Homework 3: Graphical models

1 α, β,γ and ξ recursions

The recursion α, β, γ and ξ are defined as follow:

$$\alpha_t(q_t) \triangleq p(q_t, u_0, \ldots, u_t)$$
$$\beta_t(q_t) \triangleq p(u_{t+1}, \ldots, u_T \mid q_t)$$
$$\gamma_t(q_t) \triangleq p(q_t \mid u_1, ..., u_T)$$
$$\xi_t(q_t, q_{t+1}) \triangleq p(q_t, q_{t+1} \mid u_1, ..., u_T)$$

And they can be computed recursively using:

$$\alpha_1(q_1) = p(u_1 \mid q_1)p(q_1)$$
$$\alpha_{t+1}(q_{t+1}) = p(u_{t+1} \mid q_{t+1}) \sum_{q_t} p(q_{t+1} \mid q_t)\alpha_t(q_t)$$
$$\beta_T(q_T) = 1$$
$$\beta_t(q_t) = \sum_{q_{t+1}} p(q_{t+1} \mid q_t)p(u_{t+1} \mid q_{t+1})\beta_{t+1}(q_{t+1})$$

Additionally we have:

$$p(q_{t+1} = j \mid q_t = i) = a_{ij}$$
$$p(u_t \mid q_t) = \frac{1}{2\pi\sqrt{|\Sigma_{q_t}|}} \exp\left(-\frac{1}{2}(u_t - \mu_{q_t})^\intercal \Sigma_{q_t}^{-1}(u_t - \mu_{q_t})\right)$$

Which enables us to compute recursively the γ recursion:

$$\gamma_t(q_t) = \frac{\alpha_t(q_t)\beta_t(q_t)}{\sum_q \alpha_t(q)\beta_t(q)}$$

And the ξ recursion:

$$\xi_t(q_t, q_{t+1}) = \frac{\alpha_t(q_t)\gamma_{t+1}(q_{t+1})p(u_{t+1} \mid q_{t+1})p(q_{t+1} \mid q_t)}{\alpha_{t+1}(q_{t+1})}$$

We take care of numerical errors by using the identity:

$$\log \sum_i x_i = \max_i \log(x_i) + \log \sum_i \exp(\log(x_i) - \max_i \log(x_i))$$

2 Plot γ for the training dataset

We can thus plot the probability $\gamma_t(q_t) = p(q_t \mid u_1, ..., u_T)$ for the first 100 points of the training dataset:

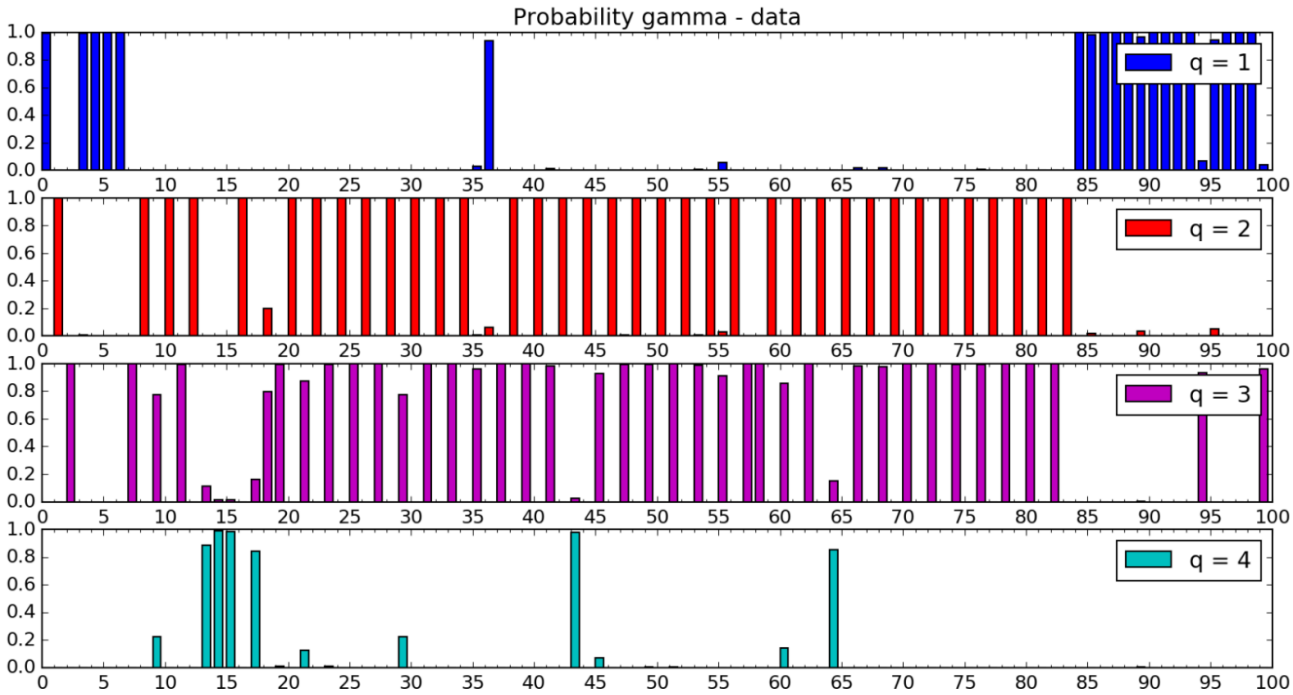

Figure 1: $\gamma_t(q_t) = p(q_t \mid u_1, ..., u_T)$ for the first 100 points of the training dataset.

3 EM algorithm

Now that we have computed the expected probability $\gamma_t(q_t)$ and $\xi_t(q_t, q_{t+1})$ in the previous section we can focus on the estimation of the parameters. We observe that:

$$\begin{aligned}\log p(U, Q \mid \theta) &= \log p(u_1, \dots, u_T, q_1, \dots, q_T \mid \theta) \\ &= \underbrace{\sum_{k=1}^{K} q_1^k \log \pi_k}_{p(q_1)} + \underbrace{\sum_{t=1}^{T-1} \sum_{i=1}^{K} \sum_{j=1}^{K} q_t^i q_{t+1}^j \log a_{ij}}_{p(q_{t+1}|q_t)} + \underbrace{\sum_{t=1}^{T} \sum_{k=1}^{K} q_t^k \log \mathcal{N}(u_t, \mu_k, \Sigma_k)}_{p(u_t|q_t)}\end{aligned}$$

From this we deduce the maximization steps:

$$\begin{aligned}\hat{\pi}_k &= \gamma_1(k) \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\ \hat{\mu}_k &= \frac{\sum_{t=1}^{T} u_t \gamma_t(k)}{\sum_{t=1}^{T} \gamma_t(k)} \\ \hat{\Sigma}_k &= \frac{\sum_{t=1}^{T} (u_t - \hat{\mu}_k)(u_t - \hat{\mu}_k)^\intercal \gamma_t(k)}{\sum_{t=1}^{T} \gamma_t(k)}\end{aligned}$$

5 log-likelihood

We plot the log-likelihood as a function of the number of iterations for both dataset and we obtain the figures below:

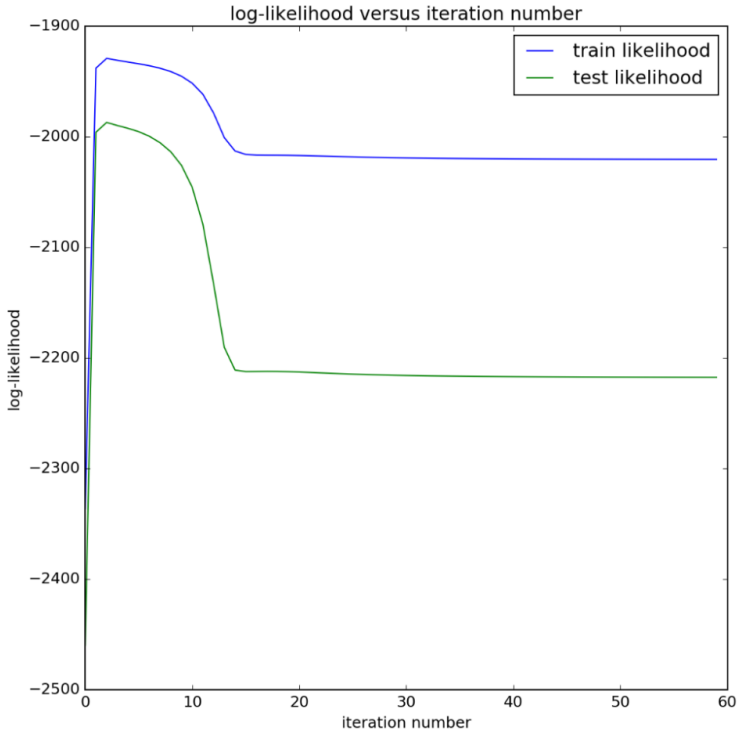

Figure 2: log-likelihood for the train and test dataset.

First we observe that the two have the same overall behavior and that only few iterations are needed for the log-likelihood to converge for both datasets. Finally the train likelihood is better than the test likelihood which isn't a big surprise has the model is specifically trained to fit the train dataset.

6 HMM vs Gaussian Mixture

We have the following

log-likelihood		
Algorithm type	**Training dataset**	**Test dataset**
HMM	-2018	-2215
Gaussian Mixture	-2328	-2409

Although the HMM use a multivariate gaussian for the observed points generation the models are totally different in the sense that the Gaussian Mixture doesn't consider the time transition from one state to the other. This implies that directly comparing the log-likelihood is not really a suited.

7 Viterbi pseudo-code

The Viterbi algorithm stems from the fact that:

$$\begin{aligned}
\max_{q_0,\dots,q_T} p(q_0,\dots,q_T \mid u_0,\dots,u_T) &= \max_{q_0,\dots,q_T} p(q_0,\dots,q_T,u_0,\dots,u_T) \\
&= \max_{q_0,\dots,q_T} \left(p(q_0)p(y_0 \mid q_0) \prod_{t=1}^{T} p(q_t \mid q_{t-1})p(u_t \mid q_t) \right) \\
&= \max_{q_T} \left(\max_{q_0,\dots,q_{T-1}} \left(p(q_0)p(y_0 \mid q_0) \prod_{t=1}^{T} p(q_t \mid q_{t-1})p(u_t \mid q_t) \right) \right) \\
&= \max_{q_T} \left(p(u_T \mid q_T) \max_{q_0,\dots,q_{T-1}} \left(p(q_T \mid q_{T-1})p(q_0)p(y_0 \mid q_0) \prod_{t=1}^{T-1} p(q_t \mid q_{t-1})p(u_t \mid q_t) \right) \right) \\
&= \max_{q_T} \left(p(u_T \mid q_T) \max_{q_{T-1}} \left(p(q_T \mid q_{T-1})p(u_{T-1} \mid q_{T-1}) \dots \left(\max_{q_0} p(q_0)p(y_0 \mid q_0)p(q_1 \mid q_0) \right) \right) \right)
\end{aligned}$$

Now considering only the part $\max_{q_0} p(q_0)p(y_0 \mid q_0)p(q_1 \mid q_0)$, for each value q_1 we compute the best q_0 that maximize this value and we store both the value and the argument.
Then for $\max_{q_1} p(u_1 \mid q_1)p(q_2 \mid q_1) \max_{q_0} p(q_0)p(y_0 \mid q_0)p(q_1 \mid q_0)$ we compute the best q_1 that maximizes it for each q_2 and we iterate which yields the pseudo-code:

for *all* q_1 **do**
  find $q_0^*(q_1)$ that maximizes $\delta_0^*(q_1) = p(q_0)p(u_0 \mid q_0)p(q_1 \mid q_0)$;
  store $\delta_0^*(q_1)$;
  store $q_0^*(q_1)$ in $state^*(q_1)$;
end
for *t = 1 to T-1* **do**
  for *all* q_{t+1} **do**
    find $q_t^*(q_{t+1})$ that maximizes $\delta_t^*(q_{t+1}) = p(u_t \mid q_t)p(q_{t+1} \mid q_t)\delta_{t-1}^*(q_t)$;
    store $\delta_t^*(q_{t+1})$;
    store $q_t^*(q_{t+1})$ by appending it to $state^*(q_{t+1})$;
  end
end
for *all* q_T **do**
  find q_T^* that maximizes $\delta_T^* = p(u_T \mid q_T)\delta_{T-1}^*(q_T^*)$;
  store q_T^* by appending it to $state^*(q_T)$;
end
return $state^*$

Algorithm 1: Viterbi pseudo-code for HMM

8 Viterbi decoding

After implementing the Viterbi decoding we can plot the most likely sequence of states and represent each datapoint of the training dataset and its cluster, in the figure below:

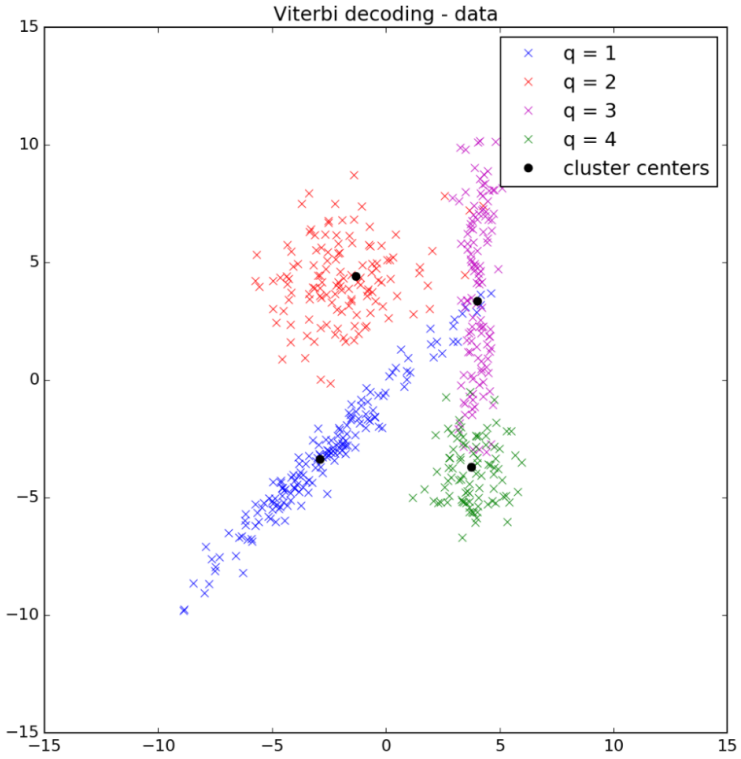

Figure 3: Viterbi decoding for the training dataset.

9 Plot γ for the testing dataset

We can thus plot the probability $\gamma_t(q_t) = p(q_t \mid u_1, ..., u_T)$ for the first 100 points of the testing dataset:

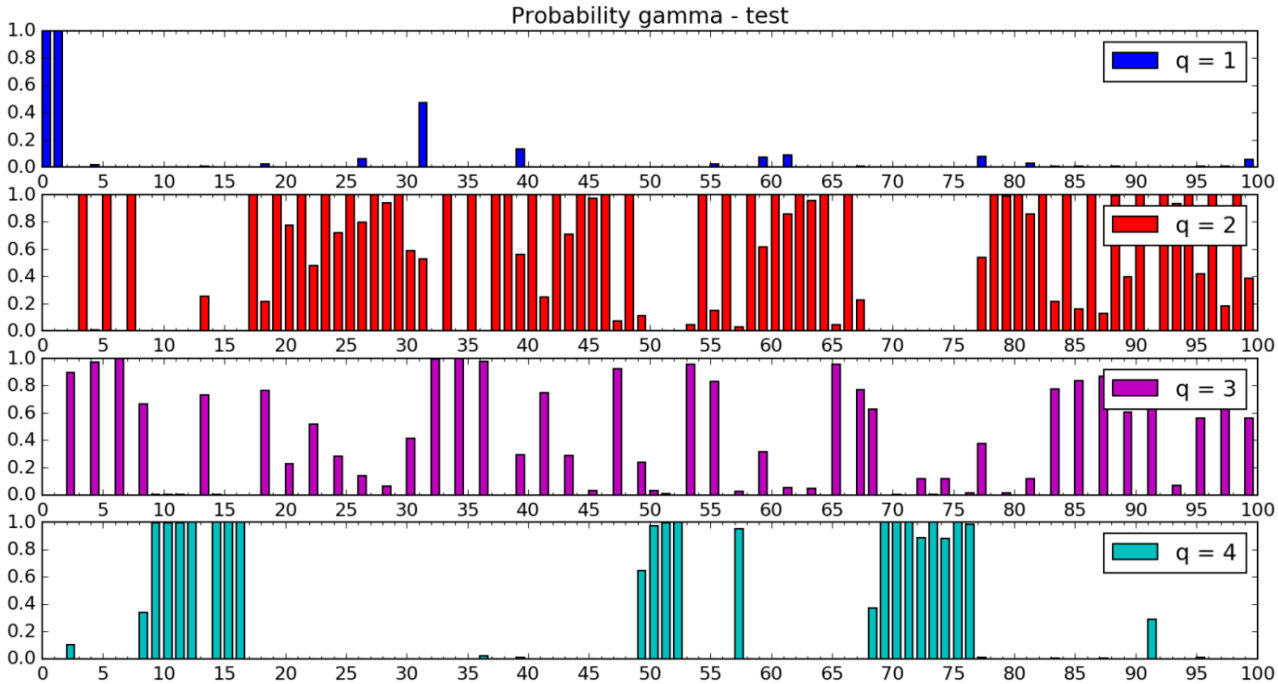

Figure 4: $\gamma_t(q_t) = p(q_t \mid u_1, ..., u_T)$ for the first 100 points of the testing dataset.

10 Marginal probability estimation

Now we plot the hidden variable state for the testing dataset using the marginal probability.

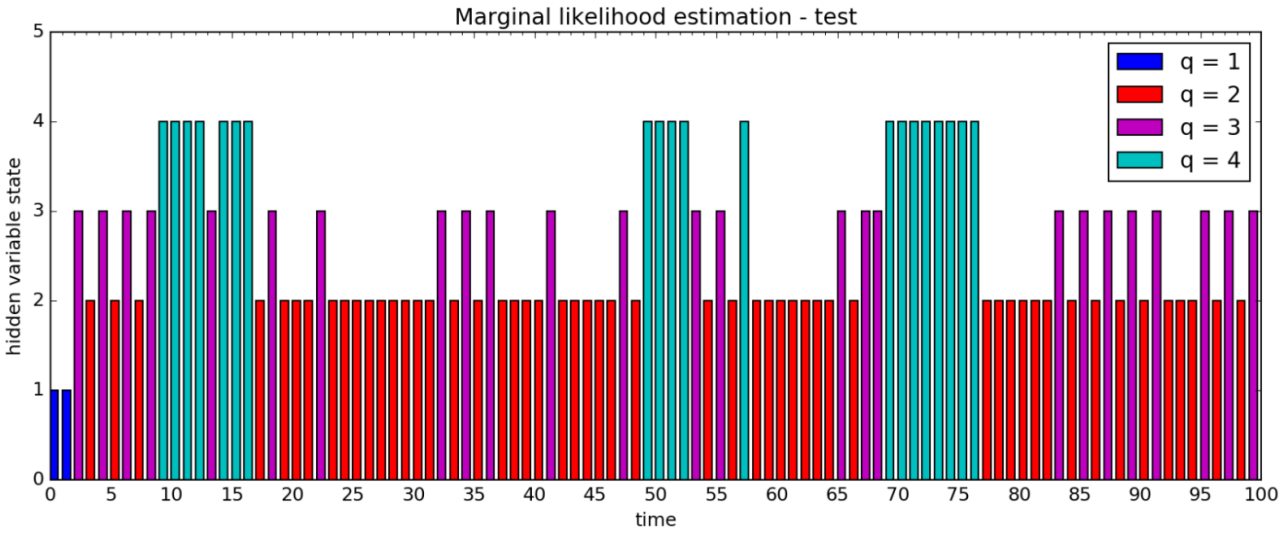

Figure 5: Hidden variable state using the marginal probability.

We can also represent the corresponding point and their cluster:

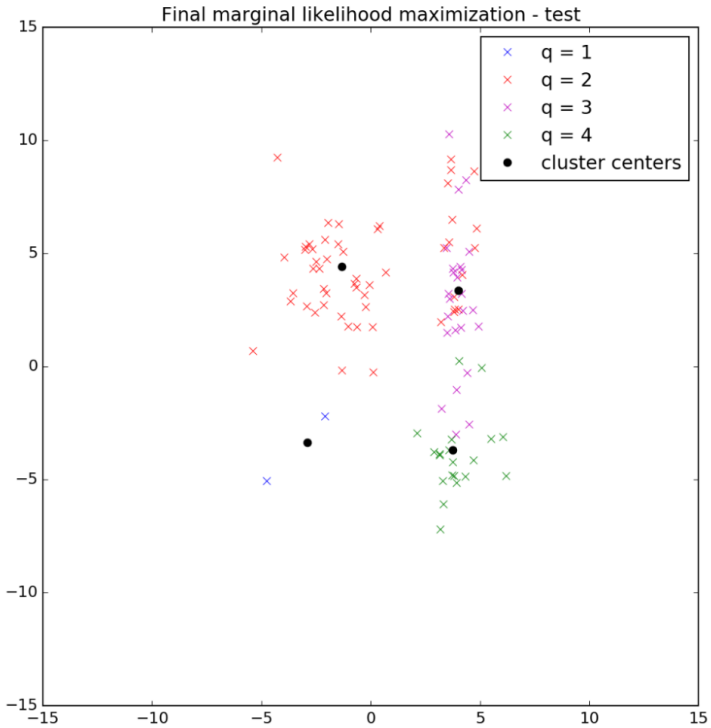

Figure 6: Hidden variable state using the marginal probability.

11 Viterbi decoding

Now we plot the hidden variable state for the testing dataset using the Viterbi decoding.

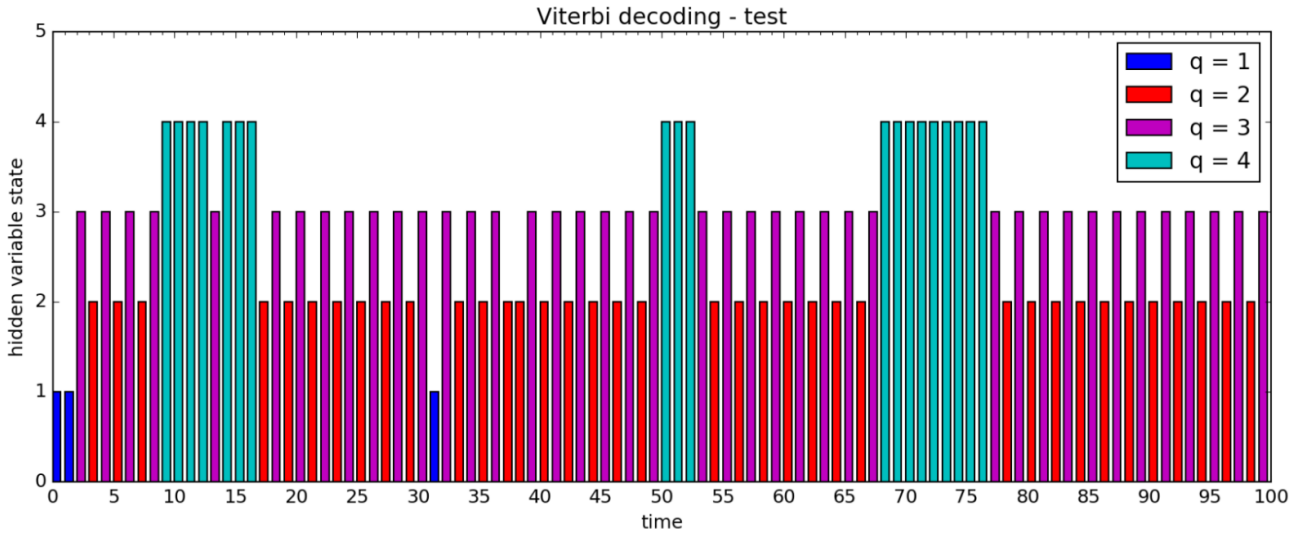

Figure 7: Hidden variable state using the viterbi decoding.

We can also represent the corresponding point and their cluster:

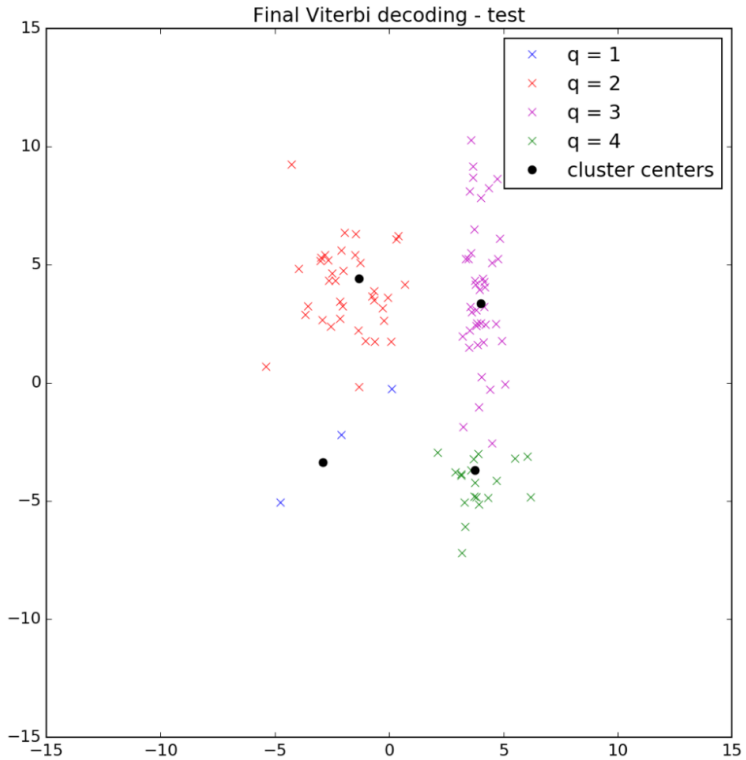

Figure 8: Hidden variable state using the viterbi decoding.

We observe that the two plots are not the same thus the most likely overall sequence is very different from the sequence of most likely states.

12 Number of states

In this exercice the number of states was a given parameter. If we didn't know it we could have used cross-validation to find the most-likely model by trying different state number.