Probabilistic Graphical Models - HW3

Gurvan L'Hostis[*], Ronan Riochet[†]

January 2016

1 HMM - Implementation

2. See figure 5.

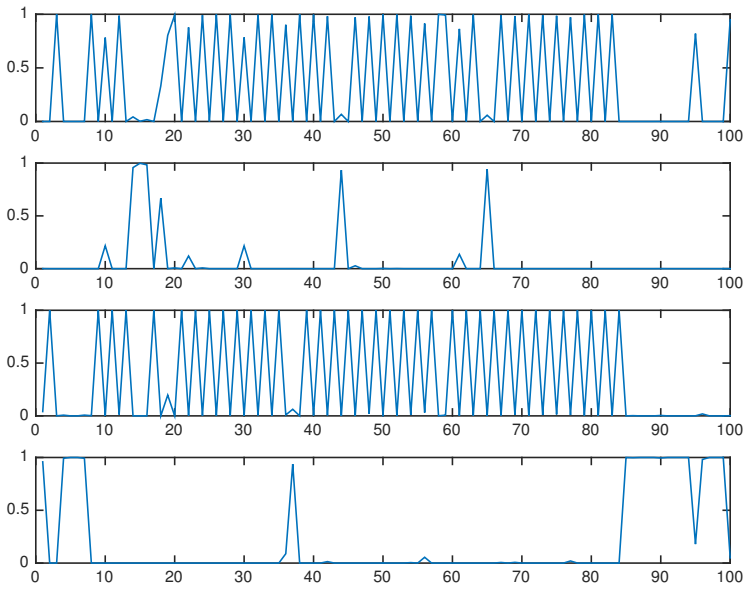

Figure 1: Probabilities of belonging to state 1 to 4 (from top to bottom for the first 100 points) drawn from EM algorithm.

[*]gurvan.lhostis@polytechnique.edu
[†]ronan.riochet@polytechnique.edu

3. We first derive the log-likelihood expression:

$$\begin{aligned}
\ell_c(\theta) &= \log\Big(p(q_0)\prod_{t=0}^{T-1} p(q_{t+1}|q_t)\prod_{t=0}^{T} p(\overline{u_t}|q_t)\Big) \\
&= \log(p(q_0)) + \sum_{t=0}^{T-1}\log p(q_{t+1}|q_t) + \sum_{t=0}^{T}\log p(\overline{u_t}|q_t) \\
&= \sum_{i=1}^{K}\delta(q_0=i)\log(\pi_i) + \sum_{t=0}^{T-1}\sum_{i,j=1}^{K}\delta(q_{t+1}=i,q_t=j)\log(A_{i,j}) + \sum_{t=0}^{T}\sum_{i=1}^{K}\delta(q_t=i)\log f(\overline{u_t},q_t)
\end{aligned}$$

The expectation step gives the following formulae:

$$\begin{aligned}
\mathbb{E}_Q[\ell_c(\theta^{k-1})] &= \sum_{i=1}^{K}\mathbb{E}[\delta(q_0=i)|\overline{u}]\log(\pi_i) + \sum_{t=0}^{T-1}\sum_{i,j=1}^{K}\mathbb{E}[\delta(q_{t+1}=i,q_t=j|\overline{u}]\log(A_{i,j}) \\
&\quad + \sum_{t=0}^{T}\sum_{i=1}^{K}\mathbb{E}[\delta(q_t=i)|\overline{u}]\log f(\overline{u_t},q_t) \\
&= \sum_{i=1}^{K} p(q_0=i|\overline{u};\theta^{k-1})\log(\pi_i) + \sum_{t=0}^{T-1}\sum_{i,j=1}^{K} p(q_{t+1}=i,q_t=j|\overline{u};\theta^{k-1})\log(A_{i,j}) \\
&\quad + \sum_{t=0}^{T}\sum_{i=1}^{K} p(q_t=i|\overline{u};\theta^{k-1})\left(-\tfrac{1}{2}\log|\Sigma_j| - \tfrac{1}{2}(u_t-\mu_j)^\top\Sigma_j^{-1}(u_t-\mu_j) - \tfrac{d}{2}\log(2\pi)\right)
\end{aligned}$$

We can then maximise on the model parameters:

$$\begin{aligned}
\pi_i^k &:= p(q_0=i|\overline{u};\theta^{k-1}) \\
\mu_i^k &:= <u_t>_{p(q_t=i|\overline{u};\theta^{k-1})} \\
\Sigma_i^k &:= <(u_t-\mu_i)(u_t-\mu_i)^T>_{p(q_t=i|\overline{u};\theta^{k-1})} \\
A_{ij}^k &:= \frac{p(q_{t+1}=i,q_t=j|\overline{u};\theta^{k-1})}{\sum_{j'}p(q_{t+1}=i,q_t=j'|\overline{u};\theta^{k-1})}
\end{aligned}$$

4. We can plot the results of EM (see figure 2) and observe that the HMM makes predictions that are not possible with the Gaussian Mixture Model. For example, violet points appear inside the blue cluster.

5. Log-likelihoods are plotted in figure 3. Likelihood is non-decreasing as it is supposed to be. We also observe that the it is smaller on the test set, which means that the model overfits. It is normal since the learnt parameters can never be those that originated the data.

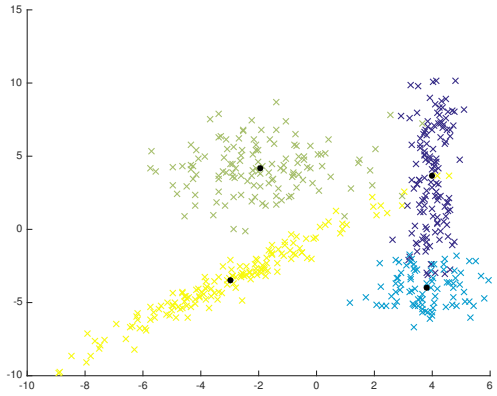

Figure 2: Plot of the EM results with colours corresponding to labels as well as cluster center.

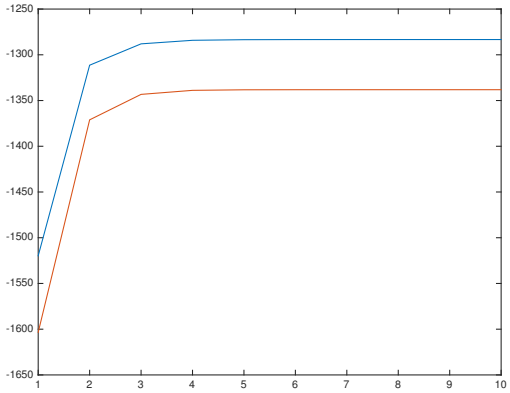

Figure 3: Log-likelihood on the train set (blue) and test set (red) during the 10 iterations of EM.

6.

	Train set	Test set
GMM	-2327	-2408
HMM	-1283	-1338

We observe two things:

- HMM produces smaller likelihoods than GMM;
- The difference between test and train is much higher for GMM than HMM.

HMM is a more elaborate model than GMM, it is thus normal to obtain a larger likelihood on the train set with it and it could only be caused by overfitting. However, it does make sense to compare likelihoods on the test sets as the likelihood is not fitted to them, it gives a measure of generalisation ability of the models.

We can conclude from this table that HMM is models our data better.

7. Viterbi algorithm uses dynamic programming to infer on the most likely sequence of hidden variables. In a first step, it stores for each time t the most likely value $v_i(t)$ of the hidden variable, knowing $v_{i=1...K}(t-1)$. In a second step, the "Viterbi path" is retrieved by saving "back pointers" that remember which state x was solution of the maximization above.

Data:
$X(1...T)$: observations
A: transition matrix
K: number of states
μ, σ: parameters of the model
π: initial probability distribution
Result: Most likely sequence of hidden variables

$v_i(t) = \pi_i * p(X(1) = x_i|y_i),\ \forall i \in 1, ..., K$

for *t=2,...,T* **do**
  for *i=1,...,K* **do**
    $v_i(t) = \max_j P(X(t) = x_i|y_i) \cdot v_j(t-1) \cdot A_{j,i}$
    $argv_i(t) = \text{argmax}_j P(X(t) = x_i|y_i) \cdot v_j(t-1) \cdot A_{j,i}$
  end
end

$result(T) = \text{argmax}_j v_j(T)$
for *t=T-1,...,1* **do**
  $result(t) = argv_{result(t+1)}(t)$
end

Algorithm 1: Viterbi algorithm

8. See figure 4. The results are the same as with EM.

9. See figure 5

10. 11. See figure 6. Here we can say that, conditionally to visible variables, the sequence of the most likely hidden variables and the most likely sequence of hidden variables are the same.

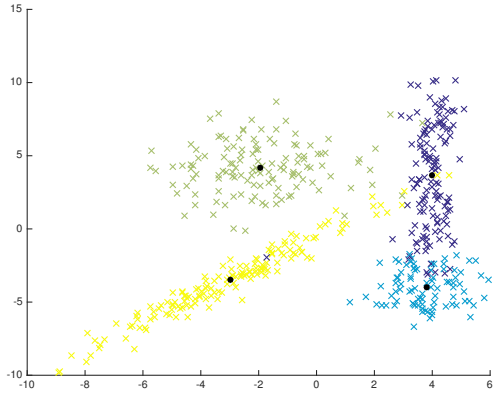

Figure 4: Plot of the Viterbi results with colours corresponding to labels as well as cluster center.

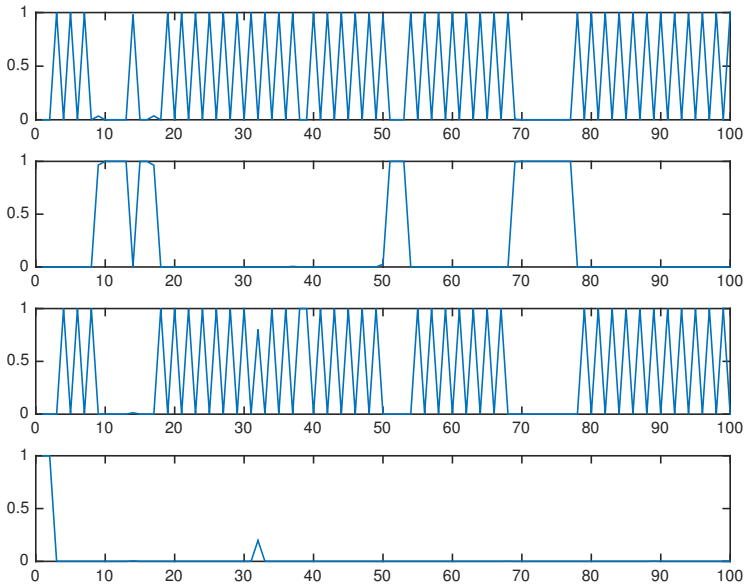

Figure 5: Probabilities of belonging to state 1 to 4 from top to bottom for the first 100 test points.

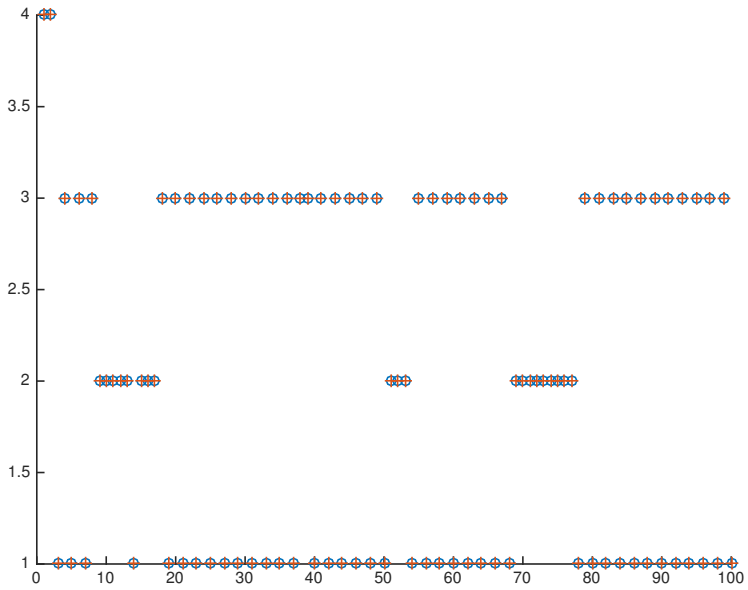

Figure 6: Most probable state for the .

12. We can apply our algorithm with several values of k and then use the elbow rule: when adding clusters improves likelihood significantly less (the elbow points), it is the best number of clusters. Unfortunately here it is very hard to use this rule. See figure 7 for values 2 to 5, we had numerical instabilities for bigger values.

The second idea is to compare the likelihood on the test data, for different k, and choose the k giving maximum value. On figure 8 we see this maximum is indeed reached in $k = 4$.

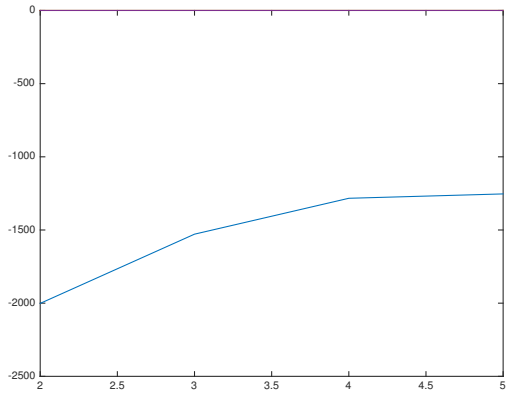

Figure 7: Log-likelihood vs number of clusters for the EM algorithm, train data.

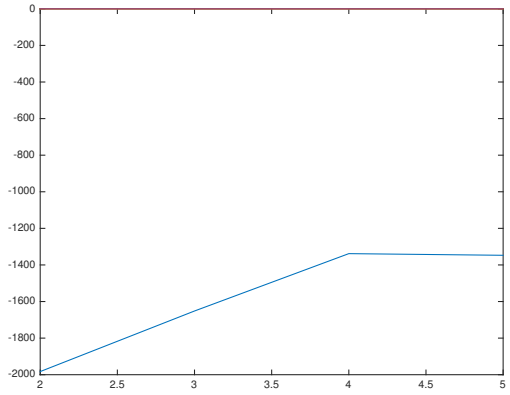

Figure 8: Log-likelihood vs number of clusters for the EM algorithm, test data.