Hidden Markov Model with multiple sequence of observations

Xinyu Gao

Department of Statistics, University of Washington

Abstract: In this report, we constructed a Hidden Markov Model (HMM) and trained the model by Baum-Welch algorithm. In practice, we added scaling method to solve computation problems. To explore the robustness of the convergence in our model, we recorded the iterations and time under the the convergence condition we defined. We found the algorithm converged in less than 500 iterations with precision (0.1) and different number of hidden states (M). We picked M=8 by cross validation and trained the model by using M and all 1000 samples. We optimized the model by Simulated Annealing Algorithm. We also predicted the most likely sequence of hidden states by Viterbi Algorithm and the distribution of next observation for training data and test data.

Keywords: HMM, Lagrange multiplier, Simulated Annealing algorithm

1. Introduction

Hidden Markov Model is an unsupervised learning method and it has been applied to many fields, such as speech tagging, DNA sequence and so on. In this report, we trained HMM by 1000 sequence of observations and made some predictions based on training parameters.

In the part of model construction, we applied Baum-Welch algorithm to maximize the log likelihood and introduced scaling methods to our computation, and used Simulated Annealing algorithm to find the global maximum. Then in results part, we chose the number of hidden states by cross validation, and find the most likely sequence for each sequence of observations by Viterbi algorithm.

2. Model

2.1 Baum-Welch algorithm and maximization the log-likelihood

The **E-step** here is what we have seen in class, i.e.

$$Q(\lambda,\bar{\lambda}) = \sum_k \sum_i \gamma_1^k(i) \ln \bar{\pi}_i + \sum_k \sum_t \sum_{i,j} \xi_t^k(i,j) \ln \bar{a}_{ij} + \sum_k \sum_t \sum_i \gamma_t^k(i) \ln \bar{b}_{io_t} \tag{1}$$

where K is the number of sequences of observations.

Followed by **M-step**, i.e. find the optimal parameters such that maximize Eq.(1):

$$\max_{\lambda} Q(\lambda,\bar{\lambda}) \tag{2}$$

Since the parameters have their constraints, because we need to follow the principle that the probability of all possible events are equal to 1:

$$\sum_{i=1}^{M} \pi_i = 1 \quad \sum_{j=1}^{M} a_{ij} = 1, \ for\ i = 1,...,M \quad \sum_{j=1}^{N} b_{i(o_t=j)} = 1 \tag{3}$$

where M is the number of hidden states, and N is the number of possible states observed.

Hence we need Lagrange multipliers to help us solve this optimization problem. However, in this project, we are going to deal with multiple sequence of observations, so we will derive the M-steps in detail in that part. We first deal with the first part in Eq.(1):

$$Q_1(\pi_i,\lambda) = \sum_k \gamma_t^k(i) \ln \pi_i + \lambda\left(\sum_i \pi_i - 1\right) \tag{4}$$

Then we find the optimal by partial derivatives:

$$\begin{cases} \dfrac{\partial Q_1(\pi_i,\lambda)}{\partial \pi_i} = \dfrac{\sum_k \gamma_1^k(i)}{\pi_i} + \lambda = 0 \\ \dfrac{\partial Q_1(\pi_i,\lambda)}{\partial \lambda} = \sum_i \pi_i - 1 = 0 \end{cases} \Rightarrow \sum_k \gamma_1^k(i) = -\lambda \pi_i \Rightarrow \sum_i \sum_k \gamma_1^k(i) = -\lambda$$

So we have

$$\bar{\pi}_i = \frac{1}{\sum_k \sum_i \gamma_1^k(i)} \sum_k \gamma_1^k(i) = \frac{1}{K} \sum_k \gamma_1^k(i) \tag{5}$$

the second equation holds due to $\sum_i \gamma_1^k(i) = 1$. Then we deal with the second part in Eq.(1)

$$Q_2 = \sum_k \sum_t \xi_t^k(i,j) \ln a_{ij} + \gamma_i \left(\sum_j a_{ij} - 1 \right) \tag{6}$$

We still find its partial derivatives:

$$\begin{cases} \dfrac{\partial Q_2}{\partial a_{ij}} = \dfrac{\sum_k \sum_t \xi_t^k(i,j)}{a_{ij}} + \gamma_i = 0 \\ \dfrac{\partial Q_2}{\partial \gamma_i} = \sum_j a_{ij} - 1 = 0 \end{cases} \Rightarrow \sum_k \sum_t \xi_t^k(i,j) = -\gamma_i a_{ij} \Rightarrow \sum_j \sum_k \sum_t \xi_t^k(i,j) = -\gamma$$

and by $\sum_j \xi_t^k(i,j) = \gamma_t^k(i)$, we have

$$\bar{a}_{ij} = \frac{1}{\sum_{k=1}^K \sum_{t=1}^{T_k - 1} \gamma_t^k(i)} \sum_{k=1}^K \sum_{t=1}^{T_k - 1} \xi_t^k(i,j) \tag{7}$$

Finally, we estimate B:

$$Q_3 = \sum_k \sum_t \gamma_t^k(i) \ln b_{io_t} + \tau \left(\sum_j b_i(o_t^k = j) - 1 \right) \tag{8}$$

Then by

$$\begin{cases} \dfrac{\partial Q_3}{\partial b_i(j)} = \dfrac{\sum_k \sum_t \gamma_t^k(i) 1_{\{o_t^k = j\}}}{b_{io_t}} + \tau = 0 \\ \dfrac{\partial Q_2}{\partial \tau} = \sum_j b_i(j) - 1 = 0 \end{cases} \Rightarrow \sum_k \sum_t \gamma_t^k(i) 1_{\{o_t^k = j\}} = -\tau b_i(j) \Rightarrow \sum_k \sum_t \gamma_t^k(i) = -\tau$$

So we have

$$\bar{b}_i(j) = \frac{1}{\sum_{k=1}^K \sum_{t=1}^{T_k} \gamma_t^k(i)} \sum_{k=1}^K \sum_{t=1, o_t^k = j}^{T_k} \gamma_t^k(i) \tag{9}$$

So far, we have found the re-estimated parameters by Eq.(5), (8) and (9), and the corresponding log likelihood can be expressed as

$$l = \sum_k \log P_k = \sum_k \log \sum_i \alpha_T^k(i) \tag{10}$$

2.2 Scaling problems

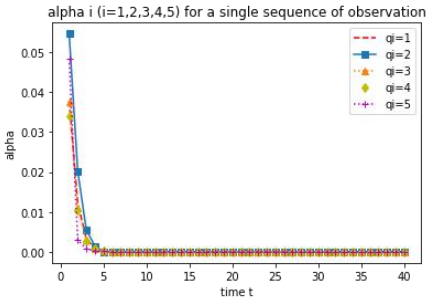

Fig 2.1 Alpha variable for single sequence

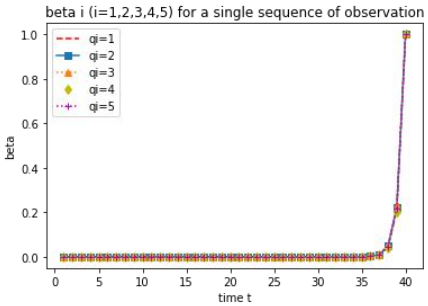

Fig 2.2 Beta variable for single sequence

Before estimating the parameters in Eq.(5),(8) and (9), we need all the values of γ and ξ , all of which are the combinations of α and β by forward and backward algorithm, and they decrease exponentially as t goes infinity, which can be verified in Fig.2.1-2.2. The solution to this problem is to scale alpha's at each time t, and the detailed process can be seen in [1] and [2]. In this report, we just displayed the modified formula as follows and programmed BW based on them:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\alpha_t^k(i)a_{ij}b_{jO_t^k}\beta_{t+1}^k(j)}{\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\beta_t^k(i)/\left(c_t^k\right)} \tag{11}$$

$$\bar{b}_i(j) = \frac{\sum_{k=1}^{K}\sum_{t=1,o_t^k=j}^{T_k}\alpha_t^k(i)\beta_t^k(i)/c_t^k}{\sum_{k=1}^{K}\sum_{t=1}^{T_k}\alpha_t^k(i)\beta_t^k(i)} \tag{12}$$

where $c_t^k = \sum_i \alpha_t^k(i)$ is the scaled factor, we then update alpha's and beta's by dividing this factor. The log likelihood should also be modified as follows:

$$l_{scaled} = \sum_k \sum_t \log c_t^k \tag{13}$$

3. Training

3.1 Programming tools and process

We used python3.6 and the following steps shows how I implemented most of the work. We first began with data cleaning to acquire 1000 sequences of observations, and each observation has the same length (40). Then we built a function called fit_BW_withscale(M,X, max_iter ,eps) to train the parameters, where M is the number of hidden states, X is the observations we will include to train the model, max_iter is the maximum iterations we want to set, and eps is the degree of precision we want to achieve. We initialized the parameters by function random_normalized we created, and then train the parameters by applying forward and backward algorithm, and we estimate the parameters.

3.2 Convergence

Then the iterations stops until the algorithm converges, and the convergence condition is as follows:

$$\left|l_{scaled}^{it+1} - l_{scaled}^{it}\right| < eps \tag{14}$$

where the scaled log likelihood we used can be seen in Eq.(13). We set the precision (eps) to be 0.1 and 1, and the maximum iteration to be 500, and then tried M=4,5,7 and 8, respectively, on all samples to find the iterations and time until it converged. We recorded the iterations and time with distinct M for several trials (Table 3.1) and picked two convergence figures to show the convergence ability of our model (Fig 3.1-3.2).

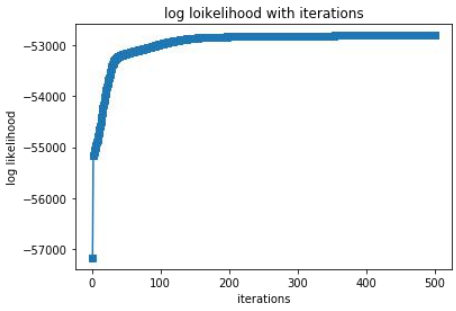

Fig 3.1 log likelihood and iterations (M=7)

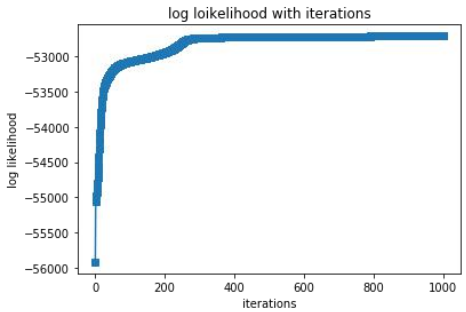

Fig 3.2 log likelihood and iterations (M=8)

From Table 3.1, we found that iterations grows dramatically with the increase of precision, but we had no evidence to conclude that the iterations grows with the increase of M under the same precision. On the other hand, there was an obvious rise of running time with the increase of M, it can be explained by the increasing number of parameters we need to estimate.

Table 3.1 The iterations and time that EM needs to converge

(eps = 1)	Trial 1	Trial 2	Trial 3
M=4	**155** iterations, **250**s	**103** iterations, **166**s	**142** iterations, **230**s
M=5	**121** iterations, **259**s	**134** iterations, **285**s	**117** iterations, **250**s
(eps = 0.1)	Trial 1	Trial 2	Trial 3
M=5	**Out of 500** iterations, **1067**s	**486** iterations, **1037**s	**392** iterations, **814**s
M=7	**Out of 500** iterations, **1886**s	**418** iterations, **1576**s	**485** iterations, **1829**s
M=8	Out of iterations, **2300s**	**478** iterations, **2198**s	**431** iterations, **1982**s

3.3 The number of hidden states (Cross validation)

The number of hidden states (M) we want to choose in our model is a hyper-parameter, which can be derived by K-fold cross validation. We used different K (K=3,4,5) to verify which M is the optimal. In each K-fold CV, we used all data, and M ranged from 4 to 10. When conducting CV, we split data into K folds randomly, trained by K-1 folds, then tested by the left fold, and repeated this process K times.

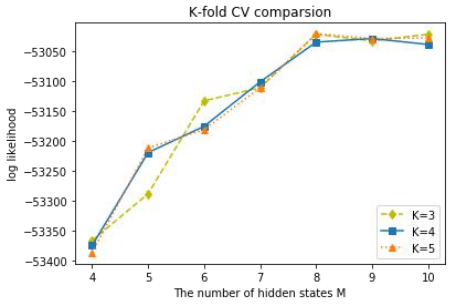

Fig 3.3 K-fold CV comparison

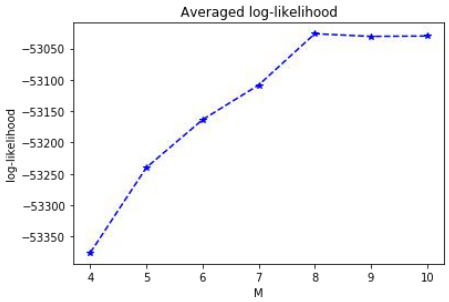

Fig 3.4 The average performance

From Fig 3.3, we can see the log likelihood has a upward trend in M, but in 4-fold and 5-fold CV, log likelihood came to a peak when M=8, so we calculated the average to find its average performance. It turned out that M=8 reached the optimal under the average level, hence we decided to use M=8.

3.4 Optimization by SA algorithm

We can find in Fig.3.1, Baum-Welch algorithm converges if given enough iterations, however, it might converges to the local maximum, which depends on the initialization. If we want to find the global maximum, global search would be a tough work, let alone we have infinity possible initials. Based on this problem, we applied Simulated Annealing algorithm to avoid trapping into the local maximum.

4. Results

4.1 Parameters

The estimated parameter A, B and π optimized by EM and SA are:

$$A_{opt} = \begin{pmatrix} 0.193 & 0.024 & 0.033 & 0.002 & 0.006 & 0.081 & 0.660 & 0.000 \\ 0.000 & 0.216 & 0.000 & 0.634 & 0.067 & 0.001 & 0.000 & 0.082 \\ 0.000 & 0.021 & 0.310 & 0.031 & 0.220 & 0.000 & 0.237 & 0.182 \\ 0.004 & 0.625 & 0.000 & 0.170 & 0.000 & 0.000 & 0.000 & 0.200 \\ 0.000 & 0.046 & 0.664 & 0.000 & 0.104 & 0.000 & 0.012 & 0.173 \\ 0.027 & 0.009 & 0.012 & 0.000 & 0.000 & 0.951 & 0.004 & 0.000 \\ 0.281 & 0.002 & 0.004 & 0.000 & 0.414 & 0.000 & 0.132 & 0.167 \\ 0.040 & 0.030 & 0.000 & 0.000 & 0.655 & 0.000 & 0.000 & 0.274 \end{pmatrix} \quad B_{opt} = \begin{pmatrix} 0.116 & 0.865 & 0.019 & 0.000 \\ 0.731 & 0.088 & 0.182 & 0.000 \\ 0.121 & 0.018 & 0.087 & 0.774 \\ 0.196 & 0.032 & 0.600 & 0.174 \\ 0.414 & 0.450 & 0.124 & 0.012 \\ 0.435 & 0.027 & 0.150 & 0.146 \\ 0.000 & 0.332 & 0.145 & 0.523 \\ 0.002 & 0.064 & 0.934 & 0.000 \end{pmatrix}$$

$$\pi_{opt} = \begin{pmatrix} 0.058 & 0.172 & 0.117 & 0.290 & 0.010 & 0.173 & 0.016 & 0.075 \end{pmatrix}$$

We can find that some of the entries either in A and B approximately equal to 0, and set them all to be 0 to be readable.

4.2 Most likely sequence of states (Viterbi algorithm)

We applied Viterbi (in practice we wrote a function called get_seq) to predict the most likely sequence of states on all the data, and the time it ran was **2.704** seconds. We took the first sample as an example, the most likely sequence of hidden states are: 1, 3, 1, 3, 7, 4, 4, 7, 4, 2, 4, 7, 4, 4, 7, 4, 2, 2, 7, 4, 2, 2, 4, 7, 4, 2, 7, 4, 4, 2, 4, 7, 4, 2, 4, 2, 2, 6, 4, 2.

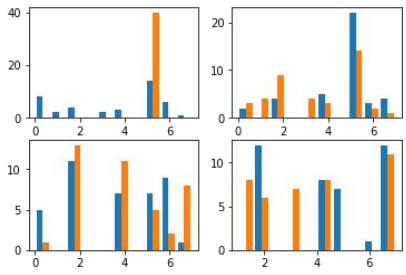

Fig 4.1 Distribution of Hidden states

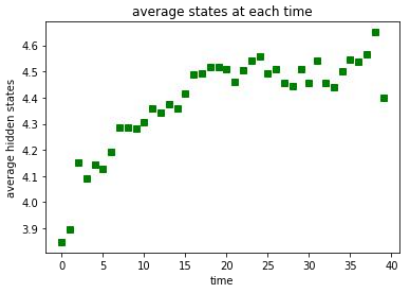

Fig 4.2 Average states at each time

Since the visualization of most likely hidden states of all 1000 sample is rather difficult, so in Fig 4.1, we randomly picked 8 sequences and plot the distribution of their hidden states by histogram, and in Fig 4.2, we plot the average states for each time. The averaged hidden states concentrated on the range of 3.9~4.6, and the averaged states rises in time t, and almost stable in the value of 4.5, which mainly because A_{55} is 0.951, and the hidden states may keep in state 5 once the previous state is in state 5.

4.3 Prediction

We can not only predict the most likely sequence of hidden states given all observations, we can also predict the most likely states of observation at next states given the previous observations based on the trained parameters. In short ,we can make a prediction by the following formula:

$$P(O_{T+1}=l \mid O_{1:T})=\sum_i \sum_j P(O_{T+1}=l \mid q_{T+1}=j)P(q_{t+1}=j \mid q_t=i)P(q_T=i \mid O_{1:T}) \tag{15}$$

which is equal to

$$p_l = P(O_{T+1}=l \mid O_{1:T})=\sum_i \sum_j \gamma_T(i)a_{ij}b_{jl} \tag{16}$$

(where l=0,1,2,...,M-1). Then the distribution of next observation would be

$$\pi_{T+1}=(p_0, p_1, ..., p_{M-1}) \tag{17}$$

Based on the above formula, we also predicted the 40th states of observation for all training samples by returning states with the maximum probability in Eq.(17), and compared the prediction and the true states. As a result, we got **0.638** error rate for training data, and **0.66** for testing data.

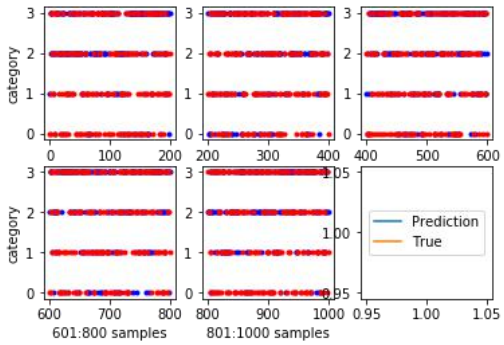

Fig 4.3 The comparison of true states and predicted states at time 40 (Training data)

The comparison of true states of observation and the predicted can be seen in Fig 4.3, and we can find the performance of prediction not well, since the red (true) and blue (prediction) points do not overlap very much.

References:

[1] Daiwei Shen, ‘Some Mathematics for HMM’, https://docplayer.net/78300910-Some-mathematics-for-hmm.html, October, 2008

[2] Lawrence R.Rabiner, ‘A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition’, Proceedings of the IEEE, Vol. 77, No.2, pp.257-286, February, 1989