STAT 534 Statistical Computing: Final Project
Training Hidden Markov Model by Baum-Welch Algorithm

Ruojin HE

June 13, 2019

1 Introduction

In this project we will fit a Hidden Markov Model to the multiple observation sequence training data. We will first derive that Baum-Welch Algorithm is able to maximize the likelihood. We will then simply describe the training data we use, and next the way we choose an appropriate number of states, and then the process about how we conduct experiments on the training data (including how we initialize the model, and which stopping criterion we use) and test data. In the last part, we will analyze the results obtained from the experiments.

2 MLE property of Baum-Welch Algorithm

2.1 Problem Setup

First suppose each sequence of a Hidden Markov Model is of length T. The observation is from V = {0,1,2,...,N-1} and the underlying states is from Q = {0,1,2,...,M-1}. The HMM is $\lambda = (A, B, \pi)$, where A is the transition matrix for underlying states: $a_{ij} = P[s_{t+1} = j | s_t = i]$, B is the emission matrix: $bi(j) = P(v_t = j | s_t = i)$ and π is the initial state matrix: $\pi_i = i = P(z_1 = i)$
We denote the multiple observation sequences of size n as $O = [O^{(1)}, O^{(2)}, ..., O^{(n)}]$, where $O^{(i)} = (o_1^{(i)}, o_2^{(i)}, ..., o_T^{(i)})$. Without given latent states $Q^{(i)}$ for $O^{(i)}$, we are supposed to find the optimal λ^*, s.t., $\lambda^* = \text{argmax}_\lambda P[O|\lambda]$.
Baum-Welch algorithm actually repeats the following steps at each iteration till some stopping point. (Note that λ is current model and $\bar{\lambda}$ is the re-estimated model at each iteration.)

Step 1. Calculate Baum's auxiliary function: $Q(\lambda, \bar{\lambda}) = \sum_Q p(Q|O, \lambda) \log P(Q, O|\bar{\lambda})$
Step 2. Maximize the above function: $\max_{\bar{\lambda}} Q(\lambda, \bar{\lambda})$

2.2 Derivation of Baum-Welch algorithm

Since $P(Q|O, \lambda) = \frac{P(Q,O|\lambda)}{P(O|\lambda)}$ and $P(O|\lambda)$ does not depend on Q and $\bar{\lambda}$, we have:

$$\underset{\bar{\lambda}}{\text{argmax}}\, Q(\lambda, \bar{\lambda}) = \underset{\bar{\lambda}}{\text{argmax}} \sum_Q P(Q|O, \lambda) \log P(O, Q|\bar{\lambda}) = \underset{\bar{\lambda}}{\text{argmax}} \sum_Q P(Q, O|\lambda) \log P(O, Q|\bar{\lambda}) \tag{1}$$

where $P(Q, O|\lambda) = \prod_{i=1}^n \pi_{q_1^{(i)}} B_{q_1^{(i)}}(o_1^{(i)}) \sum_{t=2}^T A_{q_{t-1}^{(i)} q_t^{(i)}} B_{q_t^{(i)}}(o_t^{(i)})$

Solution to (1) is given as follows:

$$\begin{aligned}
\bar{\pi_i} &= \gamma_1(i) \\
\bar{a_{ij}} &= \frac{\sum_{k=1}^n \frac{1}{p_k} \sum_{t=1}^{T-1} \alpha_t^k(i) a_{ij} b_j(O_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^n \frac{1}{p_k} \sum_{t=1}^{T} \alpha_t^k(i) \beta_{t+1}^k(j)} \\
\bar{b_j}(l) &= \frac{\sum_{k=1}^n \frac{1}{p_k} \sum_{t=1, O_t=l}^{T-1} \alpha_t^k(j) \beta_t^k(j)}{\sum_{k=1}^n \frac{1}{p_k} \sum_{t=1}^{T} \alpha_t^k(j) \beta_t^k(j)}
\end{aligned}$$

Now we need to prove that for the above re-estimated model $\bar{\lambda} = (\bar{\pi}, \bar{A}, \bar{B})$ we have P(O|$\bar{\lambda}$) ≥P(O|λ), i.e., the likelihood for re-estimated model is larger than that of current model.
Note that $P(O|\bar{\lambda}) = \sum_Q P(O, Q|\bar{\lambda})$ and $Q(\lambda, \bar{\lambda}) = E_{Q|O,\lambda}[\log P(O, Q|\bar{\lambda})]$ By Jensen's inequality, we have:

$$\begin{aligned}
\log P(O|\bar{\lambda})/P(O|\lambda) &= \log \sum_Q P(O, Q|\bar{\lambda})/P(O|\lambda) \\
&= \log \sum_Q \frac{P(O, Q|\lambda)}{P(O|\lambda)} \frac{P(O, Q|\bar{\lambda})}{P(O, Q|\lambda)} \\
&\geq \sum_Q \frac{P(O, Q|\lambda)}{P(O|\lambda)} \log \frac{P(O, Q|\bar{\lambda})}{P(O, Q|\lambda)} \\
&= \sum_Q \frac{P(O, Q|\lambda)}{P(O|\lambda)} (\log P(O, Q|\bar{\lambda}) - \log P(O, Q|\lambda)) \\
&= \frac{1}{P(O|\lambda)} \{\sum_Q P(O, Q|\lambda) \log P(O, Q|\bar{\lambda}) - \sum_Q P(O, Q|\lambda) \log P(O, Q|\lambda)\}
\end{aligned} \tag{2}$$

By equation (1), maximization of $Q(\lambda, \bar{\lambda})$ is equal to maximization of $\sum_Q P(O, Q|\lambda) \log P(O, Q|\bar{\lambda})$. Therefore, for solution of (1), we have:

$$\sum_Q P(O, Q|\lambda) \log P(O, Q|\bar{\lambda}) - \sum_Q P(O, Q|\lambda) \log P(O, Q|\lambda) \geq 0 \tag{3}$$

which indicates that $\log P(O|\bar{\lambda})/P(O|\lambda) \geq 1$, and therefore, $P(O|\bar{\lambda}) \geq P(O|\lambda)$. The equality holds iff $\lambda = \bar{\lambda}$.
Therefore, at each iteration, when we maximize $Q(\lambda = \bar{\lambda})$ and obtain a new model $\bar{\lambda}$, we will increase the likelihood. When some stopping point is reached, or the algorithm converges, ideally we will obtain the optimal λ^* s.t. $P(O|\lambda^*)$ is maximal.

3 Experiments

3.1 Initialization & Stopping Criterion

For initialization for $\lambda = (\pi, A, B)$, it has been experimentally proven that either random or uniform values of A and π is adequate for good re-estimation. However for B random or uniform initial estimates may not be enough. K-means segmentation with clustering is suggested by Rabiner in his paper. But in this project, for simplicity, we use random estimates for HMM.
For stopping point, we look at the log-likelihood of training data given the re-estimated model. That is, at the end of each iteration, we obtain a new model, and use the forward algorithm to compute the probability of the observation sequences given the new model. When the improvement of the log-likelihood is below a certain threshold, the learning process stops. In our project, we define

$$\Delta_{\%} = \frac{|\log p_k - log p_{k-1}| \times 100}{|\log p_{k-1}|} \tag{4}$$

When $\Delta_{\%}$ is below 0.005, the algorithm stops.

3.2 Normalization

Note that α and β could get extremely small (may be zero when it exceeds the precision range) and cause underflow problem. To avoid this problem, we perform the scaling procedure when computing the value of α, β. That is, scale α, β using a common normalization factor: $\sum_{i=0}^{M-1} \alpha_i(t)$ at each time t. That is,

$$\begin{aligned}
\hat{\alpha}_i(t) &= \frac{\alpha_i(t)}{\sum_{i=0}^{M-1} \alpha_i(t)} \\
\hat{\beta}_i(t) &= \frac{\beta_i(t)}{\sum_{i=0}^{M-1} \alpha_i(t)}
\end{aligned} \tag{5}$$

3.3 Implementation of Baum-Welch Algorithm for multiple sequences

To implement the Baum-Welch Algorithm we define a class for Hidden Markov Model. This class mainly contains the following functions: initialize a Hidden Markov Model by loading a json file(this file includes initial A, B and π), transform likelihood to log-likelihood, compute the log-likelihood using backward and forward algorithm, compute the normalization factor, choose the optimal latent states using viterbi algorithm, re-estimate the model using Baum-Welch Algorithm, simulate a HMM which returns the latent states and corresponding observations, and predict next observation given a sequence.

3.4 Simulation

In this part, we first define a Hidden Markov Model $\lambda = (\pi, A, B)$, where:

$$\begin{aligned} \pi &= (0.13, 0.23, 0.34, 0.3) \\ A &= \begin{bmatrix} 0.1 & 0.3 & 0.3 & 0.3 \\ 0.4 & 0.13 & 0.07 & 0.4 \\ 0.3 & 0.3 & 0.2 & 0.2 \\ 0.1 & 0.15 & 0.25 & 0.5 \end{bmatrix} \\ B &= \begin{bmatrix} 0.15 & 0.15 & 0.3 & 0.4 \\ 0.5 & 0.3 & 0.1 & 0.1 \\ 0.23 & 0.37 & 0.2 & 0.3 \\ 0.2 & 0.3 & 0.4 & 0.1 \end{bmatrix} \end{aligned} \tag{6}$$

And then we generate a 1000 observation sequence and the corresponding latent state sequence. Using these 1000 observations, next, we train the HMM by B-W algorithm and then use Viterbi algorithm to get the most likely state sequence for each observation sequence.
The result of B-W algorithm is

$$\begin{aligned} \bar{\pi} &= (0, 0.96, 0.04, 0) \\ rA &= \begin{bmatrix} 0.1020 & 0.3054 & 0.2975 & 0.2950 \\ 0.3961 & 0.1398 & 0.0713 & 0.3927 \\ 0.3045 & 0.2970 & 0.2021 & 0.1963 \\ 0.1038 & 0.1573 & 0.2563 & 0.4824 \end{bmatrix} \\ \bar{B} &= \begin{bmatrix} 0.1701 & 0.1417 & 0.2856 & 0.4024 \\ 0.5379 & 0.2665 & 0.0900 & 0.1055 \\ 0.2621 & 0.3323 & 0.1907 & 0.2148 \\ 0.2327 & 0.2859 & 0.3660 & 0.1151 \end{bmatrix} \end{aligned} \tag{7}$$

We then compare the true state sequence with state sequence obtained from Viterbi algorithm, and the correct rate is 0.378.

3.5 Training Data

The training data set contains n=1000 sequences, each of which is of length T=40. The observation space is V = {0,1,2,3}. Now we are supposed to pick a latent state space {0,1,...,M-1} by cross-validation. To use cross-validation we need to know at first how to evaluate a Hidden Markov Model. The first idea is to split the training data into two parts for training and test, and then compute the likelihood of test data. The second idea is to use some of the data to predict the other. That is, use $o_1, ..., o_t$ to predict o_{t+1}, and then measure the difference between the true and the predicted. Here we would use the first method, since it's more straightforward and easier to conduct.
The range of M we consider is C = {3,4,5,6,7,8}. Since increase in the number of states will also lead to a huge increase in the number of parameters to be estimated in all matrices (O(M^2)), we do not consider too large numbers.
For cross-validation, we randomly split these training data into 10 folds. We then use the folds other than the k-th fold, k=1,2,3,...,8,9,10 as the new training sample and obtain a model λ, and then we use the forward algorithm and λ to obtain the log-likelihood of the test sample (k-th

fold). Finally we average the log-likelihood as the result of each number of latent states. For each choice in C, we conduct the above procedure. And then we pick the number with the largest log-likelihood. The cross-validation gives M=6 as the number of latent states.

3.6 Results & Analysis

3.6.1 Model Estimates

We first split the training data into two parts. Each one contains 500 sequences. And we use the first part as our train set, and use the other as our test set. We conduct forward-backward algorithm on the first 500-sequence training data set. The estimates of transition matrix A, emission matrix B and initial probability π are

$$\bar{\pi} = (0.0181, 0.1458, 0.1396, 0.3668, 0.2108, 0.1187)$$

$$\bar{A} = \begin{bmatrix} 0.0996 & 0.1702 & 0.3143 & 0.1369 & 0.1929 & 0.0859 \\ 0.0356 & 0.0191 & 0.0105 & 0.0803 & 0.1990 & 0.6553 \\ 0.1330 & 0.2378 & 0.4878 & 0.0049 & 0.0575 & 0.0789 \\ 0.0625 & 0.0442 & 0.0087 & 0.6797 & 0.1964 & 0.0083 \\ 0.0688 & 0.4308 & 0.0376 & 0.1106 & 0.3516 & 0.0005 \\ 0.0497 & 0.1785 & 0.2476 & 0.0462 & 0.2167 & 0.2612 \end{bmatrix} \tag{8}$$

$$\bar{B} = \begin{bmatrix} 0.3285 & 0.3490 & 0.2574 & 0.0650 \\ 0.4803 & 0.4462 & 0.0270 & 0.0464 \\ 0.0702 & 0.5626 & 0.1555 & 0.2116 \\ 0.5995 & 0.0588 & 0.2657 & 0.0758 \\ 0.0840 & 0.0473 & 0.8525 & 0.0162 \\ 0.0249 & 0.0069 & 0.0727 & 0.8954 \end{bmatrix}$$

The Baum Welch algorithm, on the training set of 500 sequences, converged in 66 iterations and took 296 seconds to run. Figure 1 shows how the log-likelihood changed when the iteration round increases. After around 60 iterations, the curve tends to flatten. And Figure 2 shows the log-likelihood of each sequence of the 500-sequence training set and of the next 500-sequence test set. From the plot, we could see that the two groups of log-likelihood are quite close, which indicates that the model performs well on the test set.

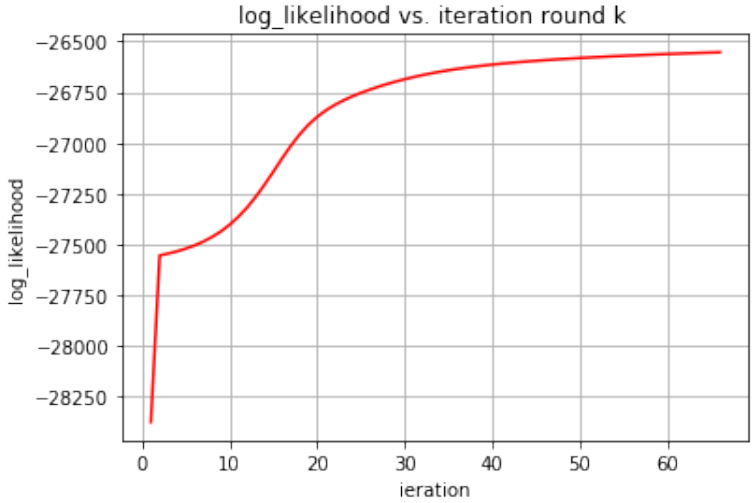

Figure 1: log likelihood vs iteration

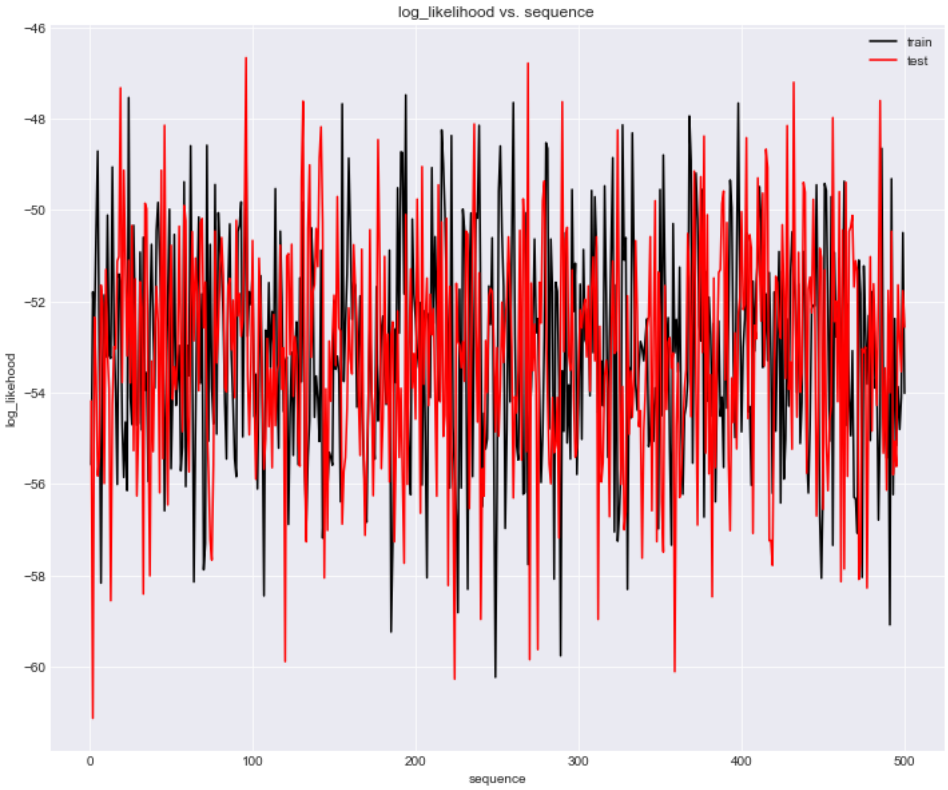

Figure 2: log likelihood vs sequence

3.6.2 Most Likely Sequence

In this part we conduct viterbi algorithm to calculate the most likely sequence of states for all 1000 sequences. The runtime is 1.337 seconds. We visualize the first 10 sequences as Figure 3 shows.

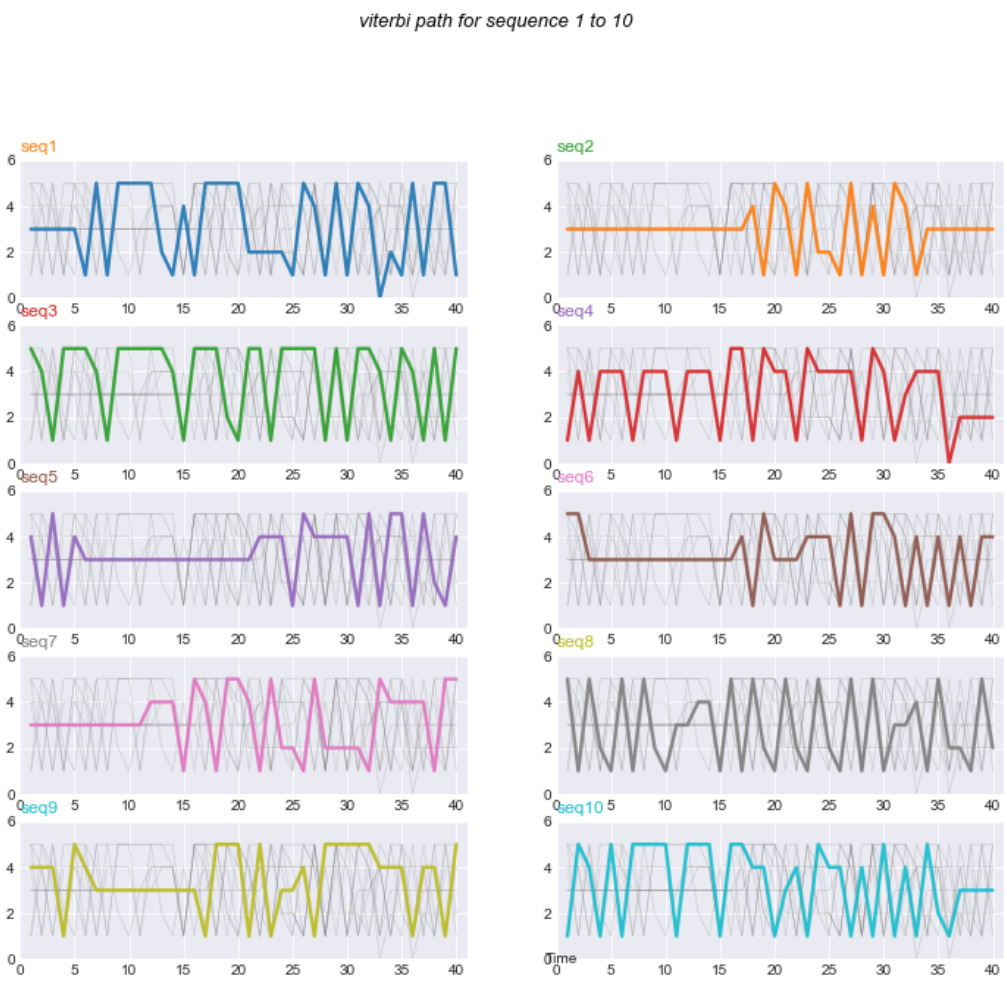

Figure 3: viterbi path: 10 sequences

3.6.3 Predict Next Output

The basic idea of predicting next output at time t_0 is to maximize $P(Y_{t_0}|Y_{1:t_0-1}, \pi)$. Since we have,

$$P(Y_{t_0}|Y_{1:t_0-1}, \pi) = \frac{P(Y_{t_0}, Y_{1:t_0-1}|\pi)}{P(Y_{1:t_0-1}|\pi)} \tag{9}$$

and $Y_{1:t_0-1}|\pi$ is not dependent of Y_{t_0} , hence we have:

$$P(Y_{t_0}|Y_{1:t_0-1}, \pi) = P(Y_{t_0}, Y_{1:t_0-1}|\pi) * C \tag{10}$$

for some C which does not depend on Y_{t_0}. Therefore, for discrete observation space as in this project, we take k as the predicted observation where k = $\text{argmax}_k P(Y_{t_0} = k, Y_{1:t_0-1}|\pi)$ for observed sequence $Y_{1:t_0-1}$ and HMM π.
We then use the trained HMM to predict the last observation of each sequence in our 500-sequence test set given the 39 observations. The results indicates that out of 500 predictions, 180 are correct, so the correct rate is 0.36.

3.7 Test Data

For our 50-sequence test data, we conduct the following experiments:

1. Output the log-likelihood of the test set

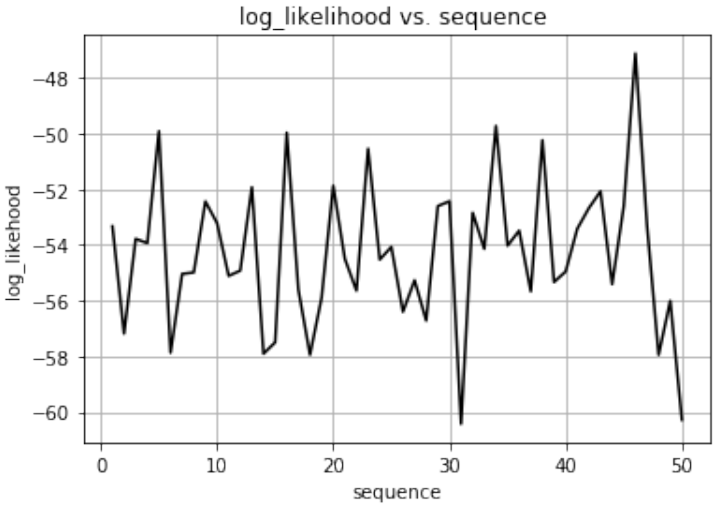

Figure 4: log likelihood for each sequence (test data)

2. Output the Viterbi sequence of states (Here we only plot the first 10 latent state sequences) as Figure 5 shows.

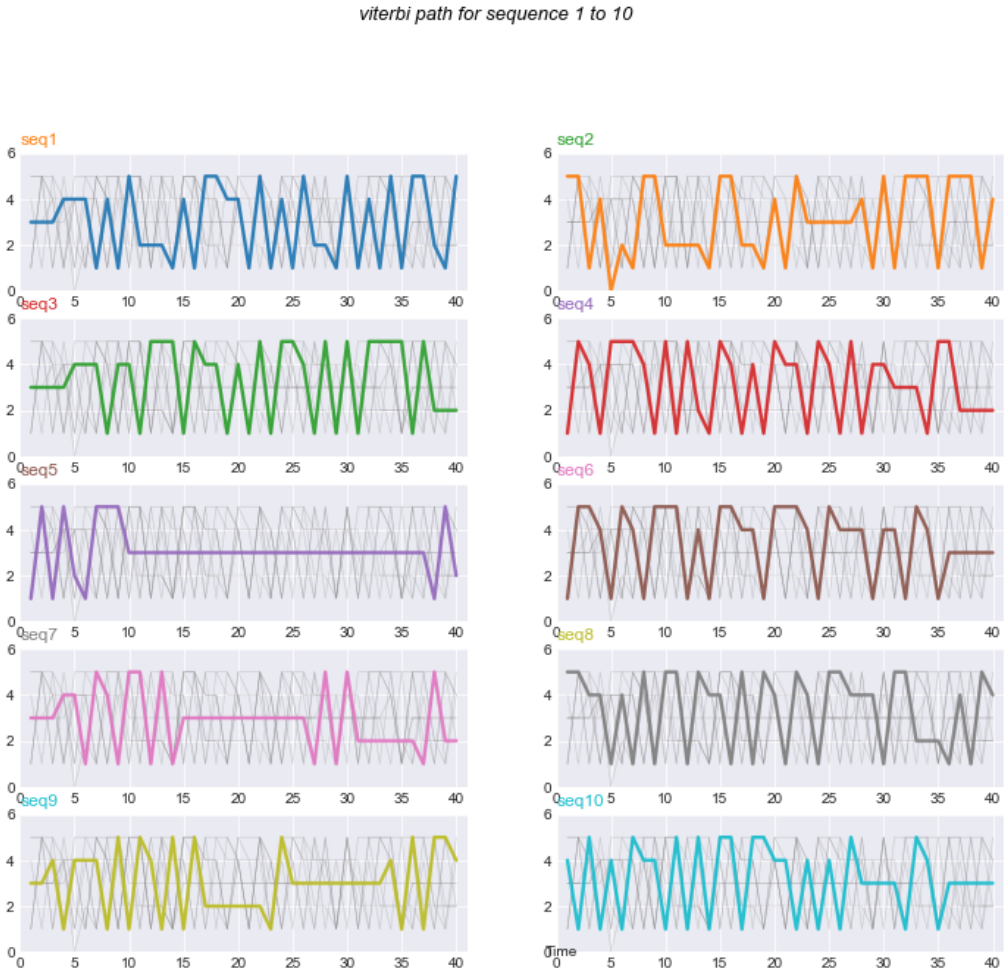

Figure 5: viterbi path: 10 test sequences

3. Predict the next output for given 50 sequences:

'seq1': '3',
'seq2': '2',
'seq3': '3',
'seq4': '3',
'seq5': '3',
'seq6': '0',
'seq7': '3',
'seq8': '2',
'seq9': '2',
'seq10': '2',
'seq11': '3',
'seq12': '2',
'seq13': '3',
'seq14': '2',
'seq15': '1',
'seq16': '2',
'seq17': '3',
'seq18': '2',
'seq19': '0',
'seq20': '3',
'seq21': '2',
'seq22': '3',
'seq23': '1',
'seq24': '3',
'seq25': '0',
'seq26': '2',
'seq27': '3',
'seq28': '0',
'seq29': '2',
'seq30': '2',
'seq31': '3',
'seq32': '2',
'seq33': '1',
'seq34': '3',
'seq35': '0',
'seq36': '3',
'seq37': '1',
'seq38': '2',
'seq39': '3',
'seq40': '2',
'seq41': '2',
'seq42': '3',
'seq43': '1',
'seq44': '2',
'seq45': '3',
'seq46': '0',
'seq47': '3',
'seq48': '2',
'seq49': '0',
'seq50': '2'