CS 4786/5786 Competition 2

Predicting the missing values in sequences

Name(netid): Haonan Liu (hl955), Nianze Liu (nl443), Yan Deng (yd256), Yunzhou Zhang (yz824)

Team: HaonanLiu

In this competition, we are given a set of sequences composed of 5 different symbols from 1 to 5. Sequence from 1 to 100 are complete and the task of the competition is to predict the missing value in the rest of 101 to 1000 sequences. The data is generated using a hidden markov model (HMM). But the twist in the tale is that a few (~5%) of the sequences are given in the reverse order of what was generated by the HMM.

1. Successful Trial: HMM

1.1 Introduction

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with hidden states. It is a generative probabilistic model, in which a sequence of observable X variable is generated by a sequence of internal hidden state Z. The hidden states can not be observed directly. The transitions between hidden states are assumed to have the form of a (first-order) Markov chain.

The HMM is completely determined by start probability vector, the transition probability matrix and emission probability distribution with parameters conditioned on the current hidden state.

There are three fundamental problems for HMMs:

1. Given the model parameters and observed data, estimate the optimal sequence of hidden states.
2. Given the model parameters and observed data, calculate the likelihood of the data.
3. Given just the observed data, estimate the model parameters.

1.2 Data Preprocessing: determining the hidden state number

Estimating the number of hidden states is critical in HMM model estimation. In order to best fit the model and choose the proper parameters, we firstly take use of the first 100 sequence as observed data to estimate the model parameters. We divided the first 100 line training data into three classes: training set, validation set and test set. The training set is used to train different type of parameter for the model; the validation set is use to choose the optimal number of hidden state; the test set is used to test how well the the model work. In the model design experiment, we choose the training set accounts for 50% of the

whole training data, the validation set accounts for 30%, and the test set accounts for 20% of the whole training data.

1.2.1 Model training and validation

With the 50% training data set, we train the model by selecting the number of hidden states ranging from 1 to 10.

Estimation method: Parameter estimation of HMM is solved by an iterative Expectation-Maximization (EM) algorithm, known as the Baum-Welch algorithm. The Baum-Welch algorithm has already implemented in the MATLAB's HMM packet or Python's sklearn.hmm packet. The most important thing we need to figure out is the appropriate parameters fed into the HMM fit function. Since the output label is discrete, we decide to use multinomial distribution to determine the emission probabilities.

Initialization: In order to avoid the converge to the local optimum, we randomly initialize the prior, transition matrix of hidden states, and the emission matrix, then normalize it to probability distribution.

Maximum iteration: We also set the maximum number of iteration to be 100 to make result more accuracy. In practice, we saw it converged after around 30-40 iterations.

1.2.2 Model validation

With the parameters estimated by training data, we evaluate the maximum log-likelihood of a set of sequences using the 30% validation data set. The Figure 1(a) shows the average of the maximum log likelihood of a seq for the model with the different number of hidden states ranging from 1 to 10. As the increase of number of hidden states, the log-likelihood increase since the more complicated the model is, the more precise it should be. When the number of state increase to 7, the log value become relatively stable.

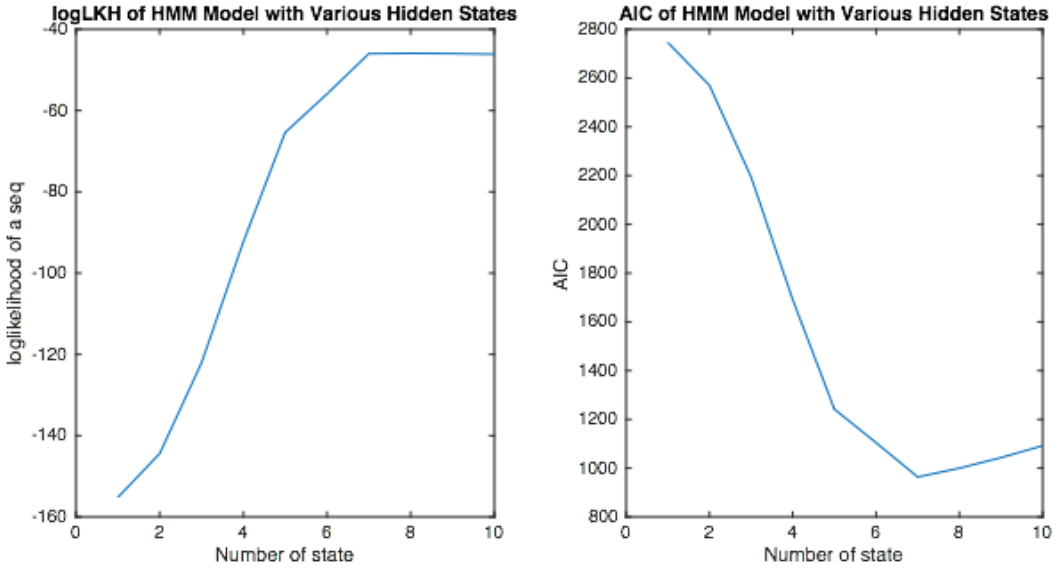
In order to avoid model overfitting, we use Akaike information criterion (AIC), which is a measure of the relative quality of statistical models for a given set of data. AIC rewards goodness of fit, as assessed by the likelihood function, but it also includes a penalty that is an increasing function of the number of estimated parameters. The penalty discourages overfitting. Expression of AIC is shown in equation 1

$$AIC = 2k - 2ln(L) \tag{1}$$

Where, k is the number of estimated parameter, L is the maximum value of the likelihood. In our model, k is quantified as the follow:

$$k = (Nstate - 1) + Nstate * (Nstate - 1) + Nstate * (Nobv - 1) \tag{2}$$

Where Nstate is the number of hidden states, Nobv is the number of observed outcome, Nobv equals to 5 in our case. The first part is the number of independent parameter in prior estimate, the second part is the number of independent parameter in Transition matrix, the third part is the number of independent parameter for emission matrix. Figure 1(b) shows the AIC of HMM model with various hidden state. The preferred model is the one with the minimum AIC value, with the number of state equals to 7.



(a) (b)

Figure 1 Optimal Number of Hidden State Evaluation

1.2.3 Model test

With the optimal number of hidden state, we test the accuracy of the model prediction by the 20% test data set. The algorithm of model test can be summarized as follow. We use the similar idea shown in the algorithm of maximize sequences log-likelihood to predict the 900 sequence with missing value.

1. For each round of simulation n, for each sequence i in test data, randomly pick one position k

2. Filling the value of 1 to 5 into that position, calculate the log likelihood of the sequence in each case, select the value which generate the biggest log likelihood.

3. Compared the observation with the estimate value from the model, quantify the percentage of the estimate value identified with the observation.

Since the number of test data set is relatively small, we run Monte Carlo simulation to test the model prediction accuracy. We randomly pick the position of observation to estimate in each sequence, in each round of simulation.

$$s = \frac{\sum_{n=1}^{N}(x - x_{avg})^2}{N-1} \quad , \quad CI = [x_{avg} - z\frac{s}{\sqrt{N}}, x_{avg} + z\frac{s}{\sqrt{N}}] \tag{3}$$

By running 1000 simulation, we get the average of accuracy as 0.895 , the confidence interval at 95% confidence level as [0.898, 0.903]. Figure 2 is the histogram of the prediction accuracy. We also compared the confidence interval of predicting accuracy from the number of states ranging from 5 to 11, shown in Table 1.

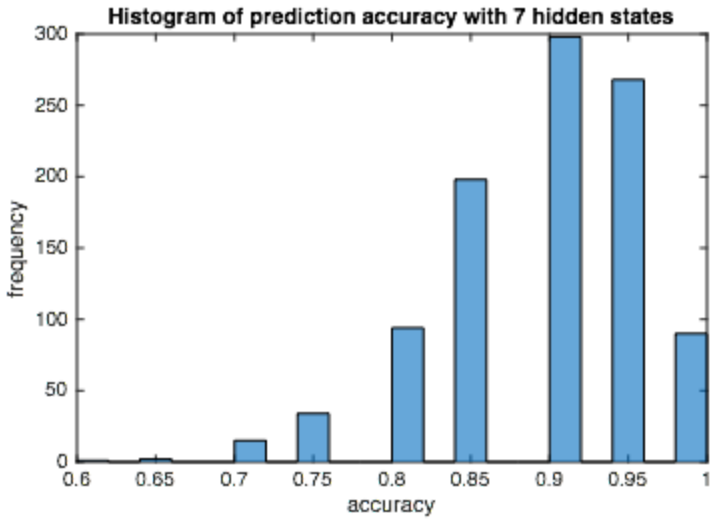


Figure 3 Histogram of prediction accuracy with 7 hidden states

Table 1. Confidence Interval of Predicting Accuracy with Different Number of State

# of state	5	6	7	8	9	10	11
Accuracy	[0.795, 0.806]	[0.8044, 0.8115]	[0.898, 0.903]	[0.9086, 0.9144]	[0.9056, 0.9139]	[0.9091, 0.9170]	[0.9106, 0.9183]

1.2.4 Summary and thought

The accuracy of predicting the test dataset increase as the number of hidden state increase. However, it remains relatively stable after the 7 hidden states. To avoid overfitting, we select 7 as the number of hidden state, and use the result from the model with more than 7 states as the “reference” information.

1.3 Missing value prediction

In the previous step, we have already figured out the appropriate hidden state number as 7 under validation and test. Using the similar method as shown in 1.2, for each new sequence from 101 to 1000 that has a missing value, we first fill the missing value with label 1,2,3,4 and 5, resulting to 5 new different sequences. We compute the log probability under the model we have already trained in the previous step and choose the specific sequence that has the largest log probability as the result of this sequence. The setting of model initialization and maximum iteration are similar with what we described in 1.2.

1.3.1 Prediction with reverse order sequence

Since there are a few (~5%) of the sequences are given in the reverse order of what was generated by the HMM, we also create another 5 new sequences that fills the missing value with label 1,2,3,4 and 5 but in the reversed order, and then calculate the log probability of these 5 reversed sequences under the model we have got in the previous step. And we choose the label that makes the largest log probability from these 10 new sequences as the final prediction result.

1.3.2 Results and discussion

The result of prediction under the HMM is 0.898 using MATLAB or 0.896 using Python. If we use the result from the model with more than 7 states as the "reference" information, that is, comparing the outcome from different models under multiple run of simulations, and select the outcome with most frequent value. The prediction accuracy increases 0.004 to 0.902.

1.4 Other prediction method: Message passing and viterbi algorithm

In addition to the maximum likelihood method mentioned above, there are others way we tried in missing value estimation by HMM. One of them is applied the idea of message passing. Our goal is the estimate the value of x_t given other observations, $x_1, x_2, \ldots, x_{t-1}, x_{t+1}, \ldots, x_n$.

$$P(x_t | x_1, x_2, \ldots, x_{t-1}, x_{t+1}, \ldots, x_n) = \sum_{St} P(x_t, S_t | x_1, x_2, \ldots, x_{t-1}, x_{t+1}, \ldots, x_n) \quad (4)$$

With a dynamic programming the viterbi algorithm, we finding the most likely sequence of hidden states, called the Viterbi path, $s_1, s_2, \ldots, s_{t-1}$ and $s_{t+1}, s_{t+2}, \ldots, s_n$ that results in a sequence of observed events $x_1, x_2, \ldots, x_{t-1}$ and $x_{t+1}, \ldots, x_n$. Based on the property of bayesian network, for the forward message, s_t is independent of all other observations given state of s_{t-1}. The idea is similarly for the backward message given state of s_{t+1}. Therefore, by viterbi algorithm we get the state of s_{t-1} and s_{t+1}, we can get $P(s_t | s_{t-1})$, $P(s_t | s_{t+1})$ and $P(x_t | s_t)$ under transition matrix and emission matrix. By sum product

algorithm of variable elimination, we eliminate s_t then we can estimate the value of x_t with the maximum probability. The result of this method is around 0.88.

2. Successful trial：Simple Markov

2.1. Introduction

Since Hidden Markov model would require certain criteria to decide optimal number of states, and using iteration to update Hidden Markov model's parameters (Transition matrix and Emission matrix) it is possible to have the problem of over-fitting. Though using AIC index may help to avoid over-fitting, another idea of approach is to use the observed sequence's frequency, develop a Markov model with observed states and predicting using conditional probabilities. This is the idea behind the "simple Markov model".

2.2 Methodology:

Parameters: Given a sequence $s_i, s_{i+1}, s_{i+2}, ...s_{i+j}$, assuming that s_{i+j} is dependent on its previous j's observations, then a frequency matrix with j+1 dimension can be created using all 1000 rows of observations (where all states are observed and not), where each dimension represents the possible states of that dimension and the entry would be the observed frequency.

Then, for any chosen dimension, given its "neighbors" we could calculate the conditional probability of observing each states on the chosen dimension, conditioning on its j "neighbors", as a ratio of frequency observing a state over the sum frequency of observing all states on that dimension. This is our probability transition matrix, with j+1 dimension.

Estimates: For an observation requiring estimation, it would fall into 2 categories below:

1. Common Case

Common case infer that this estimating observation has j observations before it and another j observation after it. By having the probability transition matrix, we could calculate the likelihood of that estimating observation of each possible state, as follow:

$$Likelihood = \prod_{i=0}^{j} P_{(s_{n-i+1}, s_{n-i+2}, s_{n-i+3}, ...s_{n-i+j})} \quad (5)$$

And by choosing the maximum likelihood we would obtain the estimate state, conditioning on its neighbor. This would result in a matrix with 2j dimension, where the first j dimension are the j observation before estimating observation, the last j dimension are the j observation after estimating

observation and the entry would be our estimation state. By input observed neighbor observations we would obtain the estimated observation.

2. Missing observation

In certain cases, when the estimating observation is at the beginning of a sequence or at the end of a sequence, we will not have full observation of all its neighbor.

Considering the speciality of beginning and end, a new probability matrix is created as follow:

Consider a case where the kth observation requires estimate, and k < j.

$P_{head}(s_1, s_2, ..., s_k, ...s_j)$ is the frequency ratio given the first j observation excepting the kth one. then :

$$Likelihood = P_{head} \times \prod_{i=0}^{k} P_{(s_{n-i+1}, s_{n-i+2}, s_{n-i+3}, ...s_{n-i+j})} \quad (6)$$

And by choosing the maximum likelihood we would obtain the estimate state, conditioning on its neighbor.

This would result in a matrix with j+k-1 dimension, where the first k-1 dimension are the k-1 observation before estimating observation, the last j dimension are the j observation after estimating observation, and the entry would be our estimation state. By input observed neighbor observations we would obtain the estimated observation.

2.3 Results

The result of prediction under the simple Markov model is 0.87778.

3. Conclusion

In competition 2, we used both HMM and simple Markov chain to predict the missing value in the sequences with the consideration of reverse sequences. Before estimating the missing values, we implemented model training, model validation and model test to determine the best fitting model with the number of states. The best prediction results we got is around 0.902.

Reference

A tutorial on Hidden Markov Models and selected applications in speech recognition, L. Rabiner, 1989, Proc. IEEE 77(2):257--286

Akaike information criterion: https://en.wikipedia.org/wiki/Akaike_information_criterion

Hidden Markov Model (HMM) Toolbox for Matlab www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html