IFT 6269, Homework 4

Adel Nabli, ID: 20121744

11 Novembre 2018

1 Entropy and Mutual Information:

1. (a) X is a discrete random variable on the finite space $\mathcal{X}$ such that $|\mathcal{X}| = k$. By definition, we can write:

$$H(X) = \mathbb{E}_{p(x)}\big[I(X)\big] = -\sum_{x\,\in\,\mathcal{X}} p(x)\log p(x)$$

As p is a pmf, $\forall\ x\ \in \mathcal{X},\ p(x)\ \in\ [0,1]$. But $x \mapsto x\log(x)$ is negative on $[0,1]$, thus $\forall\ x\ \in \mathcal{X},\ p(x)\log p(x) \leq 0$ and $H(X) \geq 0$.

We have that:

$$\begin{aligned} H(X) = 0 &\Leftrightarrow \sum_{x\,\in\,\mathcal{X}} \underbrace{p(x)\log p(x)}_{\leq 0} = 0 \\ &\Leftrightarrow \forall\ x\ \in\ \mathcal{X},\ p(x)\log p(x) = 0 \\ &\Leftrightarrow \forall\ x\ \in\ \mathcal{X},\ p(x) = 0 \textbf{ or } p(x) = 1 \end{aligned}$$

And as p is a pmf, we also have $\sum_{x\,\in\,\mathcal{X}} p(x) = 1$, which means that $\exists\ !\ x^*\ \in\ \mathcal{X}$ s.t $p(x^*) = 1$ and $\forall\ x \neq x^*,\ p(x) = 0$. Hence, X is a constant.

(b) Let p be the distribution of X and q be the uniform distribution on $\mathcal{X}$. As we know that $|\mathcal{X}| = k$, we can deduce that $\forall\ x\ \in\ \mathcal{X},\ q(x) = \frac{1}{k}$. Thus, we can write:

$$D(p||q) = \sum_{x\,\in\,\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} = \sum_{x\,\in\,\mathcal{X}} p(x)\log p(x) + \underbrace{\sum_{x\,\in\,\mathcal{X}} p(x)}_{=1}\log k = -H(X) + \log(k)$$

(c) **Jensen's inequality:** *$\forall\ f : I \to \mathbb{R}$ convex, $\forall\ k \geq 1$, $(y_1, ..., y_k)\ \in\ I^k$, $\forall\ (t_1, ..., t_k)\ \in\ \mathbb{R}_+^k$ s.t $\sum_i t_i = 1$, we have $f(\sum_i t_i y_i) \leq \sum_i t_i f(y_i)$.*

Thus, by using $f = -\log$, $y_i = \frac{q(x_i)}{p(x_i)}$ and $t_i = p(x_i)$, we have:

$$D(p||q) = \sum_{x\,\in\,\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} \geq \log\Big(\sum_{x\,\in\,\mathcal{X}} p(x)\frac{q(x)}{p(x)}\Big) = \log\Big(\sum_{x\,\in\,\mathcal{X}} q(x)\Big) = 0$$

But as we know that $D(p||q) = \log(k) - H(X)$, we deduce that $H(X) \leq \log(k)$.

2. (a) If we define $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$ (which is a finite set), $p = p_{1,2}$ (which is a pmf on $\mathcal{X}$), $q : (x_1, x_2) \mapsto p_1(x_1)p_2(x_2)$ (which is a pmf on $\mathcal{X}$), we have that $I(X_1, X_2) = D(p||q)$. Thus, by using the fact that $\forall\ p, q\ ,\ D(p||q) \geq 0$ (property of the KL divergence prooven in question 1.c), we deduce that $I(X_1, X_2) \geq 0$.

(b) We have:

$$\begin{aligned}
I(X_1, X_2) &= \sum_{(x_1,x_2)\,\in\,\mathcal{X}_1\times\mathcal{X}_2} p_{1,2}(x_1,x_2)\log\frac{p_{1,2}(x_1,x_2)}{p_1(x_1)p_2(x_2)} \\
&= \underbrace{\sum_{(x_1,x_2)\,\in\,\mathcal{X}_1\times\mathcal{X}_2} p_{1,2}(x_1,x_2)\log p_{1,2}(x_1,x_2)}_{=-H(X_1,X_2)} \\
&\quad - \sum_{(x_1,x_2)\,\in\,\mathcal{X}_1\times\mathcal{X}_2} p_{1,2}(x_1,x_2)\log p_1(x_1) - \sum_{(x_1,x_2)\,\in\,\mathcal{X}_1\times\mathcal{X}_2} p_{1,2}(x_1,x_2)\log p_2(x_2) \\
&= -H(X_1,X_2) - \sum_{x_1\,\in\,\mathcal{X}_1}\log p_1(x_1)\underbrace{\sum_{x_2\,\in\,\mathcal{X}_2} p_{1,2}(x_1,x_2)}_{=p_1(x_1)} - \sum_{x_2\,\in\,\mathcal{X}_2}\log p_2(x_2)\underbrace{\sum_{x_1\,\in\,\mathcal{X}_1} p_{1,2}(x_1,x_2)}_{=p_2(x_2)} \\
&= H(X_1) + H(X_2) - H(X_1,X_2)
\end{aligned}$$

(c)
- We want to maximize $H(X_1, X_2)$ given p_1 and p_2.
- As $H(X_1, X_2) = H(X_1) + H(X_2) - I(X_1, X_2)$, this is equivalent to minimizing $I(X_1, X_2)$ ($H(X_1)$ and $H(X_2)$ being given).
- But we know by 2.a that $I(X_1, X_2) \geq 0$, which means that $I(X_1, X_2)$ is lower-bounded by 0.
- Given that, by definition, $I(X_1, X_2) = \sum_{(x_1,x_2)\,\in\,\mathcal{X}_1\times\mathcal{X}_2} p_{1,2}(x_1,x_2)\log\frac{p_{1,2}(x_1,x_2)}{p_1(x_1)p_2(x_2)}$, we can say that the case $I(X_1, X_2) = 0$ is attainable i.i.f $\forall\ x_1, x_2,\ p_{1,2}(x_1,x_2) = p_1(x_1)p_2(x_2)$ ($\Leftrightarrow X_1 \perp\!\!\!\perp X_2$) (*case of equality in the Jensen's inequality*)
- Thus, the joint distribution of maximum entropy is $p_{1,2} = p_1 p_2$.

2 HMM – Implementation:

1. In this question, we will try to rewrite the asked probabilities using "known" quantities.
   - As we know that $\forall\ t\ \in [\![1, T-1]\!],\ x_{1:t} \perp\!\!\!\perp x_{t+1:T}|z_t$ by HMM property, we can write:

$$\begin{aligned}
p(z_t|x_{1:T}) &= \frac{p(x_{1:T}|z_t)p(z_t)}{p(x_{1:T})} = \frac{p(x_{1:t}|z_t)\overbrace{p(x_{t+1:T}|z_t)}^{=\beta(z_t)}p(z_t)}{p(x_{1:T})} = \frac{\overbrace{p(x_{1:t}, z_t)}^{=\alpha(z_t)}\beta(z_t)}{p(x_{1:T})} \\
&= \frac{\alpha(z_t)\beta(z_t)}{\sum_{z_t} p(x_{1:T}, z_t)} = \frac{\alpha(z_t)\beta(z_t)}{\sum_{z_t} p(x_{1:T}, z_t)} = \frac{\alpha(z_t)\beta(z_t)}{\sum_{z_t}\alpha(z_t)\beta(z_t)}
\end{aligned}$$

   - To compute $p(z_t, z_{t+1}|x_{1:T})$, we will use several other HMM properties:

$$\begin{aligned}
p(z_t, z_{t+1}|x_{1:T}) &= \frac{p(x_{1:t}, x_{t+1:T}, z_t, z_{t+1})}{p(x_{1:T})} \\
(x_{1:t} \perp\!\!\!\perp x_{t+1:T}|z_t, z_{t+1}) \rightarrow &= \frac{p(x_{1:t}|z_t, z_{t+1})p(x_{t+1:T}|z_t, z_{t+1})p(z_t, z_{t+1})}{p(x_{1:T})} \\
(x_{1:t} \perp\!\!\!\perp z_{t+1}|z_t) \rightarrow &= \frac{\overbrace{p(x_{1:t}|z_t)p(z_t)}^{=\alpha(z_t)}\,p(x_{t+1}, x_{t+2:T}|z_t, z_{t+1})\,\overbrace{p(z_{t+1}|z_t)}^{=A_{z_{t+1},z_t}}}{p(x_{1:T})} \\
(x_{t+1} \perp\!\!\!\perp x_{t+2:T}|z_t, z_{t+1}) \rightarrow &= \frac{\alpha(z_t)A_{z_{t+1},z_t}p(x_{t+2:T}|z_t, z_{t+1})p(x_{t+1}|z_t, z_{t+1})}{p(x_{1:T})}
\end{aligned}$$

$$(x_{t+2:T} \perp\!\!\!\perp z_t | z_{t+1}) \rightarrow = \frac{\alpha(z_t) A_{z_{t+1},z_t} \overbrace{p(x_{t+2:T}|z_{t+1})}^{=\beta(z_{t+1})} p(x_{t+1}|z_t, z_{t+1})}{p(x_{1:T})}$$

$$(x_{t+1} \perp\!\!\!\perp z_t | z_{t+1}) \rightarrow = \frac{\alpha(z_t) A_{z_{t+1},z_t} \beta(z_{t+1}) p(x_{t+1}|z_{t+1})}{\sum_{z_t} \alpha(z_t)\beta(z_t)}$$

And we know that $x_{t+1}|z_{t+1} = k \sim \mathcal{N}(x_{t+1}|\mu_k, \Sigma_k)$ *(in class, we used the notation $o(z_t)$ to refer to $p(x_t|z_t)$).*

Now, we can go back to the initial question which was to implement the α and β-recursions to compute the 2 quantities. As we know by taking what we've seen in class, we have:

$$\begin{cases} \alpha(z_1) = p(z_1)p(x_1|z_1) = \pi o(z_1),\ \forall t \in [\![1, T-1]\!],\ \alpha(z_{t+1}) = o(z_{t+1}) \sum_{z_t} A_{z_{t+1},z_t} \alpha(z_t) \\ \beta(z_T) = 1,\ \forall t \in [\![1, T-1]\!],\ \beta(z_t) = \sum_{z_{t+1}} A_{z_{t+1},z_t} o(z_{t+1}) \beta(z_{t+1}) \end{cases}$$

But, as we can see, there might be an underflow problem if we compute the values of α and β directly from these formulas as we are multiplying values smaller than one recursively. Two methods were presented in class to overcome this problem (a log trick and normalization). We will use here the normalized versions of α and β. Thus, we define the quantities:

$$\tilde{\alpha}(z_t) = \frac{\alpha(z_t)}{p(x_{1:t})} \quad ; \quad \tilde{\beta}(z_t) = \frac{\beta(z_t)}{p(x_{t+1:T}|x_{1:t})} \quad ; \quad c_t = p(x_t|x_{1:t-1})$$

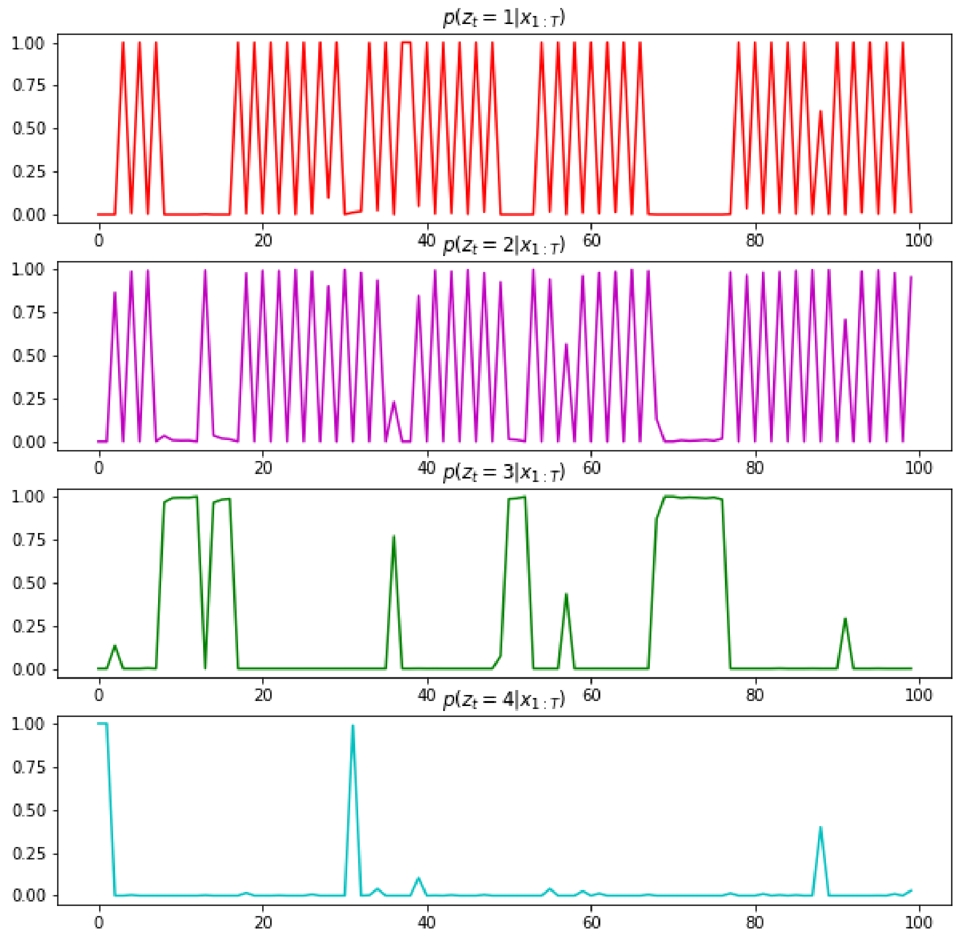
As whe have the product rule $\forall\ t \in\ [\![1, T-1]\!],\ p(x_{1:t}) = \prod_{k=1}^{t} c_k$, we can derive the new recursions by recurrence:

$$\begin{cases} \tilde{\alpha}(z_1) = \pi o(z_1),\ \forall t \in [\![1, T-1]\!],\ \tilde{\alpha}(z_{t+1}) = \dfrac{o(z_{t+1}) \sum_{z_t} A_{z_{t+1},z_t} \tilde{\alpha}(z_t)}{c_{t+1}} \\ \textit{with } c_{t+1} = \sum_{z_{t+1}} o(z_{t+1}) \sum_{z_t} A_{z_{t+1},z_t} \tilde{\alpha}(z_t) \\ \tilde{\beta}(z_T) = 1,\ \forall t \in [\![1, T-1]\!],\ \tilde{\beta}(z_t) = \dfrac{\sum_{z_{t+1}} A_{z_{t+1},z_t} o(z_{t+1}) \tilde{\beta}(z_{t+1})}{c_{t+1}} \end{cases}$$

By computing and storing the c_t alongside the $\tilde{\alpha_t}$ during a forward pass, we can compute the $\tilde{\beta_t}$ during a backward pass *(contrary to the un-normalized recursion, we can't compute the backward pass independently of the forward pass as we need the values of the c_t).*
Using these new values, we now have to re-compute the probabilities $p(z_t|x_{1:T})$ and $p(z_t, z_{t+1}|x_{1:T})$ which leads to:

$$\begin{cases} p(z_t|x_{1:T}) = \dfrac{\alpha(z_t)\beta(z_t)}{p(x_{1:T})} = \dfrac{\tilde{\alpha}(z_t) \overbrace{\cancel{p(x_{1:t})p(x_{t+1:T}|x_{1:t})}}^{=p(x_{1:T})} \tilde{\beta}(z_t)}{\cancel{p(x_{1:T})}} = \tilde{\alpha}(z_t)\tilde{\beta}(z_t) \\ p(z_t, z_{t+1}|x_{1:T}) = \dfrac{\alpha(z_t) A_{z_{t+1},z_t} \beta(z_{t+1}) o(z_{t+1})}{p(x_{1:T})} = \dfrac{A_{z_{t+1},z_t} \tilde{\alpha}(z_t) \cancel{p(x_{1:t})p(x_{t+2:T}|x_{1:t+1})} \tilde{\beta}(z_{t+1}) o(z_{t+1})}{\cancel{p(x_{1:t})} \underbrace{p(x_{t+1}|x_{1:t})}_{=c_{t+1}} \cancel{p(x_{t+2:T}|x_{1:t+1})}} \end{cases}$$

2. After implementation (cf *Notebook*), we plotted $p(z_t|x_{1:T})$ for each of the 4 states for the first 100 time steps in the test set:



3. We have $\theta = (\pi, A, (\mu_k, \Sigma_k)_{k\in[\![1,4]\!]})$. We write the complete log-likelihood $l(z, x|\theta)$:

$$l(x, z, |\theta) = \log\left(p(z_1) \prod_{t=1}^{T-1} p(z_{t+1}|z_t) \prod_{t=1}^{T} p(x_t|z_t)\right)$$
$$= \sum_{k=1}^{4} \delta(z_1 = k)\log(\pi_k) + \sum_{t=1}^{T-1}\sum_{i=1}^{4}\sum_{j=1}^{4} \delta(z_{t+1} = i, z_t = j)\log(A_{ij})$$
$$+ \sum_{t=1}^{T}\sum_{k=1}^{4} \delta(z_t = k)\Big(-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(|\Sigma_k|) - \frac{1}{2}(x_t - \mu_k)^T\Sigma_k^{-1}(x_t - \mu_k)\Big)$$

We want to find $\underset{\theta}{\text{argmax}}\ \mathbb{E}_{z|x}\big[l(z, x|\theta)\big]$ (E-M algorithm).

E step: We compute $\mathbb{E}_{z|x}\big[l(z, x|\theta)\big]$ with θ fixed. We have:

$$\mathbb{E}_{z|x}\big[\delta(z_1 = k)\big] = p(z_1 = k|x, \theta)$$
$$\mathbb{E}_{z|x}\big[\delta(z_t = k)\big] = p(z_t = k|x, \theta)\ (smoothing\ distribution)$$
$$\mathbb{E}_{z|x}\big[\delta(z_{t+1} = i, z_t = j)\big] = p(z_{t+1} = i, z_t = j|x, \theta)\ (pair - marginals)$$

Thus, we already computed all the values necessary to the E-step in question 2.1.

M step: We maximize $\mathbb{E}_{z|x}\big[l(z,x|\theta)\big]$ with respect to θ to find $\hat{\theta}$:

- **For** π: We have the condition $\sum_{k=1}^4 \pi_k = 1$ and we know that $x \mapsto \log(x)$ is concave. Thus, we can write the Lagrangian function:

$$\mathcal{L}(\pi,\lambda) = \sum_{k=1}^{4} p(z_1 = k|x)\log(\pi_k) + \lambda\Big(1 - \sum_{k=1}^{4}\pi_k\Big)$$

  Thus, we have:

$$\forall\, k \in [\![1,4]\!], \frac{\partial \mathcal{L}(\pi,\lambda)}{\partial \pi_k} = 0 \Leftrightarrow \lambda\pi_k = p(z_1 = k|x)$$

  Having $\sum_{k=1}^4 \pi_k = 1$ leads to $\lambda \underbrace{\sum_{k=1}^{4}\pi_k}_{=1} = \underbrace{\sum_{k=1}^{4} p(z_1 = k|x)}_{=1}$ which means that:

$$\forall\, k \in [\![1,4]\!], \hat{\pi}_k = p(z_1 = k|x)$$

- **For A**: As A is a stochastic matrix, we have $\forall\, j \in [\![1,4]\!], \sum_{i=1}^4 A_{ij} = 1$. We again use the fact that log is a concave function to justify the use of the Lagrangian method on every A_j. Hence, we can write:

$$\forall\, j \in [\![1,4]\!],\ \mathcal{L}(A_j,\lambda) = \sum_{t=1}^{T-1}\sum_{i=1}^{4} p(z_{t+1} = i, z_t = j|x)\log(A_{ij}) + \lambda\Big(1 - \sum_{i=1}^{4} A_{ij}\Big)$$

$$\forall\, i \in [\![1,4]\!],\ \frac{\partial \mathcal{L}(A_j,\lambda)}{\partial A_{ij}} = 0 \Leftrightarrow \lambda A_{ij} = \sum_{t=1}^{T-1} p(z_{t+1} = i, z_t = j|x)$$

  Having $\sum_{i=1}^4 A_{ij} = 1$ leads to $\lambda = \sum_{i=1}^4 \sum_{t=1}^{T-1} p(z_{t+1} = i, z_t = j|x)$ and then:

$$\forall\, i,j \in [\![1,4]\!]^2,\ \hat{A}_{ij} = \frac{\sum_{t=1}^{T-1} p(z_{t+1} = i, z_t = j|x)}{\sum_{i=1}^4 \sum_{t=1}^{T-1} p(z_{t+1} = i, z_t = j|x)} = \frac{\sum_{t=1}^{T-1} p(z_{t+1} = i, z_t = j|x)}{\sum_{t=1}^{T-1} p(z_t = j|x)}$$

- **For** μ **and** Σ: We can directly take the derivations we've already made in Homework 2 with the QDA model as the only change with regards to the μ_k and Σ_k is that we have here $k \in [\![1,4]\!]$ instead of having $k \in \{1,2\}$. Adapting the formulas to our case leads to:

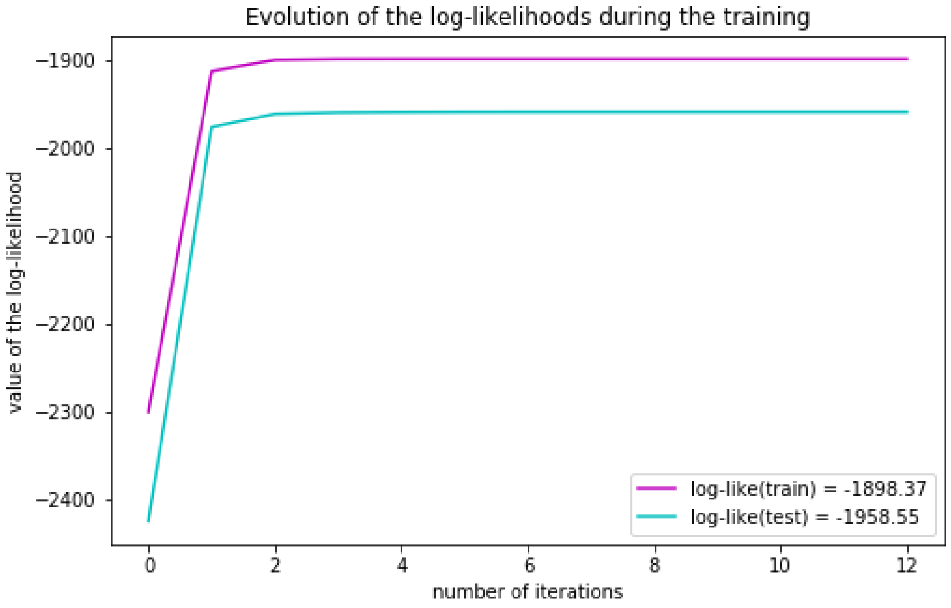
$$\forall\, k \in [\![1,4]\!],\ \hat{\mu}_k = \frac{\sum_{t=1}^{T} x_t p(z_t = k|x)}{\sum_{t=1}^{T} p(z_t = k|x)} \quad ; \quad \hat{\Sigma}_k = \frac{\sum_{t=1}^{T}(x_t - \hat{\mu}_k)(x_t - \hat{\mu}_k)^T p(z_t = k|x)}{\sum_{t=1}^{T} p(z_t = k|x)}$$

4. After having implemented the E-M algorithm (cf *Notebook*), the values of the parameters learnt on the train set are:

$$\pi = \begin{pmatrix} 2.497e-18 \\ 9.584e-89 \\ 3.435e-31 \\ 1.000e+00 \end{pmatrix} \quad A = \begin{pmatrix} 0.016 & 0.930 & 0.042 & 0.007 \\ 0.874 & 0.026 & 0.045 & 0.073 \\ 0.047 & 0.011 & 0.879 & 0.020 \\ 0.063 & 0.033 & 0.034 & 0.900 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -1.959 \\ 4.190 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} 3.994 \\ 3.650 \end{pmatrix} \quad \mu_3 = \begin{pmatrix} 3.790 \\ -3.964 \end{pmatrix}$$

$$\mu_4 = \begin{pmatrix} -2.972 \\ -3.449 \end{pmatrix} \quad \Sigma_1 = \begin{pmatrix} 2.904 & 0.207 \\ 0.207 & 2.756 \end{pmatrix} \quad \Sigma_2 = \begin{pmatrix} 0.210 & 0.290 \\ 0.290 & 12.239 \end{pmatrix} \quad \Sigma_3 = \begin{pmatrix} 0.921 & 0.057 \\ 0.057 & 1.866 \end{pmatrix} \quad \Sigma_4 = \begin{pmatrix} 6.241 & 6.050 \\ 6.050 & 6.183 \end{pmatrix}$$

5. By definition, the log-likelihood is $\log p_\theta(x)$. In our case, as we defined $\forall\, t \in [\![1, T-1]\!]$, $c_t = p(x_t|x_{1:t-1})$, we simply have $\log p_\theta(x) = \log\big(p(x_{1:T})\big) = \sum_{t=2}^{T} \log(c_t)$.



We see that, with the results of the GMM as inputs, only 3 steps are necessary to the E-M algorithm to converge. We also have that the log-likelihood on the training data is higher than the one on the test data, which can mean that our model doesn't overfit.

6. By taking the results of the previous homework, we have:

Models / Sets	**Train set**	**Test set**
isotropic GMM	-2646	-2692
general GMM	-2328	-2409
HMM	-1898	-1959

Log-likelihood on both the training and test sets for the models tested

Thus, we can clearly see that the highest log-likelihood is obtained with the HMM on both the train and test sets. Indeed, the HMM is the model with the highest capacity amoung those tested (the HMM being a generalisation of the general GMM and the isotropic GMM being a special case of the general one (*this fact also means that comparing the log-likelihoods here makes sense*)), which means that it has more *degrees of freedom* we can tune to fit the training data. But this fact alone doesn't allow the HMM to perform better on the test set: indeed, if the data was truly generated by a GMM, then we would have expected the HMM to perform as well as the GMM on the test set or even worse (it could have overfitted). This leads us to think that the data on which we are testing our algorithms was truly generated by a HMM and not a GMM.

7. The Viterbi algorithm allows us to compute the sequence of hidden states that are jointly most probable (in general, it doesn't give the same result as giving the sequence of hidden states that are individualy most probable). Thus, we want to compute $\underset{z}{\text{argmax}}\, p(z|x)$. To do that, and to prevent underflow, wee can compute instead $\underset{z}{\text{argmax}} \log\big(p(z|x)\big)$ as log is an increasing function. To prevent us from computing through an exponentionally growing number of combinations , we can use exactly the same tricks we already used in the α-β *recursions* (instead of implementing a *sum-product* algorithm, we implement a *max-sum* one). For the forward step, we want to compute the probability of

the most probable sequence. Thus, we just have to re-write the α-recursion, changing all the "sums" with "maximums", all the "products" with "sums", and all the probabilities with log-probabilities. We also have to store the values of z_t that maximizes the message at z_{t+1} (we do that using a variable named *memory*).

This leads us to consider the following recursion (we named the "messages" that passes γ):

$$\gamma(z_1) = \log p(z_1) + \log p(x_1|z_1), \ \forall t \in [\![1, T-1]\!], \ \gamma(z_{t+1}) = \log p(x_{t+1}|z_{t+1}) + \max_{z_t} \Big(\log A_{z_{t+1},z_t} + \gamma(z_t) \Big)$$

$$memory(z_{t+1}) = \underset{z_t}{\text{argmax}} \Big(\log A_{z_{t+1},z_t} + \gamma(z_t) \Big)$$

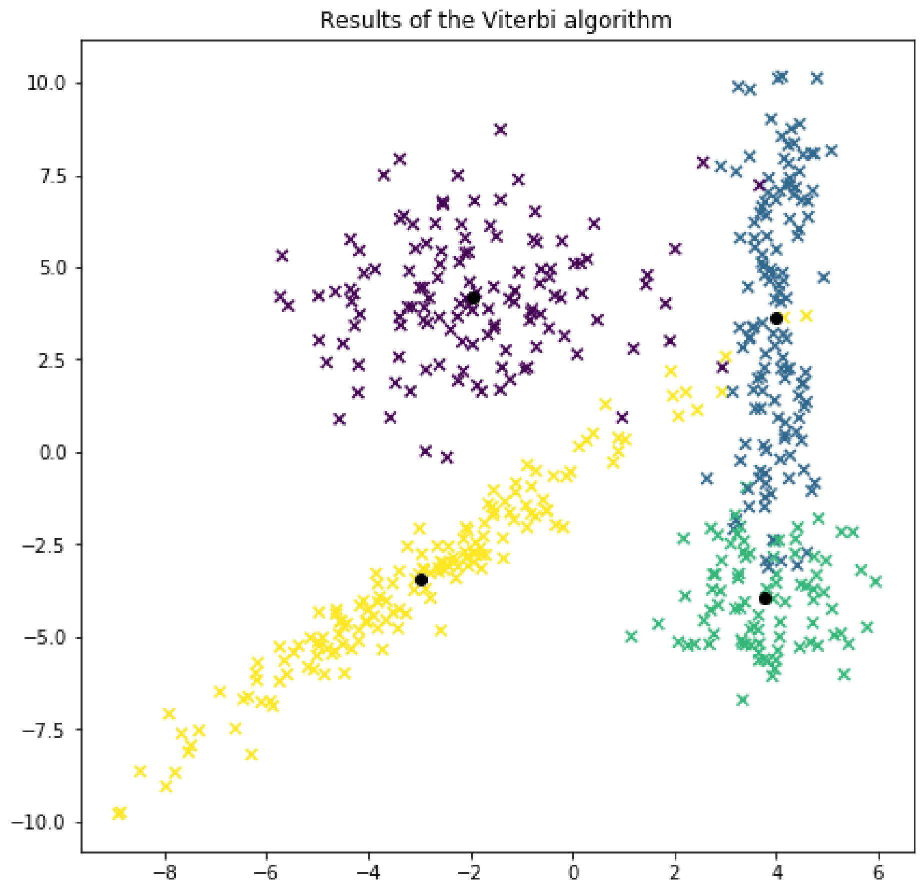
For the backward step, we want to find the states z_t that have built those messages:

$$z_T^* = \underset{z_T}{\text{argmax}} \gamma(z_T), \ \forall t \in [\![1, T-1]\!], z_t^* = memory(z_{t+1}^*)$$

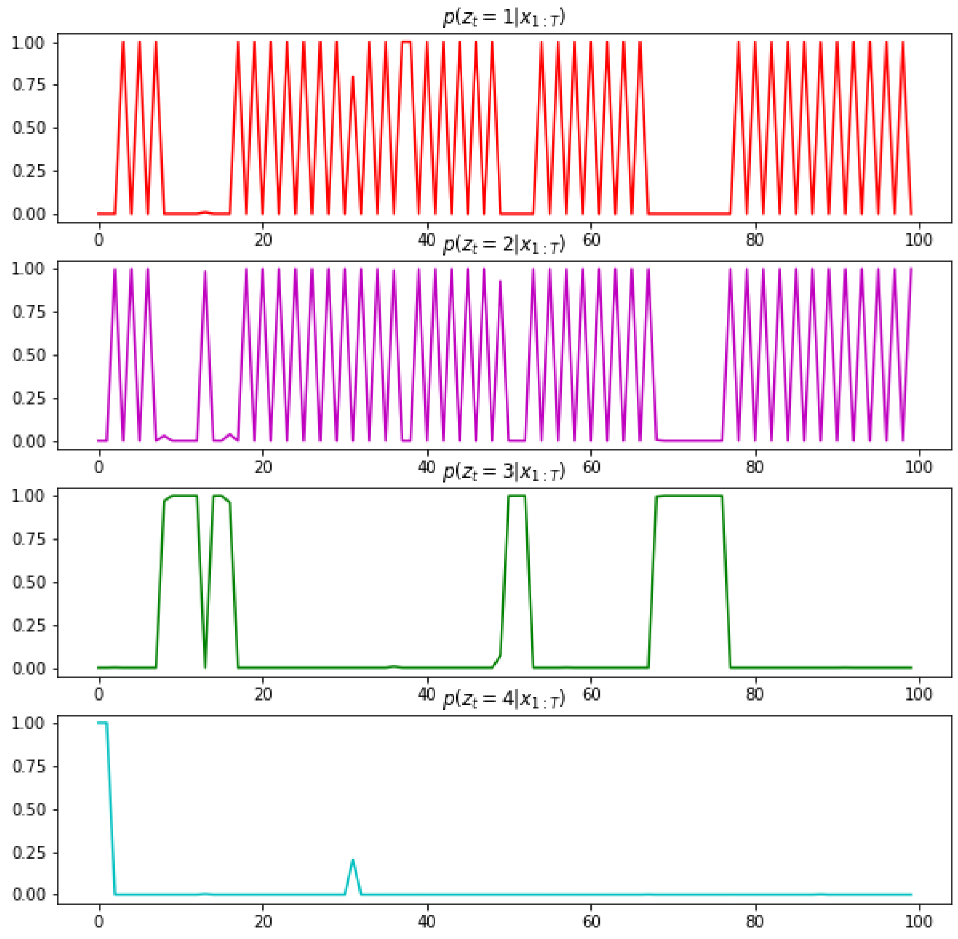
Hence, the pseudo-code of the Viterbi algorithm is:

- **Initialize** the variables γ and *memory*
- **Compute** $\gamma(z_1)$
- **For** $t \in [\![1, T-1]\!]$ **compute** $\gamma(z_{t+1})$ and $memory(z_{t+1})$
- **Initialize** the variable z^*
- **Compute** z_T^*
- **For** $t \in \{T-1, ..., 1\}$ **compute** z_t^*
- **Return** z^*

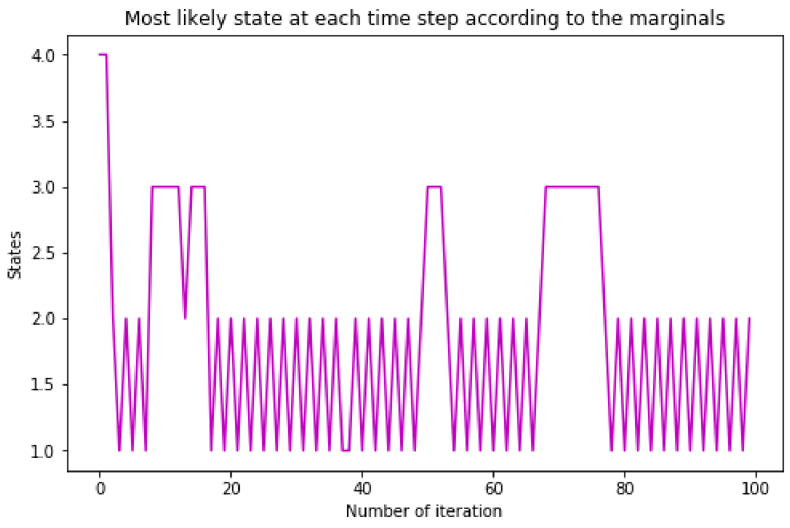
8. After implementation (cf *Notebook*) that's the result we get by running the Viterbi algorithm:



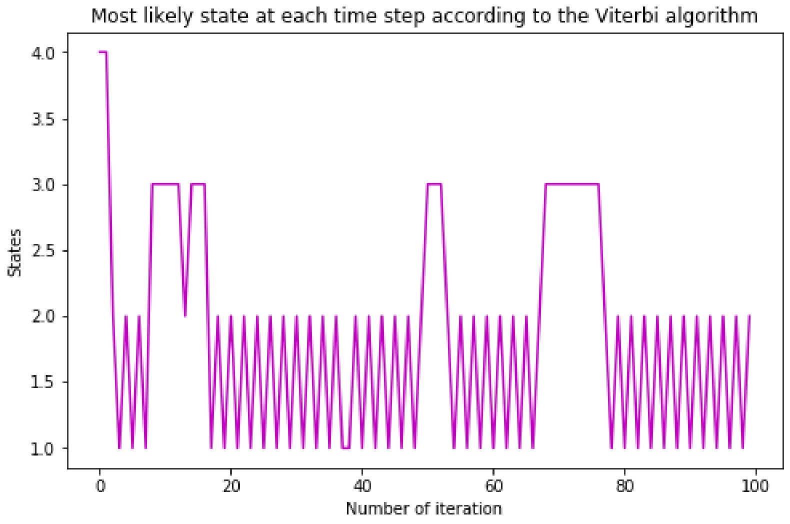
9. After implementation (cf *Notebook*) that's the graphic we get for the first 100 time steps on the test set:



10. By taking $\forall\, t \in [\![1, 100]\!], \underset{z_t}{\operatorname{argmax}}\, p(z_t|x_{test})$ we produce this graph:



11. this time, we compute the most likely states on the test data using the Viterbi algorithm. That leads to the following graph:



We can see that the Viterbi algorithm has produced exactly the same sequence of states as the maximization of the marginals. As we've said earlier in the question 2.7, this is not compulsery in general, but it seems to be the case here.

12. If I didn't know the number K in advance, one strategy would have been to treat K as an hyper-parameter and take the value of K that maximizes the likelihood on the test-set.
An other strategy would have been to maximize with respect to K a function mixing some kind of measure of the between and within class coherence *(**ex:** we could have tried to maximize the average of the silouhette statistic)*