

An Introduction to Hidden Markov Models

L. R. Rabiner
B. H. Juang

The basic theory of Markov chains has been known to mathematicians and engineers for close to 80 years, but it is only in the past decade that it has been applied explicitly to problems in speech processing. One of the major reasons why speech models, based on Markov chains, have not been developed until recently was the lack of a method for optimizing the parameters of the Markov model to match observed signal patterns. Such a method was proposed in the late 1960's and was immediately applied to speech processing in several research institutions. Continued refinements in the theory and implementation of Markov modelling techniques have greatly enhanced the method, leading to a wide range of applications of these models. It is the purpose of this tutorial paper to give an introduction to the theory of Markov models, and to illustrate how they have been applied to problems in speech recognition.

INTRODUCTION

ASSUME YOU ARE GIVEN the following problem. A real world process produces a sequence of observable symbols. The symbols could be discrete (outcomes of coin tossing experiments, characters from a finite alphabet, quantized vectors from a codebook, etc.) or continuous (speech samples, autocorrelation vectors, vectors of linear prediction coefficients, etc.). Your job is to build a signal model that explains and characterizes the occurrence of the observed symbols. If such a signal model is obtainable, it then can be used later to identify or recognize other sequences of observations.

In attacking such a problem, some fundamental decisions, guided by signal and system theory, must be made. For example, one must decide on the form of the model, linear or non-linear, time-varying or time-invariant, deterministic or stochastic. Depending on these decisions, as well as other signal processing considerations, several possible signal models can be constructed.

To fix ideas, consider modelling a pure sinewave. If we have reason to believe that the observed symbols are from a pure sinewave, then all that would need to be measured is the amplitude, frequency and perhaps phase of the sinewave and an exact model, which explains the observed symbols, would result.

Consider next a somewhat more complicated signal—namely a sinewave imbedded in noise. The noise components of the signal make the modelling problem more complicated because in order to properly estimate the sinewave parameters (amplitude, frequency, phase) one has to take into account the characteristics of the noise component.

In the above examples, we have assumed the sinewave part of the signal was stationary—i.e. not time varying. This may not be a realistic assumption. If, for example, the unknown process produces a sinewave with varying amplitude, then clearly a non-linear model, e.g. amplitude-modulation, may be more appropriate. Similarly, if we assume that the frequency, instead of the amplitude, of the sinewave is changing, a frequency-modulation model might be most appropriate.

Linear system models

The concepts behind the above examples have been well studied in classical communication theory. The variety and types of real world processes, however, does not stop here. Linear system models, which model the observed symbols as the output of a linear system excited by an appropriate source, form another important class of processes for signal modeling and have proven useful for a wide variety of applications. For example, "short time" segments of speech signals can be effectively modeled as the output of an all-pole filter excited by appropriate sources with essentially a flat spectral envelope. The signal modeling technique, in this case, thus involves determination of the linear filter coefficients and, in some cases, the excitation parameters. Obviously, spectral analyses of other kinds also fall within this category.

One can further incorporate temporal variations of the signal into the linear system model by allowing the filter coefficients, or the excitation parameters, to change with time. In fact, many real world processes cannot be meaningfully modeled without considering such temporal variation. Speech signals are one example of such processes. There are several ways to address the problem of modeling temporal variation of a signal.

As mentioned above, within a "short time" period, some physical signals, such as speech, can be effectively modeled by a simple linear time-invariant system with the

appropriate excitation. The easiest way then to address the time-varying nature of the process is to view it as a direct concatenation of these smaller "short time" segments, each such segment being individually represented by a linear system model. In other words, the overall model is a synchronous sequence of symbols where each of the symbols is a linear system model representing a short segment of the process. In a sense this type of approach models the observed signal using representative tokens of the signal itself (or some suitably averaged set of such signals if we have multiple observations).

Time-varying processes

Modeling time-varying processes with the above approach assumes that every such short-time segment of observation is a unit with a prechosen duration. In general, however, there doesn't exist a precise procedure to decide what the unit duration should be so that both the time-invariant assumption holds, and the short-time linear system models (as well as concatenation of the models) are meaningful. In most physical systems, the duration of a short-time segment is determined empirically. In many processes, of course, one would neither expect the properties of the process to change synchronously with every unit analysis duration, nor observe drastic changes from each unit to the next except at certain instances. Making no further assumptions about the relationship between adjacent short-time models, and treating temporal variations, small or large, as "typical" phenomena in the observed signal, are key features in the above direct concatenation technique. This template approach to signal modeling has proven to be quite useful and has been the basis of a wide variety of speech recognition systems.

There are good reasons to suspect, at this point, that the above approach, while useful, may not be the most efficient (in terms of computation, storage, parameters etc.) technique as far as representation is concerned. Many real world processes seem to manifest a rather sequentially changing behavior; the properties of the process are usually held pretty steadily, except for minor fluctuations, for a certain period of time (or a number of the above-mentioned duration units), and then, at certain instances, change (gradually or rapidly) to another set of properties. The opportunity for more efficient modeling can be exploited if we can first identify these periods of rather steadily behavior, and then are willing to assume that the temporal variations within each of these steady periods are, in a sense, statistical. A more efficient representation may then be obtained by using a common short time model for each of the steady, or well-behaved parts of the signal, along with some characterization of how one such period evolves to the next. This is how hidden Markov models (HMM) come about. Clearly, three problems have to be addressed: 1) how these steadily or distinctively behaving periods can be identified, 2) how the "sequentially" evolving nature of these periods can be characterized, and 3) what typical or common short time model should be chosen for each of these periods. Hid-

den Markov models successfully treat these problems under a *probabilistic* or *statistical* framework.

It is thus the purpose of this paper to explain what a hidden Markov model is, why it is appropriate for certain types of problems, and how it can be used in practice. In the next section, we illustrate hidden Markov models via some simple coin toss examples and outline the three fundamental problems associated with the modeling technique. We then discuss how these problems can be solved in Section III. We will not direct our general discussion to any one particular problem, but at the end of this paper we illustrate how HMM's are used via a couple of examples in speech recognition.

DEFINITION OF A HIDDEN MARKOV MODEL

An HMM is a doubly stochastic process with an underlying stochastic process that is *not* observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. We illustrate HMM's with the following coin toss example.

Coin toss example

To understand the concept of the HMM, consider the following simplified example. You are in a room with a barrier (e.g., a curtain) through which you cannot see what is happening. On the other side of the barrier is another person who is performing a coin (or multiple coin) tossing experiment. The other person will not tell you anything about what he is doing exactly; he will only tell you the result of each coin flip. Thus a sequence of *hidden* coin tossing experiments is performed, and you only observe the results of the coin tosses, i.e.

$$O = \mathcal{H} \mathcal{H} \mathcal{T} \mathcal{H} \mathcal{T} \mathcal{H} \dots \mathcal{T}$$

$$O_1 O_2 O_3 \dots O_T$$

where \mathcal{H} stands for heads and \mathcal{T} stands for tails.

Given the above experiment, the problem is how do we build an HMM to explain the observed sequence of heads and tails. One possible model is shown in Fig. 1a. We call this the "1-fair coin" model. There are two states in the model, but each state is uniquely associated with either heads (state 1) or tails (state 2). Hence this model is *not hidden* because the observation sequence uniquely defines the state. The model represents a "fair coin" because the probability of generating a head (or a tail) following a head (or a tail) is 0.5; hence there is no bias on the current observation. This is a degenerate example and shows how independent trials, like tossing of a fair coin, can be interpreted as a set of sequential events. Of course, if the person behind the barrier is, in fact, tossing a single fair coin, this model should explain the outcomes very well.

A second possible HMM for explaining the observed sequence of coin toss outcomes is given in Fig. 1b. We call this model the "2-fair coin" model. There are again 2 states in the model, but neither state is uniquely associated with

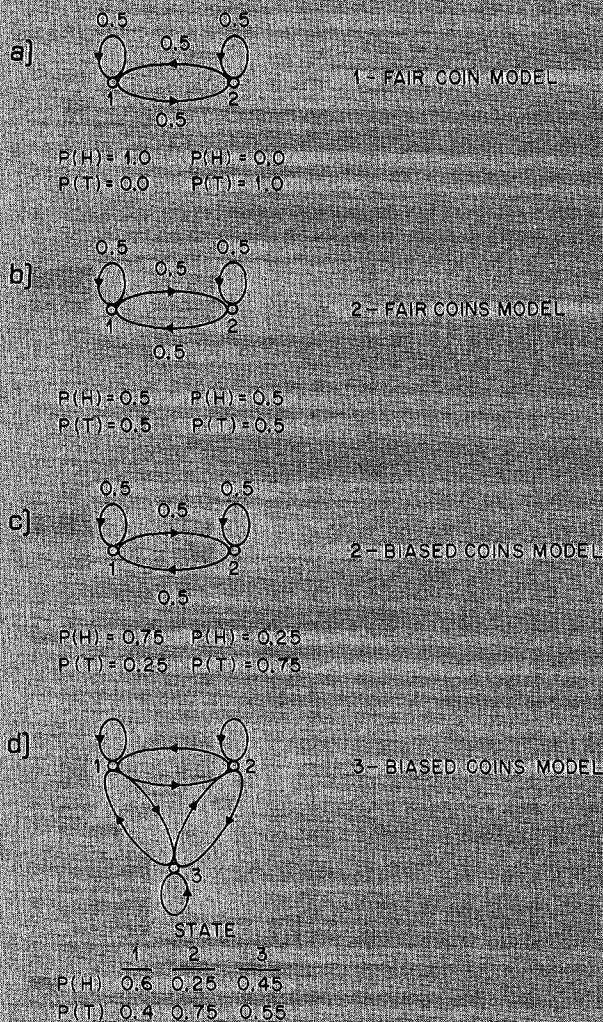


Figure 1. Models which can be used to explain the results of hidden coin tossing experiments. The simplest model, shown in part (a), consists of a single fair coin with the outcome heads corresponding to one state and tails to the other state. The model of part (b) corresponds to tossing two fair (unbiased) coins, with the first coin being used in state 1 and the second coin being used in state 2. An independent "fair" coin is used to decide which of the other two fair coins is flipped at each trial. The model of part (c) corresponds to tossing two biased coins, with the first coin being heavily biased towards heads, and the second coin heavily biased towards tails. Again a "fair" coin is used to decide which biased coin is tossed at each trial. Finally the model of part d corresponds to the case of 3 biased coins being used.

either heads or tails. The probabilities of heads (or tails) in either state is 0.5. Also the probability of leaving (or remaining in) either state is 0.5. Thus, in this case, we can associate each state with a fair (unbiased) coin. Although the probabilities associated with remaining in, or leaving, either of the two states are all 0.5, a little thought should convince the reader that the statistics of the observable output sequences of the 2-fair coins model are independent of the state transitions. The reason for this is that this

model is hidden (i.e. we cannot know exactly which fair coin (state) led to the observed heads or tails at each observation), but is essentially indistinguishable (in a statistical sense) from the 1-fair coin model of Fig. 1a.

Figures 1c and 1d show two more possible HMM's which can account for the observed sequence of heads and tails. The model of Fig. 1c, which we call the 2-biased coins model, has two states (corresponding to two different coins). In state 1, the coin is biased strongly towards heads. In state 2, the coin is biased strongly towards tails. The state transition probabilities are all equal to 0.5. This 2-biased coins model is a hidden Markov model which is distinguishable from the two previously discussed models. Interestingly, the reader should be able to convince himself that the long time statistics (e.g. average number of heads or tails) of the observation sequences from the HMM of Fig. 1c are the same as those from the models of Figs. 1a and 1b. This model is very appropriate if what is happening behind the barrier is as follows. The person has three coins, one fair and the other two biased according to the description in Fig. 1c. The two biased coins are associated with the two faces of the fair coin respectively. To report the outcome of every mysterious coin flip, the person behind the barrier first flips the fair coin to decide which biased coin to use, and then flips the chosen biased coin to obtain the result. With this model, we thus are able to look into and explain the above subtle characteristic changes (i.e. switching the biased coins).

The model of Fig. 1d, which we call the 3-biased coins model, has three states (corresponding to three different coins). In state 1 the coin is biased slightly towards heads; in state 2 the coin is biased strongly toward tails; in state 3 the coin is biased slightly toward tails. We have not specified values of the state transition probabilities in Fig. 1d; clearly the behavior of the observation sequences produced by such a model are strongly dependent on these transition probabilities. (To convince himself of this, the reader should consider two extreme cases, namely when the probability of remaining in state 3 is large (>0.95), or small (<0.05). Very different sequence statistics will result from these two extremes because of the strong bias of the coin associated with state 3). As with the 2-biased coin model, some real scenario behind the barrier, corresponding to such a model can be composed; the reader should find no difficulty doing this himself.

There are several important points to be learned from this discussion of how to model the outputs of the coin tossing experiment via HMM's. First we note that one of the most difficult parts of the modeling procedure is to decide on the size (the number of states) of the model. Without some a priori information, this decision often is difficult to make and could involve trial and error before settling on the most appropriate model size. Although we stopped at a 3-coin model for the above illustration, even this might be too small. How do we decide on how many coins (states) are really needed in the model? The answer to this question is related to an even larger question, namely how do we choose model parameters (state transi-

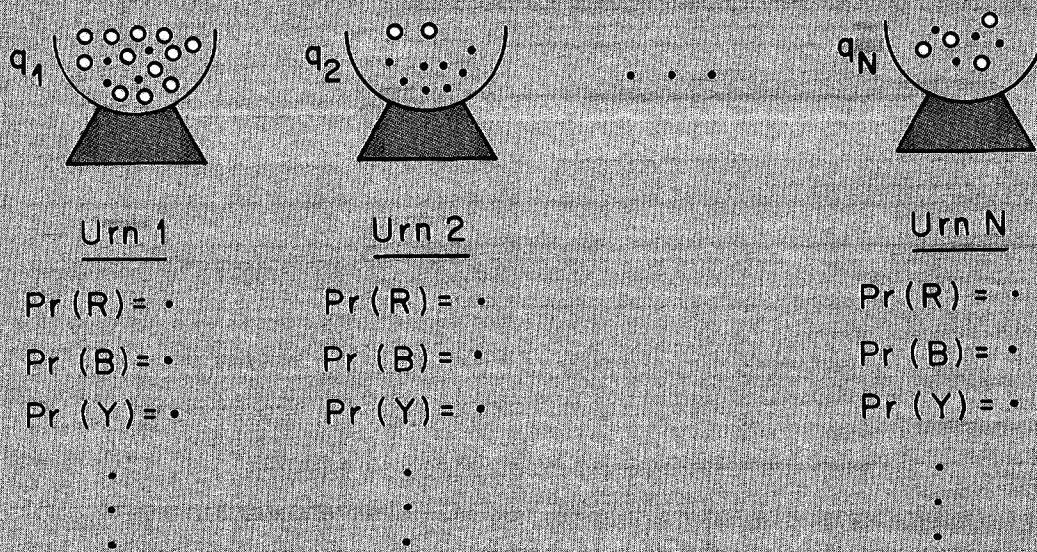


Figure 2. An urn and ball model which illustrates the general case of a discrete symbol hidden Markov model. Each of N urns (the N states of the model) contains a large number of colored balls. The proportion of each colored ball in each urn is different, and is governed by the probability density of colors for each urn. The observations from the urn and ball model consists of announcing the color of the ball drawn at random from a selected urn, replacing the ball, and then choosing a new urn from which to select a ball according to the state transition density associated with the originally selected urn.

tion probabilities, probabilities of heads and tails in each state) to optimize the model so that it best explains the observed outcome sequence. We will try to answer this question in the section on Solutions to the Three HMM Problems as this is the key to the successful use of HMM's for real world problems. A final point concerns the size of the observation sequence. If we are restricted to a small finite observation sequence we may not be able to reliably estimate the optimal model parameters. (Think of the case of actually using 10 coins but be given a set of 50-100 observations). Hence, in a sense, depending on the amount of model training data we are given, certain HMM's may not be statistically, reliably different.

Elements of an HMM

We now explain the elements and the mechanism of the type of HMM's that we discuss in this paper:

1. There are a *finite number*, say N , of states in the model; we shall not rigorously define what a state is but simply say that within a state the signal possesses some measurable, distinctive properties.
2. At each clock time, t , a new state is entered based upon a *transition probability distribution which depends on the previous state* (the Markovian property). (Note that the transition may be such that the process remains in the previous state.)
3. After each transition is made, an observation output symbol is produced according to a probability distribution which depends on the current state. This probability distribution is held fixed for the state regardless of when and how the state is entered. There are thus N such *observation probability distributions* which, of course, represent random variables or stochastic processes.

To fix ideas, let us consider the "urn and ball" model of Fig. 2. There are N urns, each filled with a large number of colored balls. There are M possible colors for each ball. The observation sequence is generated by initially choosing one of the N urns (according to an initial probability distribution), selecting a ball from the initial urn, recording its color, replacing the ball, and then choosing a new urn according to a transition probability distribution associated with the current urn. Thus a typical observation sequence might be:

clock time	1 2 3 4 \cdots T
urn (hidden) state	$q_3 q_1 q_1 q_2 \cdots q_{N-2}$
color (observation)	R B Y Y \cdots R

We now formally define the following model notation for a discrete observation HMM:

- T = length of the observation sequence (total number of clock times)
- N = number of states (urns) in the model
- M = number of observation symbols (colors)
- $Q = \{q_1, q_2, \dots, q_N\}$, states (urns)
- $V = \{v_1, v_2, \dots, v_M\}$ discrete set of possible symbol observations (colors)
- $A = \{a_{ij}\}$, $a_{ij} = \Pr(q_j \text{ at } t + 1 | q_i \text{ at } t)$, state transition probability distribution
- $B = \{b_j(k)\}$, $b_j(k) = \Pr(v_k \text{ at } t | q_j \text{ at } t)$, observation symbol probability distribution in state j
- $\pi = \{\pi_i\}$, $\pi_i = \Pr(q_i \text{ at } t = 1)$, initial state distribution

Using the model, an observation sequence, $O = O_1 O_2, \dots, O_T$, is generated as follows:

1. Choose an initial state, i_1 , according to the initial state distribution, π ;
2. Set $t = 1$;
3. Choose O_t according to $b_{i_t}(k)$, the symbol probability distribution in state i_t ;
4. Choose i_{t+1} according to $\{a_{i_t i_{t+1}}\}$, $i_{t+1} = 1, 2, \dots, N$, the state transition probability distribution for state i_t ;
5. Set $t = t + 1$; return to step 3 if $t < T$; otherwise terminate the procedure.

We use the compact notation $\lambda = (A, B, \pi)$ to represent an HMM. Specification of an HMM involves choice of the number of states, N , and the number of discrete symbols M , (we will briefly discuss continuous density HMM's at the end of this paper), and specification of the three probability densities A , B , and π . If we try to specify the relative importance of the three densities, A , B , and π , then it should be clear that for most applications π is the least important (this represents initial conditions), and B is the most important (since it is directly related to the observed symbols). For some problems the distribution A is also quite important (recall the 3-biased coins models discussed earlier), whereas for other problems (e.g. isolated word recognition problems) it is of less importance.

The three problems for HMM's

Given the form of the HMM discussed in the previous section, there are three key problems of interest that must be solved for the model to be useful in real world applications. These problems are the following:

- Problem 1 — Given the observation sequence $O = O_1, O_2, \dots, O_T$, and the model $\lambda = (A, B, \pi)$, how we compute $\Pr(O|\lambda)$, the probability of the observation sequence.
- Problem 2 — Given the observation sequence $O = O_1, O_2, \dots, O_T$, how we choose a state sequence $I = i_1, i_2, \dots, i_T$ which is optimal in some meaningful sense.
- Problem 3 — How we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $\Pr(O|\lambda)$.

Problem 1 is the evaluation problem: given a model and a sequence of observations, how we can compute the probability that the observed sequence was produced by the model. We can also view the problem as: given a model and a sequence of observations, how we "score" or evaluate the model. The latter viewpoint is very useful. If we think of the case in which we have several competing models (e.g. the four models of Fig. 1 for the coin tossing experiment), the solution to problem 1 allows us to choose the model which best matches the observations.

Problem 2 is the one in which we attempt to uncover the hidden part of the model, i.e. the state sequence. This is a typical estimation problem. We usually use an optimality criterion to solve this problem as best as possible. Unfortunately, as we will see, there are several possible optimality criteria that can be imposed and hence the choice of criterion is a strong function of the intended use

for the uncovered state sequence. A typical use of the recovered state sequence is to learn about the structure of the model, and to get average statistics, behavior, etc. within individual states.

Problem 3 is the one in which we attempt to optimize the model parameters so as to best describe how the observed sequence comes about. We call this a training sequence in this case since it is used to train the model. The training problem is the crucial one for most applications of HMM's since it allows us to optimally adapt model parameters to observed training data—i.e. to create best models for real phenomena.

To fix ideas, consider the following speech recognition scheme. We want to design an N -state HMM for each word of a V -word vocabulary. Using vector quantization (VQ) techniques, we represent the speech signal by a sequence of VQ codebook symbols derived from an M -word codebook. Thus we start with a training sequence, for each vocabulary word, consisting of a number of repetitions of the spoken word (by one or more talkers). We use the solution to *Problem 3* to optimally get model parameters for each word model. To develop an understanding of the physical meaning of the model states, we use the solution to *Problem 2* to segment each of the word training sequences into states, and then study the observations occurring in each state. The result of this study may lead to further improvements on the model. We shall discuss this in later sections. Finally to do recognition on an unknown word, we use the solution to *Problem 1* to score each word model based upon the given test observation sequence, and select the word whose word model score is the highest.

We now present the formal mathematical solutions to each of the three fundamental problems for HMM's. And, as we shall see, these three problems may be linked together under our probabilistic framework.

SOLUTIONS TO THE THREE HMM PROBLEMS

Problem 1

We wish to calculate the probability of the observation sequence O , given the model λ . The most straightforward way of doing this is through enumerating every possible state sequence of length T (the number of observations). For every fixed state sequence $I = i_1 i_2 \dots i_T$, the probability of the observation sequence O is $\Pr(O|I, \lambda)$, where

$$\Pr(O|I, \lambda) = b_{i_1}(O_1)b_{i_2}(O_2) \dots b_{i_T}(O_T).$$

The probability of such a state sequence I , on the other hand, is

$$\Pr(I|\lambda) = \pi_{i_1} a_{i_1 i_2} a_{i_2 i_3} \dots a_{i_{T-1} i_T}.$$

The joint probability of O and I , i.e., the probability that O and I occur simultaneously, is simply the product of the above two terms, $\Pr(O, I|\lambda) = \Pr(O|I, \lambda) \Pr(I|\lambda)$. The probability of O then is obtained by summing this joint probability over all possible state sequences:

The forward-backward procedure

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

$$\alpha_i(i) = \Pr(O_1, O_2, \dots, O_i, i_t = q_i | \lambda)$$

i.e. the probability of the partial observation sequence (until time t) and state q_i at time t , given the model λ . We can solve for $\alpha_i(i)$ inductively, as follows:

1. $\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N;$
2. for $t = 1, 2, \dots, T-1, \quad 1 \leq j \leq N$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1});$$

3. then, $\Pr(O | \lambda) = \sum_{i=1}^N \alpha_T(i).$

Step 1 initiates the forward probabilities with the joint probability of state q_i and initial observation O_1 . Step 2 is illustrated in Fig. 3a which shows how the state q_j is reached at time $t+1$ from the N possible states $q_i, i = 1, 2, \dots, N$, at time t . Since $\alpha_t(i)$ is the probability of the joint event that O_1, O_2, \dots, O_t are observed and the state stops at q_i at time t , the product $\alpha_t(i) a_{ij}$ is then the probability of the joint event that O_1, O_2, \dots, O_t are observed and state q_j is reached at time $t+1$ via state q_i at time t . Summing this product over all the N possible states $q_i, i = 1, 2, \dots, N$, at time t results in the probability of q_j at time $t+1$ with all the accompanying previous partial observations. Once this is done and q_j is known, it is easy to see that $\alpha_{t+1}(j)$ is obtained by augmenting multiplicatively the summed quantity with the probability $b_j(O_{t+1})$. Finally step 3 gives the desired calculation of $\Pr(O | \lambda)$ as the sum of the terminal forward variables $\alpha_T(i)$. This is so because $\alpha_T(i) = \Pr(O_1, O_2, \dots, O_T, i_T = q_i | \lambda)$.

If we examine the computation involved in the calculation of $\alpha_t(j), 1 \leq t \leq T, 1 \leq j \leq N$, we see that it requires on the order of $N^2 T$ calculations, rather than $2TN^T$ as required by the direct calculation. (Again to be precise, we need $N(N+1)(T-1) + N$ multiplications and $N(N-1)(T-1)$ additions.) For $N = 5, T = 100$, we

need about 3000 computations for the forward method versus 10^{72} for the direct calculation, a savings of about 69 orders of magnitude!

This forward-backward procedure is in effect based upon the lattice structure shown in Fig. 3b. The key is that since there are only N states (nodes), all the possible state sequences will re-merge into these N nodes no matter how long the observation sequence. At time $t = 1$, we need to calculate values of $\alpha_1(i), 1 \leq i \leq N$. At times $t = 2, 3, \dots, T$ we only need to calculate values of $\alpha_t(j), 1 \leq j \leq N$, where each calculation involves N previous values of $\alpha_{t-1}(i)$ because each of the N grid points is reached from the same N grid points at the previous moment.

In a similar manner we can consider a backward variable, $\beta_t(i)$ defined as:

$$\beta_t(i) = \Pr(O_{t-1}, O_{t-2}, \dots, O_T | i_t = q_i, \lambda)$$

i.e. the probability of the partial observation sequence from $t+1$ to the end, given state q_i at time t and the model λ . Again we can solve for $\beta_t(i)$ inductively, as follows:

1. $\beta_T(i) = 1, \quad 1 \leq i \leq N;$
2. for $t = T-1, T-2, \dots, 1, \quad 1 \leq j \leq N$

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}(j));$$

Step 1 arbitrarily defines $\beta_T(i)$ to be 1 for all i . Step 2, which is illustrated in Figure 4, shows that in order to have been in state q_i at time t , and to account for the rest of the observation sequence, you had to make a transition to every one of the N possible states at time $t+1$, account for the observation symbol O_{t+1} in that state, and then account for the rest of the observation sequence.

Again the computation of $\beta_t(i)$ for $1 \leq t \leq T, 1 \leq i \leq N$, requires on the order of $N^2 T$ calculations, and can be computed in a lattice structure similar to that of Fig. 3b.

Box 1

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

The interpretation of the computation in the above equation is the following. Initially (at time $t = 1$) we are in state i_1 with probability π_{i_1} , and generate the symbol O_1 with probability $b_{i_1}(O_1)$. We then make a transition to state i_2 with probability $a_{i_1 i_2}$, and generate symbol O_2 with probability $b_{i_2}(O_2)$. This process continues until we make the last transition from state i_{T-1} to state i_T with probability $a_{i_{T-1} i_T}$

and generate symbol O_T with probability $b_{i_T}(O_T)$.

A little thought should convince the reader that the calculation of $\Pr(O | \lambda)$, according to its direct definition, involves on the order of $2T \cdot i$ calculations, since at every time $t = 1, 2, \dots, T$, there are N possible states to go through and for each summand about $2T$ calculations are required. (To be precise, we need $(2T-1)N^T$ multiplications and $N^T - 1$ additions.) This calculation is computationally unfeasible, even for small values of N and T ; e.g. for $N = 5, T = 100$, there are on the order of $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ computations! Clearly a more efficient procedure is required to solve problem 1. Such a procedure exists and is sometimes called the forward-backward procedure. (See Box 1)

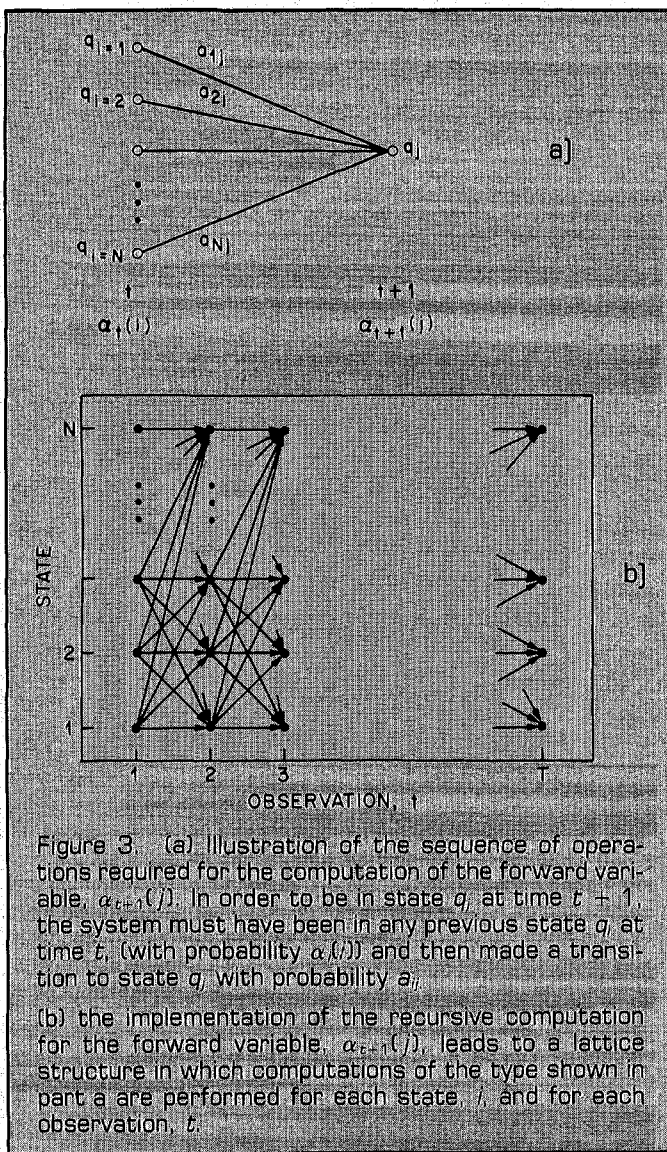


Figure 3. (a) Illustration of the sequence of operations required for the computation of the forward variable, $\alpha_{t+1}(j)$. In order to be in state q_j at time $t + 1$, the system must have been in any previous state q_i at time t , (with probability $\alpha_t(i)$) and then made a transition to state q_j with probability a_{ij} . (b) the implementation of the recursive computation for the forward variable, $\alpha_{t+1}(j)$, leads to a lattice structure in which computations of the type shown in part a are performed for each state, i , and for each observation, t .

Problem 2

There are several possible ways of solving Problem 2, namely finding the optimal state sequence associated with the given observation sequence, since there are several possible optimality criteria. One possible optimality criterion is to choose the states, i_t , which are individually most likely. This maximizes the expected number of correct individual states. To implement this solution we define the variable

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

i.e. the probability of being in state q_i at time t , given the observation sequence O and the model λ . A little thought should convince the reader that $\gamma_t(i)$ is trivially expressed in terms of the α 's and β 's as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\Pr(O | \lambda)}$$

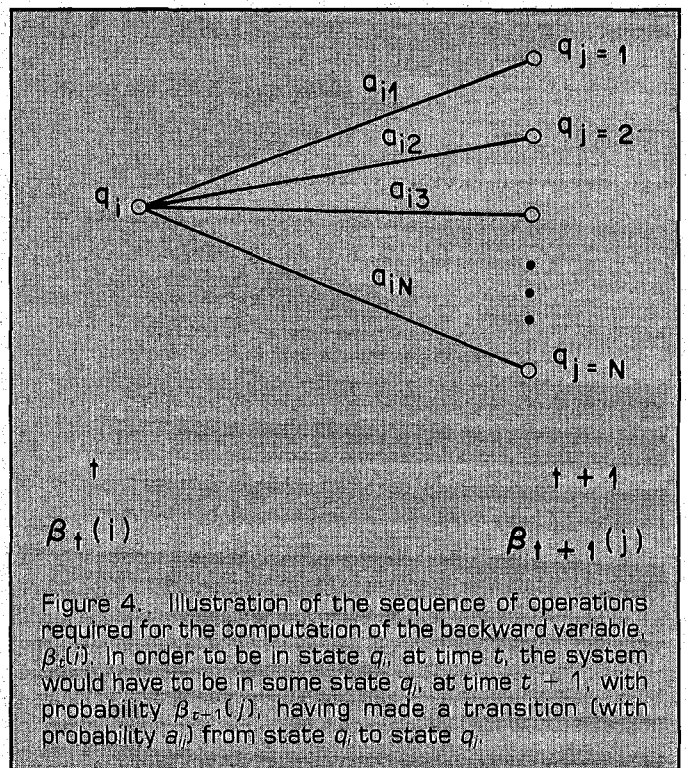


Figure 4. Illustration of the sequence of operations required for the computation of the backward variable, $\beta_t(i)$. In order to be in state q_i at time t , the system would have to be in some state q_j at time $t + 1$, with probability $\beta_{t+1}(j)$, having made a transition (with probability a_{ij}) from state q_j to state q_i .

since $\alpha_t(i)$ accounts for O_1, O_2, \dots, O_t and state q_i at t , and $\beta_t(i)$ accounts for O_{t+1}, \dots, O_T given state q_i at t . The normalization factor, $\Pr(O | \lambda)$, makes $\gamma_t(i)$ a conditional probability, so that $\sum_{i=1}^N \gamma_t(i) = 1$.

Using $\gamma_t(i)$, the individually most likely state, i_t , at time t is

$$i_t = \operatorname{argmax}_{1 \leq i \leq N} [\gamma_t(i)] \quad 1 \leq t \leq T$$

There might be some problems with the above criterion and solution, however. When there are disallowed transitions—i.e. $a_{ij} = 0$ for some i and j , the obtained state sequence may in fact be an impossible state sequence. The solution simply determines the most likely state at every instance without regard to the global trellis structure, the neighboring (in time) state, and the length of the observation sequence. It is still useful, though, as in practice such undesirable situations do not usually occur, and this instantaneous optimality provides insights for a theoretically tractable parameter smoothing scheme.

The drawback of the above approach points to the necessity of global constraints, of some type, on the derived optimal state sequence. Almost trivially, an optimality criterion of this type is to find the single best path (state sequence) with the highest probability, i.e. to maximize $\Pr(O, I | \lambda)$. A formal technique for finding this single best state sequence exists and is called the Viterbi algorithm. (See Box 2)

Problem 3

The third problem is to adjust the model parameters (A, B, π) to maximize the probability of the observation

sequence given the model. This is the most difficult of the three problems we have discussed. There is no known way to solve for a maximum likelihood model analytically. Therefore an iterative procedure, such as the Baum-Welch method, or gradient techniques for optimization must be used. Here we will only discuss the iterative procedure. It appears that with this procedure, the physical meaning of various parameter estimates can be easily visualized.

To describe how we (re)estimate HMM parameters, we first define $\xi_t(i, j)$ as

$$\xi_t(i, j) = \Pr(i_t = q_i, i_{t+1} = q_j | O, \lambda)$$

i.e. the probability of a path being in state q_i at time t and making a transition to state q_j at time $t + 1$, given the observation sequence and the model. From Fig. 5 it should be clear that we can write $\xi_t(i, j)$ as

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\Pr(O | \lambda)}$$

In the above, $\alpha_t(i)$ accounts for the first t observations, ending in state q_i at time t , the term $a_{ij} b_j(O_{t+1})$ accounts for the transition to state q_j at time $t + 1$ with the occurrence of symbol O_{t+1} , and the term $\beta_{t+1}(j)$ accounts for

Box 2

Viterbi algorithm

The formal steps in the Viterbi algorithm for finding the single best state sequence are as follows:

Step 1 — Initialization

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

$$\Psi_1(j) = 0$$

Step 2 — Recursion

$$\text{For } 2 \leq t \leq T, \quad 1 \leq j \leq N$$

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t)$$

$$\Psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}]$$

Step 3 — Termination

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

$$i_T^* = \operatorname{argmax}_{1 \leq i \leq N} [\delta_T(i)]$$

Step 4 — Path (state sequence) backtracking

$$\text{For } t = T - 1, T - 2, \dots, 1$$

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

The Viterbi algorithm is similar (without the backtracking steps) in implementation to the forward-backward calculation; however, a maximization over previous states is used in place of the summing procedure used previously. Again a trellis structure efficiently implements the computation.

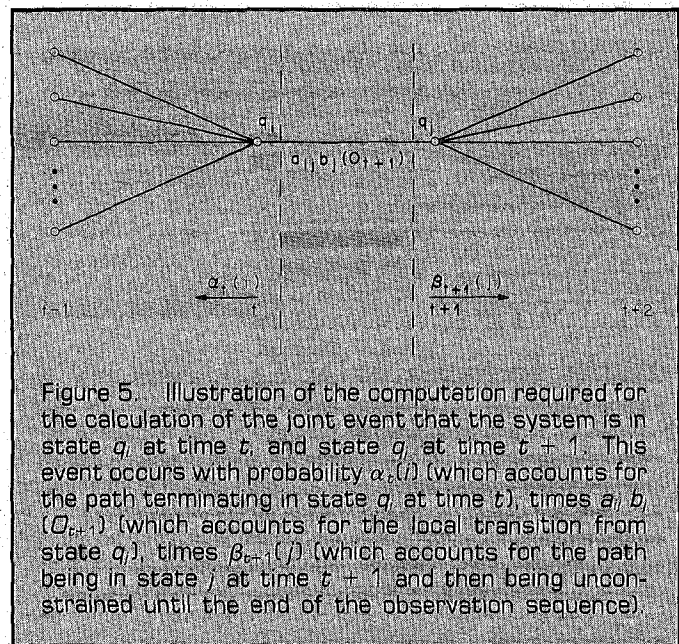


Figure 5. Illustration of the computation required for the calculation of the joint event that the system is in state q_i at time t , and state q_j at time $t + 1$. This event occurs with probability $\alpha_t(i)$ (which accounts for the path terminating in state q_i at time t), times $a_{ij} b_j(O_{t+1})$ (which accounts for the local transition from state q_i to state q_j at time $t + 1$ with the occurrence of symbol O_{t+1}), times $\beta_{t+1}(j)$ (which accounts for the path being in state j at time $t + 1$ and then being unconstrained until the end of the observation sequence).

Box 3

The Baum-Welch reestimation formulas

The reestimation formulas for π , A , and B are:

$$1. \bar{\pi}_i = \gamma_1(i), \quad 1 \leq i \leq N$$

$$2. \bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$3. \bar{b}_j(k) = \frac{\sum_{t=1}^T \gamma_t(j) \delta_{jk}}{\sum_{t=1}^T \gamma_t(j)}$$

The reestimation formula for π_i is trivially the probability of being in state q_i at $t = 1$. The reestimation formula for a_{ij} is the ratio of the expected number of transitions from state q_i to q_j , divided by the expected number of transitions out of state q_i . Finally the reestimation formula for $b_j(k)$ is the ratio of the expected number of times of being in state j and observing symbol k divided by the expected number of times of being in state j . Note that the summation for $b_j(k)$ is from $t = 1$ to $t = T$.

If we define the initial model as λ and the reestimation model as $\bar{\lambda}$, consisting of the above $\bar{\pi}_i$, \bar{a}_{ij} , and $\bar{b}_j(k)$, then it can be proven that either:

1. The initial model λ defines a critical point of the likelihood function, in which case $\bar{\lambda} = \lambda$, or
2. Model $\bar{\lambda}$ is more likely in the sense that $\Pr(O | \bar{\lambda}) > \Pr(O | \lambda)$, i.e. we have found another model $\bar{\lambda}$, from which the observation sequence is more likely to be produced.

Therefore, if we iteratively use $\bar{\lambda}$ in place of λ and repeat the above reestimation calculation, we then can improve the probability of O being observed from the model until some limiting point is reached. The result is the estimated model.

the remainder of the observation sequence. The normalization factor $\text{Pr}(O|\lambda)$ provides the proper normalization for $\xi_t(i, j)$.

Recall that we have previously defined $\gamma_t(i)$ as the probability of being in state q_i at time t , given the observation sequence and the model; hence we can relate $\gamma_t(i)$ to $\xi_t(i, j)$ by summing $\xi_t(i, j)$ over j , giving

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

If we sum $\gamma_t(i)$ over the time index t , we get a quantity which can be interpreted as the expected (over time) number of times that state q_i is visited, or equivalently, the expected number of transitions made from state q_i , if we exclude the last moment, T , in the summation. Similarly, summation of $\xi_t(i, j)$ over t (from $t = 1$ to $t = T - 1$) can be regarded as the expected number of transitions from state q_i to state q_j . That is

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{Expected number of transitions made from } q_i$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{Expected number of transitions from state } q_i \text{ to state } q_j$$

Using the above formulas (and the concept of counting event occurrences) we can use the Baum-Welch method to reestimate values of the HMM parameters. (See Box 3)

Summary of results

We have shown how to define an HMM, how to score it on an observation sequence (Problem 1), how to make a best guess as to the hidden state sequence (Problem 2), and how to optimize model parameters to best match a given training sequence (Problem 3). In the next section we discuss some properties of the models, issues involved in practical implementation, and some special cases of the B parameters. Finally, in Section V, we illustrate the application of HMM's to a simple speech recognition system to show one possible way of applying the concepts discussed here.

ISSUES WITH HMM'S

In this section we discuss several issues related to types of HMM's, issues in implementation, and extensions of the basic model to more advanced forms. We will not be rigorous here, but will only give indications of the kinds of problems people have been concerned with. More detail on the mathematical aspects are given in the references.

Types of HMM's

The general HMM we have been dealing with until now is assumed to have essentially a full state transition matrix, i.e. transitions can be made from any state in some way to any other state. Such models are often ergodic in the sense that any state will be revisited with probability one and that such revisits are not required to take place at periodic intervals of time. We show an example of one

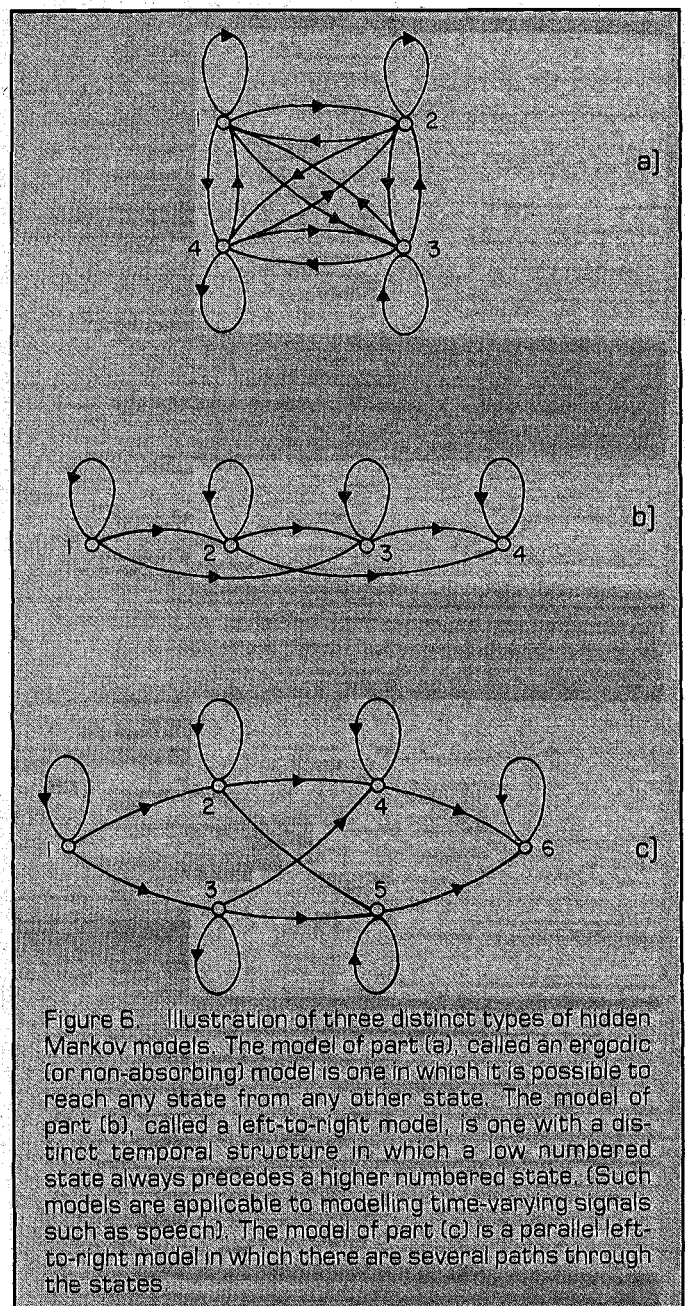


Figure 6. Illustration of three distinct types of hidden Markov models. The model of part (a), called an ergodic (or non-absorbing) model is one in which it is possible to reach any state from any other state. The model of part (b), called a left-to-right model, is one with a distinct temporal structure in which a low numbered state always precedes a higher numbered state. (Such models are applicable to modelling time-varying signals such as speech). The model of part (c) is a parallel left-to-right model in which there are several paths through the states.

such model in Fig. 6a. (Here $N = 4$ states). For some applications we are interested in non-ergodic models where we impose constraints on the state transition matrix. For example, Figs. 6b and 6c show two examples of non-ergodic HMM's. For these cases the state transition matrix is upper triangular (i.e. transitions can only be made to a state whose index is as large or larger than the index of the current state). Such models have been called left-to-right models since the state sequence which produced the observation sequence must always proceed from the leftmost state to the rightmost state. Such left-to-right models inherently impose a temporal order to the HMM since lower numbered states account for observations occurring prior to those for higher numbered states. We shall see how we use this feature to our advantage in our discussion of how we apply HMM's to speech recognition.

Implementation issues

In the section on Solutions to the Three HMM Problems, we outlined several simple and straightforward procedures for working with HMM's. For the most part the procedures work exactly as discussed. However there is at least one computational issue of significance, and a couple of practical aspects that must be kept in mind, for the procedures to be maximally useful.

The computational issue concerns the implementation of the forward-backward computation. A quick glance will convince the reader that both $\alpha_t(i)$ and $\beta_t(i)$ tend to zero geometrically fast (recall that all probabilities are less than 1.0). Hence a scaling technique of the α 's and β 's is required to avoid mathematical underflow. The details of such scaling procedures are beyond the scope of this paper.

A second issue concerns the use of a finite set of training data for estimating the HMM parameters. If we look at the reestimation formulas we see that a parameter will be set to 0 if there are no occurrences in the training set—i.e. if a symbol does not occur often enough in the observation sequence, then the probability for that symbol will be 0 in some states. If this effect is due to the small size of the training observation sequence, then special effort must be made to insure that no HMM parameter becomes too small. If it is a real effect, then a zero probability parameter is perfectly reasonable. In any case care must be taken to guarantee (perhaps via constraints on the parameter space) that the estimated HMM parameters are reasonable.

Finally we point out that all the formulas presented in this paper for a single observation sequence can be modified to handle the case of multiple observation sequences. Hence one could do training of an HMM from a long single sequence, or from a set of multiple observation sequences (particularly useful for non-ergodic models).

Special cases of the B parameters

Until now we have only considered the case of discrete symbol HMM's, i.e. where the observation sequence was one of a set of M discrete symbols. The model can readily be extended to the case where the observations are continuous symbols, or more generally, continuous vectors, \mathbf{x} . For such a model the $b_j(k)$ probability density is replaced by the continuous density, $b_j(\mathbf{x})$, $1 \leq j \leq N$, where

$b_j(\mathbf{x})d\mathbf{x}$ = probability that observation vector, O , lies between \mathbf{x} and $\mathbf{x} + d\mathbf{x}$.

There are several special forms for $b_j(\mathbf{x})$ which have been proposed, including:

1. Gaussian M -component mixture densities of the form

$$b_j(\mathbf{x}) = \sum_{k=1}^M c_{jk} \mathcal{N}[\mathbf{x}, \mu_{jk}, \mathbf{U}_{jk}]$$

where c_{jk} is the mixture weight, \mathcal{N} is the normal density and μ_{jk} and \mathbf{U}_{jk} are the mean vector and covariance matrix associated with state j , mixture k .

2. Gaussian autoregressive M -component mixture densities of the form

$$b_j(\mathbf{x}) = \sum_{k=1}^M c_{jk} b_{jk}(\mathbf{x})$$

where

$$b_{jk}(\mathbf{x}) = \frac{e^{-\delta(\mathbf{x}; \mathbf{a}_{jk})/2}}{(2\pi)^{K/2}}$$

$$\delta(\mathbf{x}; \mathbf{a}) = r_a(0)r_x(0) + 2 \sum_{i=1}^p r_a(i)r_x(i)$$

$\delta(\mathbf{x}; \mathbf{a})$ is the standard LPC distance between a vector \mathbf{x} (of dimension K) with autocorrelation r_x , and an LPC vector \mathbf{a} (of dimension p) with autocorrelation r_a .

These alternate density functions have been used to good advantage in several speech recognition systems.

EXAMPLE OF THE USE OF HMM'S—ISOLATED WORD RECOGNITION

Hidden Markov models have been found to be extremely useful for ecology, cryptanalysis, and a wide spectrum of speech applications. Here we consider the case of trying to use HMM's to build an isolated word recognizer. Assume we have a vocabulary of V words to be recognized. We have a training set of L tokens of each word (spoken by 1 or more talkers), and an independent testing set. To do speech recognition we perform the following steps:

1. First we build an HMM for each word in the vocabulary. We use the observations from the set of L tokens to estimate the optimum parameters for each word, giving model λ^v , for the v^{th} vocabulary word, $1 \leq v \leq V$.
2. For each unknown word in the test set, characterized by observation sequence $O = O_1, O_2, \dots, O_T$, and for each word model, λ^v , we calculate $P_v = \Pr(O | \lambda^v)$ according to the procedure of the section on Solution to the Three HMM Problems.
3. We choose the word whose model probability is highest, i.e.

$$v^* = \operatorname{argmax}_{1 \leq v \leq V} [P_v]$$

The HMM based recognizer has been applied to several word recognition tasks using both a discrete symbol observation set (VQ codebook symbols), and at least two continuous observation models. The table below (based on experiments performed at AT&T Bell Laboratories) gives some performance characteristics for a speaker independent system using a vocabulary of 10 digits.

Recognizer	Recognition Accuracy (%)
Template Based Using Dynamic Time Warping	98.2%
HMM using Discrete Symbols	97.1%
HMM using Continuous Densities	98.1%

rational information is often represented in a normalized form for word models, (since the word boundary is essentially known), in the form:

$P_j(I/T)$ = probability of being in state j for exactly (I/T) of the word, where T is the number of frames in the word and I is the number of frames spent in state j .

A typical set of histograms of $P_j(I/T)$ for a 5-state model for the word "six" is shown in Fig. 9. As seen from the figure, the first state is generally very brief; the second and third states have longer duration; the fourth state has a well-defined peak in the density with an average duration of about 20 percent of the word and is never skipped over (i.e. $I/T = 0$); the final state (the stop plus the fricative) covers about 50 percent of the word length and is also always present in the utterances.

It is found that this durational information is rather robust under different channel conditions and is quite useful for word recognition. The main effect appears to be from the resultant constraint that certain states must be present for some minimum duration.

Score evaluation

In the section on Solutions to the Three HMM Problems, we already explained how the forward-backward procedure works in obtaining the quantity $\Pr(O|\lambda)$. This quantity is the summation of $\Pr(O, I|\lambda)$ over all possible state sequences I . Since the Viterbi algorithm efficiently finds the maximum of $\Pr(O, I|\lambda)$ over all I , a question is then: what is the relationship between $\Pr(O|\lambda)$ and $\max_I \Pr(O, I|\lambda)$?

Interestingly enough, for speech signals and with some properly chosen model specifications, the dynamic range

of $\Pr(O, I|\lambda)$ is usually very large and $\max_I \Pr(O, I|\lambda)$ is usually the only significant term in the summation for $\Pr(O|\lambda)$. Therefore, in such cases, either the forward-backward procedure or the Viterbi algorithm works equally well in the word recognition task.

Other considerations

HMM's provide a framework based upon which higher level structures in continuous speech signals may be integrally modelled. Care, however, must be taken in implementing such an extension.

The above left-to-right word models effectively exploit such a priori information as the word boundaries. Direct concatenation of the above word model may or may not be viable for continuous speech recognition, particularly when the vocabulary is large. Constructing a global HMM from small HMM's based upon such units as phonemes, etc. has been and is still being pursued.

Another consideration relates to the robustness of the modeling technique. Different assumptions on the form of observation density, as well as the a priori Markov structure constraints lead to different levels of robustness in performing the recognition task. This robustness issue, of course, is compounded by the various representations of the short-time speech symbols (spectra). Some representations may be better characterized as Gaussian multivariates and some may be less susceptible to channel fluctuations, speaker variations, and noise contamination etc. It is yet unknown what the best combination is.

The above considerations in no way discourage the use of HMM in speech recognition. On the contrary, these are the main directions that research effort is pointing to for solving the ultimate recognition problem with HMM's.

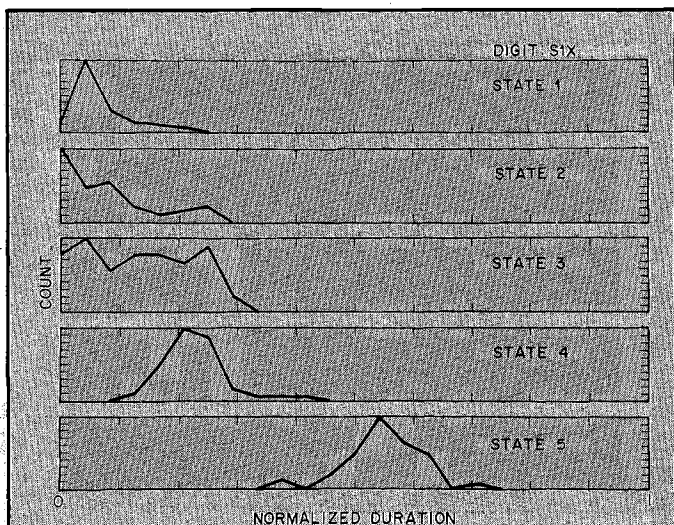


Figure 9. Illustration of the measured normalized duration density histograms for the 5 states of a hidden Markov model for the word /six/. The plots show that state 1 represents a transient state, whereas state 5 (which corresponds to the final fricative /s/), has an average normalized duration of over 0.5.

REFERENCES

- [1] Baker, J. K., "The Dragon System—An Overview," *IEEE Trans. on Acoustics Speech Signal Processing*, Vol. ASSP-23, No. 1, pp. 24-9, February 1975.
- [2] Jelinek, F., "Continuous Speech Recognition by Statistical Methods," *Proc. IEEE*, Vol. 64, pp. 532-556, April 1976.
- [3] Poritz, A. B., "Linear Predictive Hidden Markov Models and the Speech Signal," *Proc. ICASSP '82*, pp. 1291-1294, Paris, France, May 1982.
- [4] Boulard, H., Wellekins, C. J., and Ney, H., "Connected Digit Recognition Using Vector Quantization," *Proc. ICASSP '84*, pp. 26.10.1-26.10.4, San Diego, CA, March 1984.
- [5] Rabiner, L. R., Levinson, S. E., and Sondhi, M. M., "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition," *Bell System Tech. J.*, Vol. 62, No. 4, pp. 1075-1105, April 1983.
- [6] Markel, J. D., and Gray, Jr., A. H., *Linear Prediction of Speech*, Springer-Verlag, New York, 1976.
- [7] Juang, B. H., "On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition—A

Unified View," *AT&T B.L.T.J.*, Vol. 63, No. 7, pp. 1213-1243, September 1984.

- [8] Levinson, S. E., Rabiner, L. R., and Sondhi, M. M., "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *B.S.T.J.*, Vol. 62, No. 4, Part 1, pp. 1035-1074, April 1983.
- [9] Forney, Jr., G. D., "The Viterbi Algorithm," *Proc. IEEE*, Vol. 61, pp. 268-278, March 1978.
- [10] Baum, L. E., and Eagon, J., "An Inequality with Applications to Statistical Prediction for Functions of Markov Processes and to a Model for Ecology," *Bull. Amer. Math. Soc.*, 73 (1963), 360-363.
- [11] Baum, L. E., Petrie, T., Soules, G., and Weiss, N., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Ann. Math. Statistic.*, 41 (1970), pp. 164-71.
- [12] Rabiner, L. R., Juang, B. H., Levinson, S. E., and Sondhi, M. M., "Recognition of Isolated Digits Using Hidden Markov Models with Continuous Mixture Densities," *AT&T B.L.T.J.*, Vol. 64, No. 3, pp. 1211-1234, July-August 1985.
- [13] Juang, B. H., and Rabiner, L. R., "Mixture Autoregressive Hidden Markov Models for Speech Signals," *Transactions on IEEE/Acoustics, Speech, and Signal Processing*, Vol. ASSP-33, No. 6, pp. 1404-1413, Dec. 1985.
- [14] Andelman, D., and Reeds, J., "On the Cryptanalysis of Rotor Machines and Substitution-Permutation Networks," *IEEE Trans. Info. Theory*, Vol. IT-28, No. 4,

pp. 578-584, July 1982.

- [15] Neuberg, E. P., "Markov Models for Phonetic Text," *J. Acoust. Soc. Am.*, Vol. 50, p. 116(A), 1971.

Lawrence R. Rabiner (S'62-M'67-SM'75-F'75) was born in Brooklyn, NY, on Sept. 28, 1943. He received the S.B. and S.M. degrees simultaneously in June 1964, and the Ph.D. degree in electrical engineering in June 1967, all from the Massachusetts Institute of Technology, Cambridge.

From 1962 through 1964 he participated in the cooperative plan in electrical engineering at Bell Laboratories, Whippany, and Murray Hill, NJ. He worked on digital circuitry, military communications problems, and problems in binaural hearing. Presently he is engaged in research on speech recognition and digital signal processing techniques at Bell Laboratories, Murray Hill. He is coauthor of the books "Theory and Application of Digital Signal Processing", (Prentice-Hall, 1975), "Digital Processing of Speech Signals", (Prentice-Hall, 1978) and "Multirate Digital Signal Processing", (Prentice-Hall, 1983).

Dr. Rabiner is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, The National Academy of Engineering, and a Fellow of both the Acoustical Society of America and the IEEE.

Biing-Hwang Juang (S'79-M'81) was born in 1951. He received the B.Sc. degree in electrical engineering from the National Taiwan University in 1973 and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of California, Santa Barbara, in 1979 and 1981, respectively.

In 1978, he joined the Speech Communications Research Laboratory, Santa Barbara, CA, and was involved in research work on vocal tract modeling. In 1979, he became affiliated with Signal Technology, Inc., Santa Barbara, CA, where his research work was in the areas of speech coding and speech interference suppression. Since 1982, he has been with AT&T Bell Laboratories. His current research interests include speech recognition, coding and stochastic processes.

Faculty Positions in Digital Signal Processing

Electrical Engineering Department, Rutgers University, Piscataway, NJ

Applications are solicited for tenure-track faculty positions in the area of Digital Signal Processing. Qualifications should include a relevant Ph.D. in Electrical Engineering, a strong academic record, research credentials in Digital Signal Processing and a commitment to teaching at both the undergraduate and graduate level. While positions are available at the Assistant Professor level, applicants with outstanding qualifications may be considered for appointments at the level of Associate or Full Professor. It is expected that appointments will begin in September 1986. Research areas of particular interest include, but are not limited to: spectral estimation, medium rate speech encoding, image restoration, parallel processing, and real number coding. Successful applicants will be expected to contribute toward the development of courses and laboratory facilities in the area of their specialization.

Resumes should be sent to: Professor Richard Mammone, Rutgers University, College of Engineering, Electrical Engineering Department, P.O. Box 909, Piscataway, N.J. 08854.

Rutgers University is an equal opportunity/affirmative action employer.