A TUTORIAL ON HIDDEN MARKOV MODELS

Rakesh Dugad* U. B. Desai

Signal Processing and Artificial Neural Networks Laboratory
Department of Electrical Engineering
Indian Institute of Technology — Bombay
Powai, Mumbai 400 076, India

dugad@uiuc.edu, ubdesai@ee.iitb.ernet.in

Technical Report No. : SPANN-96.1

May 1996

Abstract

In this tutorial we present an overview of (i) what are HMMs, (ii) what are the different problems associated with HMMs, (iii) the Viterbi algorithm for determining the optimal state sequence, (iv) algorithms associated with training HMMs, and (v) distance between HMMs.

1 Introduction

[1]

Suppose a person has say three coins and is sitting inside a room tossing them in some sequence– this room is closed and what you are shown (on a display outside the room) is only the outcomes of his tossing TTHTHHTT. . . this will be called the **observation sequence** . You do not know the sequence in which he is tossing the different coins, nor do you know the bias of the various coins. To appreciate how much the outcome depends on the individual biasing and the order of tossing the coins, suppose you are given that the third coin is highly biased to produce heads and all coins are tossed with equal probability. Then, we naturally expect there to be far greater number of heads than tails in the output sequence. Now if it be given that besides the bias the probability of going to the third coin (state) from either the first or the second coin (state) is zero; then assuming that we were in the first or second state to begin with the heads and tails will appear with almost equal probability in spite of the bias. So we see that the output sequence depends very much on the individual bias, the transition probabilities between various states, as well as on which state is chosen to begin the observations. The three sets, namely, the set of individual bias of the three coins, the set of transition probabilities from one coin to the next and the set of initial probabilities of choosing the states characterize what is called as the **HIDDEN MARKOV MODEL(HMM)** for this coin tossing experiment. We shall shortly see the problems and their solutions that come under the framework of HMMs.

*Rakesh Dugad is currently with Beckman Institute, ECE Department, University of Illinois, 405 N. Mathews Avenue, Urbana IL 61801, USA

2 Notations

In the above experiment the outcomes, of tossing of each coin, T or H are called the **observation symbols**. Because we considered coins, only two observation symbols were possible. To be more general, consider a set of N urns each consisting of a number of marbles. The marbles are of M distinct colors. Within each urn the marbles are of distinct colors. Our experiment consists of drawing marbles from these urns in some sequence; only the sequence of marbles drawn is shown to us. Marbles, here, correspond to the T or H and urns to coins in the above experiment. We now define the following notation for our model :

N = number of states (urns) in the model

M = total number of distinct observation symbols (marbles of M distinct colors)

T = length of observation sequence i.e. the number of symbols observed

1,2,...,N will denote the N urns respectively

i_t denotes the state in which we are at time t

$V = \{v_1,\dots,v_M\}$ the discrete set of possible observation symbols

$\pi = \{\pi_{\rm i}\}$, $\pi_{\rm i}$ = P(i_1 = i), the probability of being in state i at the beginning of the experiment i.e. at t=1

$A = \{a_{ij}\}$ where a_{ij} = P(i_{t+1}=j | i_t= i), the probability of being in state j at time t+1 given that we were in state i at time t. We assume that a_{ij}'s are independent of time.

$B = \{b_j(k)\}$,$b_j(k)$ = P(v_k at t | i_t = j), the probability of observing the symbol v_k given that we are in state j.

O_t will denote the observation symbol observed at instant t

$\lambda = (A,B,\pi)$ will be used as a compact notation to denote an HMM

Using the model, an observation sequence $O = O_1,O_2,\dots,O_T$ is generated as follows : We start our experiment by choosing one of the urns (according to the initial probability distribution π), then we choose a marble (observation symbol) from this urn – this beginning instant of time is taken as t=1 and the state and observation symbol chosen at this t=1 are denoted by i_1 and O_1 respectively. After this we choose an urn (may be same or different from the urn at t=1) according to the transition probability distribution A and again select a marble (denoted by O_2) from this urn depending on the observation symbol probability distribution $b_j(k)$ for that urn (state). Continuing this up to time t=T, generates the observation sequence $O = O_1,O_2,\dots,O_T$.

3 The Three Problems for HMMs

Most applications of HMMs are finally reduced to solving three main problems. These are :

Problem 1 : Given the model $\lambda = (A,B,\pi)$ how do we compute P(O| λ) , the probability of occurrence of the observation sequence $O = O_1,O_2,\dots,O_T$.

Problem 2 : Given the model $\lambda = (A,B,\pi)$ how do we choose a state sequence $I = i_1,i_2,\dots,i_T$ so that P(O,I| λ) ,the joint probability of the observation sequence $O = O_1,O_2,\dots,O_T$ and the state sequence given the model is maximized.

Problem 3 : How do we adjust the HMM model parameters $\lambda = (A,B,\pi)$ so that P$(O|\ \lambda)$ (or P$(O,I|\ \lambda)$) is maximized.

Problems 1 and 2 can be viewed as analysis problems while Problem 3 is a typical synthesis (or model identification or training) problem.

Solutions to the Three Problems

3.1 Problem 1

A most straightforward way to determine P$(O|\ \lambda)$ is to find P$(O \mid I,\lambda)$ for a fixed state sequence $I = i_1,i_2,\ldots,i_T$ then multiply it by P$(I|\ \lambda)$ and then sum up over all possible I's. We have

$$\mathrm{P}(O \mid I,\lambda) = b_{i_1}(O_1)b_{i_2}(O_2)\cdots b_{i_T}(O_T). \tag{1}$$

$$\mathrm{P}(I|\ \lambda) = \pi_{i_1}a_{i_1i_2}a_{i_2i_3}\cdots a_{i_{T-1}i_T}. \tag{2}$$

Hence we have:

$$\mathrm{P}(O|\ \lambda) = \sum_I \mathrm{P}(O \mid I,\lambda)\ \mathrm{P}(I|\ \lambda) \tag{3}$$

$$\sum_I \pi_{i_1}b_{i_1}(O_1)a_{i_1i_2}b_{i_2}(O_2)\cdots a_{i_{T-1}i_T}b_{i_T}(O_T). \tag{4}$$

where $I = i_1,i_2,\ldots,i_T$.

From Eq. 4 we see that the summand of this equation involves $2T$–1 multiplications and there exists N^T distinct possible state sequences I. Hence a direct computation from (4) will involve of the order of $2TN^T$ multiplications. Even for small values, N =5 and T=100, this means approximately 10^{72} multiplications which would take eons to complete even for a supercomputer. Hence we see that a more efficient procedure is required to solve Problem 1; such a procedure exists and is called the forward-backward procedure.

Forward-Backward Procedure

Consider the forward variable $\alpha_t(i)$ defined as :

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, i_t = i\ \mid \lambda) \tag{5}$$

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

1.
$$\alpha_1(i) = \pi_i b_i(O_1)\ , 1 \leq i \leq N \tag{6}$$

2. for t = 1,2,...,T–1, $1 \leq j \leq N$
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(O_{t+1}) \tag{7}$$

3. then we have:
$$\mathrm{P}(O|\ \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{8}$$

In Step 2 we want to compute the probability of partial observation sequence up to time t+1 and state j at time t+1; state j can be be reached (with probability a_{ij}) independently from any of the N states at time t. The summation in Eq. 7 refers to this fact. Also the summand gives observation sequence up to time t; hence the $b_j(O_{t+1})$ outside the brackets. In Step 3 we just sum up all possible (independent) ways of realizing the given observation sequence. Let us illustrate this by taking the case of

$$\alpha_2(j) = P(O_1, O_2, i_2 = j\ )$$

(We have dropped dependence on λ for convenience). The sequence O_1,O_2,i_2=j occurs as: first O_1, then state i_2, then object O_2. But O_1 can occur through any of the following mutually exclusive and exhaustive ways : state 1 then O_1, state 2 then O_1, and so on up to state N. We know that if $\{S_i\}$ be a set of mutually exclusive and exhaustive events then for any event E we have

$$P(E) = \sum_i P(E \mid S_i)P(S_i) \tag{9}$$

Hence here we can write

$$\begin{aligned}
\alpha_2(j) &= P(O_1, O_2, i_2 = j\ ) \\
&= \sum_i P(O_2, i_2 = j \mid O_1\text{from state } i)P(O_1\text{from state } i) \\
&= \sum_i ([P(O_2 \mid i_2 = j, O_1\text{from state } i)P(i_2 = j \mid O_1\text{from state } i)]\,[P(O_1 \mid i_1 = i)P(i_1 = i)]) \\
&= \sum_i ([P(O_2 \mid i_2 = j)P(i_2 = j \mid i_1 = i)]\,[P(O_1 \mid i_1 = i)P(i_1 = i)]) \\
&= \sum_i ([b_j(O_2)\ \mathrm{a}_{ij}]\,[b_i(O_1)\ \pi_i]) \\
&= \left[\sum_i (\pi_i b_i(O_1)\ )\mathrm{a}_{ij}\right] b_j(O_2) \\
&= \left[\sum_i \alpha_1(i)\mathrm{a}_{ij}\right] b_j(O_2) \\
&\ \text{(using (6) above)}
\end{aligned}$$

which is the same as (7) above. Hence this simple example gives an insight into the mathematical justification of (7).

As for (8) using (9) we can write

$$\begin{aligned}
P(O) &= \sum_i P(O \mid i_T = i)P(i_T = i) \\
&= \sum_i P(O, i_T = i) \\
&= \sum_i \alpha_T(i) \\
&\ \text{(using the definition given in (5))}
\end{aligned}$$

Hence proved.

Now let us examine the number of computations involved in this algorithm. Step 1 involves N multiplications. In Step 2 the summation involves N multiplications plus one for the out of bracket $b_j(O_{t+1})$ term—this has to be done for j=1 to N and t = 1 to T−1, making the total number of multiplications in Step 2 as (N+1)N(T–1). Step 3 involves no multiplications. Hence total number of multiplications is N+N(N+1)(T–1) i.e. of the order of N^2T as compared to $2T \cdot N^T$ required for the direct method[1]. For N=5 and T=100 we need about 3000 computations for the forward method as compared to 10^{72} required by the direct method–a saving of about 69 orders of magnitude!

Backward Procedure

In a similar manner we may define a backward variable $\beta_t(i)$ as:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T \mid i_t = i, \lambda) \tag{10}$$

i.e. the probability of the observation sequence from t+1 to T given the state i at time t and the model λ. Note that here i_t=i has already been given (it wasn't in the case of the forward variable). This distinction has been made to be able to combine the forward and the backward variables to produce useful results, as we shall see soon. We can easily solve for $\beta_t(i)$ as done for $\alpha_t(i)$:

1.
$$\beta_T(i) = 1, \quad 1 \le i \le N \tag{11}$$

2. for t = T–1,T–2,...,1, $1 \le i \le N$
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\ \beta_{t+1}(j) \tag{12}$$

3.
$$\mathrm{P}(O|\ \lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1)\ \beta_1(i) \tag{13}$$

The proof of (12) and (13) is similar to the one given for (7) and (8) and is left as an exercise for the reader.

The computation of P(O| λ) using $\beta_t(i)$ also involves of the order of N^2T calculations. Hence both the forward as well as the backward method is equally efficient for the computation of P(O| λ) . This solves Problem 1.

3.2 Problem 2

Here we have to find a state sequence $I = i_1, i_2, \ldots, i_T$ such that probability of occurrence of the observation sequence $O = O_1, O_2, \ldots, O_T$ from this state sequence is greater than that from any other state sequence. In other words, our problem is to find I that will maximize P(O,I| λ) . There is a famous algorithm to do this, called the **Viterbi Algorithm** [2]. It is an inductive algorithm in which at each instant you keep the best (i.e. the one giving maximum probability) possible state sequence for each of the N states as the intermediate state for the desired observation sequence $O = O_1, O_2, \ldots, O_T$. In this way you finally have the best path for each of the N states as the last state for the desired observation sequence. Out of these, we select the one which has highest probability.

In order to get an idea of the Viterbi algorithm as applied to the optimum state estimation problem a simple reformulation of the problem will be useful

Consider the expression for $P(O,I|\lambda)$; from (1)-(2) we have

[1]On similar lines it can be seen that the number of additions involved is N(N–1)(T–1) + (N–1).

$$\begin{aligned} P(O,I|\lambda) &= P(O|I,\lambda)P(I|\lambda) \\ &= \pi_{i_1} b_{i_1}(O_1) a_{i_1 i_2} b_{i_2}(O_2) \cdots a_{i_{T-1} i_T} b_{i_T}(O_T) \end{aligned}$$

Now define

$$U(\ i_1,i_2,\ldots,i_T\ ) = -\left[\ln\left(\pi_{i_1} b_{i_1}(O_1)\right) + \sum_{t=2}^{T} \ln\left(a_{i_{t-1} i_t} b_{i_t}(O_t)\right)\right] \tag{14}$$

then it is easily seen that

$$P(O,I|\lambda) = exp(-U(\ i_1,i_2,\ldots,i_T\ ))$$

consequently the problem of optimal state estimation, namely,

$$\max_{\{i_t\}_{t=1}^{T}} P(O, i_1 \cdots i_T|\lambda)$$

becomes equivalent to

$$\min_{\{i_t\}_{t=1}^{T}} U(\ i_1,i_2,\ldots,i_T\ ) \tag{15}$$

This reformulation now enables us to view terms like $-\ln(a_{i_j i_k} b_{i_k}(O_t))$ as the cost associated in going form state i_j to state i_k at time t.

The Viterbi algorithm to find the optimum state sequence can now be described as follows: Suppose we are currently in state i and we are considering visiting state j next. We shall say that the weight on the path from state i to state j is $-\ln(a_{ij} b_j(O_t)\ )$ (i.e. the negative of logarithm of probability of going from state i to state j and selecting the observation symbol O_t in state j) where O_t is the observation symbol selected after visiting state j–this is the same symbol that appears in the given observation sequence $O = O_1,O_2,\ldots,O_T$ .When the initial state is selected as state i the corresponding weight is $-\ln(\pi_i b_i(O_1)\ )$– we shall call this the initial weight. We define the weight of a sequence of states as the sum of the weights on the adjacent states. Note that this actually corresponds to multiplying the corresponding probabilities. Now finding the optimum sequence is merely a matter of finding the path (i.e. a sequence of states) of minimum weight through which the given observation sequence occurs.

Note that the Viterbi Algorithm is essentially a *dynamic programming* approach for minimizing U($i_1,i_2,\ldots,i_T$).

We shall illustrate with an example on how the minimization is done. Consider a three state HMM. Fig.1(a) shows the weights assigned to various paths as described above. The numbers in circles show the initial weights assigned to the corresponding states. To simplify understanding of the algorithm we have assumed that the weights are symmetrical (i.e. the weight from state i to state j is same as the weight from state j to state i). Also we have assumed the weights do not change with time which in general will not be true but then once this simplified case is understood the extension to practical case is straight forward.

Consider Fig. 1(b). We take an observation sequence $O_1,\ldots,O_4$ of length four. At time t=2 state 1 can be visited from any of the three states that we had at time t=1. We find out the weights on each of these paths and add corresponding weights to the initial weights (we shall call this cumulative weight at state 1 as the cost at state 1 at time t=2). Thus going from state 1 to state 1 the cost at state 1 is $3+6=9$. Similarly the cost at state 1 in going from state 2 to state 1

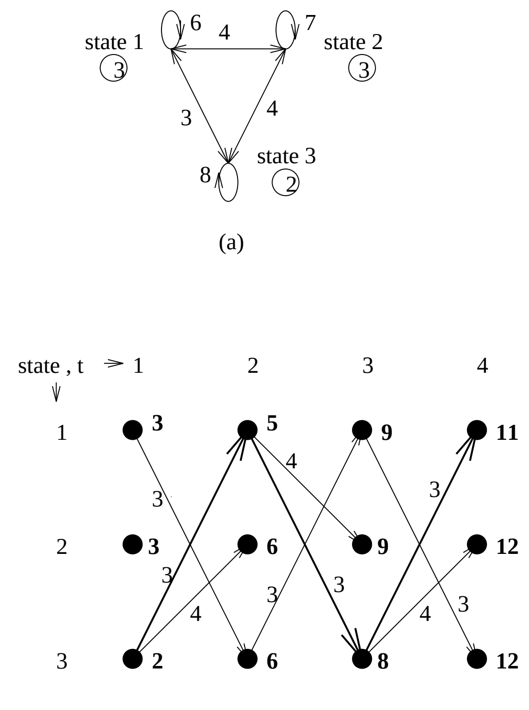

Figure 1: (a) The state machine. The weights for transitions between the states have been shown. The circled numbers are the initial weights of the corresponding states. (b) How a path of minimum cost is traced out using the Viterbi algorithm. The cumulative weights are shown in bold and the final minimum cost path is also shown in bold.

is $3+4=7$ and the cost in going from state 3 to state 1 is $2+3=5$. Of these the cost 5 is minimum. Hence we retain this cost at state 1 for further calculations. The minimum cumulative weight paths are shown in the figure by arrowed lines. The cumulative weights have been shown in bold alongside the respective states at each time instant. We repeat the same procedure for state 2 and state 3. We see that the minimum costs at state 2 and state 3 are 6 (through state 3) and 6 (through state 1) respectively.

We repeat the above procedure again for t=3 but now using the costs calculated above for each state rather than the initial weights (we used them above because we started with t=1). And the same procedure is repeated for t=4. Now select out the state which has minimum cost of all the states. We see that state 1 is the required state with a cost of 11. Back tracing the sequence of states through which we got at state 1 at time t=4 gives the required sequence of states through which the given observation sequence has highest probability of occurrence. As can be seen from the figure this state sequence is state 3,state 1,state 3,state 1 . This sequence has been shown in bold in the figure.

To prove our point suppose you were given that the length of observation sequence is required to be two and that the last state is to be state 3. What path would you choose to minimize the cost (i.e. to maximize the probability of the observation sequence O_1O_2) ? The procedure outlined above clearly shows that we would choose the path beginning at state 1 and ending at state 3 (at t=2) as that gave the minimum cost (viz. 6) at state 3. All other paths have a higher cost. Similar argument applies if any other state were required to be the last state. Similarly if the observation sequence were to be of length three and ending in say state 1 we would choose the path state 1,state 3,state 1 as outlined in the figure and described above. This means that at any given instant the path tracked upto any state by the above procedure is the minimum cost path if we were to stop at that instant at that state. Proceeding to t=4 we see that we have the minimum cost paths corresponding to stopping in state 1,state 2 or state 3 respectively. We just pick up the smallest of these three because we are interested in the minimum cost and hence we choose to stop in state 1 which gives us the minimum cost.

Now that we know how the Viterbi Algorithm works here is how it can be implemented [1, 3] :

Note : $\delta_t(i)$ denotes the weight accumulated when we are in state i at time t as the algorithm proceeds and $\psi_t(j)$ represents the state at time $t-1$ which has the lowest cost corresponding to the state transition to state j at time t. For example in Fig. 1(b),

$$\delta_3(1) = 9,\ \delta_3(2) = 9,\ \delta_3(3) = 8 \tag{16}$$

and

$$\psi_4(1) = 3,\ \psi_4(2) = 3,\ \psi_4(3) = 1 \tag{17}$$

1. Initialization
   For $1 \leq i \leq N$

$$\delta_1(i) = -\ln(\pi_i) - \ln(b_i(O_1)) \tag{18}$$
$$\psi_1(i) = 0 \tag{19}$$

2. Recursive computation
   For $2 \leq t \leq T$ for $1 \leq j \leq N$

$$\delta_t(j) = \min_{1\leq i\leq N}[\delta_{t-1}(i) - \ln(a_{ij})] - \ln(b_j(O_t)) \tag{20}$$
$$\psi_t(j) = \arg\min_{1\leq i\leq N}[\delta_{t-1}(i) - \ln(a_{ij})] \tag{21}$$

3. Termination

$$P^* = \min_{1\leq i\leq N}[\delta_T(i)] \tag{22}$$
$$q_T^* = \arg\min_{1\leq i\leq N}[\delta_T(i)] \tag{23}$$

4. Tracing back the optimal state sequence
   For $t = T-1, T-2, \ldots, 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{24}$$

Hence $\exp(-P^*)$ gives the required state-optimized probability , and $Q^* = \{q_1^*, q_2^*, \ldots, q_T^*\}$ is the optimal state sequence.

A little reflection over the above steps will show that computationally the Viterbi Algorithm is similar to the forward (-backward) procedure except for the comparisons involved for finding the maximum value. Hence its complexity is also of the order of N^2T. This solves Problem 2.

3.3 Problem 3

This problem deals with training the HMM such that it encodes the observation sequence in such a way that if a observation sequence having many characteristics similar to the given one be encountered later it should be able to identify it. There are two methods that we shall discuss depending on which probability is chosen for identification:

The Segmental K-means Algorithm : In this method the parameters of the model $\lambda = (A,B,\pi)$ are adjusted to maximize P(O,I| λ) where I here is the optimum sequence as given by the solution to Problem 2 above.

The Baum-Welch re-estimation Formulas : Here parameters of the model $\lambda = (A,B,\pi)$ are adjusted so as to increase P(O| λ) until a maximum value is reached. As seen before calculating P(O| λ) involves summing up P(O,I| λ) over all possible state sequences I. Hence here our focus is not on a particular state sequence.

3.3.1 The Segmental K-means Algorithm ([3], [4])

This method takes us from λ^k to λ^{k+1} such that $P(O,I_k^* \mid \lambda^k) \leq P(O,I_{k+1}^* \mid \lambda^{k+1})$ where, I_k^* is the optimum sequence for $O = O_1,O_2,\ldots,O_T$ and λ^k, found according to the solution to Problem 2 . This criterion of optimization is called the **maximum state optimized likelihood criterion.** This function $P(O,I^* \mid \lambda) = \max_I P(O,I \mid \lambda)$ is called the **state optimized likelihood function.** In this method to train the model a number of (training) observation sequences are required. Let there be ω number of such sequences available. Each sequence $O = O_1,O_2,\ldots,O_T$ consists of T observation symbols. Instead of ω number of such sequences we might be given just one very long sequence; in that case we segment it into some conveniently chosen ω number of short sequences each of length T. Each observation symbol (O_i) is assumed[2] to be a vector of dimension D (≥ 1). The algorithm then consists of the following steps:

1. Randomly choose N observation symbols (vectors of dimension D) and assign each of the ωT observation vectors to one of these N vectors from which its Euclidean distance is minimum. Hence we have formed N clusters each called a state (1 to N). This initial choice of clustering vectors doesn't decide the final HMM that we get but can decide the number of iterations required for the HMM training. In our case we just pick up N equally spaced sequences of feature vectors and pick one vector from each of these sequences. The first vector is taken as the first vector of the first of these sequences, the second as the second of the second of these sequences and so on. Of course this is just to make the initial choice of clusters as widely distributed as possible and one is at liberty to choose the vectors as per his need.

2. Calculate the initial probabilities and the transition probabilities : For $1 \leq i \leq N$:

$$\hat{\pi}_i = \frac{\text{Number of occurrences of } \{O_1 \in \; i\}}{\text{Total number of occurrences of } O_1 \text{ (i.e. } \omega)} \tag{25}$$

For $1 \leq i \leq N$ and $1 \leq j \leq N$:

$$\hat{a}_{ij} = \frac{\text{Number of occurrences of } \{O_t \in \; i \text{ and} O_{t+1} \in \; j\} \text{ for all } t}{\text{Number of occurrences of } \{O_t \in \; i\} \text{ for all t}} \tag{26}$$

[2]If the observation symbols aren't actually given as numbers or vectors we shall have to characterize each of them with a vector

3. Calculate the mean vector and the covariance matrix for each state : For $1 \leq i \leq N$

$$\hat{\mu}_i = \frac{1}{N_i} \sum_{O_t \in i} O_t \tag{27}$$

$$\hat{V}_i = \frac{1}{N_i} \sum_{O_t \in i} (O_t - \hat{\mu}_i)^T (O_t - \hat{\mu}_i) \tag{28}$$

4. Calculate the symbol probability distributions for each training vector for each state as (we assume Gaussian distribution – there is no particular reason for this ,just change the formulas below for the particular probability distribution that sui ts your problem): For $1 \leq i \leq N$

$$\hat{b}_i(O_t) = \frac{1}{(2\pi)^{D/2} |\hat{V}_i|^{1/2}} \exp[-\frac{1}{2}(O_t - \hat{\mu}_i)\hat{V}_i^{-1}(O_t - \hat{\mu}_i)^T] \tag{29}$$

5. Find the optimal state sequence I^* (as given by the solution to Problem 2) for each training sequence using $\hat{\lambda}_i = (\hat{A}_i, \hat{B}_i, \hat{\pi}_i)$ computed in steps 2 to 4 above. A vector is reassigned a state if its original assignment is different from the corresponding estimated optimum state; for example suppose O_2 of the 5th training sequence was assigned to state 3 (i.e. the 3rd cluster) and now we find that the in the optimal state sequence I^* corresponding to the 5th training sequence, ${i_2}^*$ is not 3 but say 4. Hence now we reassign O_2 of the 5th training sequence to state 4. Hence we assign O_t (of say kth training sequence) to state i if ${i_t}^*$ (corresponding to the kth training sequence) is state i and this is done for all training sequences (i.e. for k=1 to ω).

6. If any vector is reassigned a new state in Step 5, use the new assignment and repeat Step 2 through Step 6; otherwise, stop.

It can be shown [4] that the segmental K-means algorithm converges to the state-optimized likelihood function for a wide range of observation density functions including the Gaussian density we have assumed. Please refer [4] for the proof.

3.3.2 The Baum-Welch re-estimation Formulas [1] :

This method assumes an initial HMM model which is improved upon by using the formulas (given below) so as to maximize P($O|\ \lambda$) . An initial HMM can be constructed in any way but we may use the first five steps of the Segmental K-means Algorithm described above to give us a reasonable initial estimate of the HMM. Henceforth we shall assume that an initial HMM is known. This algorithm maximizes P($O|\ \lambda$) by adjusting the parameters of λ. This optimization criterion is called the **maximum likelihood criterion**. The function P($O|\ \lambda$) is called the **likelihood function**. Before we get down to the actual formulas we shall introduce some concepts and notations that shall be required in the final formulas. Consider $\gamma_t(i)$ defined as follows:

$$\gamma_t(i) = P(i_t = i | O, \lambda)$$

i.e. the probability of being in state i at time t given the observation sequence $O = O_1, O_2, \ldots, O_T$ and the model $\lambda = (A, B, \pi)$. By Bayes law we have :

$$\gamma_t(i) = \frac{P(i_t = i, O|\lambda)}{\mathrm{P}(O|\ \lambda)} \tag{30}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\mathrm{P}(O|\ \lambda)} \tag{31}$$

where $\alpha_t(i)$ and $\beta_t(i)$ are defined by (5) and (10) respectively. $\alpha_t(i)$ accounts for $O_1,\ldots,O_t$ and state i at time t, and $\beta_t(i)$ accounts for $O_{t+1},\ldots,O_T$ given state i at time t .

We also define $\xi_t(i,j)$ as :

$$\xi_t(i,j) = P(i_t = i, i_{t+1} = j \mid O, \lambda)$$

i.e. the probability of being in state i at time t and making a transition to state j at time $t+1$, given the observation sequence $O = O_1,O_2,\ldots,O_T$ and the model $\lambda = (A,B,\pi)$. Using Bayes law it is seen that

$$\xi_t(i,j) = \frac{P(i_t = i, i_{t+1} = j, O \mid \lambda)}{\mathrm{P}(O|\ \lambda)} \tag{32}$$

But $O = O_1,O_2,\ldots,O_T$. Hence

$$\begin{aligned} \text{The Numerator} &= P(i_t = i,\ O_1,\ldots,O_t\ ,\ O_{t+1},\ldots,O_T\ , i_{t+1} = j \mid \lambda) && (33)\\ &= P(i_t = i,\ O_1,\ldots,O_t\ \mid \lambda) P(\ O_{t+1},\ldots,O_T\ , i_{t+1} = j \mid \lambda) && (34)\\ & \quad \text{(since the Markov chain is causal)} && (35)\end{aligned}$$

Consider the second term. The observation symbol at time $t+1$ is O_{t+1} at which time we are required to be in state j; this means that the observation symbol O_{t+1} has to be picked up from state j.Hence we have

$$\begin{aligned} P(\ O_{t+1},\ldots,O_T\ , i_{t+1} = j \mid \lambda) &= P(i_{t+1} = j, O_{t+1} \mid \lambda) P(\ O_{t+1},\ldots,O_T\ \mid i_{t+1} = j, \lambda) && (36)\\ &= \mathrm{a}_{ij} b_j(O_{t+1})\ \beta_{t+1}(j) && (37)\\ & \quad \text{(since } i_t\text{=i is known from (34))} && (38)\end{aligned}$$

Hence combining (5),(32),(34) and (37) we get

$$\xi_t(i,j) = \frac{\alpha_t(i)\mathrm{a}_{ij} b_j(O_{t+1})\ \beta_{t+1}(j)}{\mathrm{P}(O|\ \lambda)} \tag{39}$$

Here $\alpha_t(i)$ accounts for $O_1,\ldots,O_t$,a_{ij} accounts for the transition to state j, $b_j(O_{t+1})$ picks up the symbol O_{t+1} from state j ,and $\beta_{t+1}(j)$ accounts for the remaining observation sequence $O_{t+2},\ldots,O_T$.

If we sum up $\gamma_t(i)$ from $t = 1$ to T we get a quantity which can be viewed as the expected number of times state i is visited or if we sum up only up to $T-1$ then we shall get the expected number of transitions out of state i (as no transition is made at $t = T$). Similarly if $\xi_t(i,j)$ be summed up from $t = 1$ to $T-1$, we shall get the expected number of transitions from state i to state j. Hence

$$\begin{aligned} \sum_{t=1}^{T-1} \gamma_t(i) &= \text{Expected number of transitions from state i}\\ \sum_{t=1}^{T-1} \xi_t(i,j) &= \text{Expected number of transitions from state i to state j}\end{aligned}$$

Now we are prepared to present the Baum-Welch re-estimation formulas:

$$\begin{aligned} \hat{\pi}_i &= \gamma_t(i), \quad 1 \le i \le N && (40)\\ \hat{\mathrm{a}}_{ij} &= \sum_{t=1}^{T-1} \xi_t(i,j) \Bigg/ \sum_{t=1}^{T-1} \gamma_t(i) && (41)\\ \hat{b}_j(k) &= \sum_{\substack{t=1\\ O_t=k}}^{T} \gamma_t(j) \Bigg/ \sum_{t=1}^{T} \gamma_t(j) && (42)\end{aligned}$$

The re-estimation formula for π_i is simply the probability of being in state i at time t. The formula for a$_{ij}$ is the ratio of expected number of time of making a transition from state i to state j to the expected number of times of making a transition out of state i. The formula for $b_k(i)$s the ratio of the expected number of times of being in state j and observing symbol O_k to the expected number of times of being in state j . Note that the summation in the last formula goes up to $t = T$.

If we denote the initial model by λ and the re-estimation model by $\hat{\lambda}$ consisting of the parameters estimated above, then it can be shown that either:

1. The initial model λ is a critical point of the likelihood function in which case $\hat{\lambda} = \lambda$, or

2. $P(O \mid \hat{\lambda}) >$ P($O|$ λ), i.e. we have found a better model from which the observation sequence $O = O_1$,O_2,...,O_T is more likely to be produced.

Hence we can go on iteratively computing until P($O|$ λ) is maximized. This solves Problem 3.

We have seen that the Baum-Welch algorithm maximizes P($O|$ λ) ; given by (4). Each term in the summand of this equation is between 0 and 1 and hence the summand (which is a product of many such terms) is going to be very small and there are as many such small terms as the number of possible state sequences – for a computer to be able to compute such terms individually, it should be able to store within its precision the smallest of these terms. If T=100 there would be around 200 numbers (each between 0 and 1) to be multiplied and hence each term could comfortably go beyond the precision of any computer. Taking logarithm does not help here since P($O|$ λ) is given by a *summation* of such small terms. Perhaps some clever scaling procedure could be used to get over this problem.

On the other hand the Segmental k-means algorithm maximizes P(O,$I|$ λ) which is calculated using the Viterbi Algorithm as given by (18)–(24). Here we can use the logarithm to take care of product of many small numbers as no summation is involved. The Segmental k-means algorithm is also preferred because most of the time we are interested in the occurrence of the observation sequence from the best possible (i.e. the optimum) state sequence and considering the occurrence of the observation sequence through all the possible state sequences is not the intended representation of the problem in most cases. Moreover the Segmental k-means algorithm requires much less computation as compared to the Baum-Welch method. For all these reasons we have used the Segmental k-means algorithm in our work.

4 Distance Between HMMs

If we want to compare two HMMs then we need a measure for distance between two HMMs. Such a measure has been developed by Juang and Rabiner [5]. Their development is based on the Kullback-Leibler distance between two probability distribution function; thus before presenting the measure for distance between tow HMMs we present a very brief introduction to Kullback-Leibler distance measure.

4.1 Kullback-Leibler Distance Measure

Let $p_1(x)$ and $p_2(x)$ be two probability density functions (or probability mass functions) then, the Kullback-Leibler distance measure can be used to judge how close the two probability distributions are.

Definition The Kullback-Leibler distance measure for determining how close the probability density function $p_2(x)$ is to $p_1(x)$ with respect to $p_1(x)$ is

$$I(p_1,p_2) = \int_{-\infty}^{\infty} p_1(x)\ln(\frac{p_1(x)}{p_2(x)})\,dx$$

In case $p_1(x)$ and $p_2(x)$ are probability mass functions then

$$I(p_1,p_2) = \sum_{\text{all } x} p_1(x)\ln(\frac{p_1(x)}{p_2(x)})$$

Note that the Kullback-Leibler distance measure is not symmetric, $I(p_1,p_2) \neq I(p_2,p_1)$. If our objective is to simply compare p_1 and p_2 we can define a symmetric distance measure as

$$I_s(p_1,p_2) = \frac{1}{2}[I(p_1,p_2) + I(p_2,p_1)]$$

A Typical Approximation Problem: Given the density functions $p(x)$, we would like to approximate it with another density function $\hat{p}(x)$ (where $\hat{p}(x)$ could be a parameterized function). Then for obtaining $\hat{p}(x)$, we consider the problem

$$\hat{p}^* = \arg\min_{\hat{p}}[I(p,\hat{p})]$$

Using $I(p,\hat{p}) = \int_x p(x)\ln p(x) - \int_x p(x)\ln\hat{p}(x)$ the above minimization is equivalent to

$$\hat{p}^* = \arg\max_{\hat{p}} \int_x p(x)\ln\hat{p}(x)$$

Moreover $I(p,\hat{p})$ can also be interpreted as

$$I(p,\hat{p}) = E_{p(x)}[\ln p(x)] - E_{p(x)}[\ln\hat{p}(x)]$$

where $E_{p(x)}$ is the expectation operator with respect to the density function $p(x)$.

4.2 Distance between HMMs

It is the use of the Kullback-Leibler distance measure which leads to the definition of the distance measure between two HMMs.

Let λ_1 and λ_2 be two HMMs. For example λ_1 could be the model HMM and λ_2 could be the HMM based on the observations. We are interested in comparing how close they are.

$$I(\lambda_1,\lambda_2) = \sum_{\mathcal{O}} P[\mathcal{O}|\lambda_1]\ln(\frac{P[\mathcal{O}|\lambda_1]}{P[\mathcal{O}|\lambda_2]})$$

In practice we cannot compute th above sum, because it would be impossible to generate all possible sequences. Suppose we have generated an observation sequence of length T using the HMM λ_i; denote this sequence by $\mathcal{O}_T^i$. Now, corresponding to this sequence we can talk about computing $P[\mathcal{O}_T^1|\lambda_2]$. For this computation we will have to assume some state sequence, which at least is not available for HMM λ_2. It is likely that we may have a state sequence for λ_1, since the observation sequence $\mathcal{O}_T^1$ was generated using HMM λ_1. Nevertheless, we do not use this information. What one does is to compute the optimal state sequence using the Viterbi algorithm for λ_1 and λ_2 corresponding to $\mathcal{O}_T^1$. Let the optimal state sequences be $I^{1*}(\mathcal{O}_T^1)$ and $I^{2*}(\mathcal{O}_T^1)$. Now, for closeness between λ_1 and λ_2, we consider

$$P[\mathcal{O}_T^1, I^{1*}(\mathcal{O}_T^1)|\lambda_1]\{\ln(P[\mathcal{O}_T^1, I^{1*}(\mathcal{O}_T^1)|\lambda_1]) - \ln(P[\mathcal{O}_T^1, I^{2*}(\mathcal{O}_T^1)|\lambda_2])\}$$

We would like to compute the above for a large value of T, the length of the observation sequence.

The probability function outside the braces can be ignored since it is common to both the terms inside the braces. The distance between two HMMs is defined as

$$D(\lambda_1, \lambda_2) \overset{\text{def}}{=} \lim_{T\to\infty} \frac{1}{T}|\ln(P[\mathcal{O}_T^1, I^{1*}(\mathcal{O}_T^1)|\lambda_1]) - \ln(P[\mathcal{O}_T^1, I^{2*}(\mathcal{O}_T^1)|\lambda_2])|$$

Once again we see that $D(\lambda_1, \lambda_2)$ is not symmetric. Nevertheless a symmetric distance between HMMs is defined as

$$D_s(\lambda_1, \lambda_2) \overset{\text{def}}{=} \frac{1}{2}[D(\lambda_1, \lambda_2) + D(\lambda_2, \lambda_1)]$$

What we have given above is a heuristic derivation of the formula for distance between two HMMs. The formal derivation is given in Juang and Rabiner [5].

4.3 How Large Should T be?

In practice one cannot have $T \to \infty$. To get over this problems we did some experimentation as suggested by [5] to find out how large a value of T should be chosen. The details of how the experiment was performed are as follows:

Let two HMMs λ_1 and λ_2, the distance between which is to be found, be given. We generate feature sequences from the HMM λ_1.

Let S be the given covariance matrix. Hence S is symmetric and semi-positive definite. Since we generate a large number of feature vectors, the covariance matrix in almost all cases is also non-singular and hence can be assumed to be positive definite. Using *Cholesky decomposition*, S can be written as

$$S = LL^T \tag{43}$$

where L is a lower triangular matrix.

Now suppose we want to generate n zero mean vectors each of dimension p with covariance matrix as S. Generate an $n \times p$ matrix A whose elements are Gaussian random variables with zero mean and standard deviation of 1 i.e. N(0,1). Then the rows of

$$B = AL \tag{44}$$

give the required n zero mean vectors with covariance matrix S. Note that L is a $p \times p$ matrix. Now simply add the required mean to all the generated vectors.

Next, we calculate the probability of the observation sequence generated from HMM λ_1 against itself and also against the HMM λ_2 and compute the asymmetric distance $D(\lambda_1, \lambda_2)$. Similarly we compute the probability of the observation sequence generated from λ_2 against itself and λ_1 and again compute the corresponding asymmetric distance $D(\lambda_2, \lambda_1)$. Taking the average of these two distances gives the required distance.

The result of such an experimentation using two HMMs λ_1 and λ_2 and a seed is shown in Fig.2.

From experimentation with different HMMs it was found that as T increases the distance approaches a limit. Also the distance has reasonably settled after $T = 2^{12}$ i.e. around 4000.

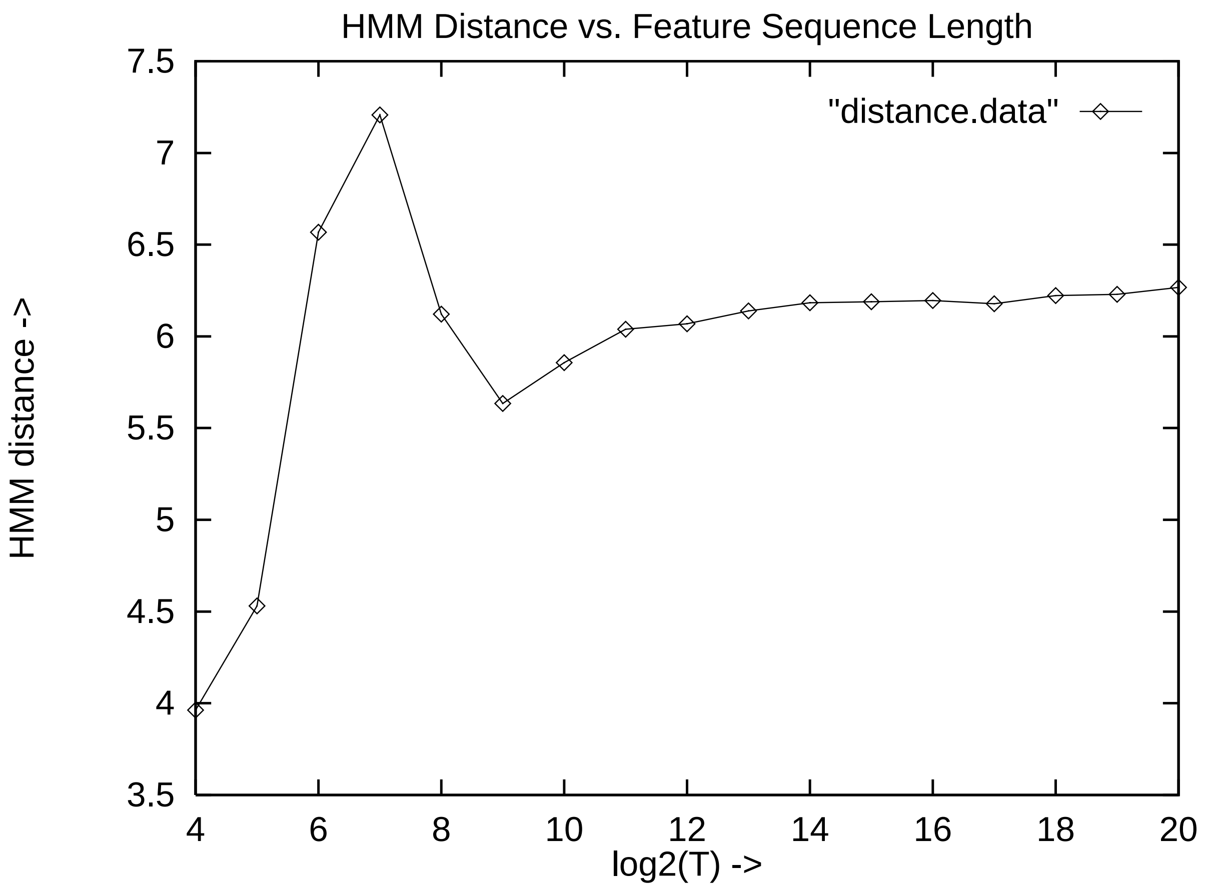

Figure 2: Convergence of the distance between two HMMs as the length of the observation sequence is increased

5 Conclusion

It is hoped that this would serve as good introduction to HMMs for anyone to venture further into the realm of HMMs.

References

[1] L. R. Rabiner and B. H. Juang, “An introduction to hidden Markov models,” *IEEE ASSP Mag.*, pp 4–16, Jun. 1986.

[2] G.D.Forney Jr., “The Viterbi Algorithm,” *Proc. IEEE*, vol. 61, no. 3, pp. 263–278, Mar. 1973.

[3] Yang He and Amlan Kundu, “2-D shape classification using HMMs,” *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 13, no. 11, pp. 1172–1184, Nov. 1991.

[4] B. H. Juang and L. R. Rabiner, “The segmental k-means algorithm for estimating the parameters of hidden Markov models,” *IEEE Trans. Accoust., Speech, Signal Processing*, vol. ASSP-38,no. 9, pp. 1639–1641, Sept. 1990.

[5] B. H. Juang and L. R. Rabiner, “A Probabilistic Distance measure between HMMs”, Vol. 64, no. 2, pp. 391-408, Feb. 1985.