

Exploring Hidden Markov Model

JIA Pei

Email: jp4work@gmail.com

Vision Open Working Group

First Edition

April 22, 2010

Contents

0.1	Conception	2
0.2	Gaussian Mixture Model	3
0.3	Evaluation - Forward and Backward Algorithm	4
0.3.1	Forward Algorithm	4
0.3.2	Backward Algorithm	5
0.3.3	Forward Backward Algorithm	6
0.4	Decoding - Posterior vs. Viterbi Algorithm	6
0.4.1	Posterior Algorithm	6
0.4.2	Viterbi Algorithm	7
0.5	Training - Baum-Welch and Viterbi Algorithm	8
0.5.1	Known Corresponding States	9
0.5.2	Baum-Welch Algorithm	10
0.5.3	Viterbi Algorithm	11

0.1 Conception

Markov models and HMM have been well investigated and introductions about both models can be found in [4]. According to [7], a HMM is precisely defined as “a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters, from the observable parameters, based on this assumption.”

Formularily, a HMM could be denoted as $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, where \mathbf{A} is the matrix of hidden states transition probabilities, \mathbf{B} is the matrix of emit probabilities relating hidden states and observation states, π is the initial probabilities of all hidden states.

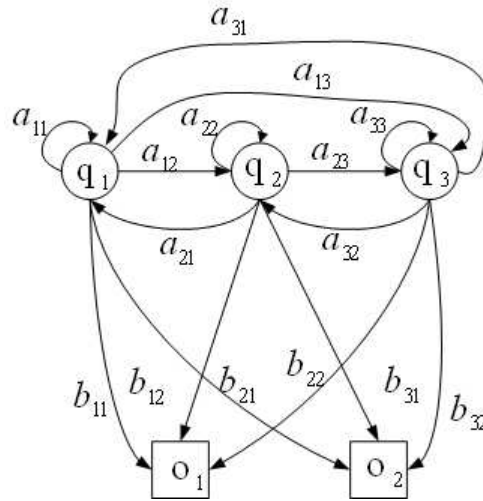


Figure 1: State transitions in an HMM

Figure 1 just shows state transitions in an HMM. \mathbf{Q} represents the \mathbf{N} hidden states; \mathbf{O} represents \mathbf{M} observation states; a_{ij} , the transition probabilities; b_{jk} , output probabilities. Apparently:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix} \quad (1)$$

$$\mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \cdots & \cdots & \cdots & \cdots \\ b_{NM} & b_{N2} & \cdots & b_{NM} \end{pmatrix} \quad (2)$$

and,

$$\sum_{j=1}^N a_{ij} = 1 \quad \forall i \quad (3)$$

$$\sum_{k=1}^M b_{jk} = 1 \quad \forall j \quad (4)$$

If the observation states can't be enumerated, namely, the observation states are infinite, we've got to adopt continuous HMMs. Continuous HMMs use a continuous probability density function, rather than the set of discrete probabilities b_{jk} . Usually the probability density is approximated by a weighted sum of S Gaussian distributions $\mathbf{G}(\mu_{js}, \Sigma_{js})$, namely, Gaussian mixture model. We will talk about GMM in a separate subsection in 0.2.

In real applications of speech recognition, in order to construct such a HMM with known state transition probabilities a_{ij} and emit probabilities b_{jk} , an entire corpus with both the audio and the transcription should be afforded like Wall Street Journal (WSJ) in LDC Corpus collection [3]. Recently, Voxforge <http://www.voxforge.org> emerged on the internet as an open source to collect all sounds with corresponding transcriptions all over the world. Anybody can contribute to this acoustic model under GPL.

There are three main problems of HMM, namely, evaluation, decoding and training. We will elaborate GMM and the three issues of HMM one by one in the following subsections.

0.2 Gaussian Mixture Model

A GMM is defined as a weighted sum of some Gaussian distributions:

$$b_{jo_t} = \sum_{s=1}^S w_{js} \varphi_{\mu_{js}, \Sigma_{js}}(o_t) \quad (5)$$

where o_t is the observation at time t , assuming the current hidden state is j , $\varphi_{\mu_{js}, \Sigma_{js}}(o_t)$ is the value of the probability density function evaluated at point o_t of the s th Gaussian distribution with mean vector μ_{js} and covariance Σ_{js} , w_{js} is the weight for this Gaussian distribution. What's more,

$$\sum_{s=1}^S w_{js} = 1 \quad \forall j \quad (6)$$

Details about how to estimate Gaussian mixture model parameters using Expectation-Maximization (EM) can be found in [1].

0.3 Evaluation - Forward and Backward Algorithm

Given HMM model $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, compute the probability of a particular output sequence \mathbf{O} , i.e., to compute $P(\mathbf{O} | \lambda) = P(o_1, o_2, \dots, o_T | \lambda)$.

Assuming the observations are independent of each other, namely, at time t , the observation o_t is only based on the hidden state q_t , then we have:

$$\begin{aligned}
 P(\mathbf{O} | \lambda) &= P(o_1, o_2, \dots, o_T | \lambda) \\
 &= \sum_{i=1}^{N^T} P(o_1, o_2, \dots, o_T | Q_i, \lambda) P(Q_i | \lambda) \\
 &= \sum_{i=1}^{N^T} P(o_1, o_2, \dots, o_T | q_1^i, q_2^i, \dots, q_T^i, \lambda) P(q_1^i, q_2^i, \dots, q_T^i | \lambda) \quad (7) \\
 &= \sum_{i=1}^{N^T} b_{q_1^i}(o_1) b_{q_2^i}(o_2) \cdots b_{q_T^i}(o_T) \pi_{q_1^i} a_{q_1^i q_2^i} \cdots a_{q_{T-1}^i q_T^i}
 \end{aligned}$$

where, \mathbf{Q}_i is a possible sequence of hidden states. Obviously, totally, there will be N^T possible sequences all together. The time complexity of the above equation is $O(TN^T)$.

With forward-backward algorithm, we are able to dramatically decrease the computation time complexity.

0.3.1 Forward Algorithm

We first define

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, q_t = i | \lambda) \quad (8)$$

It's easy to have the following recursion,

$$\alpha_1(i) = P(o_1, q_1 = i | \lambda) = \pi_i b_i(o_1) \quad (9)$$

and

$$\alpha_t(i) = \left[\sum_{j=1}^N \alpha_{t-1}(j) a_{ji} \right] b_i(o_t) \quad t \neq 1 \quad (10)$$

Then, using total probability, we have:

$$\begin{aligned}
P(\mathbf{O} \mid \lambda) &= P(o_1, o_2, \dots, o_T \mid \lambda) \\
&= \sum_{i=1}^N P(o_1, o_2, \dots, o_T, q_T = i \mid \lambda) \\
&= \sum_{i=1}^N \alpha_T(i)
\end{aligned} \tag{11}$$

The time complexity of forward algorithm is $O(TN^2)$.

0.3.2 Backward Algorithm

We define

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T \mid q_t = i, \lambda) \tag{12}$$

It's easy to have the following recursion,

$$\beta_T(i) = 1 \tag{13}$$

and

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{14}$$

Apparently,

$$\begin{aligned}
P(\mathbf{O} \mid \lambda) &= P(o_1, o_2, \dots, o_T \mid \lambda) \\
&= \sum_{i=1}^N P(o_2, \dots, o_T \mid o_1, q_1 = i, \lambda) P(q_1 = i \mid \lambda) P(o_1 \mid q_1 = i, \lambda) \\
&= \sum_{i=1}^N P(o_2, \dots, o_T \mid q_1 = i, \lambda) \pi_i b_i(o_1) \\
&= \sum_{i=1}^N \pi_i b_i(o_1) \beta_1(i)
\end{aligned} \tag{15}$$

where $1 \leq t \leq T$.

0.3.3 Forward Backward Algorithm

Similarly, it's not hard to have

$$\begin{aligned} P(\mathbf{O} | \lambda) &= P(o_1, o_2, \dots, o_T | \lambda) \\ &= \sum_{i=1}^N P(o_1, o_2, \dots, o_T, q_t = i | \lambda) \\ &= \sum_{i=1}^N P(o_1, o_2, \dots, o_t, q_t = i | \lambda) P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = i, \lambda) \\ &= \sum_{i=1}^N \alpha_t(i) \beta_t(i) \end{aligned} \tag{16}$$

where $1 \leq t \leq T$.

0.4 Decoding - Posterior vs. Viterbi Algorithm

Decoding problem of HMM is to find the hidden states sequence that best explains the observation sequence. Normally, there are two criteria, corresponding to two decoding methods, posterior algorithm, and prestigious Viterbi algorithm.

0.4.1 Posterior Algorithm

Posterior algorithm chooses states that are individually most likely at time t when a symbol o_t is emitted. Let $\delta_t(i)$ be the probability of the model being in state i at time t given an observation sequence \mathbf{O} , namely,

$$\delta_t(i) = P(q_t = i | \mathbf{O}, \lambda) \tag{17}$$

Use the same induction trick as in (16), we have

$$\delta_t(i) = \frac{P(q_t = i, \mathbf{O} | \lambda)}{P(\mathbf{O} | \lambda)} = \frac{\alpha_t(i) \beta_t(i)}{P(\mathbf{O} | \lambda)} \tag{18}$$

Then, posterior decoding determines the t th state as

$$q_t = \arg \max_i [\delta_t(i)] \tag{19}$$

Posterior decoding assumes that the stochastic process is ergodic, which is not a must for general processes. Therefore, Viterbi decoding is introduced next.

0.4.2 Viterbi Algorithm

Viterbi decoding was first proposed by Andrew Viterbi in 1967 [5] and well investigated during 70's of last century [2]. It has been successfully used to decode/recognize the speech.

Given HMM model $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, find the most likely sequence of the hidden states \mathbf{Q} which could have generated a given output sequence \mathbf{O} , i.e., to maximize $P(\mathbf{Q} | \mathbf{O}, \lambda) = P(q_1, q_2, \dots, q_T, | o_1, o_2, \dots, o_T, \lambda)$.

According to Bayes' Theorem,

$$\begin{aligned} P(\mathbf{Q} | \mathbf{O}, \lambda) &= P(q_1, q_2, \dots, q_T, | o_1, o_2, \dots, o_T, \lambda) \\ &= \frac{P(q_1, q_2, \dots, q_T, o_1, o_2, \dots, o_T | \lambda)}{P(o_1, o_2, \dots, o_T | \lambda)} \end{aligned} \quad (20)$$

Apparently, the denominator of (20) is a constant given the observed sequence $\mathbf{O} = o_1, o_2, \dots, o_T$, which could be computed by forward-backward algorithm in the previous subsection 0.3. So, we may ignore it in the following deduction.

If we define $\delta_t(i)$ is the highest probability path ending in hidden state i :

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1, q_2, \dots, q_t = i, o_1, o_2, \dots, o_t | \lambda) \quad (21)$$

then, to maximize $P(\mathbf{Q} | \mathbf{O}, \lambda)$, is just equal to compute the maximum $\delta_T(i), 1 \leq i \leq N$. Namely, the probability of the most likely sequence is:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \quad (22)$$

Besides, it's not hard to have the following recursion:

$$\delta_1(i) = \pi_i b_i(o_1) \quad (23)$$

and

$$\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(o_t) \quad (24)$$

The above is the recursion to calculate the biggest probability given a sequence of observations. To find the best path under Viterbi decoding, an additional matrix ψ of size $T * N$ needs to be introduced. Viterbi decoding can be summarized in the following main parts.

1. Initialization:

- $\psi_1(i) = 0$
- $\delta_1(i) = \pi_i b_i(o_1)$

2. Recursion:

- $\psi_t(j) = \arg \max_i [\delta_{t-1}(i) a_{ij}]$
- $\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(o_t)$

3. Termination:

- $P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$
- $q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)]$

4. Path Backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad t = T - 1, T - 2 \dots 1 \quad (25)$$

As shown, Viterbi algorithm is quite similar to forward algorithm, only except two differences [4]:

1. Viterbi algorithm is using maximization in place of summation during the recursion.
2. Viterbi algorithm keeps track of the arguments that maximize $\delta_t(i)$ for each time t , and store the data in the matrix ψ . ψ is just the matrix to retrieve the most probable state sequence during backtracking, using equation (25).

0.5 Training - Baum-Welch and Viterbi Algorithm

In a nutshell, training problem of HMM is to find the most suitable HMM model based on given output sequences $\mathbf{O}_i, 0 \leq i < L$, where L is the number of training sequences. It can be categorized into two problems: the one with known state sequences \mathbf{Q}_i corresponding to \mathbf{O}_i , and the one without. Baum-Welch training, together with Viterbi training are both for the second problem.

We first enumerate some preliminary definitions of HMM.

1. λ_0 – the initial HMM model

2. $\xi_t(i, j)$ – the probability of being in state i at time t and in state j at time $t + 1$. Given the current estimated HMM model λ and one particular sequence of output observations $\mathbf{O} = o_1, o_2, \dots, o_T$. Apparently,

$$\begin{aligned}
\xi_t(i, j) &= P(q_t = i, q_{t+1} = j \mid o_1, o_2, \dots, o_T, \lambda) \\
&= \frac{P(q_t = i, q_{t+1} = j, o_1, o_2, \dots, o_T \mid \lambda)}{P(o_1, o_2, \dots, o_T \mid \lambda)} \\
&= \frac{P(o_1, o_2, \dots, o_t, q_t = i, q_{t+1} = j, o_{t+1}, o_{t+2}, \dots, o_T \mid \lambda)}{P(\mathbf{O} \mid \lambda)} \quad (26) \\
&= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} \mid \lambda)}
\end{aligned}$$

3. $\gamma_t(i)$ – the probability of being in state i at time t , given one sequence of output observations $\mathbf{O} = o_1, o_2, \dots, o_T$. Apparently,

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (27)$$

0.5.1 Known Corresponding States

If we know sequences of states \mathbf{Q}_i corresponding to the sequences of observations \mathbf{O}_i , it will be quite easy for us to build up a HMM immediately.

Given corresponding states and observations, obviously, we may compute:

1. $\sum_{t=1}^T \gamma_t(i)$ – the expected number of times that the hidden state i is visited;
2. $\sum_{t=1}^{T-1} \xi_t(i, j)$ – the expected number of times that the transition from the hidden state i to the hidden state j is visited.

Now, we are able to construct the HMM model by calculating the following statistics:

1. $\bar{\pi}_i$ – the expected frequency in state i at time $t = 1$. Apparently,

$$\bar{\pi}_i = \gamma_1(i) \quad (28)$$

2. \bar{a}_{ij} – (the expected number of times that the transition from the hidden state i to the hidden state j) / (the expected number of all transitions from state i). Apparently,

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (29)$$

3. $\bar{b}_i(o_t)$ – (the expected number of times being in the hidden state i with the observation state o_t) / (the expected number of times being in state i). Apparently,

$$\bar{b}_i(o_t) = \frac{\sum_{t=1, o_t}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (30)$$

However, it's not always the case that the sequences of states are known. The problem will come to the following one:

Given output sequences \mathbf{O}_i without known corresponding state sequences \mathbf{Q}_i , calculate the most likely set of state transition and output probabilities, i.e., find the most suitable HMM model $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ to maximize the output probabilities $P(\mathbf{O} | \lambda) = P(o_1, o_2, \dots, o_T | \lambda)$.

In fact, there is no analytic solution for this problem. Two classical methods, namely Baum-Welch algorithm [6] and Viterbi training are two iterative methods to approximate the most probable HMM model.

0.5.2 Baum-Welch Algorithm

Baum-Welch algorithm is briefly summarized in algorithm 1.

Algorithm 1 Baum-Welch Algorithm

1: Initialization:

- Randomly initialize a HMM as λ_0
- $diff = MAX, DELTA, EPOCH = 50, count = 0$

2: **while** ($diff > DELTA \ \&\& \ count < EPOCH$) **do**

3: Update $\xi_t(i, j)$ and $\gamma_t(i)$ according to equations (26) and (27) progressively, based on the current HMM model λ_0 and observations $\mathbf{O}_i \ 0 \leq i < L$.

4: Compute new λ based on the current λ_0 , using equations (28), (29) and (30). Make sure the calculated $\bar{\pi}_i, \bar{a}_{ij}$ and $\bar{b}_i(o_t)$ are all normalized.

5: Calculate the log likelihood by

$$diff = \| \log P(\mathbf{O} | \lambda) - \log P(\mathbf{O} | \lambda_0) \| \quad (31)$$

6: $\lambda_0 = \lambda$

7: $count++$;

8: **end while**

9: **return** λ_0

0.5.3 Viterbi Algorithm

Viterbi is also possible to train a HMM. Refer to algorithm 2.

Algorithm 2 Viterbi Algorithm

1: Initialization:

- Randomly initialize a HMM as λ_0
- $diff = MAX, DELTA, EPOCH = 50, count = 0$

2: **while** ($diff > DELTA \ \&\& \ count < EPOCH$) **do**

3: Find the most probable state sequences \mathbf{Q}_i for corresponding observation sequences \mathbf{O}_i , using Viterbi decoding algorithm, based on the current HMM model λ_0 .

4: Compute new λ based on the derived \mathbf{Q}_i and the observation sequences \mathbf{O}_i . Make sure the calculated $\bar{\pi}_i, \bar{a}_{ij}$ and $\bar{b}_i(o_t)$ are all normalized.

5: Calculate the log likelihood by

$$diff = \|\log P(\mathbf{O}|\lambda) - \log P(\mathbf{O}|\lambda_0)\| \quad (32)$$

6: $\lambda_0 = \lambda$

7: $count++$;

8: **end while**

9: **return** λ_0

Bibliography

- [1] J. A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, University of Berkeley, Berkeley CA, 94704, USA, April 1997.
- [2] G. D. Forney. The viterbi algorithm. In *Proceedings of the IEEE*, volume 61 of 3, pages 268 – 278, March 1973.
- [3] J. Garofalo, D. Graff, D. Paul, and D. Pallett. Csr-i (wsj0) complete. Linguistic Data Consortium, Philadelphia, 1993.
- [4] V. Petrushin. Hidden markov models: Fundamentals and applications. In *Online Symposium for Electronics Engineers*, 2000.
- [5] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260269, April 1967.
- [6] L. R. Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4), December 2003.
- [7] Wikipedia. Hidden markov model — wikipedia, the free encyclopedia, 2006. [Online; accessed 28-December-2007].