MATHÉMATIQUES
VISION
APPRENTISSAGE

Computational Statistics - TD1

Louis Martinez
louis.martinez@telecom-paris.fr

(*The notebook of all implementations can be found here*)

Exercise 1: Discrete distributions

1)

Let $n \in \mathbb{N}^*$ and $X = \{x_1, ..., x_n\}$ where x_i are real values. To make the practical implementation of the algorithm, and without any loss of generality, we assume that x_i is an integer. Up to a permutation, and as every x_k are distinct, we can build a bijection such that $\tilde{x}_i = i$ for $i \in (1, ..., n)$.

We can generate a discrete random variable having the same distribution as X using the same method as in TP1.

Let $F_X : k \mapsto \sum_{i=1}^{k} p_i$ the cdf of X. If $U \sim \mathcal{U}[0, 1]$ then $F(U) \sim X$. But the inverse of F_X is not well defined as it's not continuous and bijective. Hence we choose the lowest k such that:

$$U \leq \sum_{i=1}^{k} p_i$$

Here U is used as the "target" probability (hence the value of X) we want to sample.

The algorithm boils down to:

sample $U \sim \mathcal{U}]0, 1[$
$k^* \leftarrow \min_k \left\{ k \in \mathbb{N}^* \mid U \leq \sum_{i=1}^{k} p_i = F_X(k) \right\}$
return k^*

2)

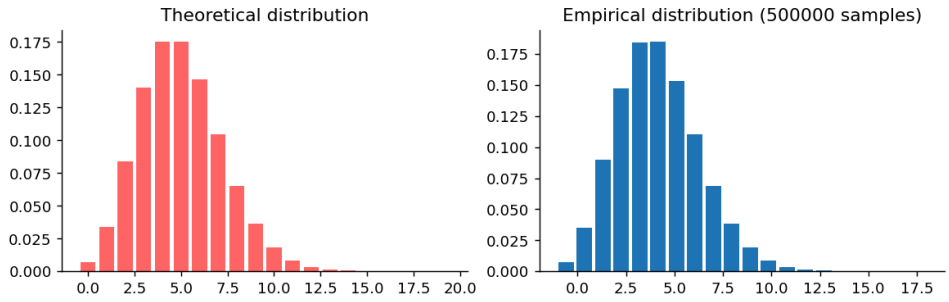
Here is the python implementation of the algorithm. The code can be found on this notebook

```
def sample(pdf, n_samples):
  """
  Samples n_samples samples from the probability density distribution pdf

  k is computed using binary search on the cdf.
  Ad the cdf is by definition an increasing array, we can apply binary search,
ensuring a O(log n)) complexity (insteaf of O(n))
  """
  u = np.random.uniform(size=n_samples)
  cdf = pdf.cumsum() # integral of the pdf
  k = np.searchsorted(cdf, u, side='right') - 1 # binary search
  return k
```

3)

The algorithm is tested to sample a Poisson distribution:



Exercise 2: Gaussian mixture model and the EM algorithm

1)

The parameters of the model are:

$$\boxed{\theta = \left(\theta_j\right)_{1 \leq j \leq m} = (\alpha_1, ..., \alpha_m, \mu_1, ...\mu_m, \Sigma_1, ..., \Sigma_m)}$$

On the one hand, we assume that $\forall j \in [\![1, m]\!], \mu_j \in \mathbb{R}^d$. Hence the pdf of $\mathcal{N}\left(\mu_j, \Sigma_j\right)$ is:

$$f_{X_i \mid Z_i=j}(x) = \varphi_j(x) = \frac{1}{\sqrt{(2\pi)^d \det\left(\Sigma_j\right)}} \exp\left(-\frac{1}{2}\left(x-\mu_j\right)^T \Sigma^{-1}\left(x-\mu_j\right)\right) \quad \left(x \in \mathbb{R}^d\right)$$

On the other hand, the law of total probabilities tells us that:

$$\begin{aligned} \forall i \in [\![1, n]\!], f_\theta(x_i) &= \sum_{j=1}^{m} f_{X_i \mid Z_i=j}(x_i) p_\theta(Z_i = j) \\ &= \sum_{j=1}^{m} \alpha_j \varphi_j(x_i) \end{aligned}$$

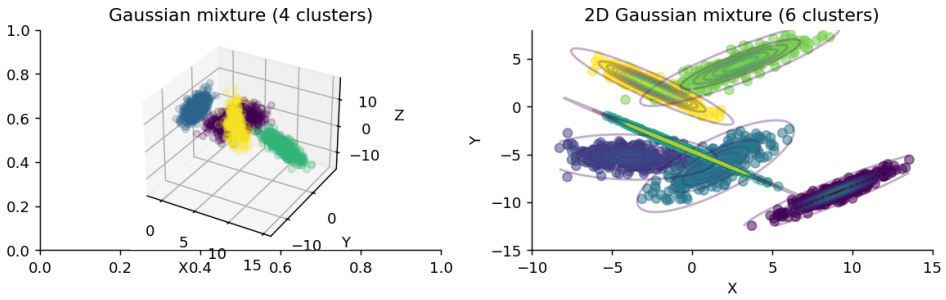
So:

$$\boxed{\mathcal{L}(x_1, ..., x_n, \theta) = \prod_{i=1}^{n} \sum_{j=1}^{m} \alpha_j \varphi_j(x_i)}$$

2)

To sample an from a Gaussian mixture, each cluster j must be populated with the corresponding proportion α_j. So using the notations from the previous exercise, $(p_i)_{i \in [\![1,n]\!]}$ corresponds the $\left(\alpha_j\right)_{j \in [\![1,m]\!]}$ and X corresponds to the random variable Z_i.

On the figure below, we sampled 2D and 3D Gaussian mixtures.



3)

We first derive the formulas to update we'll use to update θ.

Expectation step

$$Q(\theta|\theta^t) = \mathbb{E}_{z\sim p_{\theta^t}(\cdot \mid x)}(\log p_\theta(x,z)) = \sum_{j=1}^{m}\sum_{i=1}^{n} p_{\theta^t}(z_i = j|x_i)\log p_\theta(x_i, z_i = j)$$

We can write that

$$\begin{aligned}\forall (i,j) \in [\![1,n]\!] \times [\![1,m]\!], \quad p_\theta(x_i, z_i = j) &= p_\theta(x_i|z_i = j)p_\theta(z_i = j) \\ &= \varphi_j(x_i)\alpha_j\end{aligned}$$

As we want to compute the expectation with respect to $p_{\theta^t}(\cdot \mid x)$, we compute $p_{\theta^t}(z_i = j \mid x_i)$.

$$\forall (i,j) \in [\![1,n]\!] \times [\![1,m]\!], \quad p_{\theta^t}(z_i = j \mid x_i) = \frac{p_{\theta^t}(x_i|z_i = j)p_{\theta^t}(z_i = j)}{p_{\theta^t}(x_i)} = \frac{\alpha_j^t\varphi_j^t(x_i)}{\sum_{k=1}^{p}\alpha_k^t\varphi_k^t(x_i)}$$

For ease of notation and because for fixed i and j the above term is constant we denote it by

$$\gamma_{ij}^t = \frac{\alpha_j^t\varphi_j^t(x_i)}{\sum_{k=1}^{p}\alpha_k^t\varphi_k^t(x_i)}$$

We used the law of total probabilities to write

$$p_{\theta^t}(x_i) = \sum_{k=1}^{m} p_{\theta^t}(x_i \mid z_i = k)p_{\theta^t}(z_i = k)$$

Wrapping up everything together we get

$$Q(\theta|\theta^t) = \sum_{j=1}^{m}\sum_{i=1}^{n}\gamma_{ij}^t\big(\log\varphi_j(x_i) + \log\alpha_j\big)$$

Maximization step

$$\theta^{t+1} = \arg\max_\theta Q(\theta|\theta^t)$$

This steps boils down to solving the following optimization problem:

$$\begin{aligned}&\min -Q(\theta|\theta^t) \\ \text{subject to } &\alpha \geq 0 \\ &\sum_{i=1}^{m}\alpha_i = 1\end{aligned}$$

The Lagrangian of the problem is

$$\mathcal{L}(\theta, \lambda, \mu) = -\sum_{j=1}^{m}\sum_{i=1}^{n} \gamma_{ij}^{t}\big(\log \varphi_j(x_i) + \log \alpha_j\big) + \lambda\left(\sum_{j=1}^{m} \alpha_i - 1\right) - \mu^T \alpha$$

The complementary slackness of KKT conditions ensure that $\mu^T \alpha = 0$. Now using the first order condition of the KKT conditions we can determine θ^{t+1}.

- Expression of α^{t+1}

$$\forall j \in [\![1, m]\!], \quad \frac{\partial \mathcal{L}}{\partial \alpha_j} = -\sum_{i=1}^{n} \frac{\gamma_{ij}^{t}}{\alpha_j} + \lambda \quad \Rightarrow \quad \alpha_j^{t+1}\lambda = \sum_{i=1}^{n} \gamma_{ij}^{t}$$

If we sum up the previous quantity with respect to j we have

$$\begin{aligned}
\sum_{j=1}^{m} \alpha_j^{t+1}\lambda = \sum_{j=1}^{m}\sum_{i=1}^{n} \gamma_{ij}^{t} \Leftrightarrow \lambda = \sum_{j=1}^{m}\sum_{i=1}^{n} \gamma_{ij}^{t} \quad \text{By definition, } \sum_{j=1}^{m} \alpha_j^{t+1} = 1 \\
\Leftrightarrow \lambda = \sum_{i=1}^{n}\sum_{j=1}^{m} \gamma_{ij}^{t} \\
\Leftrightarrow \lambda = \sum_{i=1}^{n} 1 = n
\end{aligned}$$

$$\boxed{\alpha_j^{t+1} = \frac{1}{n}\sum_{i=1}^{n} \gamma_{ij}^{t}}$$

- Expression of μ^{t+1}

$$\begin{aligned}
\forall j \in [\![1, m]\!], \quad \frac{\partial \mathcal{L}}{\partial \mu_j} = \sum_{i=1}^{n} \gamma_{ij}^{t}\Sigma_j^{-1}(x_i - \mu_j) \quad \Rightarrow \quad \sum_{i=1}^{n} \gamma_{ij}^{t}\Sigma_j^{-1}x_i = \sum_{i=1}^{n} \gamma_{ij}^{t}\Sigma_j^{-1}\mu_j^{k+1} \\
\Leftrightarrow \sum_{i=1}^{n} \gamma_{ij}^{t}x_i = \sum_{i=1}^{n} \gamma_{ij}^{t}\mu_j^{k+1} \\
\Leftrightarrow \mu_j^{k+1} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{t}x_i}{\sum_{i=1}^{n} \gamma_{ij}^{t}}
\end{aligned}$$

$$\boxed{\mu_j^{k+1} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{t}x_i}{\sum_{i=1}^{n} \gamma_{ij}^{t}}}$$

- Expression of Σ^{k+1}

With the same type of computations, we finally get:

$$\boxed{\Sigma_j^{k+1} = \frac{\sum_{i=1}^{n} \gamma_{ij}^{t}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{n} \gamma_{ij}^{t}}}$$

As the parameter update only depends on the previous parameters $\left(\alpha_j^t, \mu_j^t, \Sigma_j^t\right)$ and γ_{ij}^t the actual implementation of the EM algorithm doesn't require to explicitly compute $Q(\theta|\theta^t)$. Hence, it boils down to:

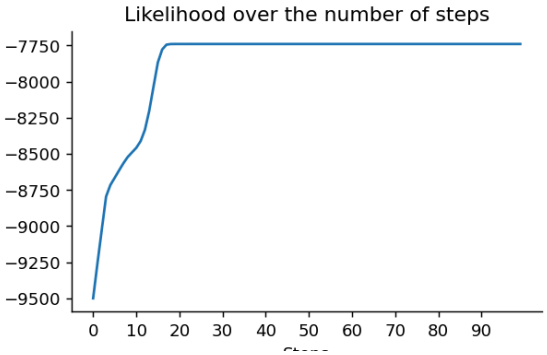
- **E step**: Compute $\gamma_{ij}^t, (i,j) \in [\![1,n]\!] \times [\![1,m]\!]$
- **M step**: Compute $\left(\alpha_j^{t+1}, \mu_j^{t+1}, \Sigma_j^{t+1}\right)$ using the aforementioned formulas

Initialization rule for the parameters

The initialization rule used here comes from this note by Prof. Thomas Bonald (Télécom Paris)

Let m the number of clusters, n the number of samples. The initialization procedure is as follows

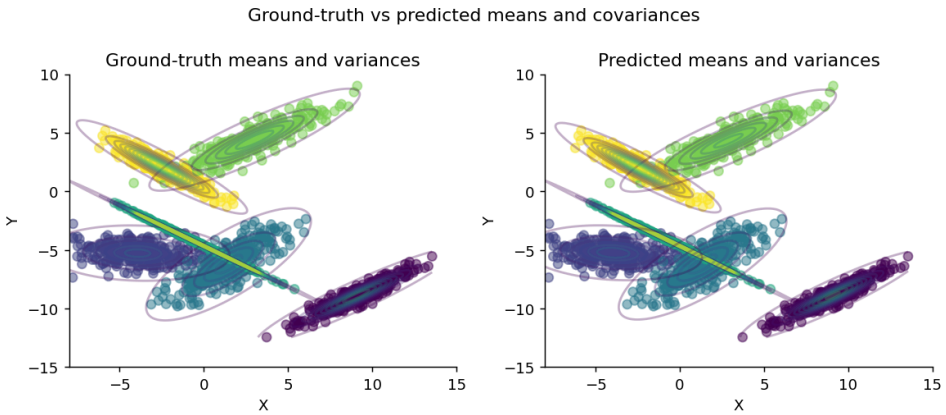
$\overline{x} \leftarrow \frac{1}{n}\sum_{i=1}^{n} x_i$
$\sigma^2 \leftarrow \frac{1}{n}\sum_{i=1}^{n} \|x_i - \overline{x}\|^2$
for $j = 1$ **to** m **do**
    $\Sigma_j \leftarrow \frac{\sigma^2}{m} I$
    $\alpha_j \leftarrow \frac{1}{m} I$



We see that the log-likelihood increases and then becomes steady. It shows that the algorithm has converged to a local maximum.

Note: We know for sure that it's a local maximum, and not the global one. If it was a global maximum, the likelihood would be 1 and the log-likelihood would be 0.

4)



Qualitatively speaking, we observe that predictions are really close to the ground-truth distribution. Ellipses correspond to the level lines of a multivariate Gaussian distribution generated for each cluster with the ground-truth parameters (left figure) and with the predicted ones.

We summarize in the table below the results obtained with the EM algorithm

	Ground-truth	Predicted
Cluster 1	$(-3.93, -5.27)$	$(-3.95, -5.24)$
Cluster 2	$(1.47, -6.22)$	$(1.35, -6.20)$
Cluster 3	$(-0.37, -4.41)$	$(-0.37, -4.40)$
Cluster 4	$(3.01, 4.41)$	$(2.88, 4.33)$
Cluster 5	$(-2.49, 2.09)$	$(-2.48, 2.02)$
Cluster 6	$(9.04, -8.93)$	$(9.08, -8.89)$

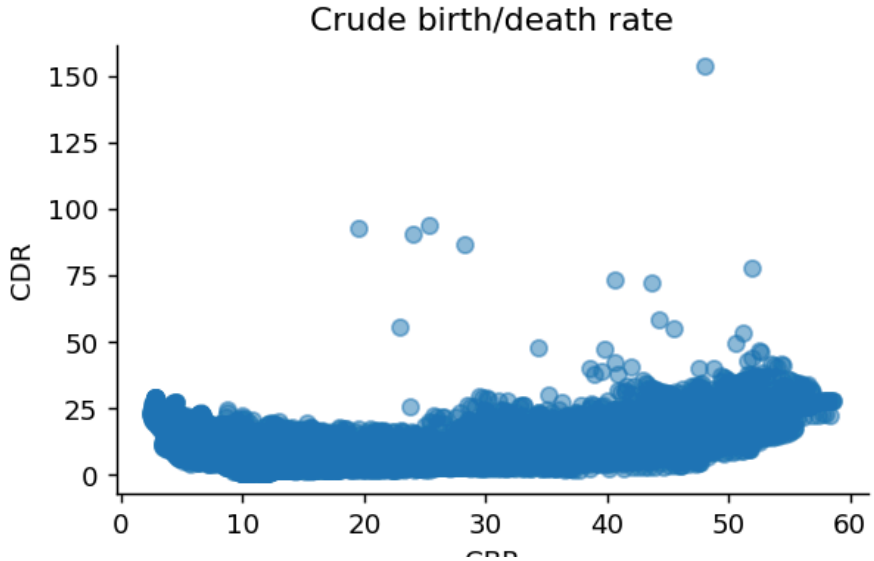
Table 1: Ground-truth and predicted μ_j

	Ground-truth	Predicted
Cluster 1	0.167	0.155
Cluster 2	0.167	0.160
Cluster 3	0.167	0.163
Cluster 4	0.167	0.161
Cluster 5	0.167	0.194
Cluster 6	0.167	0.167

Table 2: Ground-truth and predicted α_j

For the sake of readability, we don't show here the ground-truth against predicted covariance matrices. The latter can be found in the notebook. However, given the previous qualitative and quantitative information provided above, predicted covariance matrices are as well very close to the true ones.

5)



At first sight, the points distribution doesn't look like a mixture. Indeed, we can't distinguish any cluster, even less a Gaussian-like structure.

Given the points distribution, the Gaussian Mixture Model seems not to be adapted to this dataset.

6)

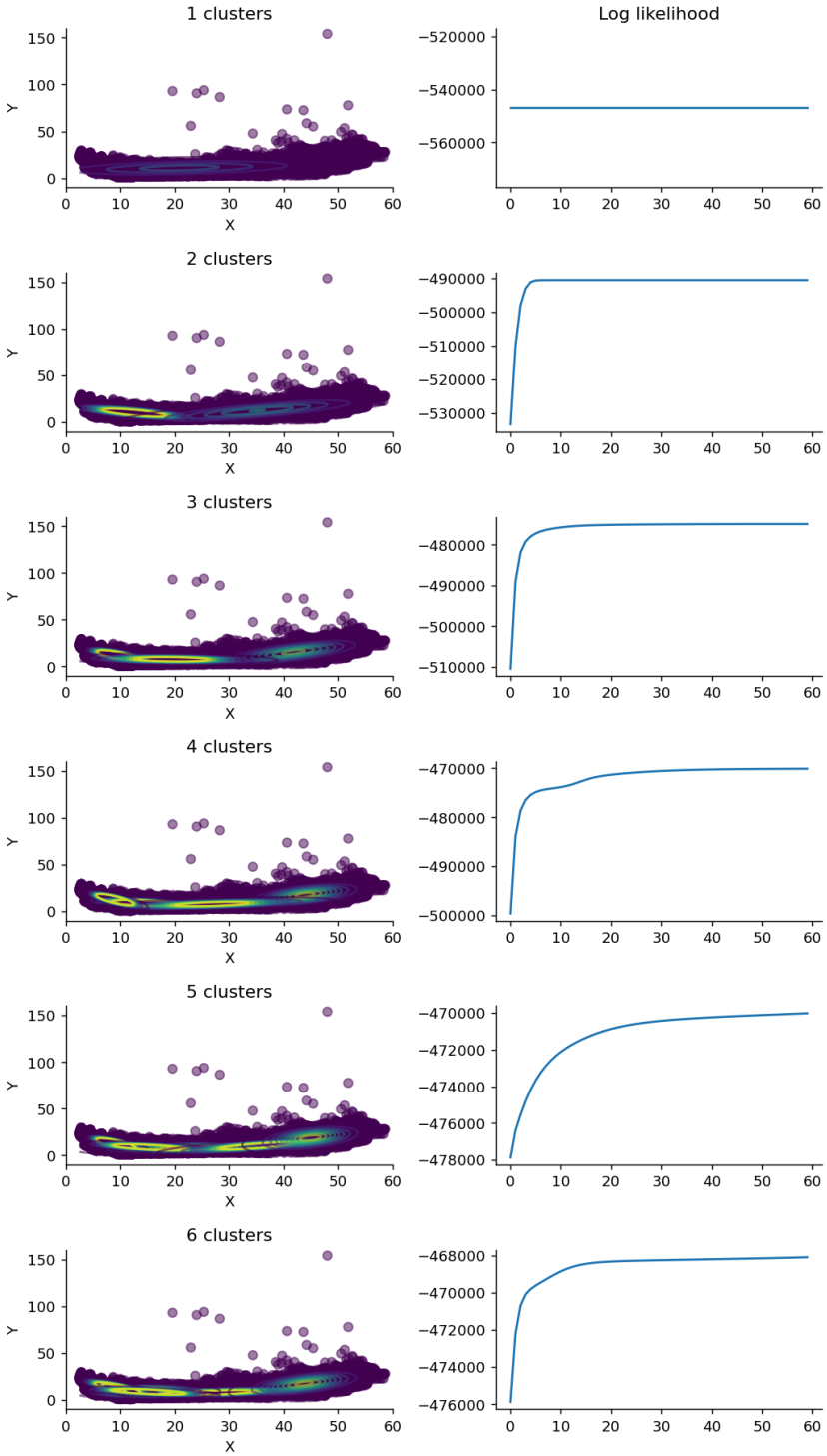


Figure 6: Convergence results with an increasing number of clusters. The left column shows the distribution of multivariate normal distributions generated using the estimated parameters μ_j and Σ_j. The right column shows the evolution of the log-likelihood with respect to the number of steps.

- For $m = 1$ cluster, the log likelihood is steady across learning steps. This illustrates the fact that the distribution as one block is not Gaussian. Thus, no matters the initial point (unless it's on the border of the block of points), the learning process is useless.
- When $m > 1$ we notice that Gaussian distributions try to cover the whole point cloud. However, from $m = 4$ and more, we don't see any major improvement in the way points are covered.

Bayesian Information Criterion

As suggested by the subject we use this criterion to find out the right number of clusters. Hence, for each value of m we compute the following quantity:

$$-\log \mathcal{L}(x_1, ..., x_n; \theta) + \frac{\mathrm{df}(m)\log(n)}{2}$$

We need to compute the number of degrees of freedom with respect to m. Let's introduce the following notations:

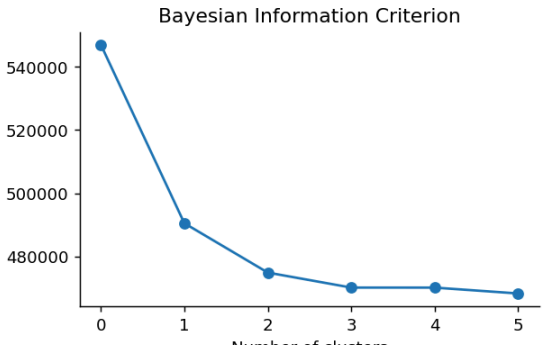
- $d \in \mathbb{N}^*$ the dimension of a sample ($x \in \mathbb{R}^d$)
- $m \in \mathbb{N}^*$ the number of clusters (already introduced)

Degrees of freedom are the number parameters we must fix to entirely define the model. Given cluster j, there are 3 sources: the mean μ_j, the covariance matrix Σ_j and the weighting coefficient α_j

- μ: each mean has the same dimensionality d as the data.
  So it brings $\mathrm{df}_\mu = m \times d$
- Σ: the only constraint on Σ is that it has to be symmetrical. (The non-negativity of the diagonal doesn't bring additional constraint). As $\Sigma \in \mathbb{R}^{d\times d}$ and $\Sigma^T = \Sigma$ we just have to fix its upper (resp.) lower triangle, diagonal included.
  So $\mathrm{df}_\Sigma = m \times \frac{d(d+1)}{2}$
- α is a real non negative value with the constraint that $\sum_{j=1}^{m} \alpha_j = 1$. So we can fix $m-1$ values of α_j.
  So $\mathrm{df}_\alpha = m - 1$

We finally have

$$\begin{aligned}\mathrm{df}(m) &= \mathrm{df}_\mu + \mathrm{df}_\Sigma + \mathrm{df}_\alpha \\ &= \frac{m}{2}(d+1)(d+2) - 1 \qquad \text{After simplification}\end{aligned}$$



The BIC seems to be a decreasing sequence with respect to m, we no clear minimum.

In this case, the BIC doesn't bring any help to find the optimal number of clusters. What's more, increasing more the number of clusters may lead to over-fitting .

Exercise 3: Importance sampling

1)

See notebook

2)

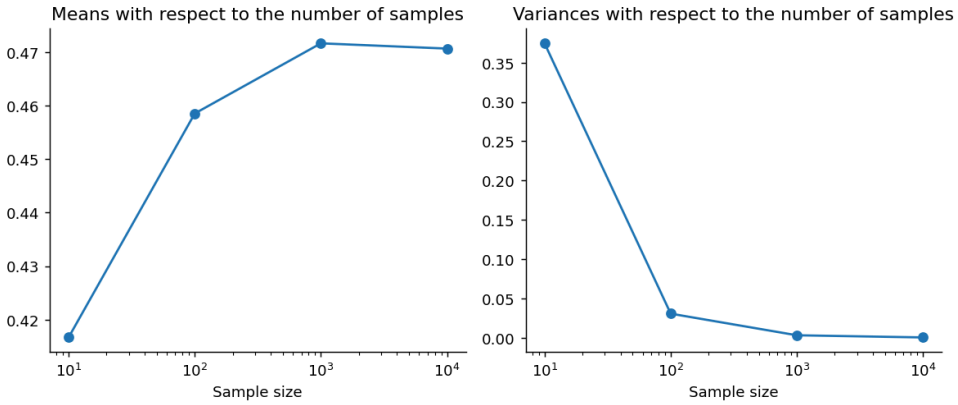


Figure 8: Evolution of mean and variance of the estimate $\mathbb{E}_p[f(X)]$, with $\mu = 0.8$ for q. For each sample size, the experiment was run $n = 500$ times.

We notice that the mean estimate converges to a value corresponding to the true value of $\mathbb{E}_p[f(X)]$. We also notice that the variance naturally decreases as the number of samples increases.

3)

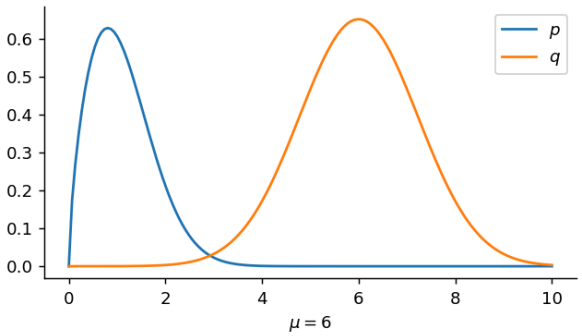


Figure 9: Plot of the function q shifted with $\mu = 6$

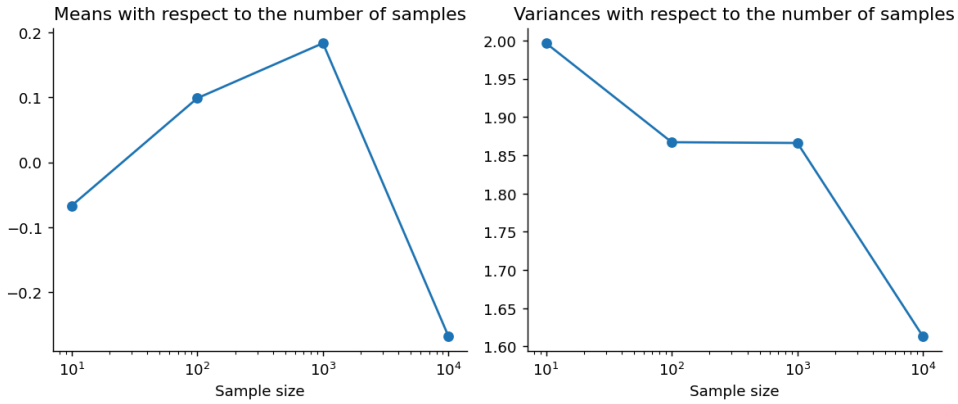


Figure 10: Evolution of mean and variance of the estimate $\mathbb{E}_p[f(X)]$, with $\mu = 6$ for q. For each sample size, the experiment was run $n = 500$ times.

Here we notice that the variance decreases a lot slower than with $\mu = 0.8$. What's more the value of the mean is not comparable with the one obtained from previous experiments.

These plots highlights the negative effects of having p and q highly misaligned.

Importance weights analysis

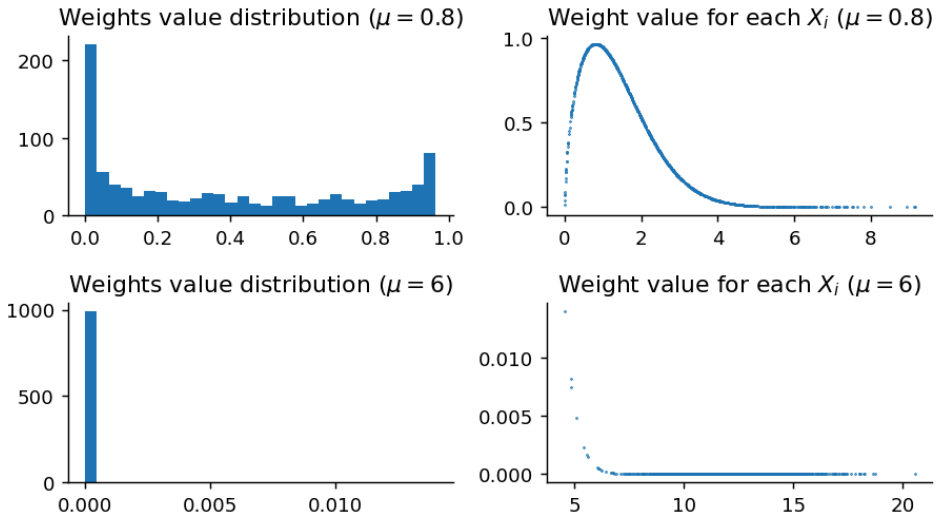


Figure 11: Differences in weights values and distribution induced by the shift of q.

For $\mu = 0.8$ we notice that most weights are non-zero, where a vast majority of them are null with $\mu = 6$. In the latter case, the estimation of $\mathbb{E}[f(X)]$ relies on just a few samples X_i, hence explaining the high variance.
What's more, as intuitively expected, importance weights have a higher value around the high values of f (i.e the ones contributing the most to the value of the expectation.) However, because p and q barely overlap when $\mu = 6$, the few non zero weights correspond to the narrow region of overlapping. So it only takes into account the regions where f has the most mass.

4)

Adaptative Importance sampling
We notice that the provided expression is a weighted version of the log-likelihood we tried to maximize in the regular version of the Gaussian Mixture Model.

$$\begin{aligned}\sum_{i=1}^{n} \tilde{\omega}_i \log\left(\sum_{j=1}^{M} \alpha_j \varphi(X_i;\theta_j)\right) &= \log\left(\prod_{i=1}^{n}\left(\sum_{j=1}^{M} \alpha_j \varphi(X_i;\theta_j)\right)^{\tilde{\omega}_i}\right) \\ &= \log\left(\prod_{i=1}^{n} f_\theta(x_i)^{\tilde{\omega}_i}\right) \qquad \text{With the notations of exercise 2}\end{aligned}$$

Using the concavity of log and the definition of $Q(\theta|\theta^t)$ as a lower bound of the likelihood, we can write

$$\begin{aligned}\sum_{i=1}^{n} \tilde{\omega}_i \log\left(\sum_{j=1}^{M} \alpha_j \varphi(X_i;\theta_j)\right) &\geq \sum_{i=1}^{n} \tilde{\omega}_i^t \sum_{j=1}^{M} \gamma_{ij}^t \log\left(\alpha_j \varphi(X_i;\theta_j)\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{M} \tilde{\omega}_i^t \gamma_{ij}^t \log\left(\alpha_j \varphi(X_i;\theta_j)\right) \\ &= \tilde{Q}(\theta|\theta^t)\end{aligned}$$

The convergence of the EM algorithm ensures that the sequence $\left(\tilde{Q}(\theta|\theta^t)\right)_t$ will increase to a local maximum. As it's a lower bound of the right hand side expression, it will converge to the same maximum.

The update rules for the parameters are almost the same as before. We just have to replace γ_{ij}^t by $\omega_i^t \gamma_{ij}^t$

$$\alpha_j^{t+1} = \frac{1}{n}\sum_{i=1}^{n} \omega_i^t \gamma_{ij}^t \qquad \mu_j^{k+1} = \frac{\sum_{i=1}^n \omega_i^t \gamma_{ij}^t x_i}{\sum_{i=1}^n \omega_i^t \gamma_{ij}^t} \qquad \Sigma_j^{k+1} = \frac{\sum_{i=1}^n \omega_i^t \gamma_{ij}^t (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^n \omega_i^t \gamma_{ij}^t}$$

The reasoning is widely inspired from these 2 articles:

- Adaptive Importance Sampling in General Mixture Classes
- Convergence of Adaptative Mictures of Importance Sampling Schemes

5)

Application to banana-shaped density

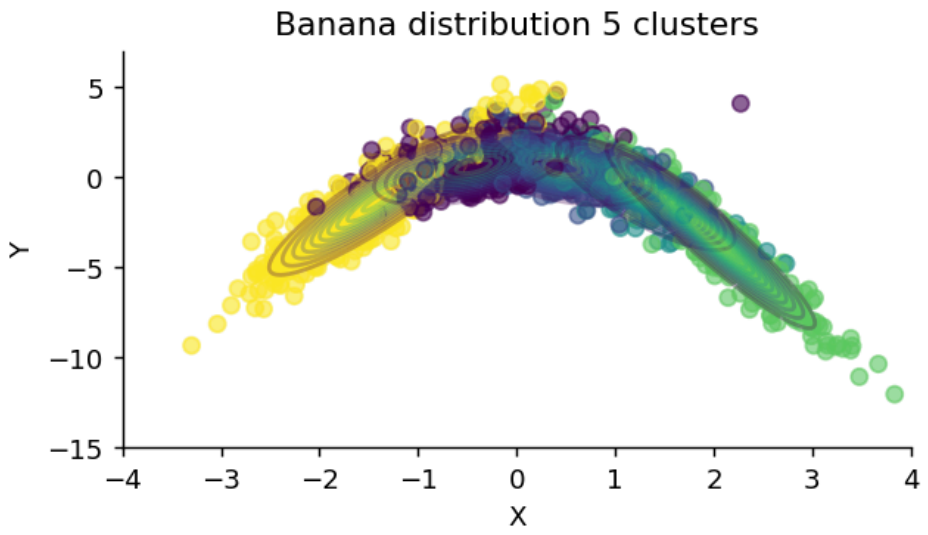


Figure 12: First two dimensions of the estimated banana distribution with $b = 1.0$. At each iteration, 2000 points are sampled and we try to fit 5 clusters. We performed 60 iterations of the algorithm to get this result.

Note: b was fixed to 1.0 because the final result looks more like a banana.
