As usual, please hand in on paper form your derivations and answers to the questions. You can use any programming language for your source code (submitted on Studium as per the website instructions). All the requested figures should be printed on paper with clear titles that indicate what the figures represent.

1 Entropy and Mutual Information (18 points)

1. Let X be a discrete random variable on a finite space $\mathcal{X}$ with $|\mathcal{X}| = k$.

   (a) Prove that the entropy $H(X) \geq 0$, with equality only when X is a constant.
   Answer: Assuming X has the distribution P, we have

   $$H(X) = -\sum_x p(x) \log p(x)$$

   where $\log p(x) \leq 0$ and $0 \leq p(x) \leq 1$. So each term in the sum will be negative, and so taking the negative of the sum is positive. If X is constant, this is sum is 0 since $p(x) = 1, \log p(x) = 0$, and so all terms in the sum will be 0. This is the only case of strict positivity. Indeed, assuming otherwise and WLOG removing points of zero probability from the support, then $0 < p(x) < 1$ and $\log p(x) < 0$. Thus each term in the sum is strictly negative and the sum will be strictly positive.

   (b) Denote by p the distribution of X and q the uniform distribution on $\mathcal{X}$. What is the relation between the Kullback-Leibler divergence $D(p\|q)$ and the entropy $H(X)$ of the distribution p?
   Answer: We have

   $$\begin{aligned} H(X) &= -\sum_x p(x) \log p(x) = -\sum_x p(x)(\log p(x) + \log(k) - \log(k)) \\ &= -\sum_x p(x)(\log p(x) + \log(k)) - \sum_x p(x)(-\log(k)) \\ &= -\sum_x (p_x) \log(\frac{p(x)}{\frac{1}{k}}) + \log(k) \sum_x p(x) \\ &= -D(p||q) + \log(k) \sum_x p(x) = -D(p||q) + \log(k) \end{aligned}$$

   (c) Deduce an upper bound on the entropy that depends on k.
   Answer: From the above, we notice that the $\log k$ term is constant in k, while the KL-divergence is minimized when p approaches q, that is, when $p(x)$ is also the uniform on K states. Then

   $$D(p||q) = -\sum_x (p_x) \log \frac{\frac{1}{k}}{\frac{1}{k}} = 0$$

   and the term $-D(p||q)$ in the RHS of the entropy equality relation is maximized, leaving us with the term $\log(k)$. This is indeed a maximum for the entropy since, as seen in class, positivity is a property of the KL-divergence. Thus $\log(k)$ is an upper bound on $H(X)$ that depends on k.

2. We consider a pair of discrete random variables (X_1, X_2) defined over the finite set $\mathcal{X}_1 \times \mathcal{X}_2$. Let $p_{1,2}$, p_1 and p_2 denote respectively the joint distribution, the marginal distribution of X_1 and the marginal distribution of X_2. The mutual information $I(X_1, X_2)$ is defined as

$$I(X_1, X_2) := \sum_{(x_1,x_2)\in\mathcal{X}_1\times\mathcal{X}_2} p_{1,2}(x_1, x_2) \log \frac{p_{1,2}(x_1, x_2)}{p_1(x_1)p_2(x_2)}.$$

(a) Prove that $I(X_1, X_2) \geq 0$.
Answer: We will use Jensen's inequality to demonstrate this. We omit the subscripts to the distributions to lighten notation.

$$\begin{aligned}
I(X_1, X_2) &= \sum_{(x_1,x_2)} p(x_1, x_2) \log \frac{p(x_1, x_2)}{p(x_1)p(x_2)} && (1)\\
&= \sum_{(x_1,x_2)} p(x_1, x_2) - \log \frac{p(x_1)p(x_2)}{p(x_1, x_2)} && (2)\\
&\geq -\log \sum_{(x_1,x_2)} p(x_1, x_2) \frac{p(x_1)p(x_2)}{p(x_1, x_2)} && (3)\\
&= -\log \sum_{(x_1,x_2)} p(x_1)p(x_2) = 0 && (4)
\end{aligned}$$

Where the equation 3 follows from Jensen's inequality because the $-\log(f(x))$ function is convex.

(b) Show that $I(X_1, X_2)$ can be expressed as a function of $H(X_1), H(X_2)$ and $H(X_1, X_2)$ where $H(X_1, X_2)$ is the entropy of the random variable $X = (X_1, X_2)$.
Answer:

$$\begin{aligned}
I(X_1, X_2) &= \sum_{(x_1,x_2)} p(x_1, x_2) \log \frac{p(x_1, x_2)}{p(x_1)p(x_2)} && (5)\\
&= \sum_{(x_1,x_2)} p(x_1, x_2) \log p(x_1, x_2) - \sum_{x_1}\sum_{x_2} p(x_1, x_2) \log p(x_1) - \sum_{x_2}\sum_{x_1} p(x_1, x_2) \log p(x_2) && (6)\\
&= \sum_{(x_1,x_2)} p(x_1, x_2) \log p(x_1, x_2) - \sum_{x_1} p(x_1) \log p(x_1) - \sum_{x_2} p(x_2) \log p(x_2) && (7)\\
&= -H(X_1, X_2) + H(X_1) + H(X_2) && (8)
\end{aligned}$$

Where equation 6 follows from developing the logarithm and equation 7 follows from marginalization.

(c) What is the joint distribution $p_{1,2}$ of maximal entropy with given marginals p_1 and p_2?
Answer: From the previous question, we have

$$H(X_1, X_2) = H(X_1) + H(X_2) - I(X_1, X_2)$$

Thus if the distributions of X_1, X_2 are fixed as marginals, we can maximize the entropy of the joint by minimizing the mutual information term. But this will occur exactly when the X_1, X_2 random variables are independent, since then we get the joint will be the product of the marginals:

$$I(X_1, X_2) = \sum_{(x_1,x_2)} p(x_1)p(x_2) \log \frac{p(x_1)p(x_2)}{p(x_1)p(x_2)} = 0$$

And this is the minimization of the mutual information term by part a).

2 HMM – Implementation (82 points)

We consider the same training data as in the previous homework (hwk 3), provided as the `EMGaussian.train` file (and we will test on the corresponding testing data from `EMGaussian.test`), but this time we use an HMM model to account for the possible temporal structure of the data. I.e. we now consider each row of the dataset to be a point $x_t \in \mathbb{R}^2$, where t is the time index (increasing with rows) going from $t = 1, \ldots, T$ rather than thinking of them as *independent* samples as we did in the last homework. The goal of this exercise is to implement the probabilistic inference algorithm (sum-product) on a HMM and its EM algorithm to estimate parameters as well as the Viterbi algorithm to do decoding. It is recommended to make use of the code of the previous homework.

We consider the following HMM model: the chain $(z_t)_{t=1}^T$ has $K = 4$ possible states, with an initial probability distribution $\pi \in \Delta_4$ and a probability transition matrix $A \in \mathbb{R}^{4\times 4}$ where

$$A_{ij} = p(z_t = i | z_{t-1} = j),$$

and conditionally on the current state z_t, we have observations obtained from Gaussian emission probabilities $x_t|(z_t = k) \sim \mathcal{N}(x_t|\mu_k, \Sigma_k)$. This is thus a generalization of a GMM with time dependence across the latent states z_t.

1. Implement the α and β-recursions seen in class (and that can be found in chapter 12 of Mike's book with slightly different notation) to compute the *smoothing* distribution $p(z_t|x_1, \ldots, x_T)$ and pair-marginals $p(z_t, z_{t+1}|x_1, \ldots, x_T)$.
   (Recall that $\alpha(z_t) := p(z_t, x_{1:t})$ and $\beta(z_t) := p(x_{(t+1):T}|z_t)$).

   Answer: The smoothing distribution is given by

   $$\gamma(t) = \frac{\alpha(t)\beta(t)}{p(x)}$$

   and the pair-marginals distribution is given by

   $$p(z_t, z_{t+1}|x_1, ..., x_T) = \frac{\alpha(z_t)\beta(z_{t+1})p(z_{t+1}|z_t)p(x_{t+1}^-|z_{t+1})}{p(x)} = \frac{\alpha(z_t)\beta(z_{t+1})p(z_{t+1}|z_t)\mathcal{N}(x_{t+1}^-|z_{t+1})}{p(x)}$$

where $\alpha(z_t) = \mathcal{N}(\bar{x_t}|z_t) \circ A\alpha(z_{t-1})$, $\beta(z_t) = ((\beta(z_{t+1}) \circ \mathcal{N}(\bar{x_{t+1}}|z_{t+1}))^T)A^T$

We implement the functions and classes in `asg4_module.py`, with the demonstrations in the `Jupyter Notebook` throughout this assignment.

2. **(Fake parameters inference).** Consider using the same parameters for the means and covariance matrix of the 4 Gaussians that you should have learned in hwk 3 (with general covariance matrices) with EM. We give them below for your convenience:

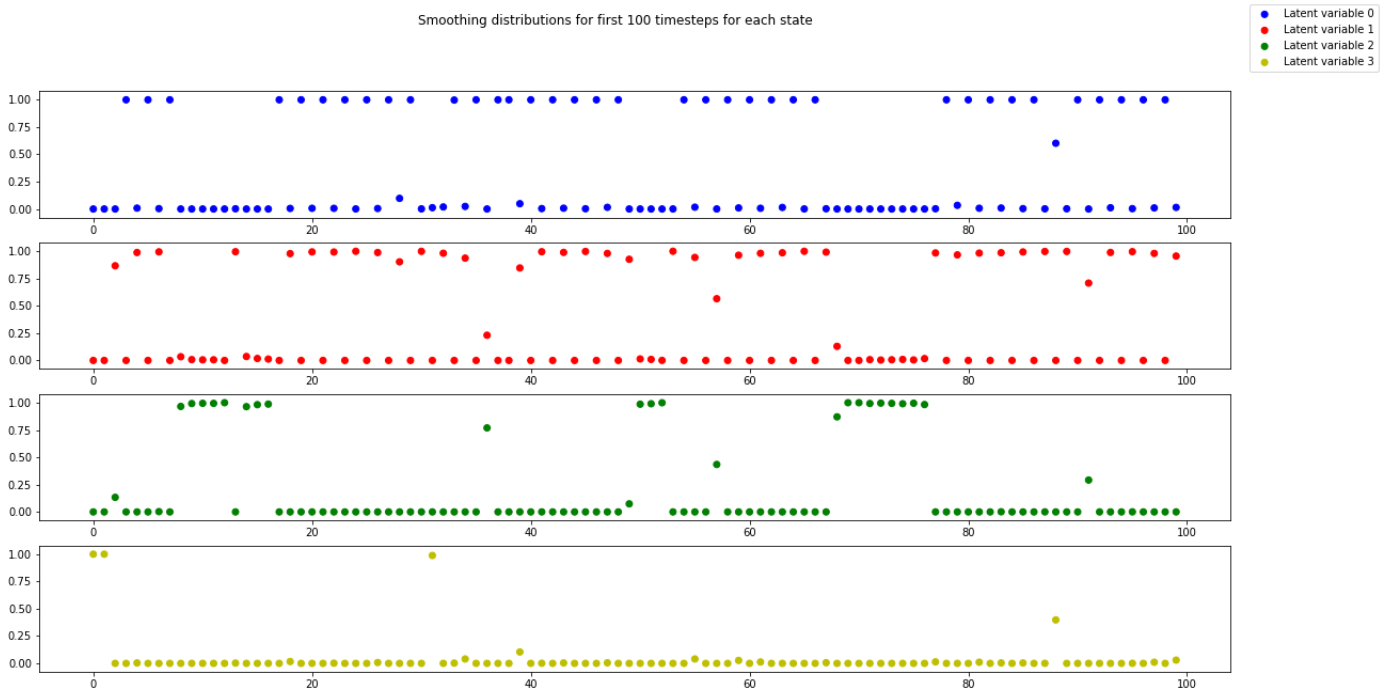
$$\mu_1 = \begin{pmatrix} -2.0344 \\ 4.1726 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} 3.9779 \\ 3.7735 \end{pmatrix} \quad \mu_3 = \begin{pmatrix} 3.8007 \\ -3.7972 \end{pmatrix} \quad \mu_4 = \begin{pmatrix} -3.0620 \\ -3.5345 \end{pmatrix}$$

$$\Sigma_1 = \begin{pmatrix} 2.9044 & 0.2066 \\ 0.2066 & 2.7562 \end{pmatrix} \qquad \Sigma_2 = \begin{pmatrix} 0.2104 & 0.2904 \\ 0.2904 & 12.2392 \end{pmatrix}$$

$$\Sigma_3 = \begin{pmatrix} 0.9213 & 0.0574 \\ 0.0574 & 1.8660 \end{pmatrix} \qquad \Sigma_4 = \begin{pmatrix} 6.2414 & 6.0502 \\ 6.0502 & 6.1825 \end{pmatrix}$$

Using a uniform initial probability distribution $\pi_k = \frac{1}{4}$, 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 $(i, j) \in \{1, \ldots, 4\}^2$, compute the vectors α_t and β_t for all t on the test data `EMGaussian.test` and compute $p(z_t|x_1, \ldots, x_T)$. Finally, represent $p(z_t|x_1, \ldots, x_T)$ for each of the 4 states as a function of t for the 100 first datapoints in the file. Note that only the 100 time steps should be plotted, but the smoothing is always done with all the data (i.e. $T = 500$). This will be the same for the subsequent questions.
(In Matlab/Scipy the command `subplot` might be handy to make multiple long horizontal plots.)

Answer:



3. Derive the M-step update for $\hat{\pi}$, $\hat{A}$, $\hat{\mu}_k$ and $\hat{\Sigma}_k$ (for $k = 1, \ldots, 4$) during the EM algorithm, as a function of the quantities computed during the E step (For the estimate of π, note that we only have *one* long chain here).

Answer:
The complete log likelihood is given by:

$$\log p(x, z|\theta) = \log \prod_{k=1}^{K} p(z_0^k) \prod_{t=0}^{T-1} \prod_{i,j=1}^{K} A_{ij}^{z_t^j z_{t+1}^i} \prod_{t=0}^{T} \prod_{k=1}^{K} N(x_t|\mu_k, \Sigma_k)^{z_t^k}$$

where A_{ij} is the probability of a state transition from state j to state i at the given time step. We denote the expected value of $z_t^k = p(z_t^k = 1|x, \theta)$ as τ_t^k, and the expected value of $z_t^j z_{t+1}^i = p(z_t^j = 1, z_{t+1}^i = 1|x, \theta)$ as $\zeta_{t,j,i}$. Pushing the log through and converting products into sums, we get thus get the expected log-likelihood as :

$$l(x, z, \theta)_{ec} = \log p(x, z|\theta) = \sum_{k=1}^{K} \tau_0^k \log \pi_0^k + \sum_{t=0}^{T-1} \sum_{i,j=1}^{K} \zeta_{t,j,i} \log A_{ij} + \sum_{t=0}^{T} \sum_{k=1}^{K} \tau_t^k \log N(x_t|\mu_k, \Sigma_k)$$

Maximizing the expected complete log-likelihood gives us the M-step updates. Now, we can use our knowledge of maximum-likelihood in exponential families to do this more concisely. The sum involving π_0 has the log-likelihood of a multinoulli distribution. The sufficient statistic for this distribution parameter is thus τ_0 and the empirical mean of the sufficient statistic gives us our maximum likelihood estimate by the properties of exponential families. Since there is only one chain here, we simply get:

$$\hat{\pi_0}_{MLE} = \tau_0$$

Similarly, the sum involving A_{ij} has the form of a multinoulli with sufficient statistic $\sum_{t=0}^{T-1} \zeta_{t,j,i}$. Thus we obtain the ML estimate by taking the empirical average of the sufficient statistic over the transitions from state j to state i :

$$\hat{A_{ij}}_{MLE} = \frac{\sum_{t=0}^{T-1} \zeta_{t,j,i}}{\sum_{i=1}^{K}\sum_{t=0}^{T-1} \zeta_{t,j,i}}$$

As for the MLE for the Gaussian likelihood term, we can use what was done in previous assignments. Indeed, this is essentially the same derivation as what was done for QDA in homework 2, but this time with multiple states instead of two classes. Thus, we get the MLE for μ_k by :

$$\begin{aligned}\nabla l_{\mu_k} &= \nabla_{\mu_k} \frac{-1}{2} \sum_{t=0}^{T} \tau_t^k (x_t - \mu_k)^T \Sigma_k^{-1} (x_t - \mu_k) \\ &= \sum_{t=0}^{T} \Sigma_k^{-1} \tau_t^k (x_t - \mu_k)\end{aligned}$$

Setting to zero gives us :

$$\sum_{t=0}^{T} \tau_t^k (x_t - \mu_k) = 0 \implies \sum_{t=0}^{T} \tau_t^k x_t = \sum_{t=0}^{T} \tau_t^k \mu_k \implies \hat{\mu_k}_{MLE} = \frac{\sum_{t=1}^{T} \tau_t^k x_t}{\sum_{t=0}^{T} \tau_t^k}$$

Similarly, we can re-use the tricks for the Gaussian to obtain:

$$\begin{aligned}\nabla l_{\Sigma_k^{-1}} &= \nabla_{\Sigma_k^{-1}} \sum_{t=0}^{T} \frac{-1}{2} \tau_t^k (x_t - \mu_k)^T \Sigma_k^{-1} (x_t - \mu_k) - \frac{\tau_t^k}{2} \log(|\Sigma_k|) \\ &= -\frac{n_k}{2} \tilde{\Sigma_k} + \frac{n_k}{2} \Sigma_k\end{aligned}$$

where $\tilde{\Sigma_k} = \frac{\sum_{t=0}^{T} \tau_t^k (x_t - \mu_k)(x_t - \mu_k)^T}{n_k}, n_k = \sum_{t=0}^{T} \tau_t^k$

Then setting to zero gives:

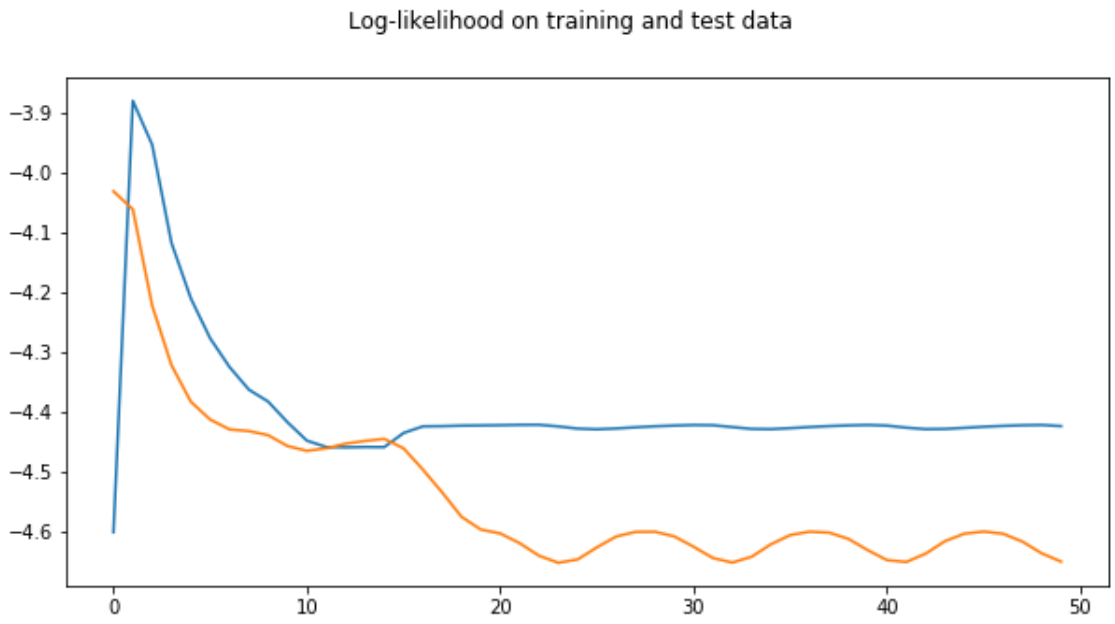
$$\hat{\Sigma_k}_{MLE} = \tilde{\Sigma_k} = \frac{\sum_{t=0}^{T} \tau_t^k (x_t - \mu_k)(x_t - \mu_k)^T}{\sum_{t=0}^{T} \tau_t^k}$$

4. Implement the EM algorithm to learn the parameters of the model $(\pi, A, \mu_k, \Sigma_k, k = 1 \ldots, 4)$. To make grading easier (avoiding the variability coming from multiple local maxima), use the parameters from question 2 for initialization. Learn the model from the training data in `EMGaussian.train`.

   Answer: See asg4_module.py and Jupyter Notebook.

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

   Answer:



   From this figure we observe that the EM algorithm initially pushes the likelihood up sharply to an optimal point, but the algorithm falls out of this maximum point before reaching convergence after slightly less than 20 iterations. This is unusual, as we would normally expect the likelihood to increase on the training set. It likely points to an implementation error, however this was not identified before submission. As expected, the log-likelihood on the test data is consistently slightly lower than that obtained on the training data.

6. Return in a table the values of the log-likelihoods of the (full-covariance) 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.

   Answer:

	Training log-likelihood	**Testing log-likelihood**
HMM	-4.421178	-4.609594
General GMM	-4.741766	-4.907124
Isotropic GMM	-5.478460	-5.437695

   The HMM can be seen as an extension to the general GMM. Thus it makes sense to make the comparison in the context of measuring how much the sequential nature of the data helps the

model. If each data point was generated completely independently from the previous ones, we would expect these two models to perform very similarly. Thus the better performance of the HMM suggests this was not the case in the data generating process. Furthermore, the isotropic GMM is clearly the least powerful of the three, as it does not have the flexibility to capture gaussians of non-spherical shapes. Thus in our case it is clear why it underperforms the general GMM and the HMM.

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. $\arg\max_z p(z_1, \ldots, z_T | x_1, \ldots, x_T)$.

Answer:
The idea of the Viterbi decoding algorithm is that we can compute the most likely sequence of states for the first t timesteps given the most likely sequence of states for the first $t-1$ timesteps and for each possible state at that timestep. Indeed, this is possible due to the recurrence relation

$$V_{t,k} = max_j p(x_t|z_k) \cdot A_{kj} \cdot V_{t-1,j}$$

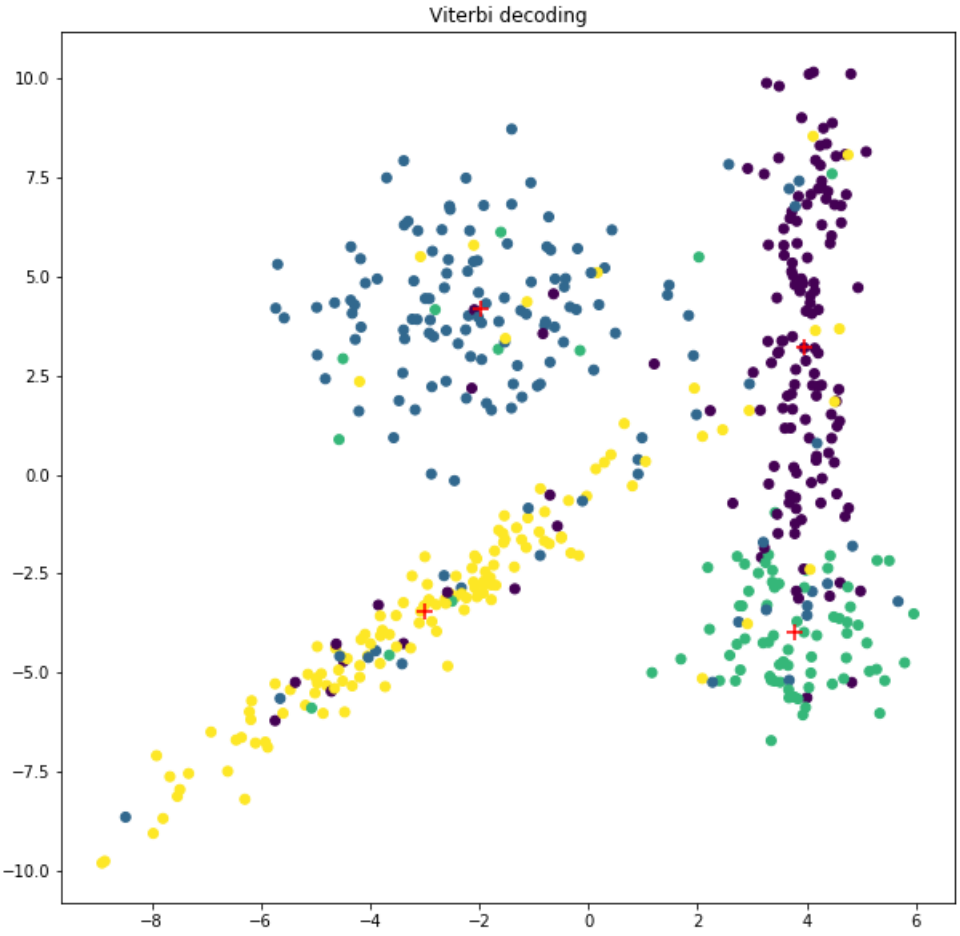
where k and j range over the possible states, and $V_{t,k}$ corresponds to the probability of the most likely sequence of states ending in state k. Thus if $t = T$, we obtain the overall most likely sequence by taking the argmax over the values $V_{T,k}$, and then following pointers back through the recursion which have been kept throughout. The initialization is simply given by the emission probabilities and the parameter π. In pseudocode,using the same notation as seen in class for the parameters and assuming these parameters are available as part of our model, this looks as follows:

Algorithm 1 Viterbi(State sequence $x_0, ..., x_T$)

$S[0, ..., T] \leftarrow null$
for *k in possible states initialize*:
  $V[0, k] \leftarrow p(x_0|z_0 = k)\pi_k$
  $V_{\text{ptr}}[0, k] \leftarrow 0$
for *all timesteps t from 1,..., T*:
  for *k in possible states*:
    $V[t, k] \leftarrow max_j\{p(x_t|z_k) \cdot Akj \cdot V[t-1, j]\}$
    $V_{ptr}[t, k] \leftarrow argmax_j\{p(x_t|z_k) \cdot Akj \cdot V[t-1, j]\}$
Reconstruct most probable state sequence:
$S[T] \leftarrow argmaxV[T, k]$
for *t from T to 1*:
  $S[t-1] \leftarrow V_{ptr}[t, S[t]]$
return S

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 datapoints belonging to different classes.

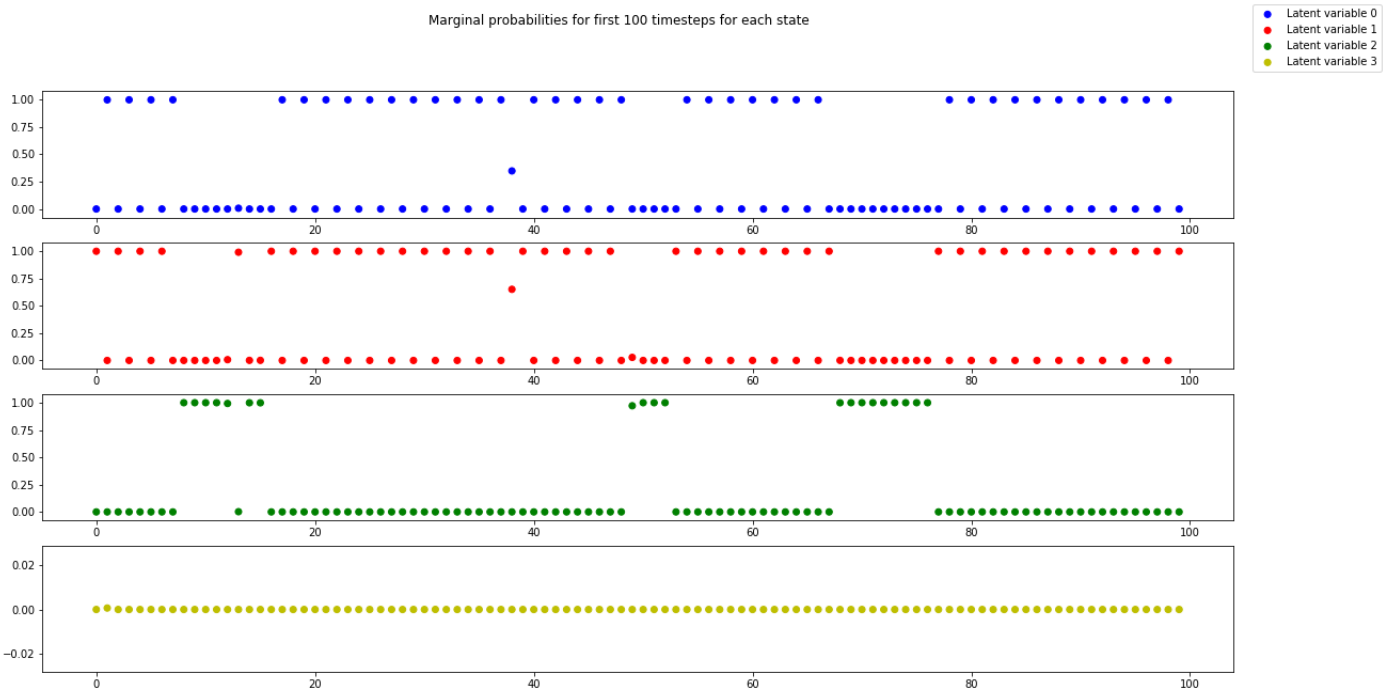
Answer:



We note that although the clustering is somewhat reasonable, it is far from perfect, which is most likely a consequence of the coding error from the EM implementation.

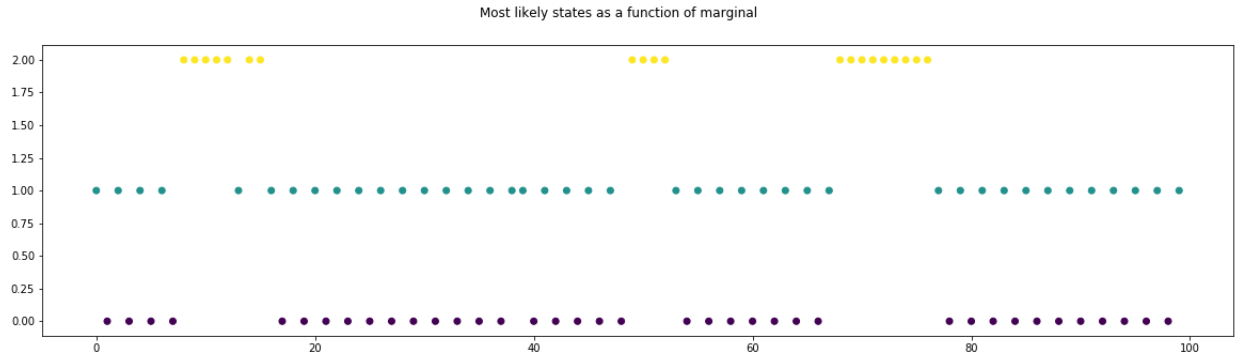
9. For the datapoints in the test file `EMGaussian.test`, compute the marginal probability $p(z_t|x_1, \ldots, x_T)$ for each point to be in state $\{1, 2, 3, 4\}$ for the parameters learned on the training set. For each state plot the probability of being in that state as a function of time for the 100 first points (i.e., as a function of the datapoint index in the file).

Answer:



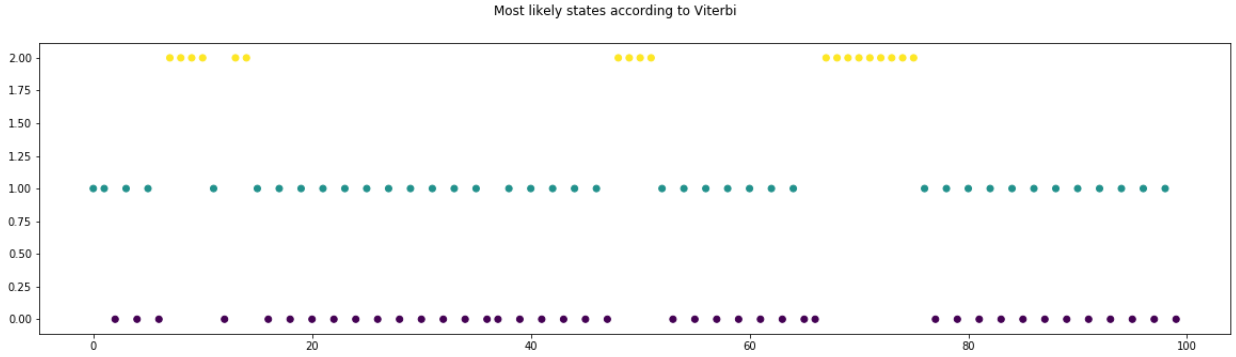
10. For each of these same 100 points, compute their most likely state according to the marginal probability computed in the previous question. Make a plot representing the most likely state in $\{1, 2, 3, 4\}$ as function of time for these 100 points.

Answer:



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.

Answer:



Although difficult to draw reliable conclusions due to the implementation error in EM, the most likely state sequences are very similar in the marginals and the Viterbi decoding. This can be understood by the fact that there are four very distinct clusters of gaussians, and thus the marginal most likely state is likely to be the correct one, as was seen from the predictions of the GMM in homework 3. Thus we conclude that an understanding of the sequential aspect of the data is not particularly useful to predicting the most likely state sequence for this problem, since each individual data point already has high probability associated with an individual cluster.

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

Answer:
A first step in determining the number of states appropriate for an HMM model should always be knowledge of the underlying phenomenon. If this provides no hints as to a reasonable number of states for the model, one way to proceed might be to treat the number of states as a hyper-parameter to the model, and compute the resulting likelihood as a function of the number of states selected. Because the likelihood is likely to be overfit by increasing the number of states too much, one could choose the smallest number of states which falls within reasonable reach of the highest likelihood obtained, implemening a form of "early stopping" on this hyperparameter search.