Probabilistic Graphical Models, Master MVA, 2017/2018

Homework 3 - HMM Implementation

Pirashanth RATNAMOGAN , Othmane SAYEM

pirashanth.ratnamogan@ens-paris-saclay.fr, othmane.sayem@ens-paris-saclay.fr

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

We have implemented the class 'EM-HMM' that contains the functions 'compute-log-alpha' and 'compute-log-beta'. These two functions compute recursively the alpha and beta in the log domain. Moreover, we defined the function Sum-Log-Exp which allows us to complete the calculations in a numerically stable way.

2- 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.

Since we observed that the recursive computations of α and β, push the values to decrease quickly and to converge to zero, we did the sums on the log domain, as seen in the course. For instance, the recursive equation of $\alpha(q_t)$ will be written :

$$\begin{aligned}\hat{\alpha}(q_{t+1}) = \log(\alpha(q_{t+1}) &= \log\left[\sum_{q_t} \alpha(q_t) A_{q_t,q_{t+1}} p(u_{t+1}|q_{t+1})\right] \\ &= \log\left[\sum_{q_t} \exp(\log(\alpha(q_t))) \exp(\log(A_{q_t,q_{t+1}} p(u_{t+1}|q_{t+1})))\right] \\ &= \log\left[\sum_{q_t} \exp(\hat{\alpha}(q_t)) \exp(\log(A_{q_t,q_{t+1}} p(u_{t+1}|q_{t+1})))\right]\end{aligned}$$

Then we use the LogSumExp function described in lecture notes, which allows us to compute in the log domain with the smallest amount of numerical errors.

For each of the 4 states, the values of $p(q_t|u_1, ..., u_T)$ are shown in the following figure :

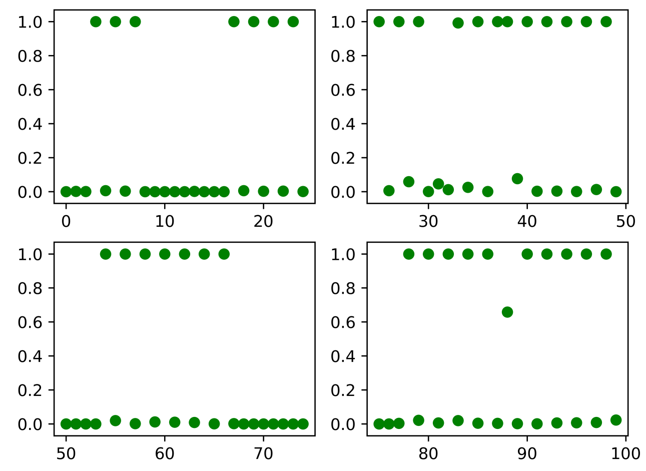

Figure 1: $p(q_t = 1|u_1, ..., u_T)$ as a function of time for the 100 first test data points

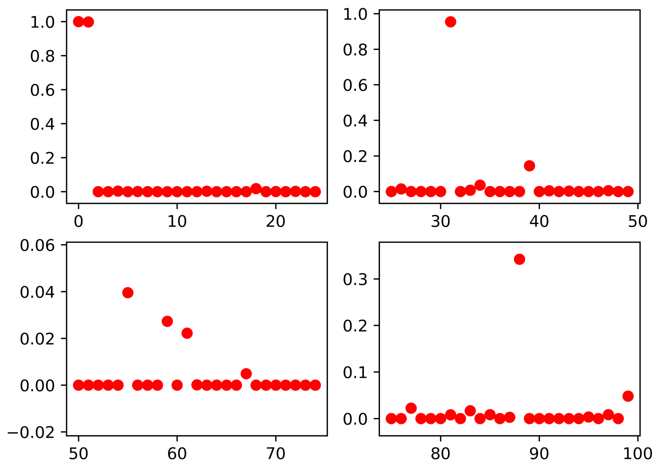

Figure 2: $p(q_t = 2|u_1, ..., u_T)$ as a function of time for the 100 first test data points

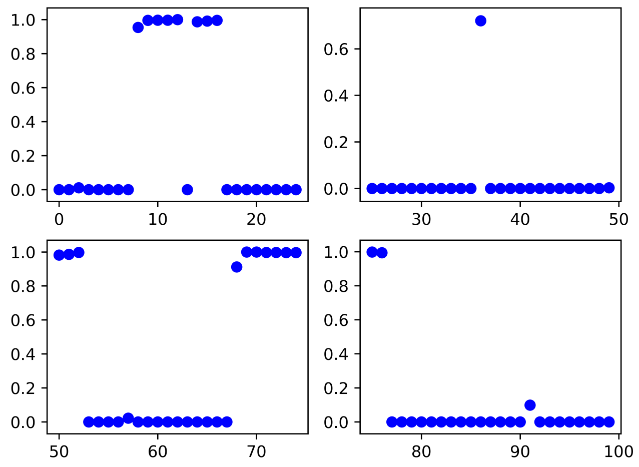

Figure 3: $p(q_t = 3|u_1, ..., u_T)$ as a function of time for the 100 first test data points

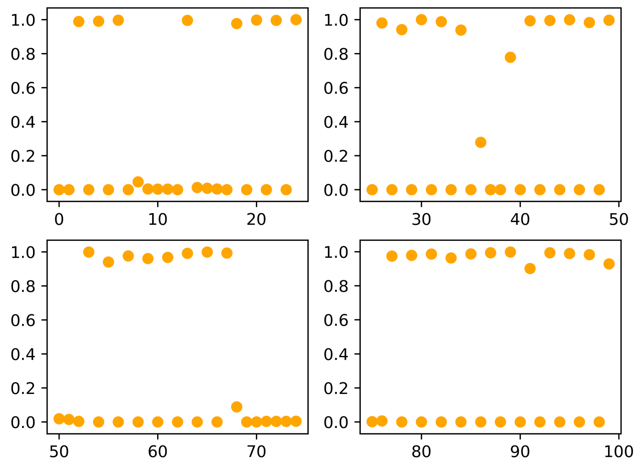

Figure 4: $p(q_t = 4|u_1, ..., u_T)$ as a function of time for the 100 first test data points

3- Derive the estimation equations of the EM algorithm.

We want to maximize the likelihood $\log(p(u_0, .., u_T)$. To do so, we apply the EM algorithm, which consists on the following steps :

1-Compute $p(q_t|u_0, .., u_T)$: using the alpha beta formulations that we used in the forward-backward algorithm, we know that :

$$\gamma(q_t) = p(q_t|u_0,..,u_T) = \frac{\alpha(q_t) \times \beta(q_t)}{\sum_{q_t} \alpha(q_t)\beta(q_t)}$$

In the same way, computing $\xi(q_t, q_{t+1}) = p(q_t = i, q_{t+1} = j|u_0..u_T)$, can be done using alpha and gammas, as following :

$$\xi(q_t, q_{t+1}) = \frac{\alpha(q_t)\gamma(q_{t+1})p(u_{t+1}|q_{t+1})A_{q_t,q_{t+1}}}{\alpha(q_{t+1})}$$

2 - Write the complete log-likelihood: Since we are considering a HMM model, the joint probability for a particular configuration $(q, u) = (q_0, q_1, .., q_T, u_0, ..., u_T)$ is given by :

$$\begin{aligned} p(u, q) &= p(q_0, q_1, .., q_T, u_0, ..., u_T) \\ &= p(q_0, q_1, .., q_T) \prod_{t=0}^{T} p(u_t|q_t) \\ &= p(q_0) \prod_{t=0}^{T-1} p(q_{t+1}|q_t) \prod_{t=0}^{T} p(u_t|q_t) \end{aligned}$$

Thus, we can write the complete likelihood of our problem as following :

$$\begin{aligned} l_c(\theta) &= \log(p(u_0.., u_T, q_0.., q_T)) \\ &= \log(p(q_0)) + \sum_{t=0}^{T-1} \log(p(q_{t+1}|q_t)) + \sum_{t=0}^{T} \log(p(u_t|q_t)) \\ &= \sum_{i=1}^{4} z_i^0 \log(\pi_i) + \sum_{i,j=1}^{4} \sum_{t=0}^{T-1} z_i^t z_j^{t+1} \log(A_{i,j}) + \sum_{t=0}^{T} \sum_{i=1}^{4} z_i^t \log(\mathcal{N}(\mu_i|\Sigma_i)) \end{aligned}$$

Where we define $z_i^t = \delta(q_t = i)$.

3- Compute E Step : We use the Jensen's inequality to derive a lower bound of the *incomplete* log likelihood. Here we chose to use $l(q_0, .., q_T)$ a probability on the states $q_0, .., q_T$:

$$\begin{aligned} \log(p(u_0, .., u_T) &= \log(\sum_{q_0} ... \sum_{q_T} p(q_0, q_1, .., q_T, u_0, ..., u_T)) \\ &= \log(\sum_{q_0} ... \sum_{q_T} l(q_0, .., q_T) \frac{p(q_0, q_1, .., q_T, u_0, ..., u_T)}{l(q_0, .., q_T)}) \\ &= \log(\mathbb{E}_l \left[\frac{p(q, u)}{l(q)} \right]) \\ &\geq \mathbb{E}_l \left[\log(p(q, u))\right] - \mathbb{E}_l \left[\log(l(q))\right] \end{aligned}$$

Maximizing this lower bound with respect to $l(q)$, is given by the distribution probability which gives equality in Jensen's inequality, which means that $l(q_t) = p(q_t|u_0, ..u_T)$. Hence, the lower bound, is given by $\mathbb{E}_{q|u}\left[\log(p(q,u))\right]$.
Hence in this step, using the model's parameters $\theta^k = (\pi^k, A^k_{i,j}, \mu^k_i, \Sigma^k_i)$, we compute :

$$\gamma^{k+1}(q_t) = \frac{\alpha(q_t) \times \beta(q_t)}{\sum_{q_t} \alpha(q_t)\beta(q_t)}$$

$$\xi^{k+1}(q_t, q_{t+1}) = \frac{\alpha(q_t)\gamma(q_{t+1})p(u_{t+1}|q_{t+1})A^k_{q_t,q_{t+1}}}{\alpha(q_{t+1})}$$

4 - Compute M Step: we need to maximize $(\mathbb{E}_{q|u}\left[\log(p(q,u))\right])$ with respect to π, A, μ_i, Σ_i, which can be written :

$$\sum_{i=1}^{4} \gamma^k(q_0 = i)\log(\pi_i) + \sum_{i,j=1}^{4}\sum_{t=0}^{T-1} \xi^k_t(i,j)\log(A_{i,j}) + \sum_{t=0}^{T}\sum_{i=1}^{4} \gamma(q_t = i)\log(\mathcal{N}(\mu_i|\Sigma_i))$$

We first derive with respect to π, as we have proved in the last homework, this can be written as following :

$$\max_{\pi} \sum_{i=1}^{4} \gamma^k(q_0 = i)\log(\pi_i) \quad \text{s.t} \quad \sum_i \pi_i = 1$$

and the optimal solution (using the lagrangien of the problem) is given by the following expression :

$$\boxed{\pi^{k+1}_i = \gamma^k(q_0 = i)}$$

Maximizing with respect to A, is equivalent to maximizing only the second term of the expected complete likelihood :

$$\max_{A} \sum_{i,j=1}^{4}\sum_{t=0}^{T-1} \xi^k_t(i,j)\log(A_{i,j}) \quad \text{s.t} \quad \sum_i A_{i,j} = 1$$

The lagrangien of the problem, if we consider λ_j the dual variables of each constraint, is written as following :

$$L(A,\lambda) = \sum_{i,j=1}^{4}\sum_{t=0}^{T-1} \xi^k_t(i,j)\log(A_{i,j}) - \sum_{j=1}^{4}\lambda_j\left(\sum_i A_{i,j} - 1\right)$$

when deriving with respect to $A_{i,j}$, we get : $\frac{\sum_{t=0}^{T-1}\xi^k_t(i,j)}{A_{i,j}} = \lambda_j$, which means that $A_{i,j} = \frac{\sum_{t=0}^{T-1}\xi^k_t(i,j)}{\lambda_j}$. The sum over all possible i's, shows that $\lambda_j = \sum_i \sum_{t=0}^{T-1}\xi^k_t(i,j)$. Thus, the optimal expression for $A_{i,j}$:

$$\boxed{A^{k+1}_{i,j} = \frac{\sum_{t=0}^{T-1}\xi^k_t(i,j)}{\sum_i\sum_{t=0}^{T-1}\xi^k_t(i,j)}}$$

Now, we derive the last term of the complete likelihood, with respect to μ_i :

$$\max_{\mu_i} \sum_{t=0}^{T} \sum_{i=1}^{4} \gamma(q_t = i) \left[-log(2\pi) - \frac{1}{2}\log(|\Sigma_i|) - \frac{1}{2}(u_t - \mu_i)^T \Sigma_i^{-1}(u_t - \mu_i) \right]$$

The condition of optimality of μ_i, gives $\sum_{t=0}^{T} \gamma(q_t = i)\Sigma_i^{-1}(u_t - \mu_i) = 0$, which means that the optimal value μ_i^{k+1} is given by :

$$\boxed{\mu_i^{k+1} = \frac{\sum_{t=0}^{T} \gamma^k(q_t=i)u_t}{\sum_{t=0}^{T} \gamma^k(q_t=i)}}$$

Computing the gradient of the same objective function along Σ_i, allows us to find easily the optimal Σ_i^{k+1} :

$$\boxed{\Sigma_i^{k+1} = \frac{\sum_{t=0}^{T} \gamma^k(q_t=i)(u_t-\mu_i^{k+1})(u_t-\mu_i^{k+1})^T}{\sum_{t=0}^{T} \gamma(q_t=i)}}$$

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

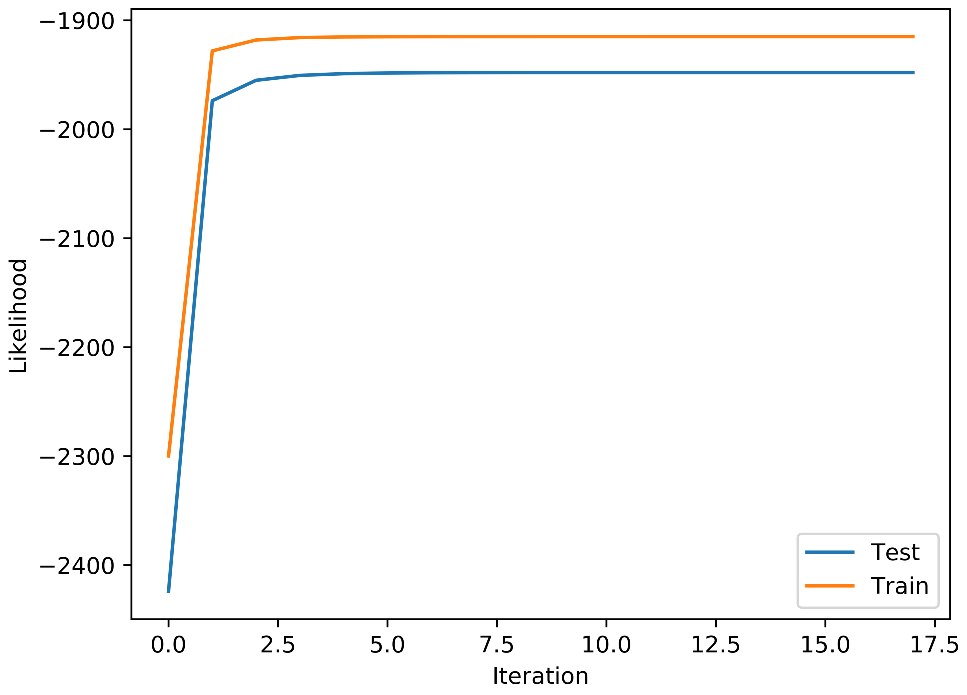

Figure 5: log-likelihood of HMM model applied to test and train data as a function of the iterations

For this question we trained our HMM model with the "EMGaussienne.dat". During this training we keep the evolution of the log-likelihood of the sets "EMGaussienne.dat" and on the test data "EMGaussienne.test". We notice that the likelihood of both train and test data increases with the number of iterations. The log-likelihood for the test set is lower than the one of the training set because the parameters are fit on the training set.

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.

Model	data set	Log-likelihood
HMM	Train	-1915.066
	Test	-1948.08
Gaussian Mixture	Train	-2327.715
	Test	-2408.985

Table 1: Final values of log-likelihood, using HMM and GM models

It doesn't make sense to compare the likelihoods given by the two models, because we are computing the maximum likelihood on different domains each time. Infact, the Gaussian Mixture model can be seen as a special case of HMM, where the hidden parameters are independent. This means that the parameters' domain Θ_1 of GM, are included in the parameters' domain of Θ_2 of HMM : $\Theta_1 \subset \Theta_2$.Hence, as we see in the results, it is normal to see that $\max_{\Theta_1} l_{\theta_1} \leq \max_{\Theta_2} l_{\theta_2}$

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. $\text{argmax}_q = p(q_1, ..., q_T|y_1, ..., y_T)$

We want to compute the most probable HMM latent variables $\{q_1, ..., q_T\}$ that allows to obtain the observed set $\{y_1, ..., y_T\}$. We know by definition that for a given set of observations we have for all $(q_i)_{ßin1,...,T}$:

$$\begin{aligned}
p(q_1, ..., q_T|y_1, ..., y_T) &= \frac{p(q_1, ..., q_T, y_1, ..., y_T)}{p(y_1, ..., y_T)} \\
&\propto p(q_1, ..., q_T, y_1, ..., y_T) \\
&= p(q_1) \prod_{t=1}^{T-1} p(q_{t+1}|q_t) \prod_{t=1}^{T} p(y_t|q_t)
\end{aligned}$$

During decoding we have the set of observation $\{\tilde{y}_1, ..., \tilde{y}_T\}$, we are looking for:

$$\text{argmax}_{q_1,...,q_T} p(q_1, ..., q_T|\tilde{y}_1, ..., \tilde{y}_T) = \text{argmax}_{q_1,...,q_T} p(q_1) \prod_{t=1}^{T-1} p(q_{t+1}|q_t) \prod_{t=1}^{T} p(\tilde{y}_t|q_t)$$

A naive idea to compute this quantity would be to compute the max for all the possible sets $(q_i)_{ßin1,...,T}$ and take the set that produce this max. However, we can rewrite the problem that we want to solve as:

$$\begin{aligned}\text{argmax}_{q_1,...,q_T} p(q_1,...,q_T|\tilde{y}_1,...,\tilde{y}_T) &= \text{argmax}_{q_1,...,q_T} p(q_1) \prod_{t=1}^{T-1} p(q_{t+1}|q_t) \prod_{t=1}^{T} p(\tilde{y}_t|q_t) \\ &= \text{argmax}_{q_1} p(q_1)p(\tilde{y}_1|q_1)\text{argmax}_{q_2}...\text{argmax}_{q_{T-1}} p(q_{T-1}|q_{T-2})p(\tilde{y}_{T-1}|q_{T-1}).. \\ &...\text{argmax}_{q_T} p(q_T|q_{T-1})p(\tilde{y}_T|q_T)\end{aligned}$$

Hence, we are facing a dynamic programming problem. We define the function v_t as :

$$v_t(s) = \max_{s'}\{v_{t-1}(s')p(s|s')p(\tilde{y}_t|s)\} \quad \forall t \in \{2,...,T\} \tag{0.1}$$

$$v_1(s) = p(s)p(\tilde{y}_1|s) \tag{0.2}$$

Maximize among the function that we want to maximize is equivalent to maximize $\max_s v_T(s)$. We can compute v_T recursively using the recursive equation below. When we have computed the maximum for v_T using the recursive equation we can directly know what is the final optimal latent state by using $q_T = \text{argmax}_s v_T(s)$. Knowing the optimal state q_T we can compute q_{T-1} using 0.1. Indeed $q_{T-1} = \text{argmax}_{s'} v_{T-1}(s')p(q_T|s')$. Using the same approach, we can compute recursively the state t knowing the optimal state $t+1$. Hence going from the state q_f that we have computed we can compute the full decoding.

You can find below a pseudo-code for the Viterbi algorithm for HMM that we have implemented. In this pseudo-code the latent state space is written Q, the number of time step is written T and the set of observations $\{\tilde{y}_1,...,\tilde{y}_T\}$ is given. The variable b will allow to keep in memory the path that allowed to obtain the optimal final value.

```
We initialize $v_1(s) = p(s)p(\tilde{y}_1|s) \quad b_1(s) = 0 \quad \forall s \in Q$
 for $t \in \{2,...,T\}$ do
    Compute and store: $v_t(s) = \max_{s'}\{v_{t-1}(s')p(s|s')p(\tilde{y}_t|s)\} \quad \forall s \in Q$
    Compute and store: $b_t(s) = \text{argmax}_{s'}\{v_{t-1}(s')p(s|s')\} \quad \forall s \in Q$
end
Compute and store latent state: $q_T = \text{argmax}_s v_T(s)$
for $t \in \{T-1,...,1\}$ do
    Compute and store latent state $q_t = b_{t+1}(q_{t+1})$
end
Return $q_1, q_2, ..., q_{T-1}, q_T$
```

Algorithm 1: Pseudo-code for Viterbi decoding of HMM latent states

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

We have imlemented the demanded algorithm in our class HMM. Here is the decoding for the training set:

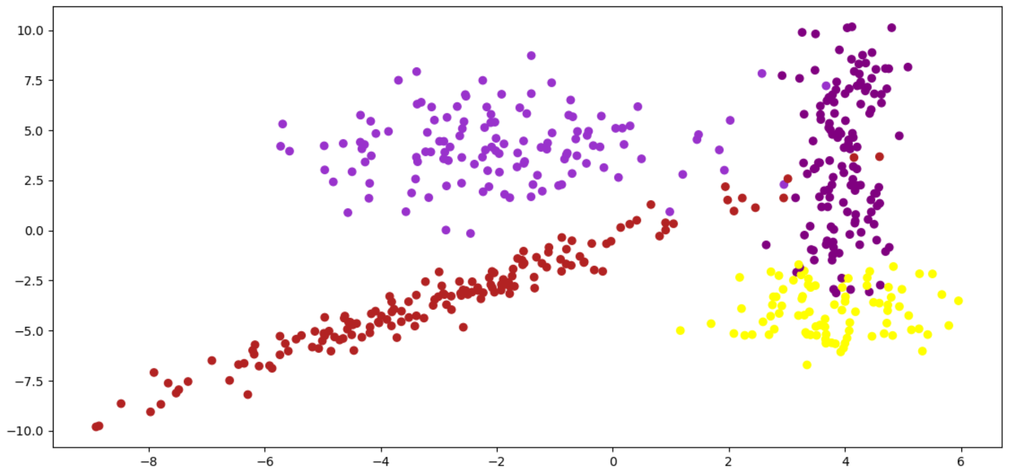

Figure 6: HMM decoding on the training dataset

9. For the datapoints in the test file "EMGaussienne.test", 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).

Here we compute the plots of this marginal probabilities:

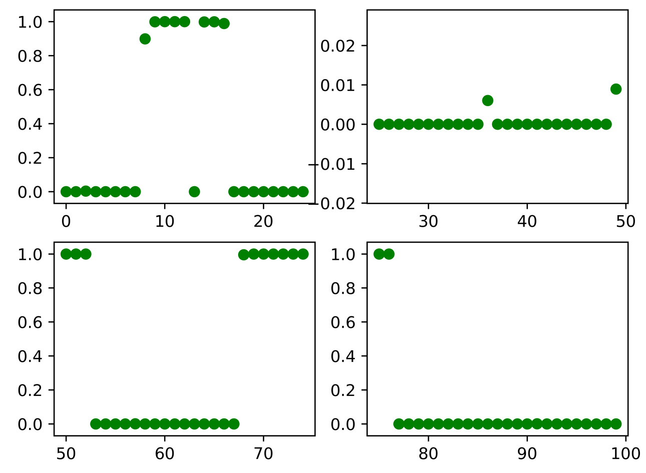

Figure 7: $p(q_t = 1|u_1, ..., u_T)$ as a function of time for the 100 first test data points

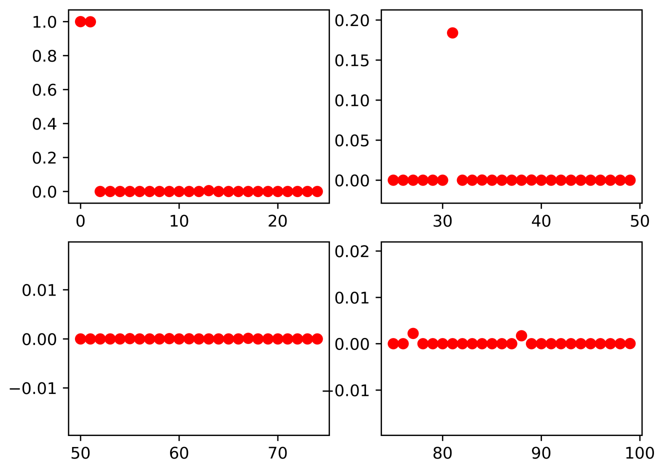

Figure 8: $p(q_t = 2|u_1, ..., u_T)$ as a function of time for the 100 first test data points

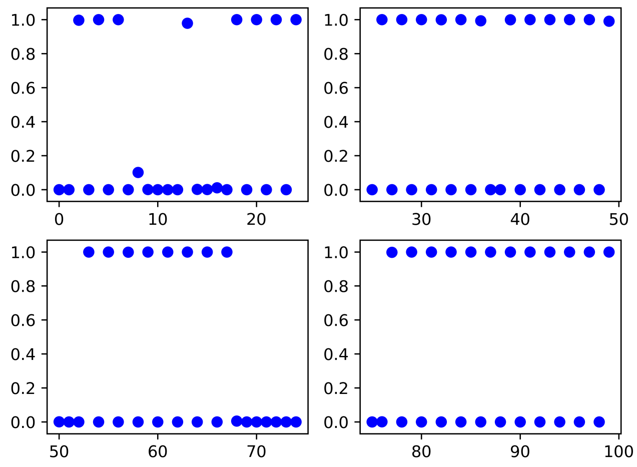

Figure 9: $p(q_t = 3|u_1, ..., u_T)$ as a function of time for the 100 first test data points

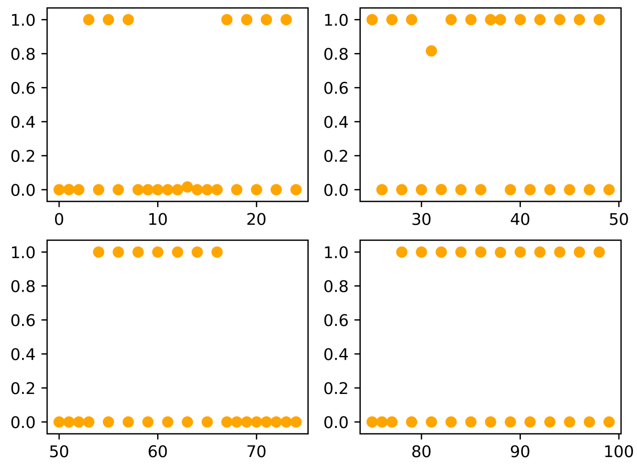

Figure 10: $p(q_t = 4|u_1, ..., u_T)$ as a function of time for the 100 first test data points

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.

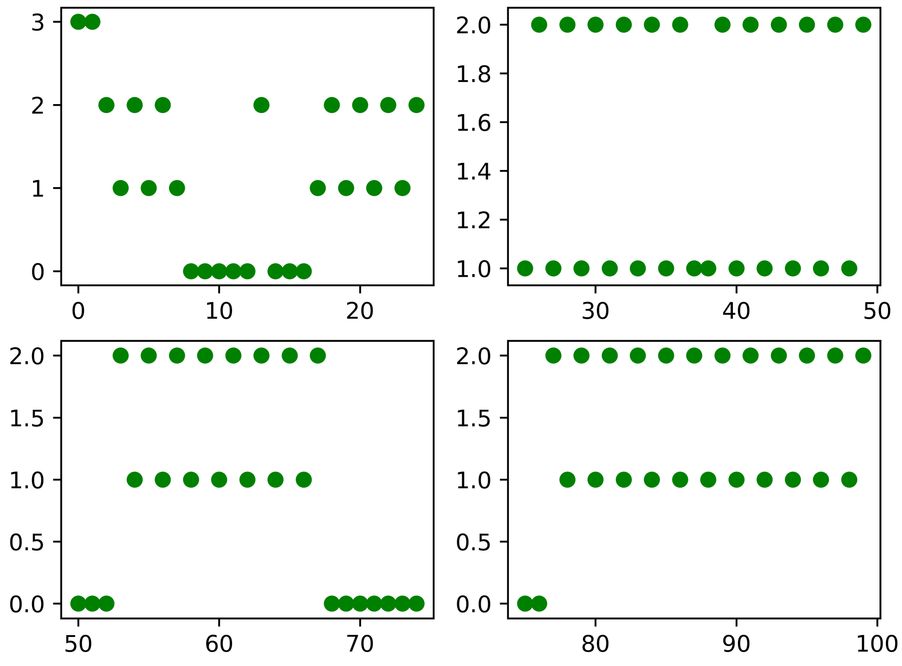

Figure 11: Evolution of the most likely sequence using marginal probabilities

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

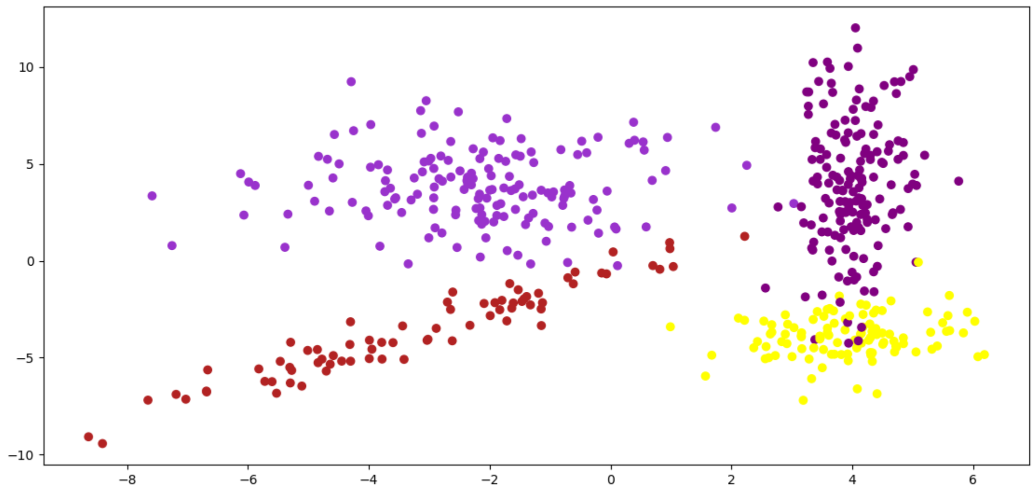

Figure 12: HMM decoding on the test dataset

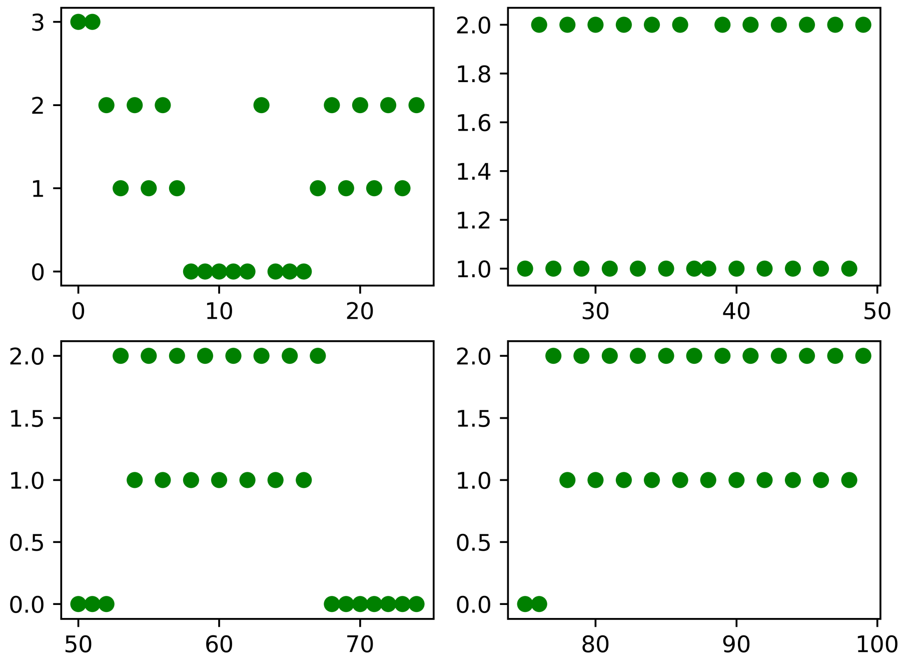

Figure 13: Evolution of the most likely sequence using decoding

As we can see, we have the exact same sequence using decoding and Viterbi in this case within the 100 first data points. Within the 500 data points we have only one variation in the 449th point.

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

One way to choose the number of states is Bayesian Information Criterion (BIC) which is a criterion of model selection that introduces a penalty over the number of parameters chosen in the mode. Actually, the more we increase the number of states, the more parameters we have and the likelihood becomes greater, however doing so could increase the risk of over fitting : that's why we introduce the penalty term.

Since the problematic in this case in a clustering, we can use some methods such as the elbow method or gap statistic, to estimate the number of clusters, and thus the number of states in our model.