Report of project

TC4 : Using Hidden Markov Model to correct words

ZHANG Xudong, PENG Qixiang

9 janvier 2018

1 First-Order HMM

In this part, we will give a brief introduction to HMM, and explain how to use HMM with Forward-Backward algorithm and Viterbi algorithm on this project.

1.1 Introduction to First-order HMM

Firstly, we note Y_t as the hidden state at time t, and X_t as the observation at time t. The First-order HMM means that :

1. The state at time t is only dependent on the previous state :
   $P(Y_t|Y_{t-1}) = P(Y_t|Y_{t-1}, Y_{t-3}, ..., Y_1)$
2. The observation at time i is only dependant on the current state :
   $P(X_t|Y_t) = P(X_t|Y_t, Y_{t-1}, ..., Y_1)$

So, as a supervised learning, we should learn the following parameters from the train-set :

$A = \{a_{ij}\}$	Transition probability distribution
$B = \{b_j(x_t)\}$	Observation probability distribution in state j
$\pi = \{\pi_i\}$	The initial probability distribution of state

1.2 Forward-backward algorithm

Here is the brief introduction to Forward-backward algorithm.

We define

$$\alpha_t(i) = P(X_{1:t}, Y_t = i, Y_{t-1} = j)$$

and

$$\beta_t(i, j) = P(X_{t+1:T}|Y_t = i, Y_{t-1} = j)$$

And for α, the equation of iteration is like :

$$\alpha_1(i) = b_i(X_1) \times \pi_i$$

and

$$\alpha_{t+1}(i) = b_i(X_{t+1}) \times [\sum_{j=1}^{k} a_{ji} \times \alpha_t(j)]$$

And for β, the equation of iteration is like :

$$\beta_T(i) = 1$$

and

$$\beta_t(i) = \sum_{j=1}^{k} a_{ij} \times b_j(X_{t+1}) \times \beta_{t+1}(j)$$

So, how don we use α and β to find the correct words, in other way, the best hidden state series ?

It's easy to find that :

$$P(Y_t = i|X_{1:T}) \propto \alpha_t(i) \times \beta_t(i)$$

so for each time t, we just consider the best hidden state as :

$$i = arg\max[\alpha_t(i) \times \beta_t(i)]$$

The result of test is shown below :

Forward Backward	baseline error rate	error rate	num corrected error	num created error	TP	FN
HMM1-test10	10.17%	6.76%	304	54	99.2%	59.2%
HMM1-test20	19.40%	13.13%	1309	262	98.05%	58.58%

TABLE 1 – Results for Forward Backward HMM1

We can see the model works well. It can reduce the error rate by one third, and correct about half errors.

1.3 Viterbi algorithm

Viterbi actually aims to find out the path with max probability by using dynamic programming, and this path corresponds to best hidden state series.

We define $\delta_t(i) = \max_{Y_{1:t-1}} P(X_{1:t}, Y_{1:t-1}, Y_t = i)$, which means the path $Y_{1:t}$ with the max probability when the state at time t is i.

We can find it's equation of iteration :

$$\delta_{t+1}(i) = \max_{1<=j<=k} [\delta_t(j)a_{ji}] \times b_i(X_{t+1})$$

We also define $\Psi_t(i) = arg\max_{1<=j<=k}[\delta_{t-1}(j)a_{ji}]$, which means the state at time t-1 in that path.

So the viterbi algorithm is like :

1. Initialization :

$$\delta_1(i) = \pi_i b_i(X_{1)}$$

$$\Psi_1(i) = 0$$

2. Iteration :

$$\delta_t(i) = \max_{1<=j<=k}[\delta_{t-1}(j)a_{ji}] \times b_i(X_t)$$

$$\Psi_t(i) = arg \max_{1<=j<=k}[\delta_{t-1}(j)a_{ji}]$$

3. Termination :

$$P^* = \max_{1<=i<=k} \delta_T(i)$$

$$i_T^* = arg \max_{<=i<=k} \delta_T(i)$$

4. For each time t,

$$i_t^* = \Psi_{t+1}(i_{t+1}^*)$$

The result of test is shown below :

Viterbi	baseline error rate	error rate	num corrected error	num created error	TP	FN
HMM1-test10	10.17%	6.8%	310	63	99.04%	58.38%
HMM1-test20	19.40%	13.13%	1366	319	97.62%	57.82%

TABLE 2 – Results for Viterbi HMM1

Compared with Forward—Backward, it can correct more errors, but meantime, can create more new errors. Overall, it performs a little better than Forward-Backward.

2 Second-Order HMM

In this part, we will explain the model of the second-order HMM. This includes the main hypothesis of this model and parameters that we need to calculate. Then we will give out its usage in Forward-Backward algorithm and Viterbi algorithm, of course with some difference to the first-order problem.

2.1 Introduction to Second-Order HMM

In this part, we give an introduction to the main idea of second-order HMM model. And the parameters of this model.

The second-order model is almost the same as the first-order model except for two difference in the hypothesis of HMM.

The first hypothesis is that we suppose the state at time t is only dependent on the first two states, which means :

$$\begin{aligned} P(Y_{1:T}) &= P(Y_1) \times P(Y_2|Y_1) \times P(Y_3|Y_{1:2}) \times ... \times P(Y_T|Y_{1:T-1}) \\ &= P(Y_1) \times P(Y_2|Y_1) \times P(Y_3|Y_1Y_2) \times ... \times P(Y_T|Y_{T-1}Y_T) \end{aligned}$$

The second hypothesis is that we suppose the observation at time t is only dependent on the first two states, which means :

$$\begin{aligned} P(X_{1:T}|Y_{1:T}) &= P(X_1|Y_{1:T}) \times P(X_2|X_1Y_{1:T}) \times ... \times P(X_T|X_{1:T-1}Y_{1:T}) \\ &= P(X_1|Y_1) \times P(X_2|Y_1Y_2) \times ... \times P(X_T|Y_{T-1}Y_T) \end{aligned}$$

The previous two equation is the crucial idea of second-order HMM. So from the previous two equation, we can find that except for the transition probability distribution matrix $A = \{a_{ij}\}$ and the observation probability distribution matrix $B = \{b_j(o_t)\}$, we need to define a second-order transition probability distribution and a second-order observation probability distribution.

The second-order transition probability distribution is defined as $\hat{A} = \{a_{ijk}\}$ with $a_{ijk} = P(Y_t = i|Y_{t-1} = j, Y_{t-2} = k)$. For the second-order observation probability distribution, we define a $\hat{B} = \{b_{ij}(o_t)\}$ with $b_{ij}(o_t) = P(X_t = o_t|Y_t = i, Y_{t-1} = j)$.

So in all, for a second-order HMM, we have the following parameters to calculate :

$A = \{a_{ij}\}$	Transition probability distribution
$B = \{b_j(o_t)\}$	Observation probability distribution in state j
$\hat{A} = \{a_{ijk}\}$	Second-order transition probability distribution
$\hat{B} = \{b_{ij}(o_t)\}$	Second-order observation probability distribution with $Y_t = i, Y_{t-1} = j$
$\pi = \{\pi_i\}$	The initial probability distribution of state

In the next two section, we will give out the implementation of HMM2 in

Forward-backward algorithm and Viterbi algorithm.

2.2 Smoothing Techniques

On problem for second order HMM is that the matrix for calculation usually becomes really sparse. So It is better that we can implementer some smoothing techniques to solve the problem of sparsity. There are many techniques of smoothing, here we just implementer the simplest one. Here we use the a_{ijk} as an example :

We estimate with

$$P(Y_t = k|Y_{t-1} = j, Y_{t-2} = i) = k_3\frac{N_3}{C_2} + (1-k_3)k_2\frac{N_2}{C_1} + (1-k_3)(1-k_2)\frac{N_1}{C_0}$$

where N_1 denote number of times k occurs

N_2 denote number of times sequence jk occurs

N_3 denote number of times sequence ijk occurs

C_0 denote number of tags appears

C_1 denote number of times j occurs

C_2 denote number of times sequence ij occurs

We define :

$$k_2 = \frac{log(N_2+1)+1}{log(N_2+1)+2}$$

$$k_3 = \frac{log(N_3+1)+1}{log(N_3+1)+2}$$

With this equation, we can well smooth the state transitional matrix. The other matrixes can be calculated with the similar form.

2.3 Forward-backward algorithm in Second-order HMM

Here we try to extend the forward-backward algorithm into the second-order form. We start by define the parameter of the algorithm.

We define

$$\alpha_t(i,j) = P(X_{1:t}, Y_t = i, Y_{t-1} = j)$$

and

$$\beta_t(i,j) = P(X_{t+1:T}|Y_t = i, Y_{t-1} = j)$$

For the first parameter, we can deduce an equation of iteration, the detail is shown below :

$$\begin{aligned}
\alpha_t(i,j) &= P(X_{1:t}, Y_t = i, Y_{t-1} = j) \\
&= P(X_t|X_{1:t-1}, Y_t = i, Y_{t-1} = j) \times P(X_{1:t-1}, Y_t = i, Y_{t-1} = j) \\
&= P(X_t|Y_t = i, Y_{t-1} = j) \times P(X_{1:t-1}, Y_t = i, Y_{t-1} = j) \\
&= b_{ij}(X_t) \times P(X_{1:t-1}, Y_t = i, Y_{t-1} = j)
\end{aligned}$$

We continue with the second term $P(X_{1:t-1}, Y_t = i, Y_{t-1} = j)$. It is not hard to find that :

$$\begin{aligned}
P(X_{1:t-1}, Y_t = i, Y_{t-1} = j) &= \sum_k P(X_{1:t-1}, Y_t = i, Y_{t-1} = j, Y_{t-2} = k) \\
&= \sum_k P(Y_t = i|X_{1:t-1}, Y_{t-1} = j, Y_{t-2} = k) \times P(X_{1:t-1}, Y_{t-1} = j, Y_{t-2} = k) \\
&= \sum_k P(Y_t = i|Y_{t-1} = j, Y_{t-2} = k) \times P(X_{1:t-1}, Y_{t-1} = j, Y_{t-2} = k) \\
&= \sum_k a_{ijk} \times \alpha_{t-1}(j,k)
\end{aligned}$$

So in all, we can write :

$$\alpha_t(i,j) = b_{ij}(X_t) \times \sum_k a_{ijk} \times \alpha_{t-1}(j,k)$$

For the parameter $\beta_t(i,j)$, we can get the similar iteration equation with the same method. The process of deduction is shown below :

$$\begin{aligned}
\beta_t(j,k) &= P(X_{t+1:T}|Y_t = j, Y_{t-1} = k) \\
&= \sum_i P(X_{t+1:T} Y_{t+1} = i|Y_t = j, Y_{t-1} = k) \\
&= \sum_i P(X_{t+1}|X_{t+2:T}, Y_{t+1} = i, Y_t = j, Y_{t-1} = k) \times P(X_{t+2:T}, Y_{t+1} = i|Y_t = j, Y_{t-1} = k) \\
&= \sum_i P(X_{t+1}|Y_{t+1} = i, Y_t = j) \times P(X_{t+2:T}|Y_{t+1} = i, Y_t = j, Y_{t-1} = k) \times P(Y_{t+1} = i|Y_t = j, Y_{t-1} = k) \\
&= \sum_i b_{ij}(X_{t+1}) \times a_{ijk} \times \beta_{t+1}(i,j)
\end{aligned}$$

So we can get :

$$\beta_t(j,k) = \sum_i b_{ij}(X_{t+1}) \times a_{ijk} \times \beta_{t+1}(i,j)$$

To calculate the parameter α_t and β_t, we need to give out the initialization of the first term.

For the parameter $\alpha_t(i,j) = P(X_{1:t}, Y_t = i, Y_{t-1} = j)$, as the index can only starts from 1, we can have a coherent definition from α_2. It can be written into the following form :

$$\begin{aligned}
\alpha_2(i,j) &= P(X_1, X_2, Y_2 = i, Y_1 = j) \\
&= P(X_2|X_1, Y_2 = i, Y_1 = j) \times P(X_1, Y_2 = i, Y_1 = j) \\
&= P(X_2|Y_2 = i, Y_1 = j) \times P(X_1|Y_2 = i, Y_1 = j) \times P(Y_2 = i, Y_1 = j) \\
&= P(X_2|Y_2 = i, Y_1 = j) \times P(X_1|Y_1 = j) \times P(Y_2 = i|Y1 = j) \times P(Y_1 = j) \\
&= b_{ij}(X_2) \times b_j(X_1) \times a_{ij} \times \pi_j
\end{aligned}$$

So in all, we can initialize the forward parameter with :

$$\alpha_2(i,j) = b_{ij}(X_2) \times b_j(X_1) \times a_{ij} \times \pi_j$$

For the backward parameter β_t, it is much easier. We can just initialize :

$$\beta_T(i,j) = 1, \quad \forall i,j \in \{Set\_of\_state\}$$

And now, we can calculate all the forward-backward parameters.

The next part is to find out what we can do with those α and β. We can try with the same process that we have done in first-order HMM and it is not hard to find that :

$$P(Y_t = i, Y_{t-1} = j|X_{1:T}) \propto \alpha_t(i,j) \times \beta_t(i,j)$$

With this equation and the previous discussion, we can predict the state two by two with forward-backward algorithm.

We realized the implementation with the previous discussion. Here there is one thing that we need to state : we predict the first two states by $\max_{i,j} \alpha_1(i,j)\beta_1(i,j)$, then we propagate through $\max_j \sum_i \alpha_1(i,j)\beta_1(i,j)$. The result of test is shown

below : We can find that the performance becomes worse with HMM2. We be-

Forward Backward	baseline error rate	error rate	num corrected error	num created error	TP	FN
HMM2-test10	10.17%	11.07%	146	212	96.77%	80.40%
HMM2-test20	19.40%	19.71%	798	849	93.68%	85.36%

TABLE 3 – Results for Forward Backward HMM2

lieve that the problem is due to the lack of data, which will make the matrix become very sparse. Even if we use some smoothing techniques, they are in fact fake probability, which may give out wrong deduction.

2.4 Viterbi in Second-order HMM

Another algorithm that is used for HMM is Viterbi algorithm. The idea of Viterbi algorithm is to $\max P(X_{1:t}, Y_{1:t})$. It is not hard to extend Viterbi for second-order HMM. Here we give necessary deductions. It is not hard to construct the Viterbi algorithm if you understand the deductions.

First we define $\delta_t(i,j) = \max P(X_{1:t}, Y_{1:t-2}, Y_{t-1} = i, Y_t = j)$ the highest probability that we can get at step t with $Y_{t-1} = i$ and $Y_t = j$. It is not hard to find that :

$$\delta_t(j,k) = [\delta_{t-1} a_{ijk}] b_{jk}(X_t)$$

The aim of Viterbi algorithm is to find the best route. So we define $\phi_t(j,k) = \arg\max_i [\delta_{t-1}(i,j) a_{ijk}]$ which calculates the best route i if j,k are given. Here we define τ_t as the best route at instant t. By calculating until δ_T, we can easily get the last two best route by $\max_{i,j} \delta_T(i,j)$. Then we use ϕ defined before to calculate the previous best route and finally have all the observations tagged.

The only thing here is how to initialize the algorithm, that is how to get $\delta_1(i,j)$. Here we initialize $\delta_1(i,j)$ with the following procedure :

$$\begin{aligned}
\delta_1(i,j) &= P(X_1, X_2, Y_1 = i, Y_2 = j) \\
&= P(X_1 | Y_1, X_2, Y_2) \times P(Y_1, X_2, Y_2) \\
&= P(X_1 | Y_1) \times P(X_2 | Y_1 Y_2) \times P(Y_2 | Y_1) \times P(Y_1) \\
&= \pi_i \times b_i(X_1) \times b_{ij}(X_2) \times a_{ij}
\end{aligned}$$

Now everything has been well explained. We implemented the Viterbi al-

gorithm with the previous discussion and the result of test is shown below :

Viterbi	baseline error rate	error rate	num corrected error	num created error	TP	FN
HMM2-test10	10.17%	7.37%	368	163	97.52%	50.60%
HMM2-test20	19.40%	13.13%	1755	708	94.73%	45.81%

TABLE 4 – Results for Forward Backward HMM2

We can see that Viterbi with HMM2 is better than Forward-Backward with HMM2, but is also much worse than HMM1.

3 Critics and evolution-Insertion of characters

The idea is whether we can change the insertion problem into the substitution problem. We can find that it is not hard to use the same method just by defining a now character empty-space.

If there happens a insertion problem, for the correct word, we add a empty-space at that insertion position to signify that here there should be no word. In this way, we change the insertion problem into the substitution problem. The same method can be used to solve this problem.

4 Critics and evolution-Omitted characters

For the omitted problem, here we suppose that there is at most one omission. It is not only for the simplicity. The problem is that if there are too many omissions, it will become impossible to correct the word. For example, if the last two character of apple is omitted, the new word app is also a correct word. In this way it will be impossible to correct the word.

The implementation will become more hard. For the training data, if there is an omission, we add a empty-space at the position where happens an omission. For prediction. we first correct the word without omission and calculate the probability of this correctness. Then, we suppose that there happens one omission at every possible position (The position between characters and at the beginning and the end of the word). We correct the word under each condition

and compare the probability of each correctness with the correctness without omission. Finally, we use the correctness with the highest probability as the answer. The details for calculating the probability should be developed. Here is just our idea for solving this problem.

5 Unsupervised training

For unsupervised training, what we need is just n observation series $\{X^1, X^2, ..., X^n\}$. So in this project, the data maybe look like :

$$[['t','h','w','e'],['u','s'],['t','h','a','n','k']]$$

Here, in order to figure out the parameters of HMM : A, B, π, we can use EM algorithm.

We don't take about EM here, and just give a brief introduction how to use EM to learn a first-order HMM . Firstly, we define some notions :

$X = \{x_1, x_2, ..., x_T\}$	An observation series, T is the length of series, which is changeable for different series
$Y = \{y_1, y_2, ..., y_T\}$	A state series, T is the length of series, which is changeable for different series
$\lambda = \{A, B, \pi\}$	The parameters which we want to learn
$\bar{\lambda}$	The parameters in current iteration

1. E-step The $Q(\lambda, \bar{\lambda})$ is :

$$Q(\lambda, \bar{\lambda}) = \sum_Y [logP(X, Y|\lambda)]P(X, Y|\bar{\lambda})$$

Because :

$$P(X, Y|\lambda) = \pi_{y_1} b_{y_1}(x_1) a_{y_1 y_2} b_{y_2}(x_2) \cdots a_{y_{T-1} y_T} b_{y_T}(x_T)$$

So we can write $Q(\lambda, \bar{\lambda})$ as :

$$\begin{aligned} Q(\lambda, \bar{\lambda}) = & \sum_{Y} log\pi_{y_1} P(X, Y|\bar{\lambda}) \\ & + \sum_{Y} [\sum_{t=1}^{T-1} loga_{y_t y_{t+1}}] P(X, Y|\bar{\lambda}) \\ & + \sum_{Y} [\sum_{t=1}^{T} logb_{y_t}(x_t)] P(X, Y|\bar{\lambda}) \end{aligned}$$

2. M-step Maximize $Q(\lambda, \bar{\lambda})$

   Because A,B,π exist in three above terms individually, so we use the method of Lagrange multipliers to maximize three terms individually. We get :

$$\pi_i = \frac{P(X, y_1 = i|\bar{\lambda})}{P(X|\bar{\lambda})}$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} P(X, y_t = i, y_{t+1} = j|\bar{\lambda})}{\sum_{t=1}^{T-1} P(X, y_t = i)|\bar{\lambda})}$$

   In this project, the value of observation $x_t \in \{v_1, v_2, ..., v_k\}$, so

$$b_j(k) = \frac{\sum_{t=1}^{T} P(X, y_t = j|\bar{\lambda}) I(x_t = v_k)}{\sum_{t=1}^{T} P(X, y_t = j|\bar{\lambda})}$$