# STAT 534 Project

Spring 2019

Ashwin T. A. N

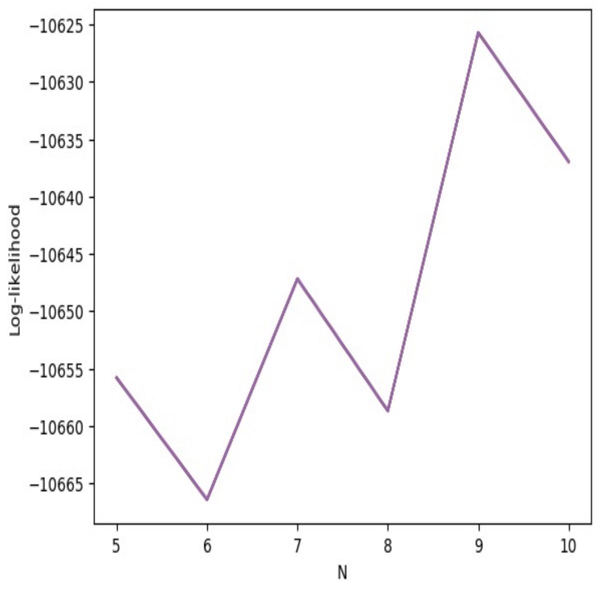
The aim of this project was to train a Hidden Markov Model (HMM) on the given training data and use it to analyze a test dataset. The training data set $\mathcal{D}$ consisted of $K = 1000$ sequences of $T = 40$ observed states, with values in $\{0, 1, 2, 3\}$. The number of hidden states $N$ was chosen by cross-validation and a best-fit Hidden Markov Model was computed using the Baum-Welch algorithm. Using this model, the most probable sequences of hidden states was computed for the given test data set $\mathcal{E}$ of $K' = 50$ sequences by the Viterbi algorithm. Moreover, I also computed the probability distribution of the missing $(T+1)^{st}$ observed states for each of the sequences in the test dataset. All analyses were implemented in Python version 2.7.

## 1 Training

The first step in training a HMM is to choose the number of hidden states $N$. This was done by cross-validation. The original training dataset was split into two parts - $\mathcal{D}_1$ consisting of the first $K_0 = 800$ sequences was used to train HMMs for different values of $N$ by the Baum-Welch algorithm. These models were then tested on the set $\mathcal{D}_2$ consisting of the remaining $K - K_0 = 200$ sequences. Both the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) indicated $N = 9$ to be the optimum choice.

Throughout the rest of this report, I will use $A, B, \pi$ to represent the transitional probability matrix, the emission probability matrix and the initial distribution of hidden states for a HMM respectively. I started by writing modules that would, given a model $(A, B, \pi)$ and a sequence of observed states $O$, compute the corresponding $\alpha_t(i), \beta_t(i)$ by the Forward-Backward algorithm. Using this module, it is easy to compute the log-likelihood of any given observed dataset.

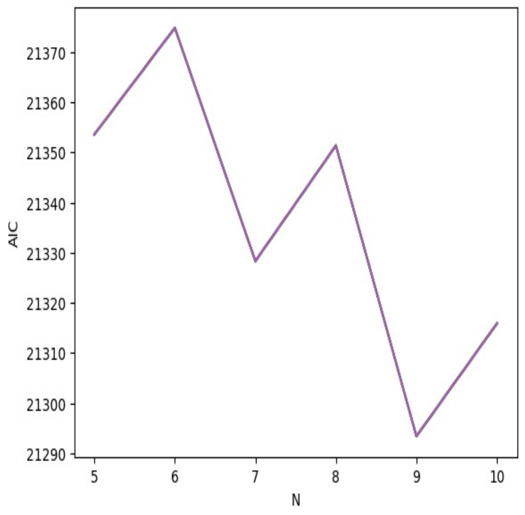
Next, I implemented the Baum-Welch algorithm as follows: The model parameters $A, B, \pi$ were initialized by choosing their entries independently from a Uniform$(0, 1)$ distribution and then normalizing by rows. Let $l_t$ denote the log-likelihood of $\mathcal{D}_1$ given the model output by the $t$-th iteration of the Baum-Welch algorithm. The iteration was made to repeat until $|l_{t+1}/l_t - 1| < 10^{-4}$ or the number of iterations reached 50. I then used this to train HMMs $\lambda_N$ for $N = 5, 6, \ldots, 10$. The log-likelihoods of the cross-validation data set $\mathcal{D}_2$ for these models is graphed below:

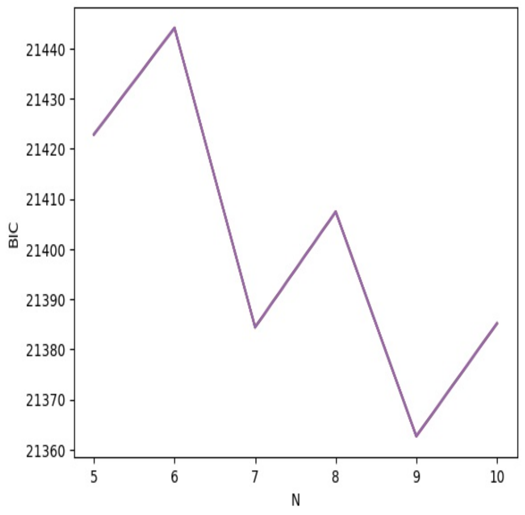


In addition, in order to account for the fact that the model increases in complexity as $N$ increases, I also computed he Akaike Information Criterion (AIC) and the Bayesian Information Criterion(BIC) statistics. Recall that the AIC and BIC values are given by

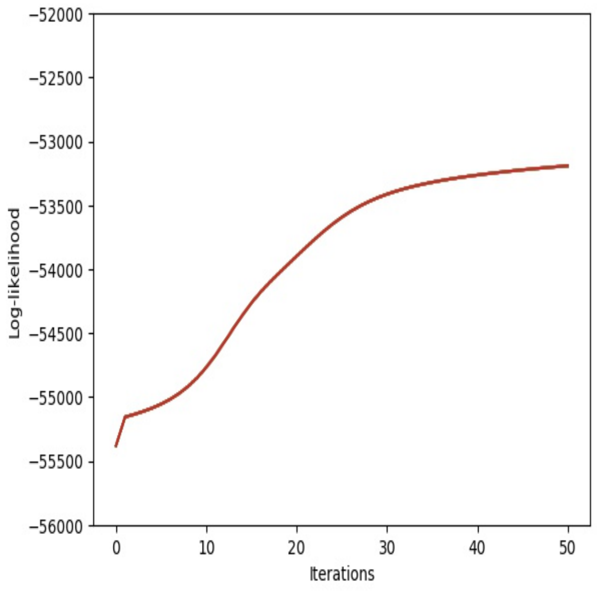
$$\begin{aligned} AIC &= 2k - 2l, \\ BIC &= k\log(n) - 2l, \end{aligned}$$

where $k$ is the number of parameters in the model, $l$ is the log-likelihood and $n$ is the sample size respectively. We note that after taking the row-sum constraints into account, the total number of real parameters in $(A, B, \pi)$ is $k = N(N-1) + N(4-1) + (N-1) = N^2 + 3N - 1$. The resulting AIC, BIC values are graphed below:



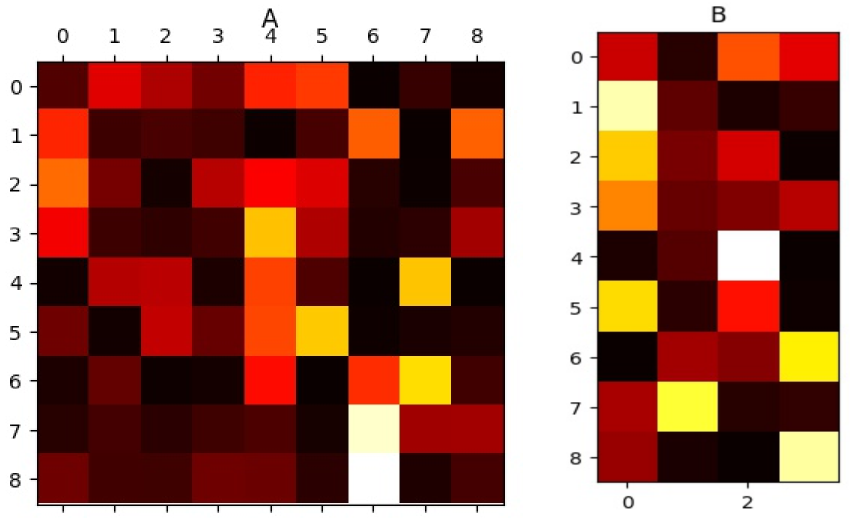


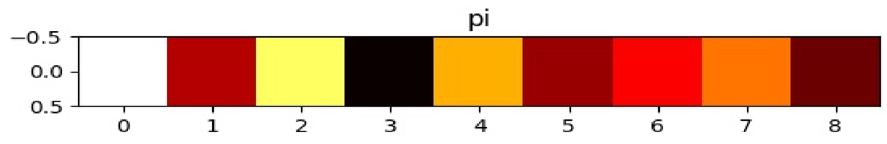
All 3 of these plots suggest that choosing $N = 9$ is optimum in among $5, 6, \ldots, 10$. Therefore, with this choice of $N$, I ran the Baum-Welch algorithm on the entire training data set $\mathcal{D}$. The program ran for 50 iterations (the maximum allowed) with a total run time of 712 seconds. This implies an average run time of roughly 14.24 seconds per iteration. The following plot shows the growth of log-likelihood of $\mathcal{D}$ with each iteration:



At the end of 50 iterations, the value of log-likelihood was $-53192.25$. Since the total log-likelihood of the sample is equal to the sum of the log-likelihoods of each of the sequences in the sample, it is relevant to consider the ratio of log-likelihood to sample size, which gives the average log-likelihood of the sequences in the training data. Since our data had 1000 samples, this ratio turns out to be around $-53.19$.

Finally, I relabeled the hidden states from $0, 1, \ldots, 8$ such that the states with more uniform emission probabilities have a lower index. This was done by permuting the rows of $A$ so that they are in ascending order of their $l^1$ norms, and then applying the same permutations to the colums of $A$, rows of $B$ and $\pi$. The heatmaps of the resulting transmission matrix, emission matrix and initial distribution are graphed below:

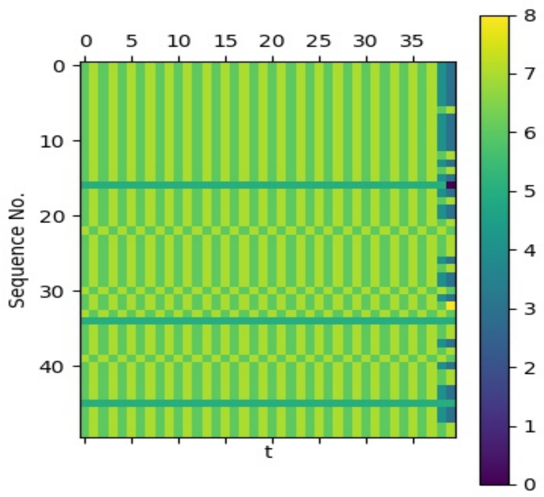




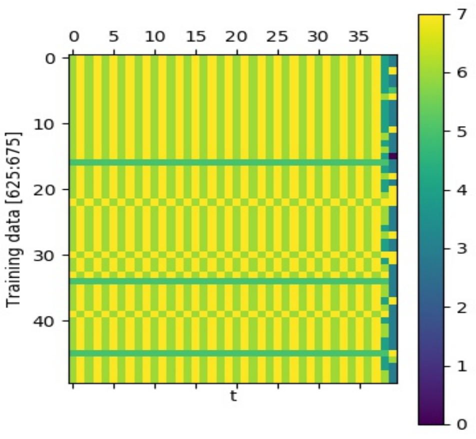
## 2 Analysis of Test data

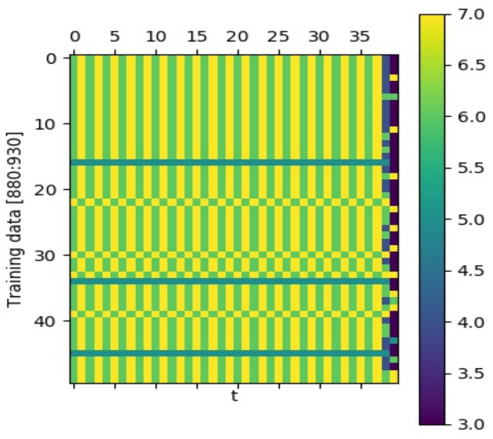
Let $\widehat{\lambda} = (\widehat{A}, \widehat{B}, \widehat{\pi})$ denote the Hidden Markov Model obtained in the previous section. The first task was to compute the log-likelihood of the test data $\mathcal{E}$ under this model. The test data set given consisted of $K' = 50$ sequences, each one of length $T = 40$ as before. I computed the log-likelihood of this test data set to be $-2719.53$. This yields a log-likelihood - to - sample size ratio of around $-54.38$, which is quite close to $-53.19$ that we obtained for the training data set.

Next, I implemented the Viterbi algorithm with parameters $\widehat{A}, \widehat{B}, \widehat{\pi}$ to compute the most likely sequence of hidden states corresponding to each sequence in the test data. The run-time of the algorithm was 0.074 seconds. The resulting output is illustrated in the plot below:



The states 6 and 7 make up a large majority of all the predicted states and most of the state transitions are between 6 and 7. 0 and 8 appear only once while 1 and 2 don't appear at all in the output. The output of the algorithm looked very similar for the given training data as well:





It would be interesting to see what happens for higher values of $N$.

Finally, I implemented an algorithm that computed the probability distribution of the missing $(T+1)^{th}$ observation in each of the sequences of the test data. Suppose $q_{1:T}$ is a sequence of hidden states and $O_{1:T}$ the corresponding observed states generated by a HMM $(A, B, \pi)$. Then

$$\begin{aligned} P[O_{T+1} = j | O_{1:T}] &= \sum_{i=1}^{N} \sum_{k=1}^{N} P[O_{T+1} = j, q_{T+1} = k, q_T = i | O_{1:T}] \\ &= \sum_{i=1}^{N} \sum_{k=1}^{N} \gamma_T(i) A_{ik} B_{kj} \\ &= (\gamma_T A B)(j). \end{aligned}$$

I already had the Forward-Backward algorithm that computes $\alpha_t(i)$ and $\beta_t(i)$ for all $t, i$. From this, it is easy to compute $\gamma_T(:)$ and further matrix multiplication by $A$ and $B$ gives us the required probability distribution. Iterating over all the sequences in the test data set now gives us the required output.