Hidden Markov Model for Multiple Observation Sequences and its Implementation

Qilong Chen
Department of Statistics, University of Washington
chenql23@uw.edu

This report illustartes how I implement Hidden Markov Model for multiple observation sequences by Python from scratch. Given a training set of multiple observation sequences, I use **Baum-Welch Algorithm** to train the HMM model, calculate the log likelihood of an observation sequence by **Forward-Backward Algorithm** and give the most likely sequence of states by **Viterbi Algorithm** when given an observation sequence. Also, I use **5-fold cross-validation** to get the most likely number of states N, and predict the next output for a given sequence using the idea of **maximum likelihood**. Section 1 is the introduction of training set and test set. Section 2 is the implementation detail for HMM model. Section 3 is the results of section 2.

1 Data Description

Training set It contains $n_0 = 1000$ sequences of $T = 40$ integers in $0, 1, 2, 3$ (one sequence per line, space separated). These sequences have been sampled independently from an unknown distribution.

Test set It contains $n_1 = 50$ sequences of $T = 40$ integers in $0, 1, 2, 3$ (one sequence per line, space separated). These sequences have been sampled independently from the same distribution for training set.

2 Model Implementation

A Hidden Markov Model λ contains three parts: transition matrix A , a $N \times N$ matrix where N is the number of hidden states, determining the probability of transition between any two states; emission probability matrix B , a $N \times M$ matrix where M is the number of observations, determining the probability from each state to each observation; initial probability π, a $N \times 1$ vector, determining the distribution of initial states (t = 0).

2.1 Determine the Number of Hidden States N

Given this dataset, we know that $M = 4$ here. But the number of hidden states N is not determined. Here I use **5-fold cross validation** to find the most possible N from { 5, 6, 7, 8, 9, 10 }.

Since the size of the training set is 1000, the size of training set for 5-fold CV is 800 and the size of validation set for 5-fold CV is 200. For each possible N, I will train the HMM model with the training set and calculate the **average log-likelihood** of each observation sequence in the validation set. I would use the 5-fold average log-likelihood as a criterion for the possibility of a certain N, which means the N with highest average log-likelihood is the most desirable and has the highest possibility of being the true N.

2.2 Initialization of A, B, π

After picking the N with highest average log-likelihood, we know the size of A, B, π. But it still remains to find a proper way to initialize them for training step in Baum-Welch algorithm. Here I use random numbers normalized by rows as the way of initialization. Actually it is a good rule of thumb.

2.3 Training with Baum-Welch Algorithm

After initialization of A, B, π, we are all set for training with Baum-Welch Algorithm. In each iteration of B-W algorithm, we will calculate the **log-likelihood** of the training set by forward-backward algorithm. For a given training set, with n samples, the formula of log-likelihood is

$$\sum_{i=1}^{n} log\Big(P\big(O_{1:T}^{(i)}|\lambda\big)\Big)$$

where $O_{1:T}^{(i)}$ is the i-th obseveration sequence. For convenience, I will use **average log-likelihood** $\frac{1}{n}\sum_{i=1}^{n} log\Big(P\big(O_{1:T}^{(i)}|\lambda\big)\Big)$.

BW algorithm is actually the EM algorithm implemented on HMM model. In each iteration, BW algorithm will assure the increase of log-likelihood of the training set. Therefore, it is natural to set a tolerance value as stopping rule for the iteration. Here I set the **stopping rule** as $AvgLogLikelihood^{(i+1)} - AvgLogLikelihood^{(i)} < tol$, i.e. the difference between log-likelihood in this iteration and in the previous iteration is smaller than a certain value **tol**. In Section 3, I will show that $tol = 0.001$ is enough for convergence when N = $5, 6, 7, 8, 9, 10$.

2.4 Predict the Sequences of States

Given the model trained by B-W algorithm, for each sequence of observations, we can come up with a most likely sequence of states using Viterbi algorithm. It is based on the idea of dynammic progamming.

2.5 Predict the Next Output

Here I use the idea of maximal likelihood to find $O_{T+1} = k$ that maximizes $P(O_{T+1} = k|O_{1:T}, \lambda)$. It is equivalent to maximize $P(O_{T+1} = k, O_{1:T}|\lambda)$ since

$$P(O_{T+1} = k|O_{1:T}, \lambda) = \frac{P(O_{T+1} = k, O_{1:T}|\lambda)}{P(O_{1:T}|\lambda)}$$

and the denominator is independent of O_{T+1}.

So we pick the k from {0,1,2,3} that maximizes $P(O_{T+1} = k, O_{1:T}|\lambda)$ as the predicted value of next state. Using forward algorithm we can easily calculate $P(O_{T+1} = k, O_{1:T}|\lambda)$.

As for testing, we can use the first 39 observations to predict the 40th observation in each sequence. Compared to the true value and predicted output at 40th observation, we can get a accuracy score for the HMM model. The results are in section 3.

3 Results

3.1 Cross-Validation

For N = 5,6,7,8,9,10, I use 5-fold CV method to pick the N with the highest average log-likelihood. Figure 1 shows the value of average log-likelihood vs. the value of N.

As we can see, the 5-fold average log-likelihood for each N is around -53. When $N = 9$, the 5-fold average log-likelihood get its maximum over this range of N. It indicates that $N = 9$ is the most likely value. For $N > 10$, the 5-fold log-likelihood might be smaller, but if N is too large, compared to $M = 4$, it might encounter a problem of overfitting. Besides, the initial values of A, B, π will affect the results, since B-W algorithm only converges into the local maximum, not the global maximum value. So $N = 9$ might not be the true N due to these reasons.

Actually, it is very time consuming to run 5-fold CV. For each N, we will have 5 training procedures to run, and each of them takes around 10 minutes and 200 iterations to converge. So due to time limit, I only consider 6 possible values of N. The total runtime is around 5 hours.

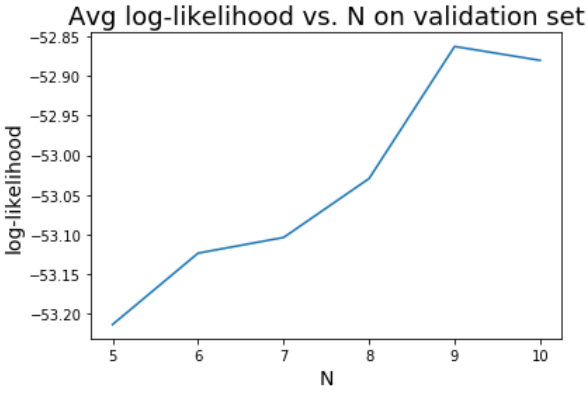

Figure 1: 5-fold Cross Validation

3.2 Training

In this parts, I will show that stopping coefficent $tol = 0.001$ ensures the convergence in training for N = {5,6,7,8,9,10}.

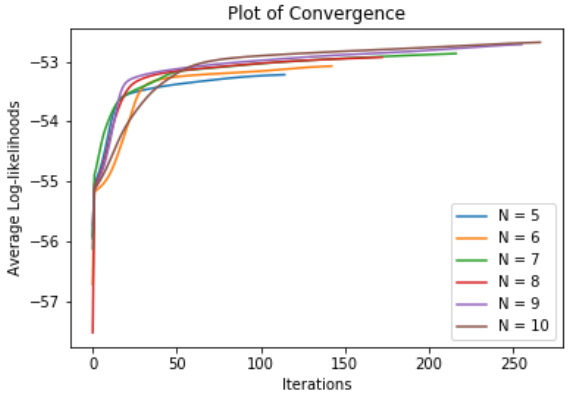

Figure 2: Plot of Convergence

As we can see, for each N, the curve of average log-likelihood increases greatly at the begining and finally converges. So the the stopping coefficent $tol = 0.001$ is enough for covergence for all N's.

For the best $N = 9$ in 5-fold CV, I will report its transition matrix A, emission probability matrix B and the initial probability distribution π as follows. Note that I number the states from 0 to 8 such that the state with the most uniform emission probability (in the sense of **variance**) is labeled 0 and etc.

```
A:
 [[0.184 0.006 0.    0.003 0.762 0.004 0.03  0.002 0.009]
 [0.002 0.039 0.    0.087 0.    0.003 0.124 0.744 0.001]
 [0.057 0.    0.278 0.    0.076 0.318 0.027 0.    0.243]
 [0.    0.    0.    0.322 0.    0.    0.346 0.324 0.008]
 [0.393 0.007 0.015 0.002 0.464 0.094 0.001 0.003 0.021]
 [0.002 0.036 0.398 0.001 0.008 0.165 0.02  0.001 0.369]
 [0.    0.456 0.    0.224 0.001 0.007 0.311 0.    0.001]
 [0.018 0.231 0.002 0.091 0.004 0.017 0.27  0.366 0.    ]
 [0.    0.026 0.001 0.001 0.051 0.535 0.101 0.    0.285]]
```

```
B:
 [[0.004 0.135 0.292 0.568]
 [0.337 0.558 0.104 0.001]
 [0.599 0.008 0.017 0.376]
 [0.664 0.048 0.25  0.038]
 [0.28  0.657 0.021 0.042]
 [0.806 0.169 0.025 0.   ]
 [0.002 0.167 0.821 0.011]
 [0.01  0.065 0.084 0.84 ]
 [0.015 0.059 0.908 0.018]]
pi:
 [0.008 0.09  0.152 0.    0.092 0.207 0.048 0.116 0.287]
```

Particularly, for $N = 9$, the B-W algorithm, on the training set of 1000 sequences, converges in 255 iterations, and it takes around 1178 seconds (around 20 minutes) to run.

3.3 Prediction

For models with $N = 5, 6, 7, 8, 9, 10$, I will give the prediction results in the following parts.

3.3.1 Log-likelihood

Since the size of training set and test set are different, here I use the **average log likelihood** (see Section 2.3) for better comparing the log-likelihood in training set and test set. Also, I will show log-likelihood. The results are as follows.

N	Training		Test	
	Avg log-likelihood	Log-likelihood	Avg log-likelihood	Log-likelihood
5	-53.220	-53220.252	-54.398	-2719.891
6	-53.077	-53077.306	-54.302	-2715.084
7	-52.868	-52868.055	-53.999	-2699.931
8	-52.929	-52929.162	-54.309	-2715.450
9	-52.715	-52715.289	-54.059	-2702.956
10	-52.680	-52679.931	-54.022	-2701.115

Table 1: Log-likelihood for training set and test set

As we can see, for log-likelihood in training set and test set, model with $N = 7$ has the best performance, having the highest log-likelihood among these models. As for the best model $N = 9$ in 5-fold CV, its performance is also not bad, having a relatively high log-likelihood. Therefore, we can see the best model from 5-fold CV is not necessarily the best model here.

3.3.2 Viterbi sequences of states

Using Viterbi algorithm, we can get the most likely sequence of states given a observation sequence and a HMM model. For saving space, I will only show the most likely sequence of states given a sequence and the HMM model with $N = 9$ here, as an example. Other Viterbi sequences of states look similar.

```
[8 5 8 8 5 8 5 8 5 8 8 5 8 8 5 2 5 8 5 2
0 4 0 4 4 4 0 4 0 4 0 0 4 5 8 5 8
 5 2 5]
```

Figure 3: An Example of Viterbi Sequence

Here the states has 9 possible values from 0 to 8. For all sequences in training set, it took 3.10 seconds to get the Viterbi sequences; for all sequences in test set, it took 0.163s to get the Viterbi sequences.

3.3.3 Next output for a given sequence

Under the formula in Section 2.5, we can make prediction for the next output given a observation sequence. Using the 40th observation as testing sample, I can present a training accuracy and test accuracy for models as follows.

N	Accuracy	
	Training	Test
5	0.336	0.220
6	0.335	0.280
7	0.359	0.240
8	0.345	0.240
9	0.361	0.320
10	0.349	0.38

Table 2: Accuracy of Predicted Next Output on Training Set and Test Set

Here, the HMM model with $N = 9$ has the highest training accuracy 0.361 and the model with $N = 10$ has the highest test accuracy 0.38. Even though models with $N < 9$ have a good training accuracy, their values test accuracy are bad. The general accuracy is around 35.0% for $N = 9$ and $N = 10$, indicating that these models are trained well.

In conclusion, based on the performance 5-fold CV, log-likelihood and accuracy of predicted output, the HMM model with $N = 9$ is my best model here.