Homework 4, Problem 3 by Amr

Yuan Liang
andrew ID: yuanlian

1 HMM[55pt, Amr] Solution Courtesy of Yuan Liang with few edits

As we discussed in the class and recitation, an HMM is used to model a sequence of observed variables as generated from a sequence of hidden states. For instance, the observed sequence might be a sentence and the hidden sequence might be the part of speech tags for each word in the sentence. However, HMM can be also used in modeling continuous observations. For instance, in speech , the observed sequence is the speech signal, and the hidden sequence represents the words. The data for this problem is available over the website `data.mat`.

In this problem you will implement an HMM with continuous observations. You will implement solutions to the three basic problems in HMM: evaluation, decoding and learning.

Recall that an HMM defines a joint probability distribution over the observed and hidden sequences $p(x_1, x_2, \cdots, x_T, y_1, y_2, \cdots, y_T)$, where x and y represent the observed and hidden sequences respectively. Each hidden state $y_i \in \{1, \cdots, K\}$, and each observation $x_i \in \mathcal{R}$. The parameters of this continuous-observation HMM are: the distribution over the initial state π, the transition probability $a_{ij} = P(Y_t = j|Y_{t-1} = i)$, and the emission *density* $p(x_t|y_t = i) = \mathcal{N}(\mu_i, \sigma_i^2)$.

Our motivating application is detecting copy number variation in the genome. The human genome is comprised of 6 billion chemical bases (or nucleotides) of DNA packaged into two sets of 23 chromosomes, one set inherited from each parent. The DNA encodes 30,000 genes. Usually genes are present in two copies (copy number) in a genome. A variation in this copy number can be linked to diseases. aCGH (Array comparative genomic hybridization) is an experimental technique used to measure copy number in a test sequence (like a genome sequence from a patient with cancer) relative to a normal sequence. The output of this procedure is represented in terms of a log2 ratio between the copy number of the test sequence and the reference (normal) sequence. For simplicity, we assume that there are three state of the copy number: 0 (normal), log2(1/2) (missing copy) , and log2(3/2) (gain). However, the aCGH procedure is noisy and thus you might get a noisy measurement as shown in Fig. 1. In this figure, the Y-axis gives the log2 ratio and the X-axis represents location over the genome. Our biological knowledge tells us that the copy number of nearby locations should be the same which suggests that an HMM model might be a good option in recovering the correct copy number from this noisy measurements (Fig. 1 has 11 segments).

Unless otherwise stated, during this problem we take $K = 3$. Moreover, in this problem we will deal only with one-sequence for simplicity, however, in practice, your EM-algorithm can be easily changed to work with multiple sequences.

1. [**2 points**] Write down the forward and backward recurrences for this continuous-observation HMM.

   Answer: The forward recurrence is:

   – initial: $\alpha_1^k = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x_1-\mu_k)}{2\sigma_k^2}} \pi_k$

   – iteration: $\alpha_t^k = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x_t-\mu_k)}{2\sigma_k^2}} \sum_i \alpha_{t-1}^i a_{ik}$

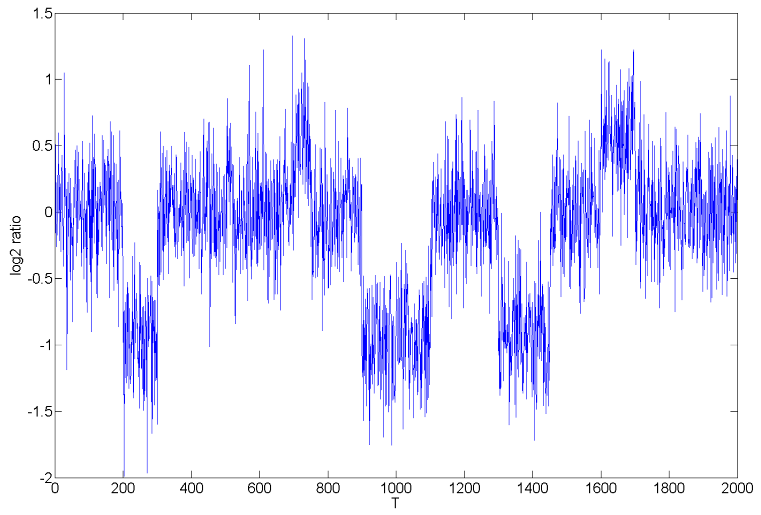


Figure 1: Copy number variation sequence. X-axis represents location over the genome. There are 11 segments

The backward recurrence is:

- initial: $\beta_T^k = 1, \ \ \forall k$
- iteration: $\beta_t^k = \sum_i a_{ki} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x_{t+1}-\mu_i)}{2\sigma_i^2}} \beta_{t+1}^i$

2. Implement the forward-backward algorithm. (you might want to refer to the book or recitation slides for how to avoid underflow using re-scaling)

3. To estimate the parameters of an HMM we use the EM algorithm. We start the EM algorithm by randomly initializing the model parameters. In the E-step we run the forward-backward algorithm over the sequence(s) using the current estimate of the model parameters, and in the M-step we re-estimate the model parameters using the expected sufficient statistics from the E-step. We repeat this two steps until the log-likelihood of the observed sequence asymptotes (i.e. converges).

   i. [**2 points**] Using the α and β probabilities, write down the MLE estimate of π_i and a_{ij}.

Answer: Let $\gamma_t^k = P(y_t = k | x_1, \cdots, x_T)$

The MLE estimate of π_i is:

$$\pi_i^{\mathrm{ML}} = \gamma_1^i = P(y_1^i = 1 | x) = \frac{\alpha_1^i \beta_1^i}{\sum_i \alpha_1^i \beta_1^i}$$

The MLE estimate of a_{ij} is:

$$a_{ij}^{\mathrm{ML}} = \frac{\sum_t \xi_t^{i,j}}{\sum_t \gamma_t^i}$$

where

$$\xi_t^{i,j} = \frac{\alpha_t^i a_{ij} P(x_{t+1} | y_{t+1} = j) \beta_{t+1}^j}{\sum_{i,j} \alpha_t^i a_{ij} P(x_{t+1} | y_{t+1} = j) \beta_{t+1}^j}$$

ii. [**5 points**]Using the α and β probabilities, in addition to the observed sequences, write down the MLE estimate of μ_k and σ_k^2. (Hint: this is somehow similar in spirit to the Gaussian Naive Bayes, i.e, it can be represented in terms of $P(y_t = i|x_1, \cdots, x_T)$)

Answer: The MLE estimate of μ_k is:

$$\mu_k^{\text{ML}} = \frac{\sum_t \gamma_t^k x_t}{\sum_t \gamma_t^k}$$

where

$$\gamma_t^k = \frac{\alpha_t^k \beta_t^k}{\sum_k \alpha_t^k \beta_t^k}$$

The MLE estimate of σ_k^2 is:

$$(\sigma_k^2)_{\text{ML}} = \frac{\sum_t \gamma_t^k (x_t - \mu_k)^2}{\sum_t \gamma_t^k}$$

iii. Implement the EM algorithm. Stop iterating when the relative change in log-likelihood (LL) falls below 1e-3, where the relative change in LL $= |\frac{\text{LL at iteration r} - \text{LL at iteration (r-1)}}{\text{LL at iteration (r-1)}}|$

iv. [**25 points**] Run your EM algorithm over the given sequence. Initialize π and the transition matrix randomly (make sure they are correct probability distributions), and initialize μ_i by drawing it randomly from $\mathcal{N}(0,1)$ and σ_i^2 by drawing it uniformly from [.01,1]. Repeat this experiment 5 times each of which with different random initialization. **Record** the final LL of each run and **write down** the EM-parameters for the best and worst LL. **Also draw** the trace of the LL over iterations for the runs with worst and best final LL. **What** do you observe and why? (**Note**: if all your runs give the same LL, try to initialize μ by drawing it from $\mathcal{N}(0,2)$, in my implementation, I was able to get different LL either way).

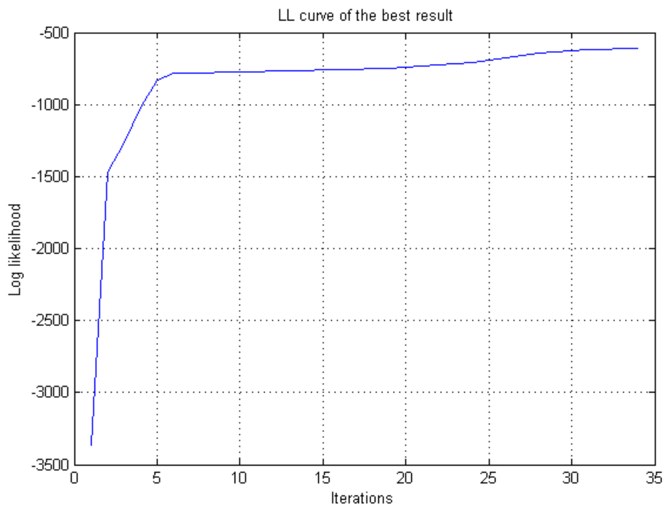
Answer: I ran it 5 times and the final LL's are:

# of run	final LL
1	-613.4561
2	-613.4365
3	-785.4694
4	-785.8705
5	-613.4763

So the maximum and the minimum of LL are -613.4365 and -785.8705. The parameters generated for those two runs are:

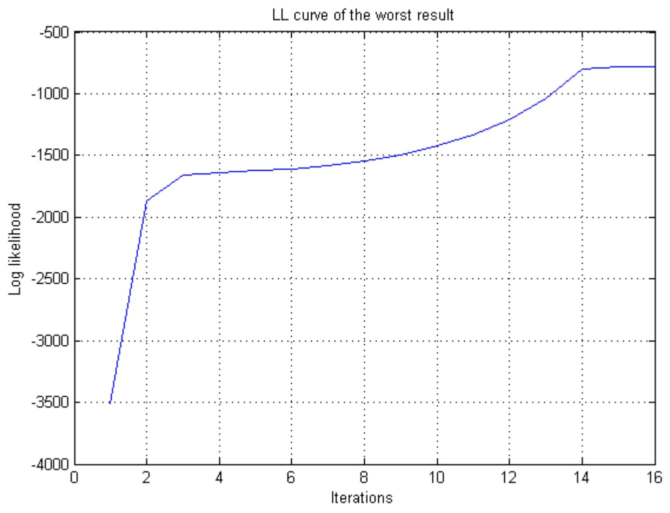
final LL	-613.4365
μ	$[0.0027,\ 0.5664,\ -0.9616]$
σ^2	$[0.1011,\ 0.1053,\ 0.1029]$
π	$[1.0000,\ 4.39\times10^{-22},\ 3.68\times10^{-104}]$
A	$\begin{bmatrix} 0.9954 & 0.0016 & 0.0022 \\ 0.0145 & 0.9854 & 2.095\times10^{-14} \\ 0.0068 & 3.908\times10^{-23} & 0.9932 \end{bmatrix}$

The LL curve is shown as follows:



final LL	-785.8705
μ	$[0.0597,\ -0.9649,\ -0.9548]$
σ^2	$[0.1304,\ 0.1013,\ 0.1062]$
π	$[1.0000,\ 7.60\times10^{-8},\ 2.64\times10^{-8}]$
A	$\begin{bmatrix} 0.9974 & 0.0008 & 0.0011 \\ 0.0080 & 0.5178 & 0.4741 \\ 0.0045 & 0.8773 & 0.1181 \end{bmatrix}$

The LL curve is shown as follows:



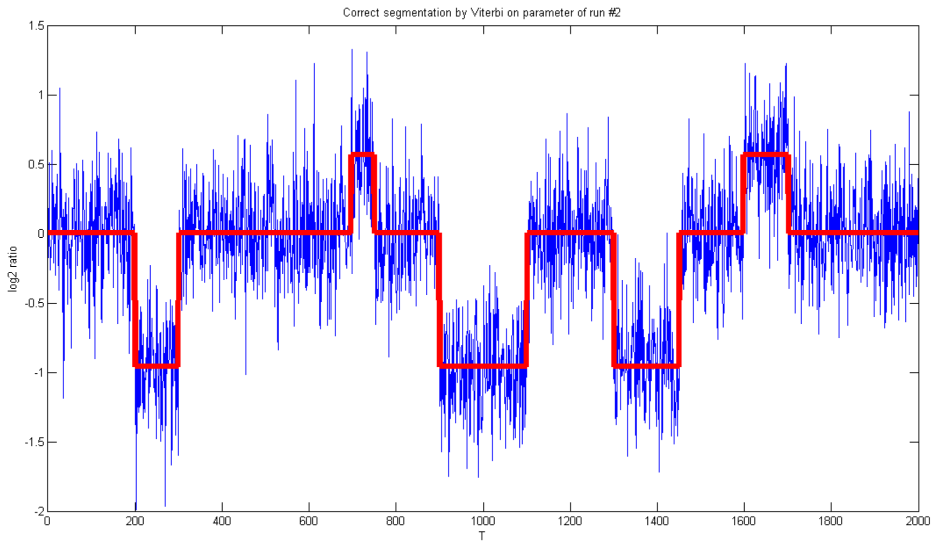
The 5 runs basically gave us 2 results: one with a larger LL the other with a smaller LL. The smaller LL corresponds to the wrong parameters as we already know what we should expect. The larger LL corresponds to the correct parameters. This is because of the difference of initial values, and since EM algorithm guarantee to converge but not guarantee to converge to global maximum, so its performance is sensitive to initialization. This tells you that you should run EM with multiple initialization and chose the one that results in the highest LL.

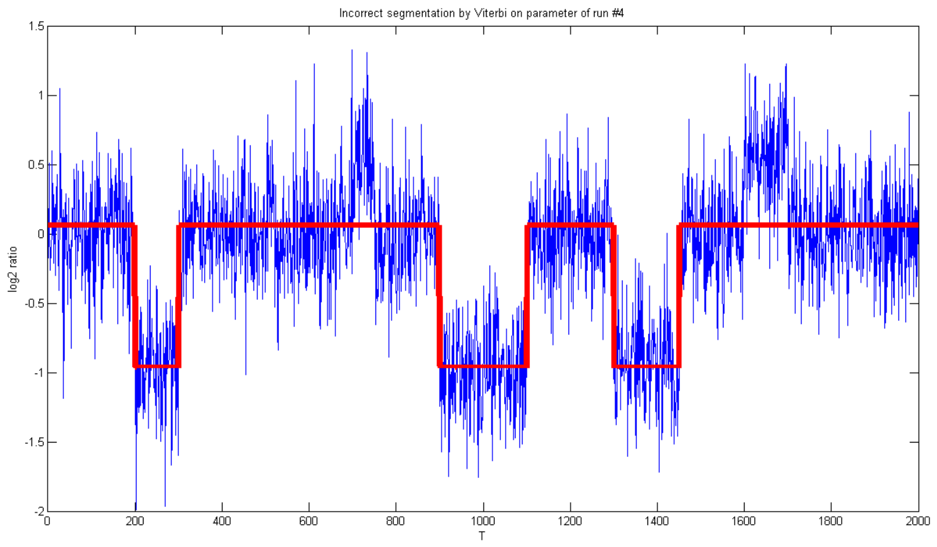
4. Decoding.

i. Implement the Viterbi decoding algorithm. (perform all operations in log-space to avoid under flow).

ii. [**15 points**] Using the estimated parameters by the EM-algorithm, run the Viterbi algorithm to get the most-probably hidden state sequence. Choose two runs from your 5-runs in (3.iv): one that gets the correct segmentation and one that doesn't. Overlay the segmentation of each run over the data in Fig. 1. To do that, for each point, draw the mean of its corresponding hidden state as recovered by Viterbi-decoding. (I am looking here for two figures, one for each run, each of which has the data and the segmentation, please draw the segmentation with a thicker marker, i.e 'linewidth'=5 in matlab).

The graph for the correct segmentation derived by the parameters on run #2, and the incorrect segmentation derived by the parameters on run #4 is shown as follows:





5. [**1 points**] Lets assume you don't know the correct segmentation, did the lowest and highest

LL in 3.iv correspond to a correct and incorrect segmentation?

Answer: YES. However you should be cautious that it won't be always the case that the solution with the highest LL will give the correct segmentation, since here the problem is unsupervised. In other words, we might choose a very large number of states that results in a very large LL but most probably we will be over fitting here. However, when we fix the number of parameters and only change the initialization, then the solution with the highest LL is always the best one for this specific choice of K (the number of states).

6. [**3 points**] Would you expect that a GMM with three mixtures would recover the correct segmentation, and why? Either argue that GMM will result in the exact segmentation or point to areas in Fig. 1 in which GMM will fail.

   Answer: No. GMM with 3 mixtures may not recover the correct segmentation because the data is noisy, and if we look at the graph where $T = 600 \sim 800$, it is very hard to distinguish the pattern by GMM, because the variance is very high.

7. [**2 points**] If we don't know the correct number of states K, suggest a scheme to select K. You shouldn't assume that you have a reference to the correct segmentation. Moreover, assume we only have one sequence.

   Answer: We can use BIC score (see lecture 8 slide 38). The idea here is to penalize complex models. We select K as follows: $k^* = arg \max_k p(\mathrm{x}|k) - C(x)$, where $p(\mathrm{x})$ is the marginal LL of the observation given the number of states k, and $C(k)$ is a complexity term that equals to the number of parameters times $logT$, where T is the number of observations, and the number of parameters for k states is: $\underbrace{k * (k-1)}_{\text{transitoin matrix}} + \underbrace{k-1}_{\text{initial state}} + \underbrace{k * 2}_{\text{mean and variance per state — emisson}}$.