M2 MVA, ENS CACHAN

Probabilistic Graphical Models - Homework Assignment 3

Chia-Man Hung, Dexiong Chen

January 4, 2017

1 HMM - IMPLEMENTATION

We consider the same training data as in the previous homework, provided as the "EMGaussienne.dat" file (and we will test on the corresponding testing data from the "EMGaussienne.test" file), but this time we use an HMM model to account for the possible temporal structure of the data. The data are of the form $u_t = (x_t, y_t) \in \mathbb{R}^2$, for $t = 1, ..., T$. The goal of th is exercise is to implement the probabilistic inference algorithm and the EM algorithm to learn parameters as well as the Viterbi algorithm. It is recommended to make use of the code of the previous homework.

We consider the following HMM model: the chain (q_t) has $K = 4$ possible states, with an initial probability distribution $\pi \in \mathbb{R}^4$ and a probability transition matrix $A \in \mathbb{R}^{4x4}$, and conditionally on the current states we have observations obtained from Gaussian emission probabilities $p(u_t|q_t) = i \sim \mathcal{N}(\mu_i, \Sigma_i)$.

Q1. Implement the recursion α and β seen in class to compute $p(q_t|u_1, ..., u_T)$ and $(q_t, q_{t+1}|u_1, ..., u_T)$.

A1. From the lecture notes,

$$\begin{aligned}
\alpha_{t+1}(q_{t+1}) &= p(u_{t+1}|q_{t+1}) \sum_{q_t} p(q_{t+1}|q_t)\alpha_t(q_t), \forall t \in \{1, ..., T-1\} \\
\beta_t(q_t) &= \sum_{q_{t+1}} p(q_{t+1}|q_t)p(u_{t+1}|q_{t+1})\beta_{t+1}(q_{t+1}), \forall t \in \{1, ..., T-1\}
\end{aligned} \tag{1.1}$$

with initial conditions $\alpha_1(q_1) = p(u_1|q_1)$ and $\beta_T(q_T) = 1$.

For computation reasons, we take the logarithm of α and β.

$$\begin{aligned} log\alpha_{t+1}(q_{t+1}) &= logp(u_{t+1}|q_{t+1}) + log\sum_{q_t} p(q_{t+1}|q_t)\alpha_t(q_t), \forall t \in \{1,...,T-1\} \\ log\beta_t(q_t) &= log\sum_{q_{t+1}} p(q_{t+1}|q_t)p(u_{t+1}|q_{t+1})\beta_{t+1}(q_{t+1}), \forall t \in \{1,...,T-1\} \end{aligned} \tag{1.2}$$

with initial conditions $log\alpha_1(q_1) = logp(u_1|q_1)$ and $log\beta_T(q_T) = 0$.

Remember that in our setting, $p(u_t|q_t) = i \sim \mathcal{N}(\mu_i, \Sigma_i)$ and $p(q_{t+1}|q_t)$ is represented by the transition matrix A. Thus, every term in the recursions is known.

There is also a small trick to compute the logarithm of a sum of a vector.

$$log\sum_i exp(a_i) = a^* + log\sum_i exp(a_i - a^*) \tag{1.3}$$

where $a^* = \underset{i}{argmax} a_i$.

The computation of $\log\alpha$ is implemented in *forward.m* and that of $\log\beta$ in *backward.m*. The small trick mentioned above is implemented in *logsumexp.m*.

Once we have α and β, we can compute $p(q_t|u_1,...,u_T)$ and $p(q_t, q_{t+1}|u_1,...,u_T)$. From the lecture notes,

$$p(q_t|u_1,...,u_T) = \frac{\alpha_t(q_t)\beta_t(q_t)}{\sum_{q_t}\alpha_t(q_t)\beta_t(q_t)}, \forall t \in \{1,...,T\} \tag{1.4}$$

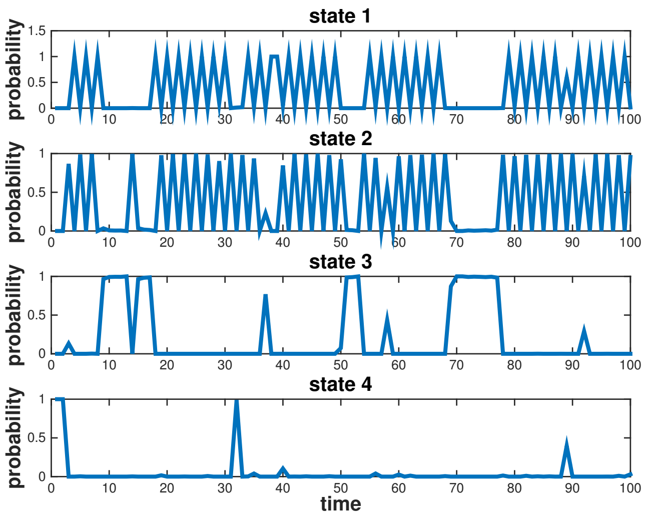
$$p(q_t, q_{t+1}|u_1,...,u_T) = \frac{\alpha_t(q_t)\beta_{q+1}(q_{t+1})p(q_{t+1}|q_t)p(u_{t+1}|q_{t+1})}{\sum_{q_t}\alpha_t(q_t)\beta_t(q_t)}, \forall t \in \{1,...,T-1\} \tag{1.5}$$

Q2. 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, and setting A to be the matrix with diagonal coefficients $A_{ii} = \frac{1}{2}$ and off-diagonal coefficients $A_{ij} = \frac{1}{6}$ for all $(i,j) \in \{1,2,3,4\}^2$, 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 datapoints in the file. Note that only the 100 first points should be plotted but that filtering should be done with all the data (i.e. T = 500). This will be the same for the subsequent questions.

A2. From the previous homework, we take the parameters $\mu_i, \Sigma_i, i \in \{1,2,3,4\}$.

$\mu_1 = \begin{pmatrix} -2.0334 \\ 4.1726 \end{pmatrix}$, $\mu_2 = \begin{pmatrix} 3.9779 \\ 3.7735 \end{pmatrix}$, $\mu_3 = \begin{pmatrix} 3.8007 \\ -3.7972 \end{pmatrix}$, $\mu_4 = \begin{pmatrix} -3.0620 \\ -3.5345 \end{pmatrix}$.

Figure 1.1: Marginal probability for test data computed by using initial parameters



$$\Sigma_1 = \begin{pmatrix} 2.9044 & 0.2066 \\ 0.2066 & 2.7562 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 0.2104 & 0.2904 \\ 0.2904 & 12.2392 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 0.9213 & 0.0574 \\ 0.0574 & 1.8660 \end{pmatrix}, \Sigma_4 = \begin{pmatrix} 6.2414 & 6.0502 \\ 6.0502 & 6.1825 \end{pmatrix}.$$

We display in Figure 1.1 the values of $p(q_t|u_1,\dots,u_T)$ as a function of time for each of the 4 states and for the 100 first datapoints, computed by using the initial parameters.

Q3. Derive the estimation equations of the EM algorithm.

A3. We can compute the complete log-likelihood $\ell(\theta)$ with θ the parameters of the model, including π, A, μ_i, Σ_i

$$\begin{aligned}\ell(\theta) &= \log\left(p(q_1)\prod_{t=1}^{T-1} p(q_{t+1}|q_t)\prod_{t=1}^{T} p(u_t|q_t)\right)\\ &= \log p(q_1) + \sum_{t=1}^{T-1}\log p(q_{t+1}|q_t) + \sum_{t=1}^{T}\log p(u_t|q_t)\\ &= \sum_{i=1}^{K}\delta(q_1=i)\log\pi_i + \sum_{t=1}^{T-1}\sum_{i,j=1}^{K}\delta(q_{t+1}=i,q_t=j)\log A_{ij} + \sum_{t=1}^{T}\sum_{i=1}^{K}\delta(q_t=i)\log\mathcal{N}(u_t,\mu_i,\Sigma_i).\end{aligned}$$

According to the Jensen's inequality, we obtain a lower bound on the log-likelihood

$$\log(u_1,\dots,u_T) \geq \mathbb{E}_{\tilde{p}}[\log(q_1,\dots,q_T,u_1,\dots,u_T)] = \mathbb{E}_{\tilde{p}}[\ell(\theta)], \tag{1.6}$$

where $\tilde{p}(q_1,\dots,q_T) = p(z_1,\dots,z_T|u_1,\dots,u_T,\theta_{k-1})$ at k-th step of the EM algorithm and θ_{k-1}

are parameters learned at step $k-1$. Thus, we obtain the following expression

$$\begin{aligned} E_{\tilde{p}}[\ell(\theta)] = \sum_{i=1}^{K} p(q_1 = i|u_1,\dots,u_T,\theta_{k-1}) \log \pi_i \\ + \sum_{t=1}^{T-1} \sum_{i,j=1}^{K} p(q_{t+1} = i, q_t = j|u_1,\dots,u_T,\theta_{k-1}) \log A_{ij} \\ + \sum_{t=1}^{T} \sum_{i=1}^{K} p(q_t = i|u_1,\dots,u_T,\theta_{k-1}) \log \mathcal{N}(u_t,\mu_i,\Sigma_i), \end{aligned} \tag{1.7}$$

where all conditional probabilities have been computed by the previous questions. We can now minimize $E_{\tilde{p}}[\ell(\theta)]$ with respect to π subject to $\sum_{i=1}^{K} \pi_i = 1$ and obtain for $i = 1,\dots,K$

$$\pi_i^k = \frac{p(q_1 = i|u_1,\dots,u_T,\theta_{k-1})}{\sum_{i=1}^{K} p(q_1 = i|u_1,\dots,u_T,\theta_{k-1})}. \tag{1.8}$$

By taking the derivative respect to A_{ij} subject to $\sum_{i=1}^{K} A_{ij} = 1$, we obtain for $i, j = 1,\dots,K$

$$A_{ij}^k = \frac{\sum_{t=1}^{T-1} p(q_{t+1} = i, q_t = j|u_1,\dots,u_T,\theta_{k-1})}{\sum_{t=1}^{T-1} p(q_t = j|u_1,\dots,u_T,\theta_{k-1})}. \tag{1.9}$$

And by setting the derivative of μ_i and Σ_i equal to 0, we obtain

$$\mu_i^k = \frac{\sum_{t=1}^{T} u_t p(q_t = i|u_1,\dots,u_T,\theta_{k-1})}{\sum_{t=1}^{T} p(q_t = i|u_1,\dots,u_T,\theta_{k-1})}, \tag{1.10}$$

$$\Sigma_i^k = \frac{\sum_{t=1}^{T} p(q_t = i|u_1,\dots,u_T,\theta_{k-1})(u_t - \mu_i^k)(u_t - \mu_i^k)^T}{\sum_{t=1}^{T} p(q_t = i|u_1,\dots,u_T,\theta_{k-1})}. \tag{1.11}$$

Q4. Implement the EM algorithm to learn the parameters of the model $(\pi, A, \mu_k, \Sigma_k, k = 1,\dots,4)$. The means and covariances could be initialized with the ones obtained in the previous homework. Learn the model from the training data.

A4. We implemented the EM algorithm using the previous question.

Q5. Plot the log-likelihood on the training data and on the test data as a function of the iterations of the algorithm. Comment.

A5. We display in Figure 1.2 the log-likelihood curves as a function of the iterations for the training and test data.

Q6. 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.

Figure 1.2: Log-likelihood as a function of the iterations of the EM algorithm for the training and test data

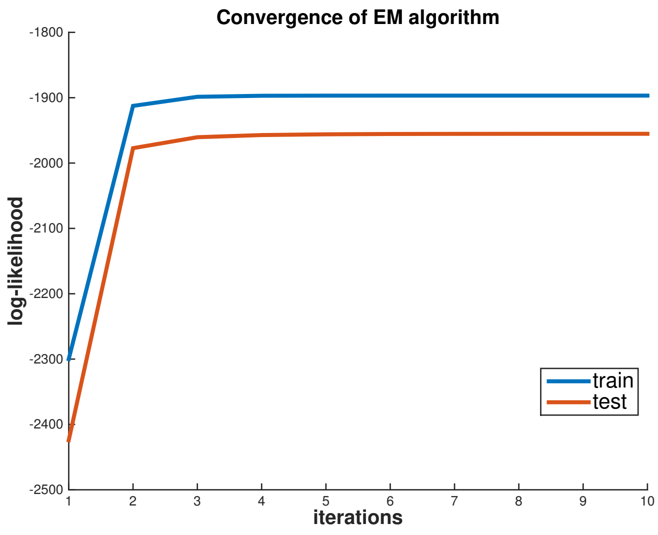


Table 1.1: Comparison of the normalized log-likelihoods of different models

Model	train	test
Gaussian mixture (isotropic)	-5.2910	-5.3882
Gaussian mixture (general)	-4.6554	-4.8180
HMM	-3.7936	-3.9105

A6. We show in Table 1.1 and 1.2 respectively the normalized log-likelihoods and log-likelihoods computed with different models. We notice that the log-likelihoods of the test data are on average lower than the training ones. The log-likelihoods obtained by using HMM are greater than the ones obtained by Gaussian mixture models. However, this comparison doesn't make sense as two models are different and the number of parameters is also different for two models.

Q7. 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

Table 1.2: Comparison of the log-likelihoods of different models

Model	train	test
Gaussian mixture (isotropic)	-2.6455×10^3	-2.6941×10^3
Gaussian mixture (general)	-2.3277×10^3	-2.4090×10^3
HMM	-1.8968×10^3	-1.9552×10^3

states, i.e. $\underset{q}{argmax} p(q_1,...,q_T|y_1,...,y_T)$.

A7. The max-product algorithm replaces the summation in the sum-product algorithm by a maximization. It solves the related problem of computing the most probable configuration of the variables, and particularly in the case of HMM it is referred as Viterbi algorithm which computes the most likely state sequence given fixed parameters. While the algorithm is very similar to the forward recursion, we also use a backpointer to track the configuration that achieves the maximum. The pseudo-code is displayed in Algorithm 1.

Algorithm 1: Viterbi algorithm

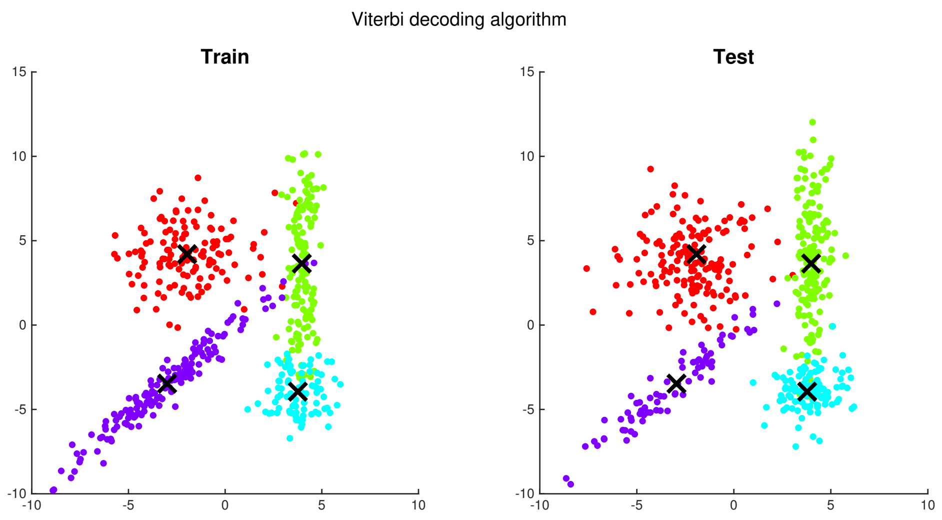
```
Input: Observations $u$, number of states $K$, initial probability distribution $\pi$, probability
       transition matrix $A$, Gaussian emission paramters $\mu_i, \Sigma_i$.
for $i = 1 : K$ do
    $\alpha_1(i) = \pi_i \mathcal{N}(u_1, \mu_i, \Sigma_i)$;
    $S_1(i) = 0$;
end
for $t = 2 : T$ do
    for $i = 1 : K$ do
        $\alpha_t(i) = \max_k \alpha_{t-1}(k) A_{ik}$;
        $S_t(i) = \arg\max_k \alpha_{t-1}(k) A_{ik}$;
        $\alpha_t(i) = \alpha_t(i) \mathcal{N}(u_t, \mu_i, \Sigma_i)$;
    end
end
$q_T = \arg\max_i \alpha_T(i)$;
for $t = (T-1) : 1$ do
    $q_t = S_{t+1}(q_{t+1})$;
end
Output: $q$
```

Q8. 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 datapoints belonging to different classes.

A8. We show in Figure 1.3 the most likely states deduced by the Viterbi algorithm for the training and test data. The black crosses correspond to the cluster centers.

Q9. 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).

Figure 1.3: Most likely states computed by the Viterbi algorithm



A9. We display in Figure 1.4 the marginal probability as a function of time for each state and for the 100 first points.

Q10. 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. Q11. 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.

A10.11. We show in Figure 1.5 the most likely sequences of states for the 100 first datapoints respectively according to the marginal probability and to the Viterbi algorithm. We notice that two curves are identical, which shows the correctness of the Viterbi algorithm.

Q12. In this problem the number of states was known. How would you choose the number of states if you did not know it?

A12. We could make a plot of log-likelihood versus the number of states and make a choice at maximum. A more robust way is to carry out a cross validation and make choice according to the log-likelihoods on the validation sets.

Figure 1.4: Marginal probability for test data with the parameters leaned by the training data

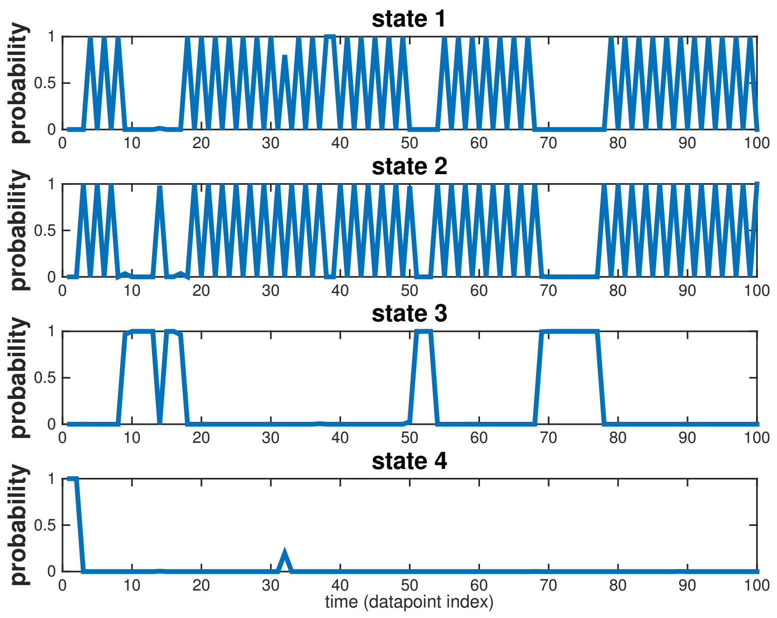
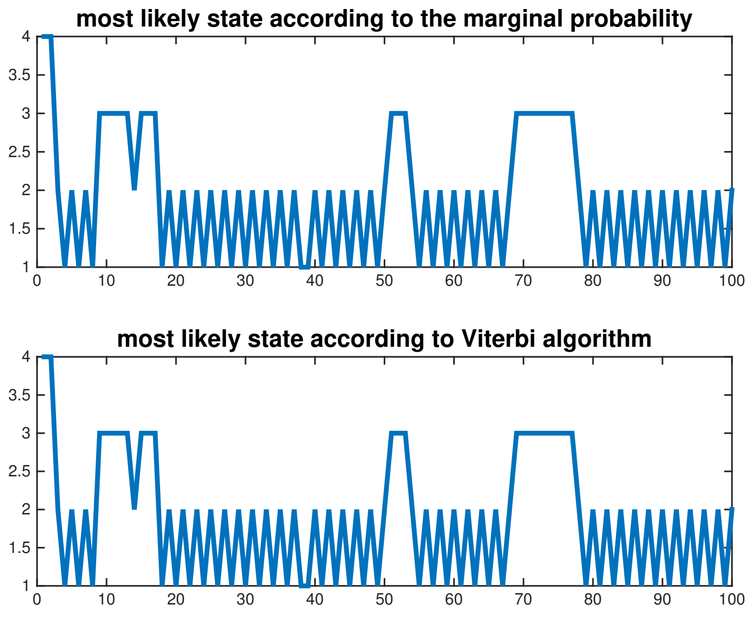


Figure 1.5: Most likely sequence of states for the 100 first datapoints according to the marginal probability and to the Viterbi algorithm


most likely state according to the marginal probability
most likely state according to Viterbi algorithm