Some Mathematics for HMM

Dawei Shen

October 13th, 2008

This document is written to help you with HMM-related homework problems. All notations used in this document are consistent with the ones used in [1].

1 Specification of an HMM model

To fully specify a hidden Markov model, you need five parameters:

1. N, the number of states in the model. We denote the set of all possible states as $S = \{S_1, S_2, \cdots, S_N\}$, the state at time t as q_t.

2. M, the number of distinct observation symbols per state, i.e., the discrete alphabet size of the output set. We denote the set of all possible output symbols as $V = \{v_1, v_2, \cdots, v_M\}$, the output symbol at time t as O_t. The sequence of observed symbols is denoted as $O = O_1 O_2 \cdots O_T$.

3. The state transition probability distribution $A = \{a_{ij}\}$, where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i, j \leq N$.

4. The observation symbol probability distribution in state $j, B = \{b_j(k)\}$, where $b_j(k) = p[O_t = v_k | q_t = S_j], \quad 1 \leq j \leq N, 1 \leq k \leq M$.

5. The initial state distributions $\pi = \{\pi_i\}$,where $\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N$.

A compact representation of a model is $\lambda = (A, B, \pi)$, where N, M are implicitly implied by A and B. A learning problem is to adjust the model parameter λ, such that $P(O|\lambda)$ is maximized. Usually partial knowledge on the model is available or reasonably assumed, e.g., the number of states N, or some transitions are impossible (some a_{ij}'s are zeros).

We do not reiterate the three problems we frequently encounter in HMM here, but emphasize that algorithms for solving these problems all depend heavily on the usage of forward and backward variables.

2 Forward variables

Forward variables are defined as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda), \tag{1}$$

i.e. the probability of the partial observation sequence, $O_1 O_2 \cdots O_t$, (up to time t) and state S_i at time t, given the model λ. $\alpha_t(i)$ can be obtained inductively as follows:

1. Initialization:
$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N$$
2. Induction:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad 1 \leq t \leq T-1, 1 \leq j \leq N. \tag{2}$$

3 Backward variables

Backward variables are defined as

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, \lambda), \tag{3}$$

i.e., the probability of the partial observation sequence from $t+1$ to the end, given state S_i at time t and the model λ. $\beta_t(i)$ can also be obtained inductively as follows:

1. Initialization:
$$\beta_T(i) = 1, \qquad 1 \leq i \leq N$$
2. Induction:
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \qquad 1 \leq t \leq T-1, 1 \leq j \leq N. \tag{4}$$

The induction step may be less obvious than the one for forward variables. Note that

$$\begin{aligned}
\beta_t(i) &= P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} P(O_{t+1} O_{t+2} \cdots O_T, q_{t+1} = S_j | q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} P(O_{t+1} O_{t+2} \cdots O_T | q_{t+1} = S_j, q_t = S_i, \lambda) P(q_{t+1} = S_j | q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} \underbrace{P(O_{t+2} \cdots O_T | q_{t+1} = S_j, \lambda)}_{\beta_{t+1}(j)} \underbrace{P(O_{t+1} | q_{t+1} = S_j, \lambda)}_{b_j(O_{t+1})} \underbrace{P(q_{t+1} = S_j | q_t = S_i, \lambda)}_{a_{ij}}
\end{aligned} \tag{5}$$

The takeaway is that both forward and backward variables can be computed very efficiently with $O(N^2T)$ calculations. The evaluation problem (the first one of three HMM problems) can be solved by both forward and backward algorithms with either $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$, or $P(O|\lambda) = \sum_{i=1}^{N} \pi_i \beta_1(i) b_i(O_1)$.

The algorithm for solving the state-tracking problem (the second one of the three HMM problems) looks very similar except that the summation symbol at each step should be replaced by a 'max' operation. In this document we won't worry about them at all.

The iterative process of computing forward or backward variables is best graphically presented using a structure called lattice or trellis, with a size of $N \times T$.

4 Two more interesting variables

From forward and backward variables, it's straightforward to calculate some other useful variables easily. The first one is the probability of being in state S_i at time t, given the observation sequence O, and the model λ, denoted using $\gamma_t(i)$:

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{6}$$

The second one is $\xi_t(i, j)$, the probability of being in state S_i at time t, and state S_j at time $t + 1$, given the model and the observation sequence, i.e.,

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} \tag{7}$$

They satisfy the relationship

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j). \tag{8}$$

It is not difficult to figure out how forward and backward variables are used here to compute $\gamma_t(i)$ and $\xi_t(i, j)$. We will skip the details here. Note that $\gamma_t(i)$ and $\xi_t(i, j)$ are both conditioned on the observation sequence O, which looks very friendly and promising, since that's exactly what a learning problem embraces: given the observation sequence, we try learn what the best parameters are.

Next, we will show that Baum-Welch algorithm simply uses these two variables to iteratively update the model until it converges to our satisfaction.

5 Baum-Welch

One key point to understand Baum-Welch algorithm, is to comprehend two interpretations. If we sum $\gamma_t(i)$ over the time index t from 1 to $T - 1$, we get a quantity

$\sum_{t=1}^{T-1}\gamma_t(i)$, which can be interpreted as the expected number of times that state S_i is visited, or the expected number of transitions made from state S_i given the model parameters and the observation sequence O. Similarly, summation of $\xi_t(i,j)$ over t from 1 to $T-1$, i.e., $\sum_{t=1}^{T-1}\xi_t(i,j)$, can be interpreted as the expected number of transitions from state S_i to state S_j given the model parameters and the observation sequence O.

Then, the Baum-Welch algorithm adjusts model parameters using these expected numbers:

- $\bar{\pi}_i = \gamma_1(i)$;
- $\bar{a}_{ij} = \sum_{t=1}^{T-1}\xi_t(i,j)/\sum_{t=1}^{T-1}\gamma_t(i)$;
- $\bar{b}_j(k) = \sum_{t=1,O_t=v_k}^{T}\gamma_t(j)/\sum_{t=1}^{T}\gamma_t(j)$.

This adjustment is intuitively reasonable in that all probabilities of our model parameters are updated by calculating corresponding ratios or proportions. After the update, we obtain a set of new model parameters $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$. We repeat the same process again and again until changes in parameters are smaller than a predefined threshold.

Conceptually, Baum-Welch algorithm can be described as:

- You have a hidden Markov model, a sequence of observed symbols, and model parameters that you believe are pretty good to explain your observations O.
- I am thinking, 'alright, if your model parameters are correct, given the observed sequence O, I should expect (1) the ratio of the number of transitions from state S_i to S_j divided by the number of transitions from state S_i,and (2) the ratio of the number of transitions from state S_j and the observed symbols are v_k divided by the number of transitions from state S_j to match model parameters you give me, i.e. a_{ij}, and $b_j(k)$.'

  How should I do it? I first construct two trellises to calculate forward and backward variables $\alpha_t(i)$'s and $\beta_t(i)$'s by using $\lambda = (A, B, \pi)$. Then, I calculate $\gamma_t(i)$ and $\xi_t(i,j)$, and use the Baum-Welch steps depicted above.
- The ratios I get don't match well the model parameters you provided. I will adjust the model parameters to be the ratios I calculated and tell you that my parameters are better than yours in that $P(O|\bar{\lambda}) > P(O|\lambda)$. My model better explains the data than your model does.
- Well, I am not satisfied with current parameters, I repeat the same process until I am happy.

The question here is, how can I guarantee the updated model is better than the previous one? This can be strictly proved and shown by using optimization theories, which is of less interest here. Just trust the conclusion for now. Note that the algorithm only converges to a local optima, which means the model parameters we obtain from the Baum-Welch algorithm are not optimal in the global sense.

6 Scaling problem

Scaling is usually necessary in the implementation of Baum-Welch re-estimation process. Consider the definition of forward variables in Eq.(1).

$$\begin{aligned}\alpha_t(i) &= P(O_1O_2\cdots O_t, q_t = S_i|\lambda) \\ &= \sum_{q_1q_2\cdots q_{t-1}} P(O_1O_2\cdots O_t, q_t = S_i|q_1q_2\cdots q_{t-1}, \lambda)P(q_1q_2\cdots q_{t-1}|\lambda) \\ &= \sum_{q_1q_2\cdots q_{t-1}} \left[\prod_{s=1}^{t} b_{q_s}(O_s) \prod_{s=1}^{t-1} a_{q_sq_{s+1}}\right]. \qquad (9)\end{aligned}$$

All $a_{q_sq_{s+1}}$ and $b_{q_s}(O_s)$ terms are significantly less than one. Thus, the above summation goes to zero very quickly, i.e., exponentially, as t goes sufficiently large. The precision of the computation of forward variables will exceed the capability of computer processors.

The solution is that at each induction step of the forward algorithm, we need to scale all $a_t(i)$'s appropriately. This scaling factor should only depend on the current time index t, but be independent of the state i.

A commonly used scaling scheme for computing forward variables is

- Initialization

$$\begin{aligned}\ddot{\alpha}_1(i) &= \alpha_1(i) \\ c_1 &= \frac{1}{\sum_{i=1}^{N} \ddot{\alpha}_1(i)} \\ \hat{\alpha}_1(i) &= c_1\ddot{\alpha}_1(i) \qquad (10)\end{aligned}$$

- Induction

$$\begin{aligned}\ddot{\alpha}_t(i) &= \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)a_{ji}b_i(O_t) \\ c_t &= \frac{1}{\sum_{i=1}^{N} \ddot{\alpha}_t(i)} \\ \hat{\alpha}_t(i) &= c_t\ddot{\alpha}_t(i) \qquad (11)\end{aligned}$$

The coefficient c_t is the scaling factor at each step, and c_t only depends on the time index t, not i. Also, the summation of modified forward variable is always 1, i.e., $\sum_{i=1}^{N} \hat{\alpha}_t(i) = 1$. By induction, we can easily prove that

$$\hat{\alpha}_t(i) = \left(\prod_{\tau=1}^{t} c_\tau\right) \alpha_t(i). \qquad (12)$$

If we use this modified forward algorithm to do the evaluation problem, at the last step, we have

$$\begin{aligned} 1 &= \sum_{i=1}^{N} \hat{\alpha}_T(i) = \sum_{i=1}^{N} \left(\prod_{t=1}^{T} c_t \right) \alpha_T(i) \\ &= \left(\prod_{t=1}^{T} c_t \right) \sum_{i=1}^{N} \alpha_T(i) = \left(\prod_{t=1}^{T} c_t \right) P(O|\lambda) \end{aligned} \tag{13}$$

Let $\mathbf{C}_t = \prod_{\tau=1}^{t} c_\tau$, we have $P(O|\lambda) = 1/\mathbf{C}_T$. In logarithm, we have

$$\log[P(O|\lambda)] = -\sum_{t=1}^{T} \log c_t. \tag{14}$$

For backward variables, we use the same scale factors for each time t for β's as were used for α's. More formally,

- Initialization

$$\begin{aligned} \ddot{\beta}_T(i) &= 1 \\ \hat{\beta}_T(i) &= c_T \ddot{\beta}_T(i) \end{aligned} \tag{15}$$

- Induction

$$\begin{aligned} \ddot{\beta}_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j) \\ \hat{\beta}_t(i) &= c_t \ddot{\beta}_t(i) \end{aligned} \tag{16}$$

We can also by induction prove that

$$\hat{\beta}_t(i) = \left(\prod_{s=t}^{T} c_s \right) \beta_t(i) = \mathbf{D}_t \beta_t(i), \tag{17}$$

where we use $\mathbf{D}_t$ to denote $\prod_{s=t}^{T} c_s$.

Now let's consider how Baum-Welch algorithm should be changed using the modified forward and backward variables:

$$\begin{aligned}
\bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
&= \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)} \\
&= \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)/\mathbf{C}_t \cdot a_{ij} b_j(O_{t+1}) \cdot \hat{\beta}_{t+1}(j)/\mathbf{D}_{t+1}}{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)/\mathbf{C}_t \cdot \hat{\beta}_t(i)/\mathbf{D}_t} \\
&= \frac{\left(\sum_{t=1}^{T-1} \hat{\alpha}_t(i) \cdot a_{ij} b_j(O_{t+1}) \cdot \hat{\beta}_{t+1}(j)\right)/\mathbf{C}_T}{\left(\sum_{t=1}^{T-1} \hat{\alpha}_t(i) \cdot \hat{\beta}_t(i)/c_t\right)/\mathbf{C}_T} \\
&= \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i) \cdot a_{ij} b_j(O_{t+1}) \cdot \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \hat{\alpha}_t(i) \cdot \hat{\beta}_t(i)/c_t}. \qquad (18)
\end{aligned}$$

Note that here we use the relationship $\mathbf{C}_t \cdot \mathbf{D}_{t+1} = \prod_{i=1}^{N} c_i = \mathbf{C}_T$, and $\mathbf{C}_t \cdot \mathbf{D}_t = \prod_{i=1}^{t} c_i \cdot \prod_{i=t}^{N} c_i = \mathbf{C}_T \cdot c_t$. Similarly, we have

$$\begin{aligned}
\bar{b}_j(k) &= \frac{\sum_{t=1, O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \\
&= \frac{\sum_{t=1, O_t=v_k}^{T} \hat{\alpha}_t(j) \cdot \hat{\beta}_t(j)/c_t}{\sum_{t=1}^{T} \hat{\alpha}_t(j) \cdot \hat{\beta}_t(j)/c_t}.
\end{aligned} \qquad (19)$$

7 Multiple sequences of observation symbols

Sometimes, we record outputs from a system for multiple times. Each time we reset the system to be in its initial state (the initial state can be random), and the clock for recording time instants is also reset to start from 0. Why do we observe for multiple times rather than observe once but for a very long time? This is because for some models, such as 'left-to-right' models, the transient nature of the states within the model only allow a small number of observations for any state (until a transition is made to a successor state). Therefore, we can not have a single long observation sequence to train the model. To have sufficient data to train the model, we have to observe for multiple times, obtaining multiple sequences of output symbols.

How should the Baum-Welch algorithm be modified?

The idea stays the same: sum up the expected number of transitions of interest and calculate ratios to replace model parameters.

Suppose we have L sequences of observed symbols, indexed by l. The observation length is denoted using T_l. The observation sequences are denotes using $\mathcal{O} = [O^1, O^2, \cdots, O^L]$.

The new learning problem becomes to adjust the parameters of the model λ to maximize

$$P(\mathcal{O}|\lambda) = \prod_{l=1}^{L} P(O^l|\lambda) = \prod_{l=1}^{L} \mathcal{P}_l, \tag{20}$$

where $\mathcal{P}_l$ represents for $P(O^l|\lambda)$, which can be calculated using the forward algorithm as an evaluation problem.

Remember the two interesting variables we talked about above? They now have new meanings and slightly different forms:

$$\begin{aligned}
\gamma_t^l(i) &= P(q_t^l = S_i|\mathcal{O}, \lambda) \\
&= P(q_t^l = S_i|O^l, \lambda) \\
&= \frac{\alpha_t^l(i)\beta_t^l(i)}{P(O^l|\lambda)} = \alpha_t^l(i)\beta_t^l(i)/\mathcal{P}_l,
\end{aligned} \tag{21}$$

and

$$\begin{aligned}
\xi_t^l(i,j) &= P(q_t^l = S_i, q_{t+1}^l = S_j|\mathcal{O}, \lambda) \\
&= P(q_t^l = S_i, q_{t+1}^l = S_j|O^l, \lambda) \\
&= \frac{\alpha_t^l(i)a_{ij}b_j(O_{t+1}^l)\beta_{t+1}^l(j)}{P(O^l|\lambda)} = \alpha_t^l(i)a_{ij}b_j(O_{t+1}^l)\beta_{t+1}^l(j)/\mathcal{P}_l,
\end{aligned} \tag{22}$$

Nothing has been significantly changed from the one-observation sequence case. Note the following remarks:

- For each observation sequence, we need to calculate a different set of forward and backward variables, which is why we put a subscript l on the shoulders of α's and β's.
- The definitions of $\gamma_t^l(i)$'s and $\xi_t^l(i,j)$'s are conditioned on all observed sequences.
- In the Baum-Welch algorithm for the single observation case, we simply ignore the term $\mathcal{P}_l$ since it will be canceled out when we take the ratio. However, as we will show later, this term has to be maintained in the calculation since we can't cancel them out any more by taking the ratio. They are different for different observation sequences.

Finally, count and divide!

The modified Baum-Welch algorithm can be listed as:

1. For each observed sequence, construct two trellises to calculate forward and backward variables $\alpha_t^l(i)$'s and $\beta_t^l(i)$'s.

2. Update!

$$\begin{aligned}
\bar{a}_{ij} &= \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\xi_t^l(i,j)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\gamma_t^l(i)} \\
&= \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\alpha_t^l(i)a_{ij}b_j(O_{t+1}^l)\beta_{t+1}^l(j)/\mathcal{P}_l}{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\alpha_t^l(i)\beta_{t+1}^l(j)/\mathcal{P}_l} \\
\bar{b}_j(k) &= \frac{\sum_{l=1}^{L}\sum_{t=1,O_t^l=v_k}^{T_l}\gamma_t^l(i)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\gamma_t^l(i)} \\
&= \frac{\sum_{l=1}^{L}\sum_{t=1,O_t^l=v_k}^{T_l}\alpha_t^l(j)\beta_t^l(j)/\mathcal{P}_l}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\alpha_t^l(j)\beta_t^l(j)/\mathcal{P}_l} \qquad (23)
\end{aligned}$$

Note that for a 'left-to-right' problem, estimating initial probabilities π_i's is not necessary.

Certainly this algorithm won't work in practice, since we need to do proper scaling. That's how it becomes simpler. The modified Baum-Welch algorithm with scaling:

1. For each observed sequence, construct two trellises to calculate modified forward and backward variables $\hat{\alpha}_t^l(i)$'s and $\hat{\beta}_t^l(i)$'s. For each l, we have a different set of scaling factors c_t^l's.

2. Update!

$$\begin{aligned}
\bar{a}_{ij} &= \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\hat{\alpha}_t^l(i)a_{ij}b_j(O_{t+1}^l)\hat{\beta}_{t+1}^l(j)/(\mathcal{P}_l\mathbf{C}_T^l)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\hat{\alpha}_t^l(i)\hat{\beta}_t^l(i)/(c_t^l\mathcal{P}_l\mathbf{C}_T^l)} \\
&= \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\hat{\alpha}_t^l(i)a_{ij}b_j(O_{t+1}^l)\hat{\beta}_{t+1}^l(j)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\hat{\alpha}_t^l(i)\hat{\beta}_t^l(i)/c_t^l} \\
\bar{b}_j(k) &= \frac{\sum_{l=1}^{L}\sum_{t=1,O_t^l=v_k}^{T_l}\hat{\alpha}_t^l(j)\hat{\beta}_t^l(j)/c_t^l}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\hat{\alpha}_t^l(j)\hat{\beta}_t^l(j)/c_t^l} \qquad (24)
\end{aligned}$$

Note that here we use the fact that $\mathcal{P}_l\mathbf{C}_T^l = 1$. By using scaling, we happily find that all $\mathcal{P}_l$'s terms are cancelled out! The resulting format looks much cleaner!

You have got everything you need for your homework now. Happy coding!

References

[1] 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