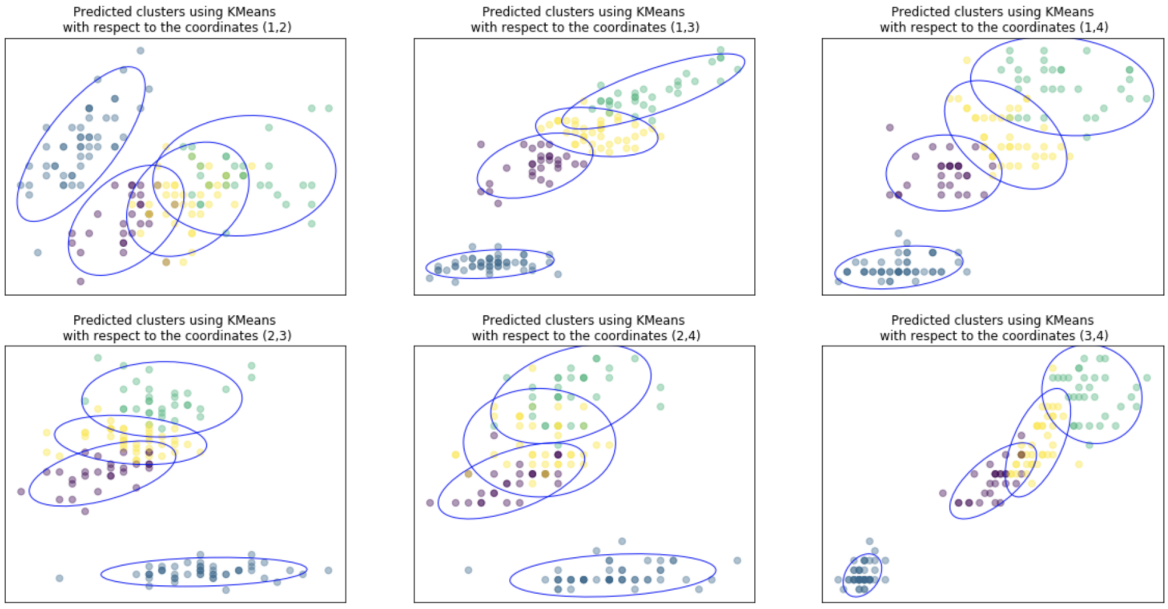

Figure 8: Kmeans Clusters (K=4)

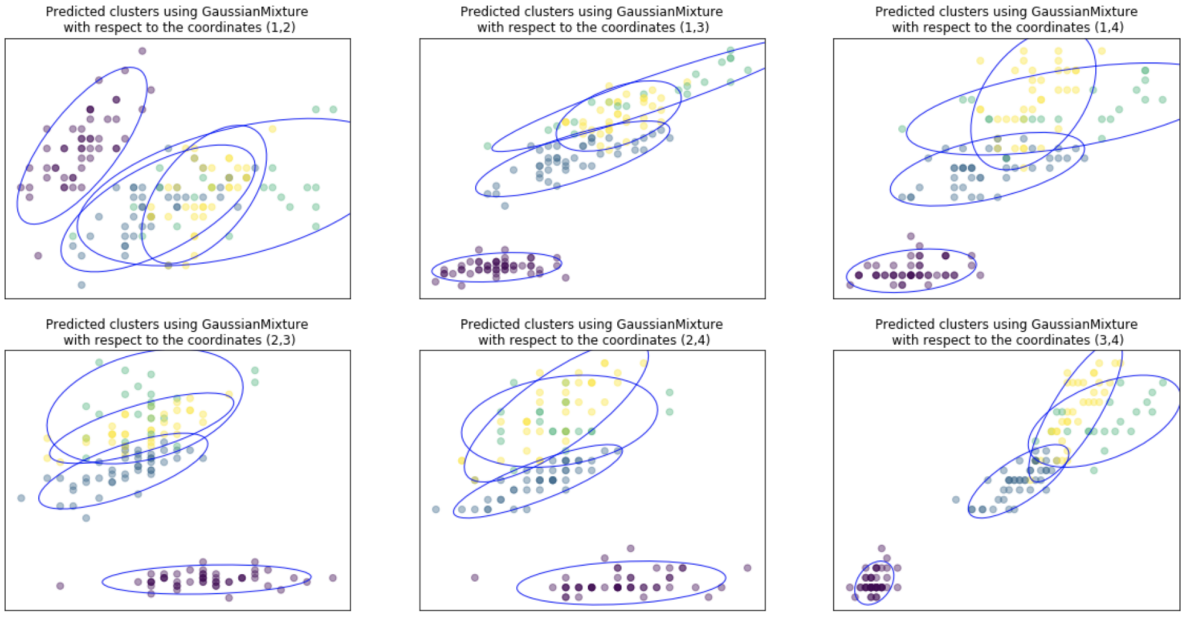

Figure 9: Gaussian mixture model Clusters (K=4)

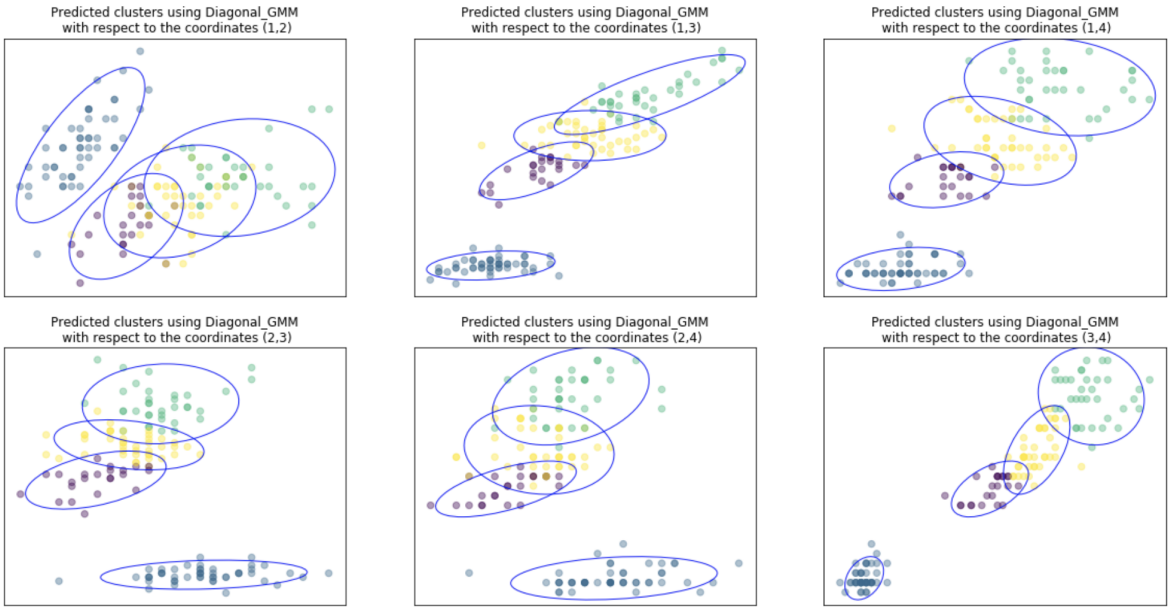

Figure 10: Diagonal Gaussian mixture model Clusters (K=4)

For $K = 4$, clustering results are not satisfying as there are only 3 different ground truth clusters. The choice of parameter K is not suitable.

1.4 Comparison with synthetic Dataset

We use a mixture of 3 different distributions in dimension $d = 2$ with diagonal covariance matrices in order to prove the EM clustering with both full and diagonal covariance matrix outperform kmeans clustering.

These are the different parameters

$$p_1 = p_2 = p_3 = \frac{1}{3}$$

$$\mu_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \mu_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$D_1 = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}, D_2 = \begin{bmatrix} 0.1 & 0 \\ 0 & 3 \end{bmatrix}, D_3 = \begin{bmatrix} 10 & 0 \\ 0 & 0.5 \end{bmatrix}$$

From the different plots (Fig 11) below we can notice the GMM clustering with EM algorithms detects the same clusters shapes as the ground truth. However, Kmeans fails to detect the clusters as it tries to construct circular clusters.

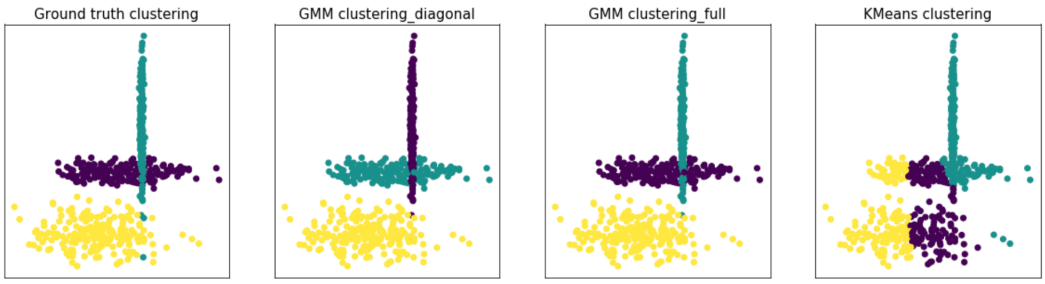

Figure 11: Different clustering methods applied on our Synthetic Data

2 Graphs, algorithms and Ising

2.1 Sum-product algorithm

2.1.1 Undirected Chain Model

Consider an undirected chain of n vertices.

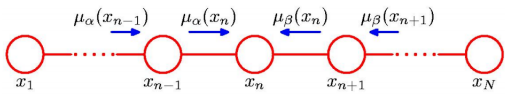

Figure 12: Undirected chain

The joint distribution on the undirected chain can be expressed as the following:

$$p(x_1, x_2, ..., x_n) = \frac{1}{Z} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=1}^{n-1} \psi_{i,i+1}(x_i, x_{i+1}).$$

To compute the marginal distribution for each $j \in [1, n]$, one can write

$$p(x_j) = \sum_{x_V/\{j\}} p(x_1, x_2, ..., x_n).$$

Another way to express $p(x_j)$ is

$$p(x_j) = \frac{1}{Z} \mu_\alpha(x_j) \psi_j(x_j) \mu_\beta(x_j)$$

with the **Forward message**

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

and the **Backward message**

$$\mu_\beta(x_j) = \sum_{x_{j+1}} \psi_{j,j+1}(x_j, x_{j+1})\mu_\beta(x_{j+1}) = \mu_{j+1\to j}(x_j).$$

The **normalization constant** Z is the following

$$Z = \sum_{x_j} \mu_{j-1\to j}(x_j)\psi_j(x_j)\mu_{j+1\to j}(x_j).$$

2.1.2 Implementation details

In this section, please refer to the notebook for the detailed implementation of the SPA class. In the SPA class, we compute the forward message, the backward message, the normalization constant, as well as the marginal distribution using the log-sum trick to avoid any overflow and underflow.
As for the following example, $\forall i, \psi_i$ and $\forall (i,j), \psi_{i,j}$ do not depend on the indices, we take as in input, X the matrix of all possible states for $X_1, ..., X_N$, with N being the length of the chain, `single_potential` $= \psi_i, \forall i$ and `joint_potential` $= \psi_{i,j}, \forall (i,j)$.

2.2 Ising Model

2.2.1 From Ising model to an undirected chain

In order to use SPA implementation from 2.1, one needs to transform the **Ising Model** into an undirected chain.
As the width w is smaller than the height h in this given example, one can be inspired from the idea behind junction tree, so as to assemble all nodes in a row into a super node.
Note that super nodes can take values from 0 to $2^w - 1$.

The joint probability in the Ising model can be written as

$$\begin{aligned}
p(x_1, ..., x_n) &= \frac{1}{Z} \exp(\alpha \sum_i x_i + \beta \sum_{i \sim j} 1_{x_i = x_j}) \\
&= \frac{1}{Z} \exp(\alpha \sum_{h_i=1}^{h} \sum_{v=1}^{w} x_{h_i,v} + \beta \sum_{h_i=1}^{h} \sum_{v=1}^{w-1} 1_{x_{h_i,v} = x_{h_i,v+1}} + \sum_{h_i=1}^{h-1} \sum_{v=1}^{w} 1_{x_{h_i,v} = x_{h_{i+1},v}}) \\
&= \frac{1}{Z} \prod_{h_i=1}^{h} \exp(\alpha \sum_{v=1}^{w} x_{h_i,v} + \beta \sum_{v=1}^{w-1} 1_{x_{h_i,v} = x_{h_i,v+1}}) \prod_{h_i=1}^{h-1} \exp(\beta \sum_{v=1}^{w} 1_{x_{h_i,v} = x_{h_{i+1},v}}) \\
&= \frac{1}{Z} \prod_{h_i=1}^{h} \psi_{h_i}(x_{h_i}) \prod_{h_i=1}^{h-1} \psi_{h_i,h_{i+1}}(x_{h_i}, x_{h_{i+1}})
\end{aligned}$$

where

$$\psi_{h_i}(x_{h_i}) = \exp(\alpha \sum_{v=1}^{w} x_{h_i,v} + \beta \sum_{v=1}^{w-1} 1_{x_{h_i,v} = x_{h_i,v+1}})$$

$$\psi_{h_i,h_{i+1}}(x_{h_i}, x_{h_{i+1}}) = \exp(\beta \sum_{v=1}^{w} 1_{x_{h_i,v} = x_{h_{i+1},v}}).$$

(a) The Ising model can be transformed into an undirected chain using super nodes x_{h_i} and the new expression of ψ_{h_i} and ψ_{h_i,h_i+1}.

(b) In this case $N = h$ and $X = \{0, 1, ..., 2^w - 1\}$. We can consider super nodes values x_{h_i} in X and $x_{h_i} = (x_{h_i,1}, ..., x_{h_i,w})$ its binary representation with w bits.

(c) Compute $Z(\alpha, \beta)$ by using SPA computed in 2.1.

2.2.2 Implementation details

The following plot presents the evolution Z over β. It is obtained using the suggested parameters $w = 10$, $h = 100$, and $\alpha = 0$.

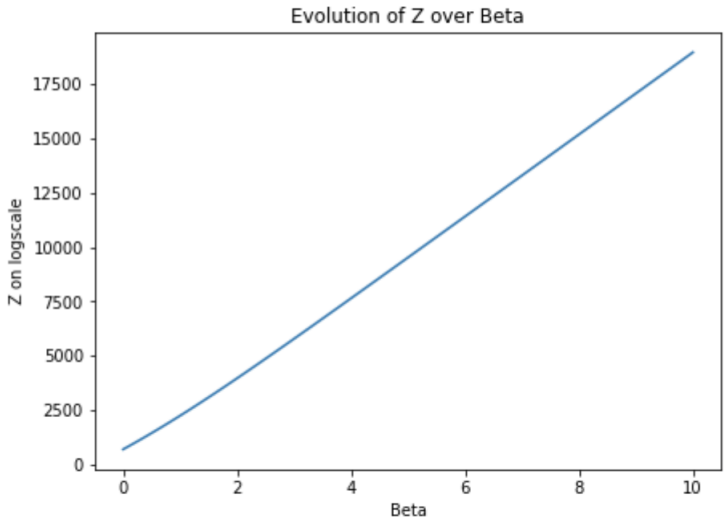

Figure 13: Z evolution wrt β

We can notice that $\log(Z(\beta))$ increases with respect to β.

2.3 Computing loopy belief propagation

2.3.1 Explanation

We consider the graphical model $G = (V, E)$ that may contain loops where the joint distribution is the following

$$P(x) = \frac{1}{Z} \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j) \prod_{i\in V} \psi_i(x_i)$$

The Loopy belief propagation updates are the following:

$$m_{i\to j} = w \sum_{x_i} \psi_{i,j}(x_i, x_j)\psi_i(x_i) \prod_{k\in N(i)/j} m_{k\to i}(x_i)$$

with w normalization factor

The approximated marginals are the following :

$$b_i(x_i) = w \prod_{j\in N(i)} m^*_{i,j}(x_i)$$

$$b_{j,i}(x_i, x_j) = w\psi_{j,i}(x_j, x_i) \prod_{k\in N(j)/i} m^*_{j,k}(x_j) \prod_{k'\in N(i)/j} m^*_{i,k'}(x_i)$$

$log(Z) = \sum_{(j,i)\in E} \sum_{(x_j,x_i)} b_{j,i}(x_i, x_j) \log(\psi_{j,i}(x_j, x_i)) - \sum_{(j,i)\in E} \sum_{(x_j,x_i)} b_{j,i}(x_i, x_j) \log(b_{j,i}(x_i, x_j)) + \sum_{i\in V} \sum_{x_i} b_i(x_i) log(b_i(x_i))$

2.3.2 Implementation details

Please find the detailed algorithm in the notebook. Unfortunately, the method that we used did not work.