Homework 3: Graphical models

Adil Rhoulam

arhoulam@ens-paris-saclay.fr

1. Implement the recursions α et β seen in class to compute $p(q_t|u_1, ..., u_T)$ and $p(q_t, q_{t+1}|u_1, ..., u_T)$.

(i) I implemented first the two functions *log_alpha_recursion / log_beta_recursion*. The first takes in arguments the data, the time t, the log of the previous value of alpha (at time t-1), the transition matrix A, the log of the initial probability distribution log_PI and finally the means, covariances of the Gaussian emission probabilities, it returns back the log of the current value of alpha (at time t).*log_beta_recursion* takes also in arguments the data, the time t, the log of the next value of beta (at time t+1), the transition matrix A and finally the means, covariances of the Gaussian emission probabilities, it returns back the log of the current value of beta (at time t).

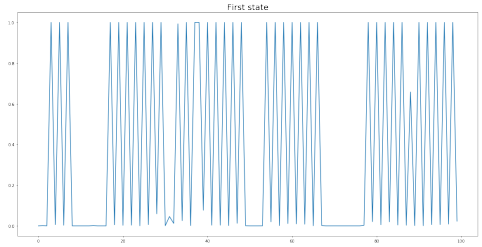
(ii) Then I implemented the two functions *log_alpha / log_beta* that manipulate the two previous described functions to calculate all the log alpha, beta w.r.t to the length of the chain (the data). The first takes in arguments the data, the transition matrix A, the log of the initial probability distribution log_PI and finally the means, covariances of the Gaussian emission probabilities and returns back a matrix of shape (T,4) where T is the length of the chain and 4 is all the possible values of a state. The same things hold true for the *log_beta* function. Note that the matrices returned from the two functions are sorted in time.

2. Using the same parameters for the means and covariance matrix of the 4 Gaussians as the ones obtained in the previous homework, taking a uniform initial probability distribution π , and setting A to be the matrix with diagonal coefficients $A_{i,i} = \frac{1}{2}$ and off-diagonal coefficients $A_{i,j} = \frac{1}{6}$ for all (i, j) $\in \{1, . . . , 4\}^2$, compute α_t and β_t for all t on the test data and compute $p(q_t|u_1, ..., u_T)$. Finally, represent $p(q_t|u_1, ..., u_T)$ for each of the 4 states as a function of time for the 100 first data points in the file.

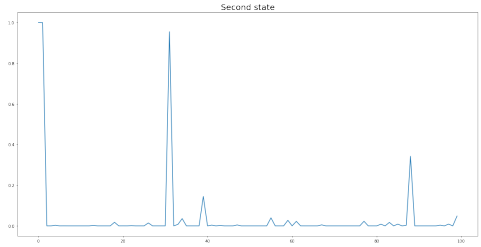
(i) For this I implemented the function *log_smoothing* that return the log values of the probabilities $p(q_t|u_1, ..., u_T)$ in a matrix of shape (T,4) and the log_likelihood which is

$p(u_1, ..., u_T)$ by taking in arguments the data and the parameters A, means, covariances and the all the log values of alpha and beta recursion.

(ii) The representations of the 4 states for the first 100 data points are shown in Figures 1 and 2

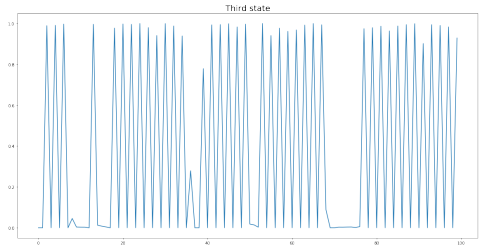


(a) State 1

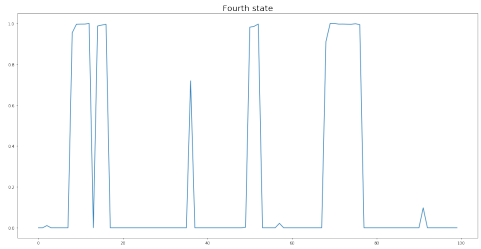


(b) State 2

Figure 1: State 1, 2 representations of $p(q_t|u_1, ..., u_T)$ for 0 $\leq$ t $\leq$ 100 using test data



(a) State 3



(b) State 4

Figure 2: State 3, 4 representations of $p(q_t|u_1, ..., u_T)$ for 0 $\leq$ t $\leq$ 100 using test data

We can notice from the figures that the first and the third states are more probable than the second and the fourth states.

3. Derive the estimation equations of the EM algorithm.

Please, see the attached images in the end of the report.

4. Implement the EM algorithm to learn the parameters of the model (π, A, μ_k, Σ_k, $k = 1..., 4$). The means and covariances could be initialized with the ones obtained in the previous homework. Learn the model from the training data in ?EMGaussienne.dat?.

(i) First, I implemented the function *log_joint_proba* that calculate a tensor of shape (T-1,4,4) containing the log of the probabilities $p(q_t, q_{t+1}|u_1, ..., u_T)$ that we will need in the EM algorithm.

(ii) The EM algorithm is initialized with the same values as in question 2, the stop criterion is used such that the difference between two consecutive log_likelihood values is small than a threshold (0.0001 used in the implementation). The EM algorithm returns the updated parameters (π, A, μ_k, Σ_k, $k = 1..., 4$) and the log_likelihood history.

(iii) When learning the model using the training data, the new parameters are:

$\log \pi$ = [-5.50195669e+01, -2.83764504e+01, -1.48889756e+02, -2.95585778e-12]

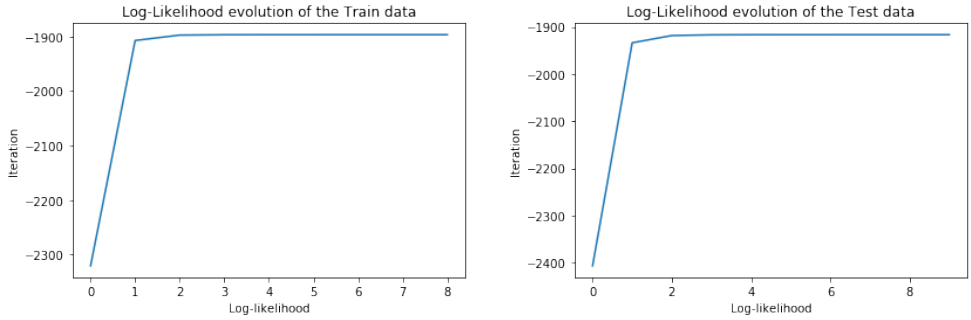
A = [[0.87860557, 0.06792749, 0.01799263, 0.03547432] ; [0.02705241, 0.01576445, 0.94904963, 0.00813356] ; [0.031911, 0.85984905, 0.02250631, 0.08573364] ; [0.01939464, 0.05234451, 0.02758432, 0.90067649]]

μ = [[3.78915942, -3.97479781] ; [-1.95013568, 4.19401969] ; [3.99447762, 3.6336008] ; [-2.96884882, -3.44596339]]

Σ = [[[0.94419417 , 0.06121408] [0.06121408, 1.55399898]] ; [[3.28362594 , 0.3046625] [0.3046625 , 2.83129284]] ; [[0.19769952 , 0.25884686] [0.25884686 , 12.34090692]] ; [[6.80922442 , 6.58541086] [6.58541086 , 6.6864791]]]

5. Plot the log-likelihood on the train data and on the test data as a function of the iterations of the algorithm. Comment.

The plots are shown in Figure 3



(a) Train data (b) Test data

Figure 3: Representations of log-likelihood evolution as a function of the iterations of the algorithm EM for a precision of 10^{-4}

We can notice that the algorithm has converged quickly for the two datasets, after 8 iterations we have already a precision of 10^{-4} for the log-likelihoods.

6. Return in a table the values of the log-likelihoods of the Gaussian mixture models and of the HMM on the train and on the test data. Compare these values. Does it make sense to make this comparison ? Conclude. Compare these log-likelihoods as well with the log-likelihoods obtained for the different models in the previous homework.

See Table 1 for comparison between GMM and HMM.

GMM + Train	HMM + Train	GMM + Test	HMM + Test
-2327.715	-1896.788	-2360.113	-1916.404

Table 1: Comparison of the returned log-likelihood values by the EM algorithm between the GMM and HMM on the train and test data

We can remark that the values of log-likelihood given by the Hidden Markov Model are higher than those given by the Gaussian Mixtures Model.
Since the likelihood represent the same probability $p(u_1, ..., u_T)$ in the two models, so the comparison has a sense.
See Table 2 for comparison between HMM, the isotropic gaussian mixture model (IGMM) and the k-means algorithm.

HMM + Train	IGMM + Train	HMM + Test	IGMM + Test
-1896.788	-3097.908	-1916.404	-3097.908

Table 2: Comparison of the returned log-likelihood values by the EM algorithm between the IGMM and HMM on the train and test data

The performance of the HMM still better on both the IGMM or GMM and gives higher values of the likelihood, we conclude that the HMM model is better than the GMM model.

7. Provide a description and pseudo-code for the Viterbi decoding algorithm (aka MAP inference algorithm or max-product algorithm) that estimates the most likely sequence of states, i.e. argmax $p(q_1, ..., q_T|y_1, ..., y_T)$

The Viterbi decoding algorithm is an algorithm that find the most likely sequence of hidden states which not in general the set of states that are most probable individually (further explanations are in question 8). Since the probabilities are all positive, we have $\log(\max p(\mathbf{x})) = \max \log p(\mathbf{x})$. Using the same formula of the sum-product algorithm that remains true for the max-product algorithm, we have:

$$\mu_{j\to i}(x_i) = \max_{x_j} \left[\psi_{i,j}(x_i, x_j)\psi_i(x_j) \prod_{x_k k\neq i} \mu_{k\to j}(x_j) \right]$$

Then when using the property mentioned before we get :

$$\log(\mu_{j\to i}(x_i)) = \max_{x_j} \left[\log\left(\psi_{i,j}(x_i, x_j)\psi_i(x_j)\right) + \sum_{x_k k\neq i} \log(\mu_{k\to j}(x_j)) \right]$$

In the HMM model we have:

$$\psi_{i,i+1}(q_i, q_{i+1})\psi_i(q_i) = p(q_{i+1}|q_i)$$

$$\prod_{q_k, y_k k \neq i+1} \mu_{q_k \to i}(q_i)\mu_{y_k \to i}(q_i) = p(y_i|q_i)\mu_{i-1 \to i}(q_i)$$

As in Sum-product let $\alpha_{i+1}(q_{i+1}) = \mu_{q_i \to q_{i+1}}(q_{i+1})\mu_{y_{i+1} \to q_{i+1}}(q_{i+1})$
if we note $w(q_{i+1}) = \log(\alpha_{i+1}(q_{i+1}))$ we get the below recursion relation:

$$w(q_{i+1}) = \log(\mu_{y_{i+1} \to q_{i+1}}(q_i)) + \max_{q_i} \left[\log\left(p(q_{i+1}|q_i)\right) + \log(\mu_{q_{i-1} \to q_i}(q_i)\mu_{y_i \to q_i}(q_i)) \right]$$

Which is equivalent to:

$$w(q_{i+1}) = \log(p(y_{i+1}|q_{i+1})) + \max_{q_i} \left[\log\left(p(q_{i+1}|q_i)\right) + w(q_i) \right].$$

with $w(q_0) = \log(p(q_0)) + \log(p(y_0|q_0))$

So finally to get the most likely sequence, we do a forward pass in which we evaluate the sequence w while keeping a record of the values of q_i that correspond to the maxima for each value of the K values q_{i+1}. Then we do a backtrack pass along the chain in which we construct the most probable chain by retrieving the state of maximal state q_{T-1} that has caused the state q_T and so on.

Pseudocode

Input $(q_1, ...q_T)$, $(u_1, ..., u_T)$, A, π, the gaussian transitions
begin: W, M forward pass $\rightarrow$
W = $[w(q_0), ..., w(q_T)]$ of size TxK , M = $[argmax(w(q_1)), ..., argmax(w(q_T))]$ of size (T-1)x4
Backward pass $\rightarrow$ Path size T-1x1 backtrack

See next question for more explanations.

8. Implement Viterbi decoding. For the set of parameters learned with the EM algorithm, compute the most likely sequence of states with the Viterbi algorithm and represent the data in 2D with the cluster centers and with markers of different colors for the data points belonging to different classes.

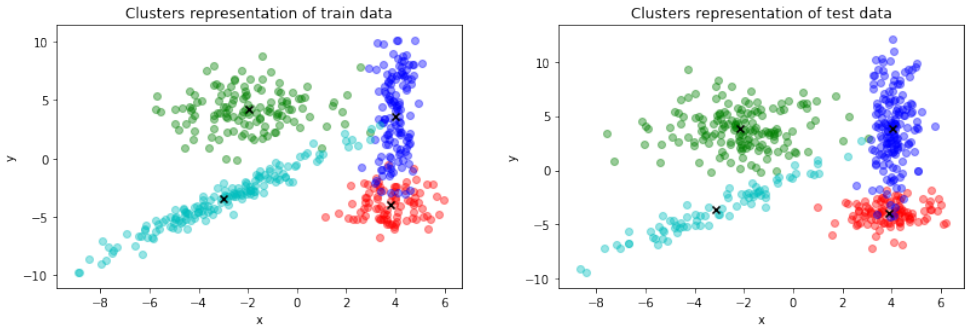
(i) I implemented first the function *log_viterbi_recursion* that takes in arguments the data, the time t, the log of the previous value of the Viterbi algorithm (at time t-1), the transition matrix A, the log of the initial probability distribution log_PI and finally the means, covariances of the Gaussian emission probabilities, it returns back the log of the current value of Viterbi Algorithm (at time t) and also the states of the previous value of the Viterbi algorithm that correspond to the maxima for each state of the 4 states of the t^{th} value of Viterbi Algorithm returned.

(ii) Then I implemented the function *log_viterbi* that manipulate the previous described function to calculate all the log of the Viterbi Algorithm values w.r.t to the length of the chain (the data) and the maximal_states which contains all the states that correspond to the maxima for each state of the 4 states of the next value of Viterbi Algorithm. It takes in arguments the data, the transition matrix A, the log of the initial

probability distribution log_PI and finally the means, covariances of the Gaussian emission probabilities and returns back a matrix of shape (T,4) containing the log values of the Viterbi Algorithm (max-prod successive values) and also a matrix of shape (T-1,4) that contains the values of states corresponding to maximal Viterbi value.

(iii) Finally, The function *most_likely_sequence* uses the previous recorded states of q_t that correspond to the maxima for each state of the 4 states of q_{t+1} for all t $\in \{1, ..., T\}$ when passing the messages to the end of the chain and uses a backtrack along the chain to found the most probable path till time t (which is equal to T by default).

(iv) The representation in 2D of the clusters of train and test datasets are in Figure 4.

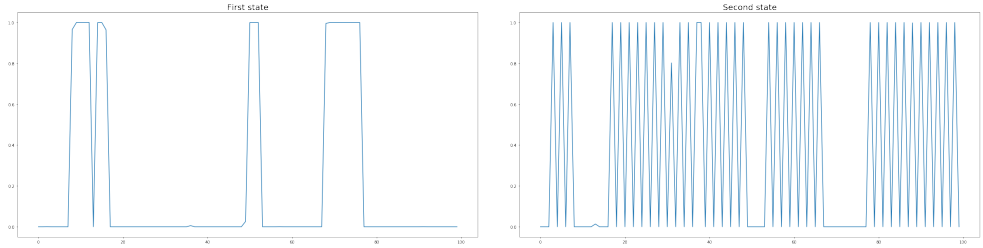


(a) Train data (b) Test data

Figure 4: Representations of clusters of different classes for the two datasets

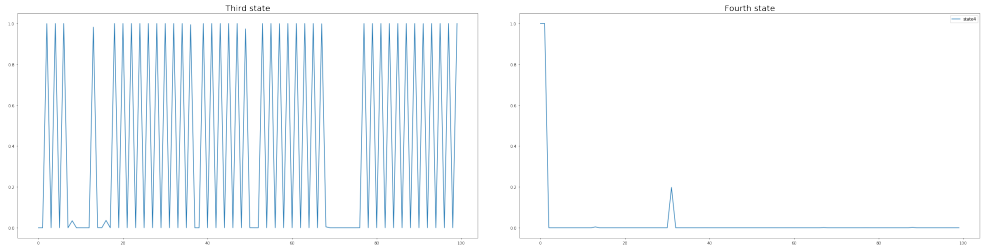
9. For the datapoints in the test file,, compute the marginal probability $p(q_t|u_1, ..., u_T)$ for each point to be in state {1,2,3,4} for the parameters learned on the training set. For each state plot the probability of being in that state as a function of time for the 100 first points (i.e., as a function of the datapoint index in the file).

The plots could be seen in Figures 5 and 6.



(a) State 1 (b) State 2

Figure 5: State 1, 2 representations of $p(q_t|u_1, ..., u_T)$ for $0 \leq t \leq 100$ of test data for parameters trained with train data



(a) State 3 (b) State 4

Figure 6: State 3, 4 representations of $p(q_t|u_1, ..., u_T)$ for $0 \leq t \leq 100$ of test data for parameters trained with train data

10. For each of these same 100 points, compute their most likely state according to the marginal probability computed in the previous question. Make a plot representing the most likely state in {1,2,3,4} as function of time for these 100 points.

The representation is in Figure 7.

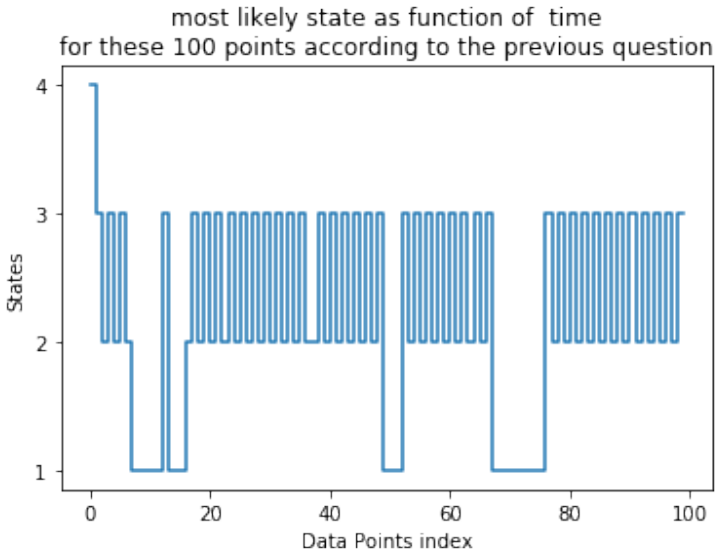


Figure 7

11. Run Viterbi on the test data. Compare the most likely sequence of states obtained for the 100 first data points with the sequence of states obtained in the previous question. Make a similar plot. Comment.

The representation is in Figure 8.

We remark that the most probable (likely) state given by calculating the marginal probability

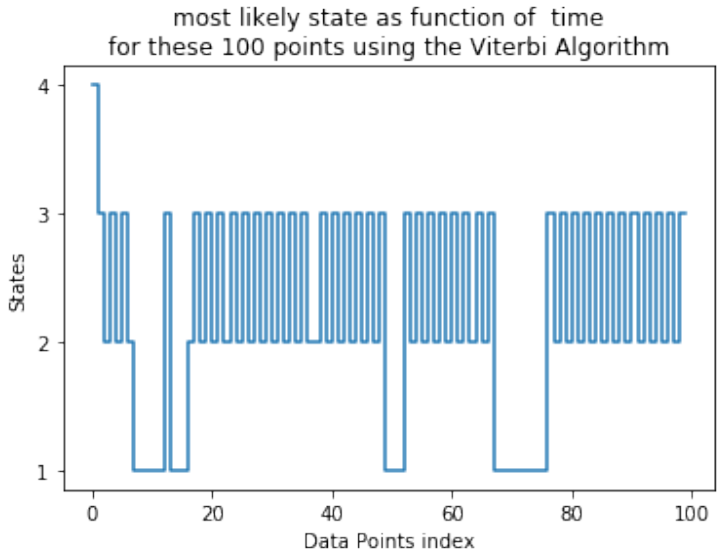


Figure 8

and the most likely sequence calculated using the Viterbi algorithm give the same most likely sequence. However, in general, the states that give the maximal marginal probability doesn't correspond to the most likely sequence of states. This can be seen if we consider two successive states that are maximal individually which means the most probable and the transition matrix element connecting them is zero.

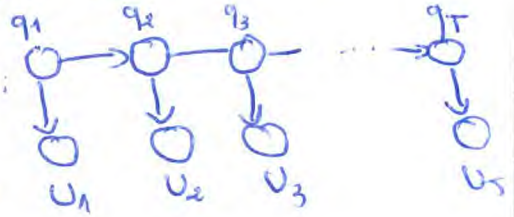
12. In this problem the number of states was known. How would you choose the number of states if you did not know it ?

If we didn't know the number of states, I would use some model selection algorithms such us cross-validation for some range of value and choose the k that gives the maximal value of the likelihood .

PGM - Home Work 3

RHOUATI ADIL

3) the complete likelihood of the HMM is : q_1 = root



$$l_c(\theta) = p(q,u|\theta) = \prod_{t=1}^{T-1} p(q_{t+1}|q_t) \prod_{t=1}^{T} p(u_t|q_t)\, p(q_1)$$

with $q = (q_1 \dots q_T)$; $u = (u_1 \dots u_T)$

$\theta = (\pi, A, \mu, \Sigma)$

$$\log(l_c(\theta)) = \log p(q_1) + \sum_{t=1}^{T-1} \log p(q_{t+1}|q_t) + \sum_{t=1}^{T} p(u_t|q_t)$$

$$p(q_1) = \prod_{k=1}^{K} \pi_i^{\delta(q_1=i)} \quad ; \quad p(q_{t+1}|q_t) = \prod_{i,j=1}^{K} p(q_{t+1}=i|q_t=j)^{\delta(q_{t+1}=i, q_t=j)}$$

with : $A_{ij} = p(q_{t+1}=i|q_t=j)$, we have :

$$\log(l_c(\theta)) = \sum_{k=1}^{K} \delta(q_1=k) \log(\pi_k) + \sum_{t=1}^{T-1} \sum_{i,j=1}^{K} \delta(q_{t+1}=i, q_t=j) \log(A_{ij}) + \sum_{t=1}^{T} \sum_{k=1}^{K} \delta(q_t=k) \log(N(u_t|\Sigma_k, \mu_k))$$

E-step :

$$Q(\theta, \theta^{old}) = E_{p(\cdot|u,\theta^{old})}(\log(l_c(\theta)))$$

$$= \sum_{k=1}^{K} p(q_1=k|u,\theta^{old}) \log(\pi_k) + \sum_{t=1}^{T-1} \sum_{i,j=1}^{K} p(q_{t+1}=i, q_t=j|u,\theta^{old}) \log(A_{ij}) + \sum_{t=1}^{T} \sum_{k=1}^{K} p(q_t=k|u,\theta^{old}) \log(N(u_t|\mu_k, \Sigma_k))$$

with $\theta^{old} = (\pi^{old}, A^{old}, \mu^{old}, \Sigma^{old})$

M-step : Maximisation of Q w.r.t to the parameters θ

w.r.t π : we have : $\sum_{k=1}^{K} \pi_k = 1$

$$L(\pi, \lambda) = -Q(\theta, \theta^{old}) + \lambda\left(\sum_{k=1}^{K} \pi_k - 1\right)$$

$\pi_k \to -Q(\pi, \theta^{old})$ is convex because $\pi_k \to -\log(\pi_k)$ are convex | $\pi_k \to \sum_{k=1}^{K} \pi_k$ convex because linear).

$$\frac{\partial L}{\partial \pi_i} = 0 \Leftrightarrow - p(q_t = k \mid v, \theta^{old}) \frac{1}{\pi_k} + \lambda = 0$$

$$\Leftrightarrow \lambda \pi_k = p(q_t = k \mid v, \theta^{old}) \Leftrightarrow \lambda \sum_{k=1}^{K} \pi_k = 1 \Leftrightarrow \lambda = 1$$

then: $\boxed{\pi_k = p(q_t = k \mid v, \theta^{old})}$

Maximisation w.r.t A: we have $\sum_{i=1}^{T} A_{ij} = 1 \; ; \; \forall j \in [1, T]$

then: if $\lambda = (\lambda_1 \dots \lambda_T) \in \mathbb{R}^T$:

$$L(A, \lambda) = - Q(A, \theta^{old}) + \sum_{j=1}^{T} \lambda_j \left(\sum_{i=1}^{T} A_{ij} - 1 \right)$$

$A \mapsto L(A, \lambda)$ is a convex function because $A \mapsto \log(A_{ij})$ and $A \mapsto \sum_{i=1}^{T} A_{ij}$ are convex

then: $$\frac{\partial L}{\partial A_{ij}} = 0 \Leftrightarrow - \sum_{t=1}^{T-1} p(q_{t+1} = i, q_t = j \mid v, \theta^{old}) + \lambda_j A_{ij} = 0$$

$$\Leftrightarrow \lambda_j \sum_{i=1}^{K} A_{ij} = \lambda_j = \sum_{t=1}^{T-1} p(q_t = j \mid v, \theta^{old})$$

then: $$\boxed{A_{ij} = \frac{\sum_{t=1}^{T-1} p(q_{t+1} = i, q_t = j \mid v, \theta^{old})}{\sum_{t=1}^{T-1} p(q_t = j \mid v, \theta^{old})}}$$

Maximisation w.r.t. μ:

$$Q(\mu, \theta^{old}) = - \frac{1}{2} \sum_{t=1}^{T} \sum_{k=1}^{K} p(q_t = k \mid v, \theta^{old}) (v_t - \mu_k)^T \Sigma_k^{-1} (v_t - \mu_k) + \text{cte}$$

$\mu_k \mapsto Q(\mu, \theta^{old})$ is concave and:

$$\frac{\partial Q}{\partial \mu_k} = 0 \Leftrightarrow - \sum_{t=1}^{T} p(q_t = k \mid v, \theta^{old}) \Sigma_k^{-1} (v_t - \mu_k) = 0$$

$$\Leftrightarrow \boxed{\mu_k = \frac{\sum_{t=1}^{T} p(q_t = k \mid v, \theta^{old}) v_t}{\sum_{t=1}^{T} p(q_t = k \mid v, \theta^{old})}}$$

Maximization w.r.t. Σ:

$\Sigma_k^{-1} \longmapsto Q(\Sigma_k, \vartheta^{old})$ is concave and:

$$\frac{\partial Q}{\partial \Sigma_k^{-1}} = 0 \Leftrightarrow \sum_{t=1}^{T} p(q_t = k \mid u, \vartheta^{old}) \left(\Sigma_k - (u_t - \mu_k^{new})(u_t - \mu_k^{new})^T\right) = 0$$

$$\Leftrightarrow \boxed{\Sigma_k = \frac{\sum_{t=1}^{T} p(q_t = k \mid u, \vartheta^{old})(u_t - \mu_k^{new})(u_t - \mu_k^{new})^T}{\sum_{t=1}^{T} p(q_t = k \mid u, \vartheta^{old})}}$$

using the fact that $\frac{\partial}{\partial A} \log\det(A) = A^{-1}$