A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition

LAWRENCE R. RABINER, FELLOW, IEEE

Although initially introduced and studied in the late 1960s and early 1970s, statistical methods of Markov source or hidden Markov modeling have become increasingly popular in the last several years. There are two strong reasons why this has occurred. First the models are very rich in mathematical structure and hence can form the theoretical basis for use in a wide range of applications. Second the models, when applied properly, work very well in practice for several important applications. In this paper we attempt to carefully and methodically review the theoretical aspects of this type of statistical modeling and show how they have been applied to selected problems in machine recognition of speech.

I. Introduction

Real-world processes generally produce observable outputs which can be characterized as signals. The signals can be discrete in nature (e.g., characters from a finite alphabet, quantized vectors from a codebook, etc.), or continuous in nature (e.g., speech samples, temperature measurements, music, etc.). The signal source can be stationary (i.e., its statistical properties do not vary with time), or nonstationary (i.e., the signal properties vary over time). The signals can be pure (i.e., coming strictly from a single source), or can be corrupted from other signal sources (e.g., noise) or by transmission distortions, reverberation, etc.

A problem of fundamental interest is characterizing such real-world signals in terms of signal models. There are several reasons why one is interested in applying signal models. First of all, a signal model can provide the basis for a theoretical description of a signal processing system which can be used to process the signal so as to provide a desired output. For example if we are interested in enhancing a speech signal corrupted by noise and transmission distortion, we can use the signal model to design a system which will optimally remove the noise and undo the transmission distortion. A second reason why signal models are important is that they are potentially capable of letting us learn a great deal about the signal source (i.e., the real-world process which produced the signal) without having to have the source available. This property is especially important when the cost of getting signals from the actual source is high. In this case, with a good signal model, we can simulate the source and learn as much as possible via simulations. Finally, the most important reason why signal models are important is that they often work extremely well in practice, and enable us to realize important practical systems—e.g., prediction systems, recognition systems, identification systems, etc., in a very efficient manner.

These are several possible choices for what type of signal model is used for characterizing the properties of a given signal. Broadly one can dichotomize the types of signal models into the class of deterministic models, and the class of statistical models. Deterministic models generally exploit some known specific properties of the signal, e.g., that the signal is a sine wave, or a sum of exponentials, etc. In these cases, specification of the signal model is generally straightforward; all that is required is to determine (estimate) values of the parameters of the signal model (e.g., amplitude, frequency, phase of a sine wave, amplitudes and rates of exponentials, etc.). The second broad class of signal models is the set of statistical models in which one tries to characterize only the statistical properties of the signal. Examples of such statistical models include Gaussian processes, Poisson processes, Markov processes, and hidden Markov processes, among others. The underlying assumption of the statistical model is that the signal can be well characterized as a parametric random process, and that the parameters of the stochastic process can be determined (estimated) in a precise, well-defined manner.

For the applications of interest, namely speech processing, both deterministic and stochastic signal models have had good success. In this paper we will concern ourselves strictly with one type of stochastic signal model, namely the hidden Markov model (HMM). (These models are referred to as Markov sources or probabilistic functions of Markov chains in the communications literature.) We will first review the theory of Markov chains and then extend the ideas to the class of hidden Markov models using several simple examples. We will then focus our attention on the three fundamental problems[1] for HMM design, namely: the

Manuscript received January 15, 1988; revised October 4, 1988.
The author is with AT&T Bell Laboratories, Murray Hill, NJ 07974-2070, USA.
IEEE Log Number 8825949.

[1]The idea of characterizing the theoretical aspects of hidden Markov modeling in terms of solving three fundamental problems is due to Jack Ferguson of IDA (Institute for Defense Analysis) who introduced it in lectures and writing.

0018-9219/89/0200-0257$01.00 © 1989 IEEE

Amazon / Zentian Limited
Exhibit 1029
Page 1

evaluation of the probability (or likelihood) of a sequence of observations given a specific HMM; the determination of a best sequence of model states; and the adjustment of model parameters so as to best account for the observed signal. We will show that once these three fundamental problems are solved, we can apply HMMs to selected problems in speech recognition.

Neither the theory of hidden Markov models nor its applications to speech recognition is new. The basic theory was published in a series of classic papers by Baum and his colleagues [1]–[5] in the late 1960s and early 1970s and was implemented for speech processing applications by Baker [6] at CMU, and by Jelinek and his colleagues at IBM [7]–[13] in the 1970s. However, widespread understanding and application of the theory of HMMs to speech processing has occurred only within the past several years. There are several reasons why this has been the case. First, the basic theory of hidden Markov models was published in mathematical journals which were not generally read by engineers working on problems in speech processing. The second reason was that the original applications of the theory to speech processing did not provide sufficient tutorial material for most readers to understand the theory and to be able to apply it to their own research. As a result, several tutorial papers were written which provided a sufficient level of detail for a number of research labs to begin work using HMMs in individual speech processing applications [14]–[19]. This tutorial is intended to provide an overview of the basic theory of HMMs (as originated by Baum and his colleagues), provide practical details on methods of implementation of the theory, and describe a couple of selected applications of the theory to distinct problems in speech recognition. The paper combines results from a number of original sources and hopefully provides a single source for acquiring the background required to pursue further this fascinating area of research.

The organization of this paper is as follows. In Section II we review the theory of discrete Markov chains and show how the concept of hidden states, where the observation is a probabilistic function of the state, can be used effectively. We illustrate the theory with two simple examples, namely coin-tossing, and the classic balls-in-urns system. In Section III we discuss the three fundamental problems of HMMs, and give several practical techniques for solving these problems. In Section IV we discuss the various types of HMMs that have been studied including ergodic as well as left-right models. In this section we also discuss the various model features including the form of the observation density function, the state duration density, and the optimization criterion for choosing optimal HMM parameter values. In Section V we discuss the issues that arise in implementing HMMs including the topics of scaling, initial parameter estimates, model size, model form, missing data, and multiple observation sequences. In Section VI we describe an isolated word speech recognizer, implemented with HMM ideas, and show how it performs as compared to alternative implementations. In Section VII we extend the ideas presented in Section VI to the problem of recognizing a string of spoken words based on concatenating individual HMMs of each word in the vocabulary. In Section VIII we briefly outline how the ideas of HMM have been applied to a large vocabulary speech recognizer, and in Section IX we summarize the ideas discussed throughout the paper.

II. Discrete Markov Processes[2]

Consider a system which may be described at any time as being in one of a set of N distinct states, $S_1, S_2, \cdots, S_N$, as illustrated in Fig. 1 (where $N = 5$ for simplicity). At regularly spaced discrete times, the system undergoes a change of state (possibly back to the same state) according to a set of probabilities associated with the state. We denote the time instants associated with state changes as $t = 1, 2, \cdots$, and we denote the actual state at time t as q_t. A full probabilistic description of the above system would, in general, require specification of the current state (at time t), as well as all the predecessor states. For the special case of a discrete, first order, Markov chain, this probabilistic description is truncated to just the current and the predecessor state, i.e.,

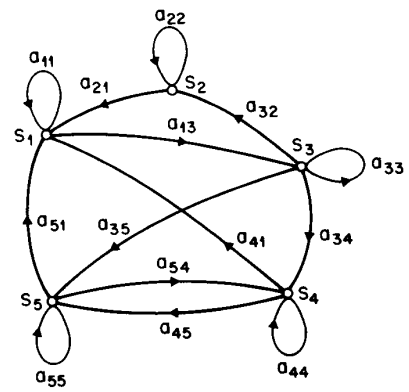

Fig. 1. A Markov chain with 5 states (labeled S_1 to S_5) with selected state transitions.

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \cdots] = P[q_t = S_j | q_{t-1} = S_i]. \quad (1)$$

Furthermore we only consider those processes in which the right-hand side of (1) is independent of time, thereby leading to the set of state transition probabilities a_{ij} of the form

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i], \quad 1 \le i, j \le N \quad (2)$$

with the state transition coefficients having the properties

$$a_{ij} \ge 0 \quad (3a)$$

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

since they obey standard stochastic constraints.

The above stochastic process could be called an observable Markov model since the output of the process is the set of states at each instant of time, where each state corresponds to a physical (observable) event. To set ideas, consider a simple 3-state Markov model of the weather. We assume that once a day (e.g., at noon), the weather is

[2]A good overview of discrete Markov processes is in [20, ch. 5].

Amazon / Zentian Limited
Exhibit 1029
Page 2

observed as being one of the following:

State 1: rain or (snow)
State 2: cloudy
State 3: sunny.

We postulate that the weather on day t is characterized by a single one of the three states above, and that the matrix A of state transition probabilities is

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}.$$

Given that the weather on day 1 ($t = 1$) is sunny (state 3), we can ask the question: What is the probability (according to the model) that the weather for the next 7 days will be "sun-sun-rain-rain-sun-cloudy-sun $\cdots$"? Stated more formally, we define the observation sequence O as $O = \{S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3\}$ corresponding to $t = 1, 2, \cdots, 8$, and we wish to determine the probability of O, given the model. This probability can be expressed (and evaluated) as

$$\begin{aligned} P(O|\text{Model}) &= P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|\text{Model}] \\ &= P[S_3] \cdot P[S_3|S_3] \cdot P[S_3|S_3] \cdot P[S_1|S_3] \\ &\quad \cdot P[S_1|S_1] \cdot P[S_3|S_1] \cdot P[S_2|S_3] \cdot P[S_3|S_2] \\ &= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ &= 1 \cdot (0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\ &= 1.536 \times 10^{-4} \end{aligned}$$

where we use the notation

$$\pi_i = P[q_1 = S_i], \qquad 1 \le i \le N \tag{4}$$

to denote the initial state probabilities.

Another interesting question we can ask (and answer using the model) is: Given that the model is in a known state, what is the probability it stays in that state for exactly d days? This probability can be evaluated as the probability of the observation sequence

$$O = \{\underset{1}{S_i}, \underset{2}{S_i}, \underset{3}{S_i}, \cdots, \underset{d}{S_i}, \underset{d+1}{S_j} \neq S_i\},$$

given the model, which is

$$P(O|\text{Model}, q_1 = S_i) = (a_{ii})^{d-1}(1 - a_{ii}) = p_i(d). \tag{5}$$

The quantity $p_i(d)$ is the (discrete) probability density function of duration d in state i. This exponential duration density is characteristic of the state duration in a Markov chain. Based on $p_i(d)$, we can readily calculate the expected number of observations (duration) in a state, conditioned on starting in that state as

$$\bar{d}_i = \sum_{d=1}^{\infty} d p_i(d) \tag{6a}$$

$$= \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}}. \tag{6b}$$

Thus the expected number of consecutive days of sunny weather, according to the model, is $1/(0.2) = 5$; for cloudy it is 2.5; for rain it is 1.67.

A. Extension to Hidden Markov Models

So far we have considered Markov models in which each state corresponded to an observable (physical) event. This model is too restrictive to be applicable to many problems of interest. In this section we extend the concept of Markov models to include the case where the observation is a probabilistic function of the state—i.e., the resulting model (which is called a hidden Markov model) is a doubly embedded 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 observations. To fix ideas, consider the following model of some simple coin tossing experiments.

Coin Toss Models: Assume the following scenario. 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, with the observation sequence consisting of a series of heads and tails; e.g., a typical observation sequence would be

$$\begin{aligned} \boldsymbol{O} &= O_1 O_2 O_3 \cdots O_T \\ &= \mathcal{H}\ \mathcal{H}\ \mathcal{T}\ \mathcal{T}\ \mathcal{T}\ \mathcal{H}\ \mathcal{T}\ \mathcal{T}\ \mathcal{H} \cdots \mathcal{H} \end{aligned}$$

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

Given the above scenario, the problem of interest is how do we build an HMM to explain (model) the observed sequence of heads and tails. The first problem one faces is deciding what the states in the model correspond to, and then deciding how many states should be in the model. One possible choice would be to assume that only a single biased coin was being tossed. In this case we could model the situation with a 2-state model where each state corresponds to a side of the coin (i.e., heads or tails). This model is depicted in Fig. 2(a).[3] In this case the Markov model is observable, and the only issue for complete specification of the model would be to decide on the best value for the bias (i.e., the probability of, say, heads). Interestingly, an equivalent HMM to that of Fig. 2(a) would be a degenerate 1-state model, where the state corresponds to the single biased coin, and the unknown parameter is the bias of the coin.

A second form of HMM for explaining the observed sequence of coin toss outcome is given in Fig. 2(b). In this case there are 2 states in the model and each state corresponds to a different, biased, coin being tossed. Each state is characterized by a probability distribution of heads and tails, and transitions between states are characterized by a state transition matrix. The physical mechanism which accounts for how state transitions are selected could itself be a set of independent coin tosses, or some other probabilistic event.

A third form of HMM for explaining the observed sequence of coin toss outcomes is given in Fig. 2(c). This model corresponds to using 3 biased coins, and choosing from among the three, based on some probabilistic event.

[3]The model of Fig. 2(a) is a memoryless process and thus is a degenerate case of a Markov model.

Amazon / Zentian Limited
Exhibit 1029
Page 3

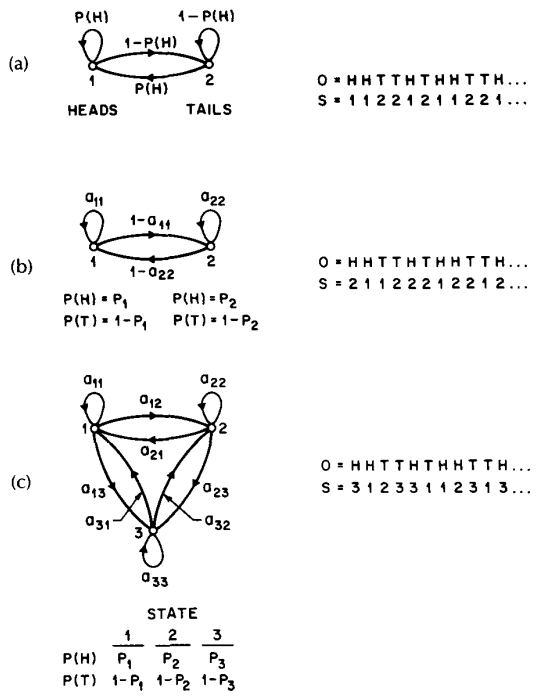

Fig. 2. Three possible Markov models which can account for the results of hidden coin tossing experiments. (a) 1-coin model. (b) 2-coins model. (c) 3-coins model.

Given the choice among the three models shown in Fig. 2 for explaining the observed sequence of heads and tails, a natural question would be which model best matches the actual observations. It should be clear that the simple 1-coin model of Fig. 2(a) has only 1 unknown parameter; the 2-coin model of Fig. 2(b) has 4 unknown parameters; and the 3-coin model of Fig. 2(c) has 9 unknown parameters. Thus, with the greater degrees of freedom, the larger HMMs would seem to inherently be more capable of modeling a series of coin tossing experiments than would equivalently smaller models. Although this is theoretically true, we will see later in this paper that practical considerations impose some strong limitations on the size of models that we can consider. Furthermore, it might just be the case that only a single coin is being tossed. Then using the 3-coin model of Fig. 2(c) would be inappropriate, since the actual physical event would not correspond to the model being used—i.e., we would be using an underspecified system.

The Urn and Ball Model[4]: To extend the ideas of the HMM to a somewhat more complicated situation, consider the urn and ball system of Fig. 3. We assume that there are N (large) glass urns in a room. Within each urn there are a large number of colored balls. We assume there are M distinct colors of the balls. The physical process for obtaining observations is as follows. A genie is in the room, and according to some random process, he (or she) chooses an initial urn. From this urn, a ball is chosen at random, and its color is recorded as the observation. The ball is then replaced in the urn from which it was selected. A new urn is then selected

[4]The urn and ball model was introduced by Jack Ferguson, and his colleagues, in lectures on HMM theory.

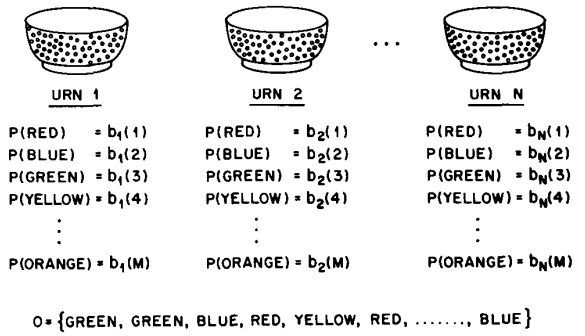

Fig. 3. An N-state urn and ball model which illustrates the general case of a discrete symbol HMM.

according to the random selection process associated with the current urn, and the ball selection process is repeated. This entire process generates a finite observation sequence of colors, which we would like to model as the observable output of an HMM.

It should be obvious that the simplest HMM that corresponds to the urn and ball process is one in which each state corresponds to a specific urn, and for which a (ball) color probability is defined for each state. The choice of urns is dictated by the state transition matrix of the HMM.

B. Elements of an HMM

The above examples give us a pretty good idea of what an HMM is and how it can be applied to some simple scenarios. We now formally define the elements of an HMM, and explain how the model generates observation sequences.

An HMM is characterized by the following:

1) N, the number of states in the model. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to sets of states of the model. Hence, in the coin tossing experiments, each state corresponded to a distinct biased coin. In the urn and ball model, the states corresponded to the urns. Generally the states are interconnected in such a way that any state can be reached from any other state (e.g., an ergodic model); however, we will see later in this paper that other possible interconnections of states are often of interest. We denote the individual states as $S = \{S_1, S_2, \cdots, S_N\}$, and the state at time t as q_t.

2) M, the number of distinct observation symbols per state, i.e., the discrete alphabet size. The observation symbols correspond to the physical output of the system being modeled. For the coin toss experiments the observation symbols were simply heads or tails; for the ball and urn model they were the colors of the balls selected from the urns. We denote the individual symbols as $V = \{v_1, v_2, \cdots, v_M\}$.

3) The state transition probability distribution $A = \{a_{ij}\}$ where

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \qquad 1 \le i, j \le N. \tag{7}$$

For the special case where any state can reach any other state in a single step, we have $a_{ij} > 0$ for all i, j. For other types of HMMs, we would have $a_{ij} = 0$ for one or more (i, j) pairs.

Amazon / Zentian Limited
Exhibit 1029
Page 4

4) The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \text{ at } t | q_t = S_j], \quad 1 \le j \le N$$
$$1 \le k \le M. \tag{8}$$

5) The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P[q_1 = S_i], \quad 1 \le i \le N. \tag{9}$$

Given appropriate values of N, M, A, B, and π, the HMM can be used as a generator to give an observation sequence

$$O = O_1 O_2 \cdots O_T \tag{10}$$

(where each observation O_t is one of the symbols from V, and T is the number of observations in the sequence) as follows:

1) Choose an initial state $q_1 = S_i$ according to the initial state distribution π.
2) Set $t = 1$.
3) Choose $O_t = v_k$ according to the symbol probability distribution in state S_i, i.e., $b_i(k)$.
4) Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state S_i, i.e., a_{ij}.
5) Set $t = t + 1$; return to step 3)if $t < T$; otherwise terminate the procedure.

The above procedure can be used as both a generator of observations, and as a model for how a given observation sequence was generated by an appropriate HMM.

It can be seen from the above discussion that a complete specification of an HMM requires specification of two model parameters (N and M), specification of observation symbols, and the specification of the three probability measures A, B, and π. For convenience, we use the compact notation

$$\lambda = (A, B, \pi) \tag{11}$$

to indicate the complete parameter set of the model.

C. The Three Basic Problems for HMMs[5]

Given the form of HMM of the previous section, there are three basic 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 $O = O_1 O_2 \cdots O_T$, and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

Problem 2: Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model λ, how do we choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$ which is optimal in some meaningful sense (i.e., best "explains" the observations)?

Problem 3: How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

[5]The material in this section and in Section III is based on the ideas presented by Jack Ferguson of IDA in lectures at Bell Laboratories.

Problem 1 is the evaluation problem, namely given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model. We can also view the problem as one of scoring how well a given model matches a given observation sequence. The latter viewpoint is extremely useful. For example, if we consider the case in which we are trying to choose among several competing models, 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., to find the "correct" state sequence. It should be clear that for all but the case of degenerate models, there is no "correct" state sequence to be found. Hence for practical situations, we usually use an optimality criterion to solve this problem as best as possible. Unfortunately, as we will see, there are several reasonable 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. Typical uses might be to learn about the structure of the model, to find optimal state sequences for continuous speech recognition, or to get average statistics of individual states, etc.

Problem 3 is the one in which we attempt to optimize the model parameters so as to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence since it is used to "train" the HMM. The training problem is the crucial one for most applications of HMMs, 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 simple isolated word speech recognizer. For each word of a W word vocabulary, we want to design a separate N-state HMM. We represent the speech signal of a given word as a time sequence of coded spectral vectors. We assume that the coding is done using a spectral codebook with M unique spectral vectors; hence each observation is the index of the spectral vector closest (in some spectral sense) to the original speech signal. Thus, for each vocabulary word, we have a training sequence consisting of a number of repetitions of sequences of codebook indices of the word (by one or more talkers). The first task is to build individual word models. This task is done by using the solution to Problem 3 to optimally estimate 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 properties of the spectral vectors that lead to the observations occurring in each state. The goal here would be to make refinements on the model (e.g., more states, different codebook size, etc.) so as to improve its capability of modeling the spoken word sequences. Finally, once the set of W HMMs has been designed and optimized and thoroughly studied, recognition of an unknown word is performed using the solution to Problem 1 to score each word model based upon the given test observation sequence, and select the word whose model score is highest (i.e., the highest likelihood).

In the next section we present formal mathematical solutions to each of the three fundamental problems for HMMs.

Amazon / Zentian Limited
Exhibit 1029
Page 5

We shall see that the three problems are linked together tightly under our probabilistic framework.

III. Solutions to the Three Basic Problems of HMMs

A. Solution to Problem 1

We wish to calculate the probability of the observation sequence, $O = O_1 O_2 \cdots O_T$, given the model λ, i.e., $P(O|\lambda)$. The most straightforward way of doing this is through enumerating every possible state sequence of length T (the number of observations). Consider one such fixed state sequence

$$Q = q_1 q_2 \cdots q_T \tag{12}$$

where q_1 is the initial state. The probability of the observation sequence O for the state sequence of (12) is

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(O_t|q_t, \lambda) \tag{13a}$$

where we have assumed statistical independence of observations. Thus we get

$$P(O|Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T). \tag{13b}$$

The probability of such a state sequence Q can be written as

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}. \tag{14}$$

The joint probability of O and Q, i.e., the probability that O and Q occur simultaneously, is simply the product of the above two terms, i.e.,

$$P(O, Q|\lambda) = P(O|Q, \lambda)\, P(Q, \lambda). \tag{15}$$

The probability of O (given the model)is obtained by summing this joint probability over all possible state sequences q giving

$$P(O|\lambda) = \sum_{\text{all } Q} P(O|Q, \lambda)\, P(Q|\lambda) \tag{16}$$

$$= \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q_1}(O_1)\, a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T). \tag{17}$$

The interpretation of the computation in the above equation is the following. Initially (at time $t = 1$) we are in state q_1 with probability π_{q_1}, and generate the symbol O_1 (in this state) with probability $b_{q_1}(O_1)$. The clock changes from time t to $t + 1$ ($t = 2$) and we make a transition to state q_2 from state q_1 with probability $a_{q_1 q_2}$, and generate symbol O_2 with probability $b_{q_2}(O_2)$. This process continues in this manner until we make the list transition (at time T) from state q_{T-1} to state q_T with probability $a_{q_{T-1} q_T}$ and generate symbol O_T with probability $b_{q_T}(O_T)$.

A little thought should convince the reader that the calculation of $P(O|\lambda)$, according to its direct definition (17) involves on the order of $2T \cdot N^T$ calculations, since at every $t = 1, 2, \cdots, T$, there are N possible states which can be reached (i.e., there are N^T possible state sequences), and for each such state sequence about $2T$ calculations are required for each term in the sum of (17). (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 $N = 5$ (states), $T = 100$ (observations), 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. Fortunately such a procedure exists and is called the forward-backward procedure.

The Forward-Backward Procedure [2], [3][6]: Consider the forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i|\lambda) \tag{18}$$

i.e., the probability of the partial observation sequence, $O_1 O_2 \cdots O_t$, (until time t) *and* state S_i at time t, given the model λ. We can solve for $\alpha_t(i)$ inductively, as follows:

1) Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N. \tag{19}$$

2) Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \le t \le T - 1, \quad 1 \le j \le N. \tag{20}$$

3) Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{21}$$

Step 1) initializes the forward probabilities as the joint probability of state S_i and initial observation O_1. The induction step, which is the heart of the forward calculation, is illustrated in Fig. 4(a). This figure shows how state S_j can be

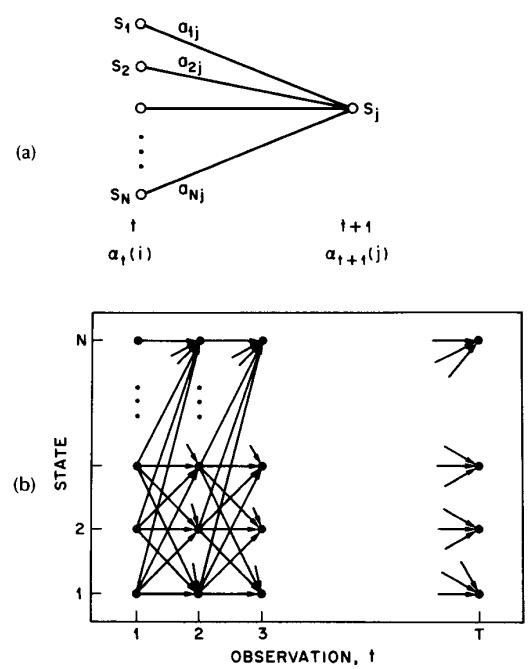

Fig. 4. (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(j)$. (b) Implementation of the computation of $\alpha_t(i)$ in terms of a lattice of observations t, and states i.

[6]Strictly speaking, we only need the forward part of the forward-backward procedure to solve Problem 1. We will introduce the backward part of the procedure in this section since it will be used to help solve Problem 3.

Amazon / Zentian Limited
Exhibit 1029
Page 6

reached at time $t + 1$ from the N possible states, S_i, $1 \le i \le N$, at time t. Since $\alpha_t(i)$ is the probability of the joint event that $O_1 O_2 \cdots O_t$ are observed, and the state at time t is S_i, the product $\alpha_t(i) a_{ij}$ is then the probability of the joint event that $O_1 O_2 \cdots O_t$ are observed, and state S_j is reached at time $t + 1$ via state S_i at time t. Summing this product over all the N possible states S_i, $1 \le i \le N$ at time t results in the probability of S_j at time $t + 1$ with all the accompanying previous partial observations. Once this is done and S_j is known, it is easy to see that $\alpha_{t+1}(j)$ is obtained by accounting for observation O_{t+1} in state j, i.e., by multiplying the summed quantity by the probability $b_j(O_{t+1})$. The computation of (20) is performed for all states j, $1 \le j \le N$, for a given t; the computation is then iterated for $t = 1, 2, \cdots, T - 1$. Finally, step 3) gives the desired calculation of $P(O|\lambda)$ as the sum of the terminal forward variables $\alpha_T(i)$. This is the case since, by definition,

$$\alpha_T(i) = P(O_1 O_2 \cdots O_T, q_T = S_i|\lambda) \tag{22}$$

and hence $P(O|\lambda)$ is just the sum of the $\alpha_T(i)$'s.

If we examine the computation involved in the calculation of $\alpha_t(j)$, $1 \le t \le T$, $1 \le j \le N$, we see that it requires on the order of N^2T 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} computations for the direct calculation, a savings of about 69 orders of magnitude.

The forward probability calculation is, in effect, based upon the lattice (or trellis) structure shown in Fig. 4(b). The key is that since there are only N states (nodes at each time slot in the lattice), all the possible state sequences will remerge into these N nodes, no matter how long the observation sequence. At time $t = 1$ (the first time slot in the lattice), we need to calculate values of $\alpha_1(i)$, $1 \le i \le N$. At times $t = 2, 3, \cdots, T$, we only need to calculate values of $\alpha_t(j)$, $1 \le j \le N$, where each calculation involves only 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 time slot.

In a similar manner,[7] we can consider a backward variable $\beta_t(i)$ defined as

$$\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T|q_t = S_i, \lambda) \tag{23}$$

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

1) Initialization:

$$\beta_T(i) = 1, \qquad 1 \le i \le N. \tag{24}$$

2) Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j),$$
$$t = T - 1, T - 2, \cdots, 1, 1 \le i \le N. \tag{25}$$

The initialization step 1) *arbitrarily* defines $\beta_T(i)$ to be 1 for all i. Step 2), which is illustrated in Fig. 5, shows that in order to have been in state S_i at time t, and to account for the observation sequence from time $t + 1$ on, you have to consider all possible states S_j at time $t + 1$, accounting for the transition from S_i to S_j (the a_{ij} term), as well as the observation O_{t+1} in state j (the $b_j(O_{t+1})$ term), and then account for the remaining partial observation sequence from state j (the $\beta_{t+1}(j)$ term). We will see later how the backward, as well as the forward calculations are used extensively to help solve fundamental Problems 2 and 3 of HMMs.

[7]Again we remind the reader that the backward procedure will be used in the solution to Problem 3, and is not required for the solution of Problem 1.

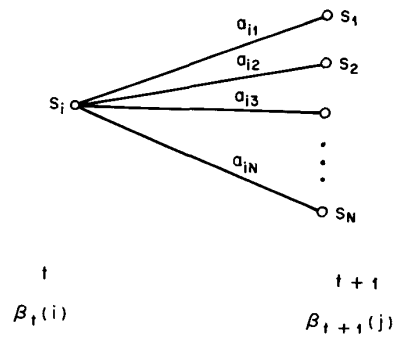

Fig. 5. Illustration of the sequence of operations required for the computation of the backward variable $\beta_t(i)$.

Again, the computation of $\beta_t(i)$, $1 \le t \le T$, $1 \le i \le N$, requires on the order of N^2T calculations, and can be computed in a lattice structure similar to that of Fig. 4(b).

B. Solution to Problem 2

Unlike Problem 1 for which an exact solution can be given, there are several possible ways of solving Problem 2, namely finding the "optimal" state sequence associated with the given observation sequence. The difficulty lies with the definition of the optimal state sequence; i.e., there are several possible optimality criteria. For example, one possible optimality criterion is to choose the states q_t which are *individually* most likely. This optimality criterion maximizes the expected number of correct individual states. To implement this solution to Problem 2, we define the variable

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) \tag{26}$$

i.e., the probability of being in state S_i at time t, given the observation sequence O, and the model λ. Equation (26) can be expressed simply in terms of the forward-backward variables, i.e.,

$$\gamma_t(i) = \frac{\alpha_t(i)\, \beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\, \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\, \beta_t(i)} \tag{27}$$

since $\alpha_t(i)$ accounts for the partial observation sequence $O_1 O_2 \cdots O_t$ and state S_i at t, while $\beta_t(i)$ accounts for the remainder of the observation sequence $O_{t+1} O_{t+2} \cdots O_T$, given state S_i at t. The normalization factor $P(O|\lambda) = \Sigma_{i=1}^{N} \alpha_t(i), \beta_t(i)$ makes $\gamma_t(i)$ a probability measure so that

$$\sum_{i=1}^{N} \gamma_t(i) = 1. \tag{28}$$

Using $\gamma_t(i)$, we can solve for the individually most likely state q_t at time t, as

$$q_t = \underset{1 \le i \le N}{\operatorname{argmax}} [\gamma_t(i)], \qquad 1 \le t \le T. \tag{29}$$

Amazon / Zentian Limited
Exhibit 1029
Page 7

Although (29) maximizes the expected number of correct states (by choosing the most likely state for each t), there could be some problems with the resulting state sequence. For example, when the HMM has state transitions which have zero probabilty ($a_{ij} = 0$ for some i and j), the "optimal" state sequence may, in fact, not even be a valid state sequence. This is due to the fact that the solution of (29) simply determines the most likely state at every instant, without regard to the probability of occurrence of *sequences* of states.

One possible solution to the above problem is to modify the optimality criterion. For example, one could solve for the state sequence that maximizes the expected number of correct pairs of states (q_t, q_{t+1}), or triples of states (q_t, q_{t+1}, q_{t+2}), etc. Although these criteria might be reasonable for some applications, the most widely used criterion is to find the *single* best state sequence (path), i.e., to maximize $P(Q|O, \lambda)$ which is equivalent to maximizing $P(Q, O|\lambda)$. A formal technique for finding this single best state sequence exists, based on dynamic programming methods, and is called the Viterbi algorithm.

Viterbi Algorithm [21], [22]: To find the single best state sequence, $Q = \{q_1 q_2 \cdots q_T\}$, for the given observation sequence $O = \{O_1 O_2 \cdots O_T\}$, we need to define the quantity

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t|\lambda] \tag{30}$$

i.e., $\delta_t(i)$ is the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state S_i. By induction we have

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}). \tag{31}$$

To actually retrieve the state sequence, we need to keep track of the argument which maximized (31), for each t and j. We do this via the array $\psi_t(j)$. The complete procedure for finding the best state sequence can now be stated as follows:

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \qquad 1 \le i \le N \tag{32a}$$

$$\psi_1(i) = 0. \tag{32b}$$

2) Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \qquad 2 \le t \le T, \quad 1 \le j \le N \tag{33a}$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \qquad 2 \le t \le T, \quad 1 \le j \le N. \tag{33b}$$

3) Termination: (34a)

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

$$q_T^* = \operatorname*{argmax}_{1 \le i \le N} [\delta_T(i)]. \tag{34b}$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \cdots, 1. \tag{35}$$

It should be noted that the Viterbi algorithm is similar (except for the backtracking step) in implementation to the forward calculation of (19)–(21). The major difference is the maximization in (33a) over previous states which is used in place of the summing procedure in (20). It also should be clear that a lattice (or trellis) structure efficiently implements the computation of the Viterbi procedure.

C. Solution to Problem 3 [1]–[5]

The third, and by far the most difficult, problem of HMMs is to determine a method to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence given the model. There is no known way to analytically solve for the model which maximizes the probability of the observation sequence. In fact, given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. We can, however, choose $\lambda = (A, B, \pi)$ such that $P(O|\lambda)$ is locally maximized using an iterative procedure such as the Baum–Welch method (or equivalently the EM (expectation-modification) method [23]), or using gradient techniques [14]. In this section we discuss one iterative procedure, based primarily on the classic work of Baum and his colleagues, for choosing model parameters.

In order to describe the procedure for reestimation (iterative update and improvement) of HMM parameters, we first define $\xi_t(i, j)$, the probability of being in state S_i at time t, and state S_j at time $t + 1$, given the model and the observation sequence, i.e.

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda). \tag{36}$$

The sequence of events leading to the conditions required by (36) is illustrated in Fig. 6. It should be clear, from the

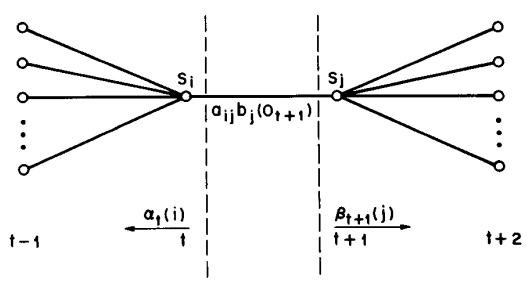

Fig. 6. Illustration of the sequence of operations required for the computation of the joint event that the system is in state S_i at time t and state S_j at time $t + 1$.

definitions of the forward and backward variables, that we can write $\xi_t(i, j)$ in the form

$$\xi_t(i, j) = \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)} \tag{37}$$

where the numerator term is just $P(q_t = S_i, q_{t+1} = S_j, O|\lambda)$ and the division by $P(O|\lambda)$ gives the desired probability measure.

Amazon / Zentian Limited
Exhibit 1029
Page 8

We have previously defined $\gamma_t(i)$ as the probability of being in state S_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 over j, giving

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j). \tag{38}$$

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 S_i is visited, or equivalently, the expected number of transitions made from state S_i (if we exclude the time slot $t = T$ from the summation). Similarly, summation of $\xi_t(i, j)$ over t (from $t = 1$ to $t = T - 1$) can be interpreted as the expected number of transitions from state S_i to state S_j. That is

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i \tag{39a}$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from } S_i \text{ to } S_j. \tag{39b}$$

Using the above formulas (and the concept of counting event occurrences) we can give a method for reestimation of the parameters of an HMM. A set of reasonable reestimation formulas for π, A, and B are

$$\bar{\pi}_i = \text{expected frequency (number of times) in state } S_i \text{ at time } (t = 1) = \gamma_1(i) \tag{40a}$$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{40b}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} = \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{40c}$$

If we define the current model as $\lambda = (A, B, \pi)$, and use that to compute the right-hand sides of (40a)–(40c), and we define the reestimated model as $\bar{\lambda} = \bar{A}, \bar{B}, \bar{\pi})$, as determined from the left-hand sides of (40a)–(40c), then it has been proven by Baum and his colleagues [6], [3] 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 than model λ in the sense that $P(O|\bar{\lambda}) > P(O|\lambda)$, i.e., we have found a new model $\bar{\lambda}$ from which the observation sequence is more likely to have been produced.

Based on the above procedure, if we iteratively use $\bar{\lambda}$ in place of λ and repeat the reestimation calculation, we then can improve the probability of O being observed from the model until some limiting point is reached. The final result of this reestimation procedure is called a maximum likelihood estimate of the HMM. It should be pointed out that the forward–backward algorithm leads to local maxima only, and that in most problems of interest, the optimization surface is very complex and has many local maxima.

The reestimation formulas of (40a)–(40c) can be derived directly by maximizing (using standard constrained optimization techniques) Baum's auxiliary function

$$Q(\lambda, \bar{\lambda}) = \sum_{Q} P(Q|O, \lambda) \log [P(O, Q|\bar{\lambda})] \tag{41}$$

over $\bar{\lambda}$. It has been proven by Baum and his colleagues [6], [3] that maximization of $Q(\lambda, \bar{\lambda})$ leads to increased likelihood, i.e.

$$\max_{\bar{\lambda}} [Q(\lambda, \bar{\lambda})] \Rightarrow P(O|\bar{\lambda}) \geq P(O|\lambda). \tag{42}$$

Eventually the likelihood function converges to a critical point.

Notes on the Reestimation Procedure: The reestimation formulas can readily be interpreted as an implementation of the EM algorithm of statistics [23] in which the E (expectation) step is the calculation of the auxiliary function $Q(\lambda, \bar{\lambda})$, and the M (modification) step is the maximization over $\bar{\lambda}$. Thus the Baum–Welch reestimation equations are essentially identical to the EM steps for this particular problem.

An important aspect of the reestimation procedure is that the stochastic constraints of the HMM parameters, namely

$$\sum_{i=1}^{N} \bar{\pi}_i = 1 \tag{43a}$$

$$\sum_{j=1}^{N} \bar{a}_{ij} = 1, \quad 1 \leq i \leq N \tag{43b}$$

$$\sum_{k=1}^{M} \bar{b}_j(k) = 1, \quad 1 \leq j \leq N \tag{43c}$$

are automatically satisfied at each iteration. By looking at the parameter estimation problem as a constrained optimization of $P(O|\lambda)$ (subject to the constraints of (43)), the techniques of Lagrange multipliers can be used to find the values of π_i, a_{ij}, and $b_j(k)$ which maximize P (we use the notation $P = P(O|\lambda)$ as short-hand in this section). Based on setting up a standard Lagrange optimization using Lagrange multipliers, it can readily be shown that P is maximized when

Amazon / Zentian Limited
Exhibit 1029
Page 9

the following conditions are met:

$$\pi_i = \frac{\pi_i \dfrac{\partial P}{\partial \pi_i}}{\sum_{k=1}^{N} \pi_k \dfrac{\partial P}{\partial \pi_k}} \tag{44a}$$

$$a_{ij} = \frac{a_{ij} \dfrac{\partial P}{\partial a_{ij}}}{\sum_{k=1}^{N} a_{ik} \dfrac{\partial P}{\partial a_{ik}}} \tag{44b}$$

$$b_j(k) = \frac{b_j(k) \dfrac{\partial P}{\partial b_j(k)}}{\sum_{\ell=1}^{M} b_j(\ell) \dfrac{\partial P}{\partial b_j(\ell)}}. \tag{44c}$$

By appropriate manipulation of (44), the right-hand sides of each equation can be readily converted to be *identical* to the right-hand sides of each part of (40a)-(40c), thereby showing that the reestimation formulas are indeed exactly correct at critical points of P. In fact the form of (44) is essentially that of a reestimation formula in which the left-hand side is the reestimate and the right-hand side is computed using the current values of the variables.

Finally, we note that since the entire problem can be set up as an optimization problem, standard gradient techniques can be used to solve for "optimal" values of the model parameters [14]. Such procedures have been tried and have been shown to yield solutions comparable to those of the standard reestimation procedures.

IV. Types of HMMs

Until now, we have only considered the special case of ergodic or fully connected HMMs in which every state of the model could be reached (in a single step) from every other state of the model. (Strictly speaking, an ergodic model has the property that every state can be reached from every other state in a finite number of steps.) As shown in Fig. 7(a), for an $N = 4$ state model, this type of model has the property that every a_{ij} coefficient is positive. Hence for the example of Fig. 7a we have

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}.$$

For some applications, in particular those to be discussed later in this paper, other types of HMMs have been found to account for observed properties of the signal being modeled better than the standard ergodic model. One such model is shown in Fig. 7(b). This model is called a left-right model or a Bakis model [11], [10] because the underlying state sequence associated with the model has the property that as time increases the state index increases (or stays the same), i.e., the states proceed from left to right. Clearly the left-right type of HMM has the desirable property that it can readily model signals whose properties change over time—e.g., speech. The fundamental property of all left-right

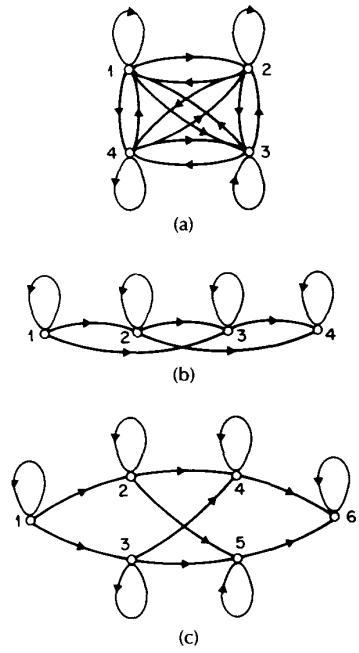

Fig. 7. Illustration of 3 distinct types of HMMs. (a) A 4-state ergodic model. (b) A 4-state left-right model. (c) A 6-state parallel path left-right model.

HMMs is that the state transition coefficients have the property

$$a_{ij} = 0, \quad j < i \tag{45}$$

i.e., no transitions are allowed to states whose indices are lower than the current state. Furthermore, the initial state probabilities have the property

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{46}$$

since the state sequence must begin in state 1 (and end in state N). Often, with left-right models, additional constraints are placed on the state transition coefficients to make sure that large changes in state indices do not occur; hence a constraint of the form

$$a_{ij} = 0, \quad j > i + \Delta \tag{47}$$

is often used. In particular, for the example of Fig. 7(b), the value of Δ is 2, i.e., no jumps of more than 2 states are allowed. The form of the state transition matrix for the example of Fig. 7(b) is thus

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}.$$

It should be clear that, for the last state in a left-right model, that the state transition coefficients are specified as

$$a_{NN} = 1 \tag{48a}$$

$$a_{Ni} = 0, \quad i < N. \tag{48b}$$

Amazon / Zentian Limited
Exhibit 1029
Page 10

Although we have dichotomized HMMs into ergodic and left-right models, there are many possible variations and combinations possible. By way of example, Fig. 7(c) shows a cross-coupled connection of two parallel left-right HMMs. Strictly speaking, this model is a left-right model (it obeys all the a_{ij} constraints); however, it can be seen that it has certain flexibility not present in a strict left-right model (i.e., one without parallel paths).

It should be clear that the imposition of the constraints of the left-right model, or those of the constrained jump model, essentially have no effect on the reestimation procedure. This is the case because any HMM parameter set to zero initially, will remain at zero throughout the reestimation procedure (see (44)).

A. Continuous Observation Densities in HMMs [24]-[26]

All of our discussion, to this point, has considered only the case when the observations were characterized as discrete symbols chosen from a finite alphabet, and therefore we could use a discrete probability density within each state of this model. The problem with this approach, at least for some applications, is that the observations are continuous signals (or vectors). Although it is possible to quantize such continuous signals via codebooks, etc., there might be serious degradation associated with such quantization. Hence it would be advantageous to be able to use HMMs with continuous observation densities.

In order to use a continuous observation density, some restrictions have to be placed on the form of the model probability density function (pdf) to insure that the parameters of the pdf can be reestimated in a consistent way. The most general representation of the pdf, for which a reestimation procedure has been formulated [24]-[26], is a finite mixture of the form

$$b_j(\boldsymbol{O}) = \sum_{m=1}^{M} c_{jm}\mathfrak{N}[\boldsymbol{O}, \boldsymbol{\mu}_{jm}, \boldsymbol{U}_{jm}], \qquad 1 \le j \le N \tag{49}$$

where $\boldsymbol{O}$ is the vector being modeled, c_{jm} is the mixture coefficient for the mth mixture in state j and $\mathfrak{N}$ is any log-concave or elliptically symmetric density [24] (e.g., Gaussian), with mean vector $\boldsymbol{\mu}_{jm}$ and covariance matrix $\boldsymbol{U}_{jm}$ for the mth mixture component in state j. Usually a Gaussian density is used for $\mathfrak{N}$. The mixture gains c_{jm} satisfy the stochastic constraint

$$\sum_{m=1}^{M} c_{jm} = 1, \qquad 1 \le j \le N \tag{50a}$$

$$c_{jm} \ge 0, \qquad 1 \le j \le N, 1 \le m \le M \tag{50b}$$

so that the pdf is properly normalized, i.e.,

$$\int_{-\infty}^{\infty} b_j(\boldsymbol{x})\, d\boldsymbol{x} = 1, \qquad 1 \le j \le N. \tag{51}$$

The pdf of (49) can be used to approximate, arbitrarily closely, any finite, continuous density function. Hence it can be applied to a wide range of problems.

It can be shown [24]-[26] that the reestimation formulas for the coefficients of the mixture density, i.e., c_{jm}, $\boldsymbol{\mu}_{jk}$, and $\boldsymbol{U}_{jk}$, are of the form

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j, k)} \tag{52}$$

$$\bar{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k) \cdot \boldsymbol{O}_t}{\sum_{t=1}^{T} \gamma_t(j, k)} \tag{53}$$

$$\bar{\boldsymbol{U}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k) \cdot (\boldsymbol{O}_t - \boldsymbol{\mu}_{jk})(\boldsymbol{O}_t - \boldsymbol{\mu}_{jk})'}{\sum_{t=1}^{T} \gamma_t(j, k)} \tag{54}$$

where prime denotes vector transpose and where $\gamma_t(j, k)$ is the probability of being in state j at time t with the kth mixture component accounting for $\boldsymbol{O}_t$, i.e.,

$$\gamma_t(j, k) = \left[\frac{\alpha_t(j)\,\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)}\right]\left[\frac{c_{jk}\mathfrak{N}(\boldsymbol{O}_t, \boldsymbol{\mu}_{jk}, \boldsymbol{U}_{jk})}{\sum_{m=1}^{M} c_{jm}\mathfrak{N}(\boldsymbol{O}_t, \boldsymbol{\mu}_{jm}, \boldsymbol{U}_{jm})}\right].$$

(The term $\gamma_t(j, k)$ generalizes to $\gamma_t(j)$ of (26) in the case of a simple mixture, or a discrete density.) The reestimation formula for a_{ij} is identical to the one used for discrete observation densities (i.e., (40b)). The interpretation of (52)-(54) is fairly straightforward. The reestimation formula for c_{jk} is the ratio between the expected number of times the system is in state j using the kth mixture component, and the expected number of times the system is in state j. Similarly, the reestimation formula for the mean vector $\boldsymbol{\mu}_{jk}$ weights each numerator term of (52) by the observation, thereby giving the expected value of the portion of the observation vector accounted for by the kth mixture component. A similar interpretation can be given for the reestimation term for the covariance matrix $\boldsymbol{U}_{jk}$.

B. Autoregressive HMMS [27], [28]

Although the general formulation of continuous density HMMs is applicable to a wide range of problems, there is one other very interesting class of HMMs that is particularly applicable to speech processing. This is the class of autoregressive HMMs [27], [28]. For this class, the observation vectors are drawn from an autoregression process.

To be more specific, consider the observation vector $\boldsymbol{O}$ with components $(x_0, x_1, x_2, \cdots, x_{K-1})$. Since the basis probability density function for the observation vector is Gaussian autoregressive (or order p), then the components of $\boldsymbol{O}$ are related by

$$\boldsymbol{O}_k = -\sum_{i=1}^{p} a_i \boldsymbol{O}_{k-i} + e_k \tag{55}$$

where e_k, $k = 0, 1, 2, \cdots, K-1$ are Gaussian, independent, identically distributed random variables with zero mean and variance σ^2, and a_i, $i = 1, 2, \cdots, p$, are the autoregression or predictor coefficients. It can be shown that for large K, the density function for $\boldsymbol{O}$ is approximately

$$f(\boldsymbol{O}) = (2\pi\sigma^2)^{-K/2} \exp\left\{-\frac{1}{2\sigma^2}\,\delta(\boldsymbol{O}, \boldsymbol{a})\right\} \tag{56}$$

where

$$\delta(\boldsymbol{O}, \boldsymbol{a}) = r_a(0)\, r(0) + 2\sum_{i=1}^{p} r_a(i)\, r(i) \tag{57a}$$

$$\boldsymbol{a}' = [1, a_1, a_2, \cdots, a_p] \tag{57b}$$

Amazon / Zentian Limited
Exhibit 1029
Page 11

$$r_a(i) = \sum_{n=0}^{p-i} a_n a_{n+i} \qquad (a_0 = 1),\ 1 \le i \le p \tag{57c}$$

$$r(i) = \sum_{n=0}^{K-i-1} x_n x_{n+i} \qquad 0 \le i \le p. \tag{57d}$$

In the above equations it can be recognized that $r(i)$ is the autocorrelation of the observation samples, and $r_a(i)$ is the autocorrelation of the autoregressive coefficients.

The total (frame) prediction residual α can be written as

$$\alpha = E\left[\sum_{i=1}^{K} (e_i)^2\right] = K\sigma^2 \tag{58}$$

where σ^2 is the variance per sample of the error signal. Consider the normalized observation vector

$$\hat{\boldsymbol{O}} = \frac{\boldsymbol{O}}{\sqrt{\alpha}} = \frac{\boldsymbol{O}}{\sqrt{K\sigma^2}} \tag{59}$$

where each sample x_i is divided by $\sqrt{K\sigma^2}$, i.e., each sample is normalized by the sample variance. Then $f(\hat{\boldsymbol{O}})$ can be written as

$$f(\hat{\boldsymbol{O}}) = \left(\frac{2\pi}{K}\right)^{-K/2} \exp\left(-\frac{K}{2}\,\delta(\hat{\boldsymbol{O}}, \boldsymbol{a})\right). \tag{60}$$

In practice, the factor K (in front of the exponential of (60)) is replaced by an *effective* frame length $\hat{K}$ which represents the effective length of each data vector. Thus if consecutive data vectors are overlapped by 3 to 1, then we would use $\hat{K} = K/3$ in (60), so that the contribution of each sample of signal to the overall density is counted exactly once.

The way in which we use Gaussian autoregressive density in HMMs is straightforward. We assume a mixture density of the form

$$b_j(\boldsymbol{O}) = \sum_{m=1}^{M} c_{jm} b_{jm}(\boldsymbol{O}) \tag{61}$$

where each $b_{jm}(\boldsymbol{O})$ is the density defined by (60) with autoregression vector $\boldsymbol{a}_{jm}$ (or equivalently by autocorrelation vector $\boldsymbol{r}_{a_{jm}}$), i.e.,

$$b_{jm}(\boldsymbol{O}) = \left(\frac{2\pi}{K}\right)^{-K/2} \exp\left\{-\frac{K}{2}\,\delta(\boldsymbol{O}, \boldsymbol{a}_{jm})\right\}. \tag{62}$$

A reestimation formula for the sequence autocorrelation, $r(i)$ of (57d), for the jth state, kth mixture, component has been derived, and is of the form

$$\bar{\boldsymbol{r}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k) \cdot \boldsymbol{r}_t}{\sum_{t=1}^{T} \gamma_t(j, k)} \tag{63a}$$

where $\gamma_t(j, k)$ is defined as the probability of being in state j at time t and using mixture component k, i.e.,

$$\gamma_t(j, k) = \left[\frac{\alpha_t(j)\,\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)}\right]\left[\frac{c_{jk} b_{jk}(\boldsymbol{O}_t)}{\sum_{k=1}^{M} c_{jk} b_{jk}(\boldsymbol{O}_t)}\right]. \tag{63b}$$

It can be seen that $\bar{\boldsymbol{r}}_{jk}$ is a weighted sum (by probability of occurrence) of the normalized autocorrelations of the frames in the observation sequence. From $\bar{\boldsymbol{r}}_{jk}$, one can solve a set of normal equations to obtain the corresponding autoregressive coefficient vector $\bar{\boldsymbol{a}}_{jk}$, for the kth mixture of state j. The new autocorrection vectors of the autoregression coefficients can then be calculated using (57c), thereby closing the reestimation loop.

C. Variants on HMM Structures—Null Transitions and Tied States

Throughout this paper we have considered HMMs in which the observations were associated with states of the model. It is also possible to consider models in which the observations are associated with the arcs of the model. This type of HMM has been used extensively in the IBM continuous speech recognizer [13]. It has been found useful, for this type of model, to allow transitions which produce no output—i.e., jumps from one state to another which produce no observation [13]. Such transitions are called null transitions and are designated by a dashed line with the symbol ϕ used to denote the null output.

Fig. 8 illustrates 3 examples (from speech processing tasks) where null arcs have been successfully utilized. The

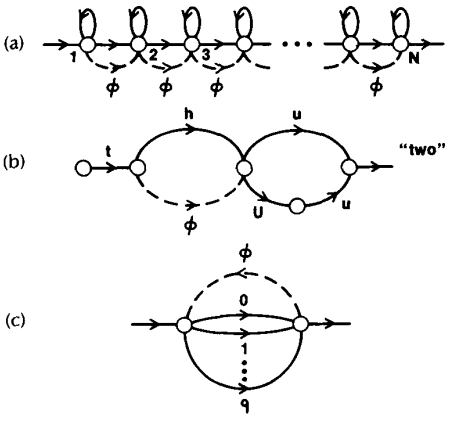

Fig. 8. Examples of networks incorporating null transitions. (a) Left-right model. (b) Finite state network. (c) Grammar network.

example of part (a) corresponds to an HMM (a left-right model) with a large number of states in which it is possible to omit transitions between any pair of states. Hence it is possible to generate observation sequences with as few as 1 observation and still account for a path which begins in state 1 and ends in state N.

The example of Fig. 8(b) is a finite state network (FSN) representation of a word in terms of linguistic unit models (i.e., the sound on each arc is itself an HMM). For this model the null transition gives a compact and efficient way of describing alternate word pronunciations (i.e., symbol delections).

Finally the FSN of Fig. 8(c) shows how the ability to insert a null transition into a grammar network allows a relatively simple network to generate arbitrarily long word (digit) sequences. In the example shown in Fig. 8(c), the null transition allows the network to generate arbitrary sequences of digits of arbitrary length by returning to the initial state after each individual digit is produced.

Another interesting variation in the HMM structure is the concept of parameter tieing [13]. Basically the idea is to set up an equivalence relation between HMM parameters in

Amazon / Zentian Limited
Exhibit 1029
Page 12

different states. In this manner the number of independent parameters in the model is reduced and the parameter estimation becomes somewhat simpler. Parameter tieing is used in cases where the observation density (for example) is known to be the same in 2 or more states. Such cases occur often in characterizing speech sounds. The technique is especially appropriate in the case where there is insufficient training data to estimate, reliably, a large number of model parameters. For such cases it is appropriate to tie model parameters so as to reduce the number of parameters (i.e., size of the model) thereby making the parameter estimation problem somewhat simpler. We will discuss this method later in this paper.

D. Inclusion of Explicit State Duration Density in HMMs[8] *[29], [30]*

Perhaps the major weakness of conventional HMMs is the modeling of state duration. Earlier we showed (5) that the inherent duration probability density $p_i(d)$ associated with state S_i, with self transition coefficient a_{ii}, was of the form

$$\begin{aligned} p_i(d) &= (a_{ii})^{d-1}(1 - a_{ii}) \\ &= \text{probability of } d \text{ consecutive observations in state } S_i. \end{aligned} \tag{64}$$

For most physical signals, this exponential state duration density is inappropriate. Instead we would prefer to explicitly model duration density in some analytic form. Fig. 9

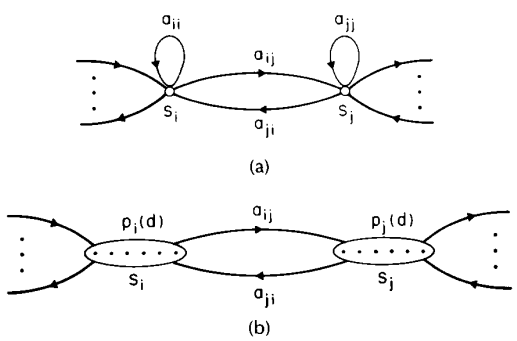

Fig. 9. Illustration of general interstate connections of (a) a normal HMM with exponential state duration density, and (b) a variable duration HMM with specified state densities and no self transitions from a state back to itself.

illustrates, for a pair of model states S_i and S_j, the differences between HMMs without and with explicit duration density. In part (a) the states have exponential duration densities based on self-transition coefficients a_{ii} and a_{jj}, respectively. In part (b), the self-transition coefficients are set to zero, and an explicit duration density is specified.[9] For this case, a transition is made only after the appropriate number of observations have occurred in the state (as specified by the duration density).

Based on the simple model of Fig. 9(b), the sequence of events of the variable duration HMM is as follows:

1) An initial state, $q_1 = S_i$, is chosen according to the initial state distribution π_i.
2) A duration d_1 is chosen according to the state duration density $p_{q_1}(d_1)$. (For expedience and ease of implementation the duration density $p_q(d)$ is truncated at a maximum duration value D.)
3) Observations $O_1 O_2 \cdots O_{d_1}$ are chosen according to the joint observation density, $b_{q_1}(O_1 O_2 \cdots O_{d_1})$. Generally we assume independent of observations so that $b_{q_1}(O_1 O_2 \cdots O_{d_1}) = \Pi_{t=1}^{d_1} b_{q_1}(O_t)$.
4) The next state, $q_2 = S_j$, is chosen according to the state transition probabilities, $a_{q_1 q_2}$, with the constraint that $a_{q_1 q_2} = 0$, i.e., no transition back to the same state can occur. (Clearly this is a requirement since we assume that, in state q_1, exactly d_1 observations occur.)

A little thought should convince the reader that the variable duration HMM can be made equivalent to the standard HMM by setting $p_i(d)$ to be the exponential density of (64).

Using the above formulation, several changes must be made to the formulas of Section III to allow calculation of $P(O|\lambda)$ and for reestimation of all model parameters. In particular we assume that the first state begins at $t = 1$ and the last state ends at $t = T$, i.e., entire duration intervals are included with the observation sequence. We then define the forward variable $\alpha_t(i)$ as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, S_i \text{ ends at } t|\lambda). \tag{65}$$

We assume that a total of r states have been visited during the first t observations and we denote the states as $q_1, q_2, \cdots, q_r$ with durations associated with each state of $d_1, d_2, \cdots, d_r$. Thus the constraints of (65) are

$$q_r = S_i \tag{66a}$$

$$\sum_{s=1}^{r} d_s = t. \tag{66b}$$

Equation (65) can then be written as

$$\begin{aligned} \alpha_t(i) = \sum_q \sum_d \pi_{q_1} \cdot p_{q_1}(d_1) \cdot P(O_1 O_2 \cdots O_{d_1}|q_1) \\ \cdot a_{q_1 q_2} p_{q_2}(d_2)\, P(O_{d_1+1} \cdots O_{d_1+d_2}|q_2) \cdots \\ \cdot a_{q_{r-1} q_r} p_{q_r}(d_r)\, P(O_{d_1+d_2+\cdots+d_{r-1}+1} \cdots O_t|q_r) \end{aligned} \tag{67}$$

where the sum is over all states q and all possible state durations d. By induction we can write $\alpha_t(j)$ as

$$\alpha_t(j) = \sum_{i=1}^{N} \sum_{d=1}^{D} \alpha_{t-d}(i)\, a_{ij} p_j(d) \prod_{s=t-d+1}^{t} b_j(\boldsymbol{O}_s) \tag{68}$$

where D is the maximum duration within any state. To initialize the computation of $\alpha_t(j)$ we use

$$\alpha_1(i) = \pi_i p_i(1) \cdot b_i(\boldsymbol{O}_1) \tag{69a}$$

$$\alpha_2(i) = \pi_i p_i(2) \prod_{s=1}^{2} b_i(\boldsymbol{O}_s) + \sum_{\substack{j=1 \\ j \neq i}}^{N} \alpha_1(j)\, a_{ji} p_i(1)\, b_i(\boldsymbol{O}_2) \tag{69b}$$

[8]In cases where a Bakis type model is used, i.e., left-right models where the number of states is proportional to the average duration, explicit inclusion of state duration density is neither necessary nor is it useful.

[9]Again the ideas behind using explicit state duration densities are due to Jack Ferguson of IDA. Most of the material in this section is based on Ferguson's original work.

Amazon / Zentian Limited
Exhibit 1029
Page 13

$$\alpha_3(i) = \pi_i p_i(3) \prod_{s=1}^{3} b_i(\boldsymbol{O}_s) + \sum_{d=1}^{2} \sum_{\substack{j=1 \\ j \neq i}}^{N} \alpha_{3-d}(j)\, a_{ji} p_i(d) \cdot \prod_{s=4-d}^{3} b_i(\boldsymbol{O}_s) \tag{69c}$$

etc., until $\alpha_D(i)$ is computed; then (68) can be used for all $t > D$. It should be clear that the desired probability of O given the model λ can be written in terms of the α's as

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{70}$$

as was previously used for ordinary HMMs.

In order to give reestimation formulas for all the variables of the variable duration HMM, we must define three more forward-backward variables, namely

$$\alpha_t^*(i) = P(O_1\, O_2 \cdots O_t,\, S_i \text{ begins at } t+1|\lambda) \tag{71}$$

$$\beta_t(i) = P(O_{t+1} \cdots O_T|S_i \text{ ends at } t, \lambda) \tag{72}$$

$$\beta_t^*(i) = P(O_{t+1} \cdots O_T|S_i \text{ begins at } t+1, \lambda). \tag{73}$$

The relationships between α, α^*, β, and β^* are as follows:

$$\alpha_t^*(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} \tag{74}$$

$$\alpha_t(i) = \sum_{d=1}^{D} \alpha_{t-d}^*(i)\, p_i(d) \prod_{s=t-d+1}^{t} b_i(\boldsymbol{O}_s) \tag{75}$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \beta_t^*(j) \tag{76}$$

$$\beta_t^*(i) = \sum_{d=1}^{D} \beta_{t+d}(i)\, p_i(d) \prod_{s=t+1}^{t+d} b_i(\boldsymbol{O}_s). \tag{77}$$

Based on the above relationships and definitions, the reestimation formulas for the variable duration HMM are

$$\bar{\pi}_i = \frac{\pi_i \beta_0^*(i)}{P(O|\lambda)} \tag{78}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T} \alpha_t(i)\, a_{ij} \beta_t^*(j)}{\sum_{j=1}^{N} \sum_{t=1}^{T} \alpha_t(i)\, a_{ij} \beta_t^*(j)} \tag{79}$$

$$\bar{b}_i(k) = \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = k}}^{T} \left[\sum_{\tau < t} \alpha_\tau^*(i) \cdot \beta_\tau^*(i) - \sum_{\tau < t} \alpha_\tau(i)\, \beta_\tau(i) \right]}{\sum_{k=1}^{M} \sum_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \left[\sum_{\tau < t} \alpha_\tau^*(i) \cdot \beta_\tau^*(i) - \sum_{\tau < t} \alpha_\tau(i)\, \beta_\tau(i) \right]} \tag{80}$$

$$\bar{p}_i(d) = \frac{\sum_{t=1}^{T} \alpha_t^*(i)\, p_i(d)\, \beta_{t+d}(i) \prod_{s=t+1}^{t+d} b_i(\boldsymbol{O}_s)}{\sum_{d=1}^{D} \sum_{t=1}^{T} \alpha_t^*(i)\, p_i(d)\, \beta_{t+d}(i) \prod_{s=t+1}^{t+d} b_i(\boldsymbol{O}_s)}. \tag{81}$$

The interpretation of the reestimation formulas is the following. The formula for $\bar{\pi}_i$ is the probability that state i was the first state, given O. The formula for $\bar{a}_{ij}$ *is almost the same as for the usual HMM except it uses the condition that the alpha terms in which a state ends at t,* join with the beta terms in which a new state begins at $t + 1$. The formula for $\bar{b}_i(k)$ (assuming a discrete density) is the expected number of times that observation $O_t = v_k$ occurred in state i, normalized by the expected number of times that any observation occurred in state i. Finally, the reestimation formula for $\bar{p}_i(d)$ is the ratio of the expected number of times state i occurred with duration d, to the expected number of times state i occurred with any duration.

The importance of incorporating state duration densities is reflected in the observation that, for some problems, the quality of the modeling is significantly improved when explicit state duration densities are used. However, there are drawbacks to the use of the variable duration model discussed in this section. One is the greatly increased computational load associated with using variable durations. It can be seen from the definition and initialization conditions on the forward variable $\alpha_i(i)$, from (68)–(69), that about D times the storage and $D^2/2$ times the computation is required. For D on the order of 25 (as is reasonable for many speech processing problems), computation is increased by a factor of 300. Another problem with the variable duration models is the large number of parameters (D), associated with each state, that must be estimated, in addition to the usual HMM parameters. Furthermore, for a fixed number of observations T, in the training set, there are, on average, fewer state transitions and much less data to estimate $p_i(d)$ than would be used in a standard HMM. Thus the reestimation problem is more difficult for variable duration HMMs than for the standard HMM.

One proposal to alleviate some of these problems is to use a parametric state duration density instead of the nonparametric $p_i(d)$ used above [29], [30]. In particular, proposals include the Gaussian family with

$$p_i(d) = \mathfrak{N}(d, \mu_i, \sigma_i^2) \tag{82}$$

with parameters μ_i and σ_i^2, or the Gamma family with

$$p_i(d) = \frac{\eta_i^{\nu_i} d^{\nu_i - 1} e^{-\eta_i d}}{\Gamma(\nu_i)} \tag{83}$$

with parameters ν_i and η_i and with mean $\nu_i \eta_i^{-1}$ and variance $\nu_i \eta_i^{-2}$. Reestimation formulas for η_i and ν_i have been derived and used with good results [19]. Another possibility, which has been used with good success, is to assume a uniform duration distribution (over an appropriate range of durations) and use a path-constrained Viterbi decoding procedure [31].

E. Optimization Criterion—ML, MMI, and MDI [32], [33]

The basic philosophy of HMMs is that a signal (or observation sequence) can be well modeled if the parameters of an HMM are carefully and correctly chosen. The problem with this philosophy is that it is sometimes inaccurate—either because the signal does not obey the constraints of the HMM, or because it is too difficult to get reliable estimates of all HMM parameters. To alleviate this type of problem, there has been proposed at least two alternatives to the standard maximum likelihood (ML) optimization procedure for estimating HMM parameters.

The first alternative [32] is based on the idea that several HMMs are to be designed and we wish to design them all at the same time in such a way so as to maximize the discrimination power of each model (i.e., each model's ability

Amazon / Zentian Limited
Exhibit 1029
Page 14

to distinguish between observation sequences generated by the correct model and those generated by alternative models). We denote the different HMMs as λ_ν, $\nu = 1, 2, \cdots, V$. The standard ML design criterion is to use a separate training sequence of observations O^ν to derive model parameters for each model λ_ν. Thus the standard ML optimization yields

$$P_\nu^* = \max_{\lambda_\nu} P(O^\nu|\lambda_\nu). \tag{84}$$

The proposed alternative design criterion [31] is the maximum mutual information (MMI) criterion in which the average mutual information I between the observation sequence O^ν and the *complete* set of models $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_V)$ is maximized. One possible way of implementing this[10] is

$$I_\nu^* = \max_\lambda \left[\log P(O^\nu|\lambda_\nu) - \log \sum_{w=1}^{V} P(O^\nu|\lambda_w)\right] \tag{85}$$

i.e., choose λ so as to separate the correct model λ_ν from all other models on the training sequence O^ν. By summing (85) over all training sequences, one would hope to attain the most separated set of models possible. Thus a possible implementation would be

$$I^* = \max_\lambda \left\{\sum_{\nu=1}^{V}\left[\log P(O^\nu|\lambda_\nu) - \log \sum_{w=1}^{V} P(O^\nu|\lambda_w)\right]\right\}. \tag{86}$$

There are various theoretical reasons why analytical (or reestimation type) solutions to (86) cannot be realized. Thus the only known way of actually solving (86) is via general optimization procedures like the steepest descent methods [32].

The second alternative philosophy is to assume that the signal to be modeled was not necessarily generated by a Markov source, but does obey certain constraints (e.g., positive definite correlation function) [33]. The goal of the design procedure is therefore to choose HMM parameters which minimize the discrimination information (DI) or the cross entropy between the set of valid (i.e., which satisfy the measurements) signal probability densities (call this set Q), and the set of HMM probability densities (call this set P_λ), where the DI between Q and P_λ can generally be written in the form

$$D(Q\|P_\lambda) = \int q(y) \ln (q(y)/p(y))\, dy \tag{87}$$

where q and p are the probability density functions corresponding to Q and P_λ. Techniques for minimizing (87) (thereby giving an MDI solution) for the optimum values of $\lambda = (A, B, \pi)$ are highly nontrivial; however, they use a generalized Baum algorithm as the core of each iteration, and thus are efficiently tailored to hidden Markov modeling [33].

It has been shown that the ML, MMI, and MDI approaches can *all* be uniformly formulated as MDI approaches.[11] The three approaches differ in either the probability density attributed to the source being modeled, or in the model effectively being used. None of the approaches, however, assumes that the source has the probability distribution of the model.

F. Comparison of HMMs [34]

An interesting question associated with HMMs is the following: Given two HMMs, λ_1 and λ_2, what is a reasonable measure of the similarity of the two models? A key point here is the similarity criterion. By way of example, consider the case of two models

$$\lambda_1 = (A_1, B_1, \pi_1) \quad \lambda_2 = (A_2, B_2, \pi_2)$$

with

$$A_1 = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix} \quad B_1 = \begin{bmatrix} q & 1-q \\ 1-q & q \end{bmatrix}$$

$$\pi_1 = [1/2 \;\; 1/2]$$

and

$$A_2 = \begin{bmatrix} r & 1-r \\ 1-r & r \end{bmatrix} \quad B_2 = \begin{bmatrix} s & 1-s \\ 1-s & s \end{bmatrix} \quad \pi_2 = [1/2 \;\; 1/2].$$

For λ_1 to be equivalent to λ_2, in the sense of having the same statistical properties for the observation symbols, i.e., $E[O_t = v_k|\lambda_1] = E[O_t = v_k|\lambda_2]$, for all v_k, we require

$$pq + (1-p)(1-q) = rs + (1-r)(1-s)$$

or, by solving for s, we get

$$s = \frac{p + q - 2pq}{1 - 2r}.$$

By choosing (arbitrarily) $p = 0.6$, $q = 0.7$, $r = 0.2$, we get $s = 13/30 \approx 0.433$. Thus, even when the two models, λ_1 and λ_2, look ostensibly very different (i.e., A_1 is very different from A_2 and B_1 is very different from B_2), statistical equivalence of the models can occur.

We can generalize the concept of model distance (dissimilarity) by defining a distance measure $D(\lambda_1, \lambda_2)$, between two Markov models, λ_1 and λ_2, as

$$D(\lambda_1, \lambda_2) = \frac{1}{T}[\log P(O^{(2)}|\lambda_1) - \log P(O^{(2)}|\lambda_2)] \tag{88}$$

where $O^{(2)} = O_1 O_2 O_3 \cdots O_T$ is a sequence of observations *generated* by model λ_2 [34]. Basically (88) is a measure of how well model λ_1 matches observations generated by model λ_2, relative to how well model λ_2 matches observations generated by itself. Several interpretations of (88) exist in terms of cross entropy, or divergence, or discrimination information [34].

One of the problems with the distance measure of (88) is that it is nonsymmetric. Hence a natural expression of this measure is the symmetrized version, namely

$$D_s(\lambda_1, \lambda_2) = \frac{D(\lambda_1\lambda_2) + D(\lambda_2, \lambda_1)}{2}. \tag{89}$$

V. Implementation Issues for HMMs

The discussion in the previous two sections has primarily dealt with the theory of HMMs and several variations on the form of the model. In this section we deal with several practical implementation issues including scaling, multiple

[10] In (85) and (86) we assume that all words are equiprobable, i.e., $p(w) = 1/V$.

[11] Y. Ephraim and L. Rabiner, "On the Relations Between Modeling Approaches for Speech Recognition," to appear in IEEE Transactions on Information Theory.

Amazon / Zentian Limited
Exhibit 1029
Page 15

observation sequences, initial parameter estimates, missing data, and choice of model size and type. For some of these implementation issues we can prescribe exact analytical solutions; for other issues we can only provide some seat-of-the-pants experience gained from working with HMMs over the last several years.

A. Scaling [14]

In order to understand why scaling is required for implementing the reestimation procedure of HMMs, consider the definition of $\alpha_t(i)$ of (18). It can be seen that $\alpha_t(i)$ consists of the sum of a large number of terms, each of the form

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(\mathbf{O}_s)\right)$$

with $q_t = S_i$. Since each a and b term is less than 1 (generally significantly less than 1), it can be seen that as t starts to get big (e.g., 10 or more), each term of $\alpha_t(i)$ starts to head exponentially to zero. For sufficiently large t (e.g., 100 or more) the dynamic range of the $\alpha_t(i)$ computation will exceed the precision range of essentially any machine (even in double precision). Hence the only reasonable way of performing the computation is by incorporating a scaling procedure.

The basic scaling procedure which is used is to multiply $\alpha_t(i)$ by a scaling coefficient that is independent of i (i.e., it depends only on t), with the goal of keeping the scaled $\alpha_t(i)$ within the dynamic range of the computer for $1 \le t \le T$. A similar scaling is done to the $\beta_t(i)$ coefficients (since these also tend to zero exponentially fast) and then, at the end of the computation, the scaling coefficients are canceled out exactly.

To understand this scaling procedure better, consider the reestimation formula for the state transition coefficients a_{ij}. If we write the reestimation formula (41) directly in terms of the forward and backward variables we get

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}. \tag{90}$$

Consider the computation of $\alpha_t(i)$. For each t, we first compute $\alpha_t(i)$ according to the induction formula (20), and then we multiply it by a scaling coefficient c_t, where

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}. \tag{91}$$

Thus, for a fixed t, we first compute

$$\alpha_t(i) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t). \tag{92a}$$

Then the scaled coefficient set $\hat{\alpha}_t(i)$ is computed as

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t)}. \tag{92b}$$

By induction we can write $\hat{\alpha}_{t-1}(j)$ as

$$\hat{\alpha}_{t-1}(j) = \left(\prod_{\tau=1}^{t-1} c_\tau\right) \alpha_{t-1}(j). \tag{93a}$$

Thus we can write $\hat{\alpha}_t(i)$ as

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \alpha_{t-1}(j) \left(\prod_{\tau=1}^{t-1} c_\tau\right) a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{t-1}(j) \left(\prod_{\tau=1}^{t-1} c_\tau\right) a_{ij} b_j(O_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)} \tag{93b}$$

i.e., each $\alpha_t(i)$ is effectively scaled by the sum over all states of $\alpha_t(i)$.

Next we compute the $\beta_t(i)$ terms from the backward recursion. The only difference here is that we use the *same* scale factors for each time t for the betas as was used for the alphas. Hence the scaled β's are of the form

$$\hat{\beta}_t(i) = c_t \beta_t(i). \tag{94}$$

Since each scale factor effectively restores the magnitude of the α terms to 1, and since the magnitudes of the α and β terms are comparable, using the same scaling factors on the β's as was used on the α's is an effective way of keeping the computation within reasonable bounds. Furthermore, in terms of the scaled variables we see that the reestimation equation (90) becomes

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)} \tag{95}$$

but each $\hat{\alpha}_t(i)$ can be written as

$$\hat{\alpha}_t(i) = \left[\prod_{s=1}^{t} c_s\right] \alpha_t(i) = C_t \alpha_t(i) \tag{96}$$

and each $\hat{\beta}_{t+1}(j)$ can be written as

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=t+1}^{T} c_s\right] \beta_{t+1}(j) = D_{t+1} \beta_{t+1}(j). \tag{97}$$

Thus (95) can be written as

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1} \beta_{t+1}(j)}. \tag{98}$$

Finally the term $C_t D_{t+1}$ can be seen to be of the form

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T \tag{99}$$

independent of t. Hence the terms $C_t D_{t+1}$ cancel out of both the numerator and denominator of (98) and the exact reestimation equation is therefore realized.

It should be obvious that the above scaling procedure applies equally well to reestimation of the π or B coefficients. It should also be obvious that the scaling procedure of (92) need not be applied at every time instant t, but can be performed whenever desired, or necessary (e.g., to prevent underflow). If scaling is not performed at some instant t, the scaling coefficients c_t are set to 1 at that time and all the conditions discussed above are then met.

The only real change to the HMM procedure because of scaling is the procedure for computing $P(O|\lambda)$. We cannot merely sum up the $\hat{\alpha}_T(i)$ terms since these are scaled already.

Amazon / Zentian Limited
Exhibit 1029
Page 16

However, we can use the property that

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1. \tag{100}$$

Thus we have

$$\prod_{t=1}^{T} c_t \cdot P(O|\lambda) = 1 \tag{101}$$

or

$$P(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t} \tag{102}$$

or

$$\log [P(O|\lambda)] = -\sum_{t=1}^{T} \log c_t. \tag{103}$$

Thus the log of P can be computed, but not P since it would be out of the dynamic range of the machine anyway.

Finally we note that when using the Viterbi algorithm to give the maximum likelihood state sequence, no scaling is required if we use logarithms in the following way. (Refer back to (32)–(34).) We define

$$\phi_t(i) = \max_{q_1, q_2, \cdots, q_t} \{\log P[q_1 q_2 \cdots q_t, O_1 O_2 \cdots O_t|\lambda]\} \tag{104}$$

and initially set

$$\phi_1(i) = \log (\pi_i) + \log [b_i(O_1)] \tag{105a}$$

with the recursion step

$$\phi_t(j) = \max_{1 \le i \le N} [\phi_{t-1}(i) + \log a_{ij}] + \log [b_j(O_t)] \tag{105b}$$

and termination step

$$\log P^* = \max_{1 \le i \le N} [\phi_T(i)]. \tag{105c}$$

Again we arrive at $\log P^*$ rather than P^*, but with significantly less computation and with no numerical problems. (The reader should note that the terms $\log a_{ij}$ of (105b) can be precomputed and therefore do not cost anything in the computation. Furthermore, the terms $\log [b_j(O_t)]$ can be precomputed when a finite observation symbol analysis (e.g., a codebook of observation sequences) is used.

B. Multiple Observation Sequences [14]

In Section IV we discussed a form of HMM called the left-right or Bakis model in which the state proceeds from state 1 at $t = 1$ to state N at $t = T$ in a sequential manner (recall the model of Fig. 7(b)). We have already discussed how a left-right model imposes constraints on the state transition matrix, and the initial state probabilities (45)–(48). However, the major problem with left-right models is that one cannot use a single observation sequence to train the model (i.e., for reestimation of model parameters). This is because the transient nature of the states within the model only allow a small number of observations for any state (until a transition is made to a successor state). Hence, in order to have sufficient data to make reliable estimates of all model parameters, one has to use multiple observation sequences.

The modification of the reestimation procedure is straightforward and goes as follows. We denote the set of K observation sequences as

$$\boldsymbol{O} = [\boldsymbol{O}^{(1)}, \boldsymbol{O}^{(2)}, \cdots, \boldsymbol{O}^{(k)}] \tag{106}$$

where $\boldsymbol{O}^{(k)} = [O_1^{(k)} O_2^{(k)} \cdots O_{T_k}^{(k)}]$ is the kth observation sequence. We assume each observation sequence is independent of every other observation sequence, and our goal is to adjust the parameters of the model λ to maximize

$$P(O|\lambda) = \prod_{k=1}^{K} P(\boldsymbol{O}^{(k)}|\lambda) \tag{107}$$

$$= \prod_{k=1}^{K} P_k. \tag{108}$$

Since the reestimation formulas are based on frequencies of occurrence of various events, the reestimation formulas for multiple observation sequences are modified by adding together the individual frequencies of occurrence for each sequence. Thus the modified reestimation formulas for $\overline{a_{ij}}$ and $\overline{b_j}(\ell)$ are

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, a_{ij} b_j(O_{t+1}^{(k)})\, \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)} \tag{109}$$

and

$$\overline{b_j}(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ \text{s.t. } O_t = v_\ell}}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)} \tag{110}$$

and π_i is not reestimated since $\pi_1 = 1$, $\pi_i = 0$, $i \neq 1$.

The proper scaling of (109)–(110) is now straightforward since each observation sequence has its own scaling factor. The key idea is to remove the scaling factor from each term before summing. This can be accomplished by writing the reestimation equations in terms of the scaled variables, i.e.,

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i)\, a_{ij} b_j(O_{t+1}^{(k)})\, \hat{\beta}_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \hat{\alpha}_t^k(i)\, \hat{\beta}_t^k(i)}. \tag{111}$$

In this manner, for each sequence $O^{(k)}$, the same scale factors will appear in each term of the sum over t as appears in the P_k term, and hence will cancel exactly. Thus using the scaled values of the alphas and betas results in an unscaled $\overline{a_{ij}}$. A similar result is obtained for the $\overline{b_j}(\ell)$ term.

C. Initial Estimates of HMM Parameters

In theory, the reestimation equations should give values of the HMM parameters which correspond to a local maximum of the likelihood function. A key question is therefore how do we choose initial estimates of the HMM parameters so that the local maximum is the global maximum of the likelihood function.

Basically there is no simple or straightforward answer to the above question. Instead, experience has shown that either random (subject to the stochastic and the nonzero value constraints) or uniform initial estimates of the π and

Amazon / Zentian Limited
Exhibit 1029
Page 17

A parameters is adequate for giving useful reestimates of these parameters in almost all cases. However, for the B parameters, experience has shown that good initial estimates are helpful in the discrete symbol case, and are essential (when dealing with multiple mixtures) in the continuous distribution case [35]. Such initial estimates can be obtained in a number of ways, including manual segmentation of the observation sequence(s) into states with averaging of observations within states, maximum likelihood segmentation of observations with averaging, and segmental k-means segmentation with clustering, etc. We discuss such segmentation techniques later in this paper.

D. Effects of Insufficient Training Data [36]

Another problem associated with training HMM parameters via reestimation methods is that the observation sequence used for training is, of necessity, finite. Thus there is often an insufficient number of occurrences of different model events (e.g., symbol occurrences within states) to give good estimates of the model parameters. One solution to this problem is to increase the size of the training observation set. Often this is impractical. A second possible solution is to reduce the size of the model (e.g., number of states, number of symbols per state, etc). Although this is always possible, often there are physical reasons why a given model is used and therefore the model size cannot be changed. A third possible solution is to interpolate one set of parameter estimates with another set of parameter estimates from a model for which an adequate amount of training data exists [36]. The idea is to simultaneously design both the desired model as well as a smaller model for which the amount of training data is adequate to give good parameter estimates, and then to interpolate the parameter estimates from the two models. The way in which the smaller model is chosen is by tieing one or more sets of parameters of the initial model to create the smaller model. Thus if we have estimates for the parameters for the model $\lambda = (A, B, \pi)$, as well as for the reduced size model $\lambda' = (A', B', \pi')$, then the interpolated model, $\tilde{\lambda} = (\tilde{A}, \tilde{B}, \tilde{\pi})$, is obtained as

$$\tilde{\lambda} = \epsilon\lambda + (1 - \epsilon)\lambda' \tag{112}$$

where ϵ represents the weighting of the parameters of the full model, and $(1 - \epsilon)$ represents the weighting of the parameters of the reduced model. A key issue is the determination of the optimal value of ϵ, which is clearly a function of the amount of training data. (As the amount of training data gets large, we expect ϵ to tend to 1.0; similarly for small amounts of training data we expect ϵ to tend to 0.0.) The solution to the determination of an optimal value for ϵ was provided by Jelinek and Mercer [36] who showed how the optimal value for ϵ could be estimated using the forward-backward algorithm by interpreting (112) as an expanded HMM of the type shown in Fig. 10. For this expanded model the parameter ϵ is the probability of a state transition from the (neutral) state $\tilde{s}$ to the model λ; similarly $(1 - \epsilon)$ is the probability of a state transition from $\tilde{s}$ to the model λ'. Between each of the models, λ and λ', and $\tilde{s}$, there is a null transition. Using the model of Fig. 9, the value of ϵ can be estimated from the training data in the standard manner. A key point is to segment the training data T into two disjoint sets, i.e., $T = T_1 \cup T_2$. Training set T_1 is first used to train models λ and λ' (i.e., to give estimates of (A, B, π) and (A', B', π')). Training set T_2 is then used to give an estimate of ϵ, assuming the models λ and λ' are fixed. A modified version of this training procedure, called the method of deleted interpolation [36], iterates the above procedure through multiple partitions of the training set. For example one might consider a partition of the training set such that T_1 is 90 percent of T and T_2 is the remaining 10 percent of T. There are a large number of ways in which such a partitioning can be accomplished but one particularly simple one is to cycle T_2 through the data, i.e., the first partition uses the last 10 percent of the data as T_2, the second partition uses the next-to-last 10 percent of the data as T_2, etc.

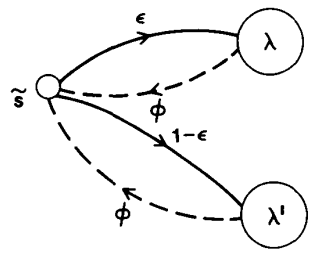

Fig. 10. Example of how the process of deleted interpolation can be represented using a state diagram.

The technique of deleted interpolation has been successfully applied to a number of problems in speech recognition including the estimation of trigram word probabilities for language models [13], and the estimation of HMM output probabilities for trigram phone models [37], [38].

Another way of handling the effects of insufficient training data is to add extra constraints to the model parameters to insure that no model parameter estimate falls below a specified level. Thus, for example, we might specify the constraint, for a discrete symbol model, that

$$b_j(k) \geq \delta \tag{113a}$$

or, for a continuous distribution model, that

$$U_{jk}(r, r) \geq \delta. \tag{113b}$$

The constraints can be applied as a postprocessor to the reestimation equations such that if a constraint is violated, the relevant parameter is manually corrected, and all remaining parameters are rescaled so that the densities obey the required stochastic constraints. Such post-processor techniques have been applied to several problems in speech processing with good success [39]. It can be seen from (112) that this procedure is essentially equivalent to a simple form of deleted interpolation in which the model λ' is a uniform distribution model, and the interpolation value ϵ is chosen as the fixed constant $(1 - \delta)$.

E. Choice of Model

The remaining issue in implementing HMMs is the choice of type of model (ergodic or left-right or some other form), choice of model size (number of states), and choice of observation symbols (discrete or continuous, single or multi-mixture, choice of observation parameters). Unfortunately, there is no simple, theoretically correct, way of making such choices. These choices must be made depending on the signal being modeled. With these comments we

Amazon / Zentian Limited
Exhibit 1029
Page 18

end our discussion of the theoretical aspects of hidden Markov models, and proceed to a discussion of how such models have been applied to selected problems in speech recognition.

VI. Implementation of Speech Recognizers Using HMMs

The purpose of this, and the following sections, is to illustrate how the ideas of HMMs, as discussed in the first 5 sections of this paper, have been applied to selected problems in speech recognition. As such, we will not strive to be as thorough or as complete in our descriptions as to what was done as we were in describing the theory of HMMs. The interested reader should read the material in [6], [10], [12], [13], [39]–[46] for more complete descriptions of individual systems. Our main goal here is to show how specific aspects of HMM theory get applied, not to make the reader an expert in speech recognition technology.

A. Overall Recognition System

Fig. 11 shows a block diagram of a pattern recognition approach to continuous speech recognition system. The key signal processing steps include the following:

1) Feature Analysis: A spectral and/or temporal analysis of the speech signal is performed to give observation vectors which can be used to train the HMMs which characterize various speech sounds. A detailed discussion of one type of feature analysis is given later in this section.

2) Unit Matching System: First a choice of speech recognition unit must be made. Possibilities include linguistically based sub-word units such as phones (or phone-like units), diphones, demisyllables, and syllables [38], as well as derivative units such as fenemes, fenones, and acoustic units [13]. Other possibilities include whole word units, and even units which correspond to a group of 2 or more words (e.g., and an, in the, of a, etc). Generally, the less complex the unit (e.g., phones), the fewer of them there are in the language, and the more complicated (variable) their structure in continuous speech. For large vocabulary speech recognition (involving 1000 or more words), the use of sub-word speech units is almost mandatory as it would be quite difficult to record an adequate training set for designing HMMs for units of the size of words or larger. However, for specialized applications (e.g., small vocabulary, constrained task), it is both reasonable and practical to consider the word as a basic speech unit. We will consider such systems exclusively in this and the following section. Independent of the unit chosen for recognition, an inventory of such units must be obtained via training. Typically each such unit is characterized by some type of HMM whose parameters are estimated from a training set of speech data. The unit matching system provides the likelihoods of a match of all sequences of speech recognition units to the unknown input speech. Techniques for providing such match scores, and in particular determining the best match score (subject to lexical and syntactic constraints of the system) include the stack decoding procedure [7], various forms of frame synchronous path decoding [37], and a lexical access scoring procedure [46].

3) Lexical Decoding: This process places constraints on the unit matching system so that the paths investigated are those corresponding to sequences of speech units which are in a word dictionary (a lexicon). This procedure implies that the speech recognition word vocabulary must be specified in terms of the basic units chosen for recognition. Such a specification can be deterministic (e.g., one or more finite state networks for each word in the vocabulary) or statistical (e.g., probabilities attached to the arcs in the finite state representation of words). In the case where the chosen units are words (or word combinations), the lexical decoding step is essentially eliminated and the structure of the recognizer is greatly simplified.

4) Syntactic Analysis: This process, much like lexical decoding, places further constraints on the unit matching system so that the paths investigated are those corresponding to speech units which comprise words (lexical decoding) and for which the words are in a proper sequence as specified by a word grammar. Such a word grammar can again be represented by a deterministic finite state network (in which all word combinations which are accepted by the grammar are enumerated), or by a statistical grammar (e.g., a trigram word model in which probabilities of sequences of 3 words in a specified order are given). For some command and control tasks, only a single word from a finite set of equiprobable is required to be recognized and therefore the grammar is either trivial or unnecessary. Such tasks are often referred to as isolated word speech recognition tasks. For other applications (e.g., digit sequences) very simple grammars are often adequate (e.g., any digit can be spoken and followed by any other digit). Finally there are tasks for which the grammar is a dominant factor and, although it adds a great deal of constraint to the recognition process, it greatly improves recognition performance by the resulting restrictions on the sequence of speech units which are valid recognition candidates.

5) Semantic Analysis: This process, again like the steps of syntactic analysis and lexical decoding, adds further constraints to the set of recognition search paths. One way in which semantic constraints are utilized is via a dynamic model of the state of the recognizer. Depending on the recognizer state certain syntactically correct input strings are eliminated from consideration. This again serves to make the recognition task easier and leads to higher performance of the system.

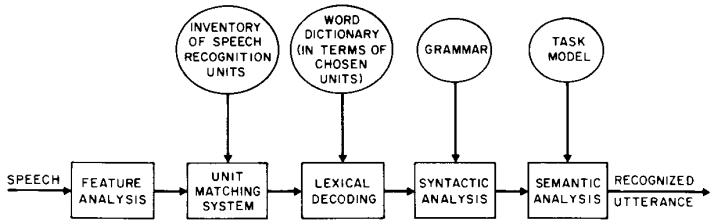

Fig. 11. Block diagram of a continuous speech recognizer.

Amazon / Zentian Limited
Exhibit 1029
Page 19

There is one additional factor that has a significant effort on the implementation of a speech recognizer and that is the problem of separating background silence from the input speech. There are at least three reasonable ways of accomplishing this task:

1) Explicitly detecting the presence of speech via techniques which discriminate background from speech on the basis of signal energy and signal durations. Such methods have been used for template-based approaches because of their inherent simplicity and their success in low to moderate noise backgrounds [48].
2) Build a model of the background silence, e.g., a statistical model, and represent the incoming signal as an arbitrary sequence of speech and background, i.e.,

$$\text{signal} = (\text{silence}) - \text{speech} - (\text{silence})$$

where the silence part of the signal is optional in that it may not be present before or after the speech [49].
3) Extend the speech unit models so that background silence is included (optionally) within the first and/or last state of the model, and therefore silence inherently gets included within all speech unit models.

All three of these techniques have been utilized in speech recognition systems.

Instead of discussing the general continuous speech recognition system further, we now present specialized applications to illustrate how HMM technology can be utilized. First we present a system where the basic speech unit is the word, where the task is to recognize a single spoken word, and where there is no task syntax or semantics to constrain the choice of words. This task is generally referred to as isolated word recognition. Next we discuss a slightly more complicated task in which the basic speech unit is still the word, but where the task is to recognize a continuous utterance consisting of words from the vocabularly. Included in such a task is the problem of recognizing a spoken string of digits. We again consider the case where there is no task syntax or semantics to constrain the choice of words, i.e., any digit can follow any other digit. Recognition tasks of this type have been referred to as connected word recognizers because the continuous speech is recognized as a concatenated sequence of word models. This is technically a misnomer because it is truly a continuous speech recognition problem. However, the terminology has become established and we continue its use.

B. Isolated Word Recognition

As our first example, consider using HMMs to build an isolated word recognizer. Assume we have a vocabulary of V words to be recognized and that each word is to be modeled by a distinct HMM.[12] Further assume that for each word in the vocabulary we have a training set of K occurrences of each spoken word (spoken by 1 or more talkers) where each occurrence of the word constitutes an observation sequence, where the observations are some appropriate representation of the (spectral and/or temporal) characteristics of the word. (We will return to the question of what specific representation is used later in this section.) In order to do isolated word speech recognition, we must perform the following:

1) For each word v in the vocabulary, we must build an HMM λ^v, i.e., we must estimate the model parameters (A, B, π) that optimize the likelihood of the training set observation vectors for the vth word.
2) For each unknown word which is to be recognized, the processing of Fig. 12 must be carried out, namely measurement of the observation sequence $O = \{O_1 O_2 \cdots O_T\}$, via a feature analysis of the speech corresponding to the word; followed by calculation of model likelihoods for all possible models, $P(O|\lambda^v)$, $1 \le v \le V$; followed by selection of the word whose model likelihood is highest, i.e.,

$$v^* = \underset{1 \le v \le V}{\operatorname{argmax}} [P(O|\lambda^v)]. \tag{114}$$

The probability computation step is generally performed using the Viterbi algorithm (i.e., the maximum likelihood path is used) and requires on the order of $V \cdot N^2 \cdot T$ computations. For modest vocabulary sizes, e.g., $V = 100$ words, with an $N = 5$ state model, and $T = 40$ observations for the

[12]An excellent description of an isolated word, large vocabulary, speech recognizer based on sub-word units is given in the description of the IBM TANGORA system [50]. Another good reference which compares the effects of continuous and discrete densities using a 60 000 word vocabulary is [46].

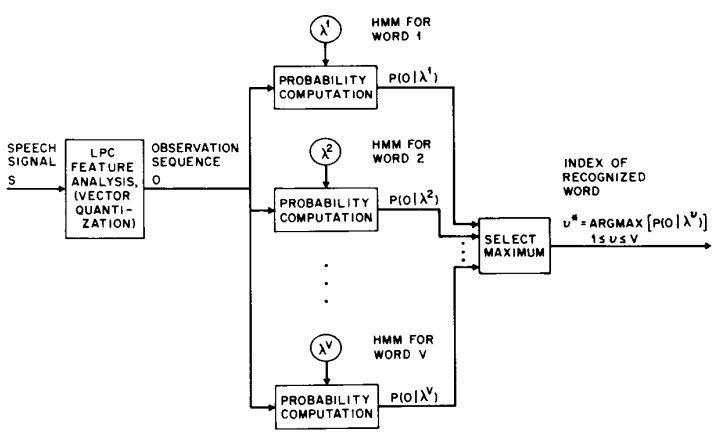

Fig. 12. Block diagram of an isolated word HMM recognizer.

Amazon / Zentian Limited
Exhibit 1029
Page 20

unknown word, a total of 10^5 computations is required for recognition (where each computation is a multiply, and add, and a calculation of observation density, $b(O)$). Clearly this amount of computation is modest as compared to the capabilities of most modern signal processor chips.

C. LPC Feature Analysis [51]–[54]

One way to obtain observation vectors O from speech samples s is to perform a front end spectral analysis. (We assume that we are processing only the speech samples corresponding to the spoken word—i.e., all background before and after the spoken word has been eliminated by an appropriate word detection algorithm.) The type of spectral analysis that is often used (and the one we will describe here) is called linear predictive coding (LPC), and a block diagram of the steps that are carried out is given in Fig. 13. The overall system is a block processing model in which a frame of N_A samples is processed and a vector of features O_t is computed. The steps in the processing are as follows:

1) Preemphasis: The digitized (at a 6.67 kHz rate for the examples to be discussed here) speech signal is processed by a first-order digital network in order to spectrally flatten the signal.

2) Blocking into Frames: Sections of N_A consecutive speech samples (we use $N_A = 300$ corresponding to 45 ms of signal) are used as a single frame. Consecutive frames are spaced M_A samples apart (we use $M_A = 100$ corresponding to 15-ms frame spacing, or 30-ms frame overlap).

3) Frame Windowing: Each frame is multiplied by an N_A-sample window (we use a Hamming window) $w(n)$ so as to minimize the adverse effects of chopping an N_A-sample section out of the running speech signal.

4) Autocorrelation Analysis: Each windowed set of speech samples is autocorrelated to give a set of $(p + 1)$ coefficients, where p is the order of the desired LPC analysis (we use $p = 8$).

5) LPC/Cepstral Analysis: For each frame, a vector of LPC coefficients is computed from the autocorrelation vector using a Levinson or a Durbin recursion method. An LPC derived cepstral vector is then computed up to the Qth component, where $Q > p$ and $Q = 12$ in the results to be described later in this section.

6) Cepstral Weighting: The Q-coefficient cepstral vector $c_\ell(m)$ at time frame ℓ is weighted by a window $W_c(m)$ of the form [55], [56]

$$W_c(m) = 1 + \frac{Q}{2} \sin\left(\frac{\pi m}{Q}\right), \qquad 1 \le m \le Q \tag{115}$$

to give

$$\hat{c}_\ell(m) = c_\ell(m) \cdot W_c(m). \tag{116}$$

7) Delta Cepstrum: The time derivative of the sequence of weighted cepstral vectors is approximated by a first-order orthogonal polynomial over a finite length window of $(2K + 1)$ frames, centered around the current vector [57], [58]. ($K = 2$ in the results to be presented; hence a 5 frame window is used for the computation of the derivative.) The cepstral derivative (i.e., the delta cepstrum vector) is computed as

$$\Delta\hat{c}_\ell(m) = \left[\sum_{k=-K}^{K} k\,\hat{c}_{\ell-k}(m)\right] \cdot G, \qquad 1 \le m \le Q \tag{117}$$

where G is a gain term chosen to make the variances of $\hat{c}_\ell(m)$ and $\Delta\hat{c}_\ell(m)$ equal. (A value of G of 0.375 was used.)

The observation vector $\mathbf{O}_\ell$ used for recognition and training is the concatenation of the weighted cepstral vector, and the corresponding weighted delta cepstrum vector, i.e.,

$$\mathbf{O}_\ell = \{\hat{c}_\ell(m), \Delta\hat{c}_\ell(m)\} \tag{118}$$

and consists of 24 coefficients per vector.

D. Vector Quantization [18], [39]

For the case in which we wish to use an HMM with a discrete observation symbol density, rather than the continuous vectors above, a vector quantizer (VQ) is required to map each continuous observation vector into a discrete codebook index. Once the codebook of vectors has been obtained, the mapping between continuous vectors and

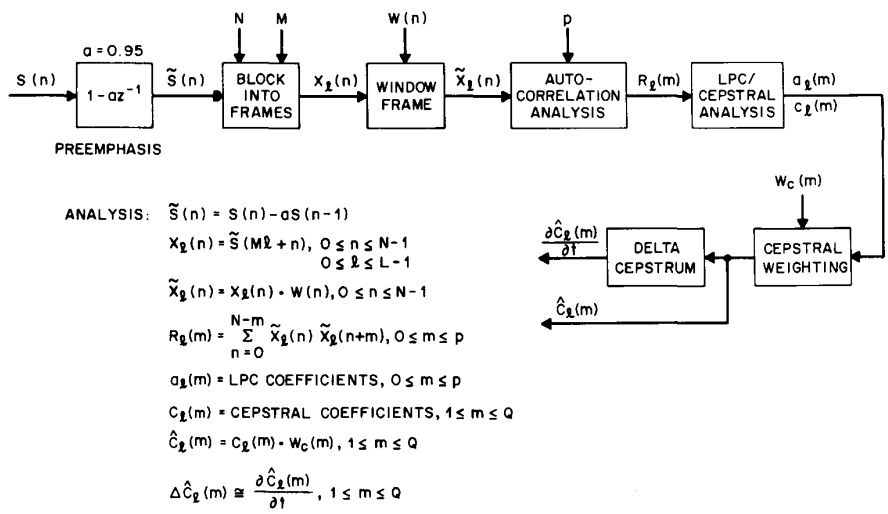

Fig. 13. Block diagram of the computations required in the front end feature analysis of the HMM recognizer.

Amazon / Zentian Limited
Exhibit 1029
Page 21

codebook indices becomes a simple nearest neighbor computation, i.e., the continuous vector is assigned the index of the nearest (in a spectral distance sense) codebook vector. Thus the major issue in VQ is the design of an appropriate codebook for quantization.

Fortunately a great deal of work has gone into devising an excellent iterative procedure for designing codebooks based on having a representative training sequence of vectors [18]. The procedure basically partitions the training vectors into M disjoint sets (where M is the size of the codebook), represents each such set by a single vector (v_m, $1 \leq m \leq M$), which is generally the centroid of the vectors in the training set assigned to the mth region, and then iteratively optimizes the partition and the codebook (i.e., the centroids of each partition). Associated with VQ is a distortion penalty since we are representing an entire region of the vector space by a single vector. Clearly it is advantageous to keep the distortion penalty as small as possible. However, this implies a large size codebook, and that leads to problems in implementing HMMs with a large number of parameters. Fig. 14 illustrates the tradeoff of quantization

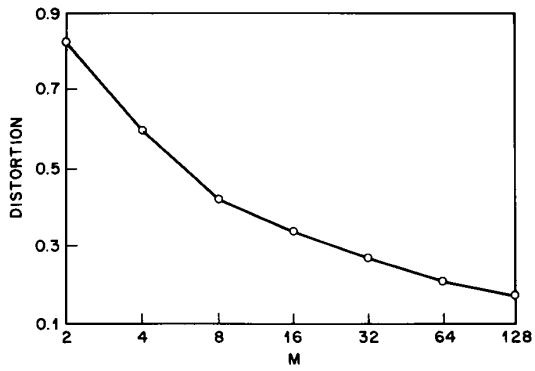

Fig. 14. Curve showing tradeoff of VQ average distortion as a function of the size of the VQ, M (shown of a log scale).

distortion versus M (on a log scale). Although the distortion steadily decreases as M increases, it can be seen from Fig. 14 that only small decreases in distortion accrue beyond a value of $M = 32$. Hence HMMs with codebook sizes of from $M = 32$ to 256 vectors have been used in speech recognition experiments using HMMs.

E. Choice of Model Parameters

We now come back to the issue that we have raised several times in this paper, namely how do we select the type of model, and how do we choose the parameters of the selected model. For isolated word recognition with a distinct HMM designed for each word in the vocabulary, it should be clear that a left-right model is more appropriate than an ergodic model, since we can then associate time with model states in a fairly straightforward manner. Furthermore we can envision the physical meaning of the model states as distinct sounds (e.g., phonemes, syllables) of the word being modeled.

The issue of the number of states to use in each word model leads to two schools of thought. One idea is to let the number of states correspond roughly to the number of sounds (phonemes) within the word—hence models with from 2 to 10 states would be appropriate. The other idea is to let the number of states correspond roughly to the average number of observations in a spoken version of the word, the so-called Bakis model [11]. In this manner each state corresponds to an observation interval—i.e., about 15 ms for the analysis we use. In the results to be described later in this section, we use the former approach. Furthermore we restrict each word model to have the same number of states; this implies that the models will work best when they represent words with the same number of sounds.

To illustrate the effect of varying the number of states in a word model, Fig. 15 shows a plot of average word error

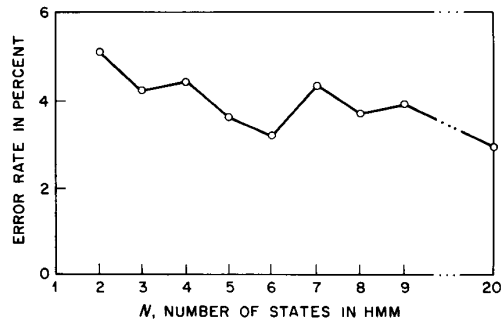

Fig. 15. Average word error rate (for a digits vocabulary) versus the number of states N in the HMM.

rate versus N, for the case of recognition of isolated digits (i.e., a 10-word vocabulary). It can be seen that the error is somewhat insensitive to N, achieving a local minimum at $N = 6$; however, differences in error rate for values of N close to 6 are small.

The next issue is the choice of observation vector and the way it is represented. As discussed in Sections VI-C and VI-D, we have considered LPC derived weighted cepstral coefficients and weighted cepstral derivatives or (for autoregressive HMMs) the autocorrelation of the LPC coefficients as the observation vectors for continuous models; for discrete symbol models we use a codebook to generate the discrete symbols. For the continuous models we use as many as $M = 9$ mixtures per state; for the discrete symbol models we use codebooks with as many as $M = 256$ codewords. Also, for the continuous models, we have found that it is preferable to use diagonal covariance matrices with several mixtures, rather than fewer mixtures with full covariance matrices. The reason for this is simple, namely the difficulty in performing reliable reestimation of the off-diagonal components of the covariance matrix from the necessarily limited training data. To illustrate the need for using mixture densities for modeling LPC observation vectors (i.e., eighth-order cepstral vectors with log energy appended as the ninth vector component), Fig. 16 shows a comparison of marginal distributions $b_j(O)|_{O=\cdots O_n\cdots}$ against a histogram of the actual observations within a state (as determined by a maximum likelihood segmentation of all the training observations into states). The observation vectors are ninth order, and the model density uses $M = 5$ mixtures. The covariance matrices are constrained to be diagonal for each individual mixture. The results of Fig. 16 are for the first model state of the word "zero." The need for values of $M > 1$ is clearly seen in the histogram of the

Amazon / Zentian Limited
Exhibit 1029
Page 22

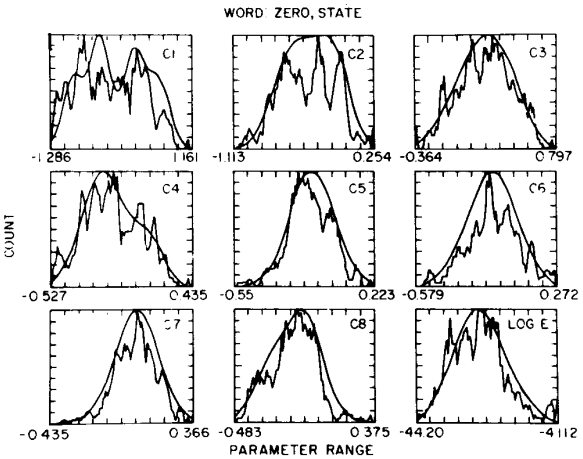

Fig. 16. Comparison of estimated density (jagged contour) and model density (smooth contour) for each of the nine components of the observation vector (eight cepstral components, one log energy component) for state 1 of the digit zero.

first parameter (the first cepstral component) which is inherently multimodal; similarly the second, fourth, and eight cepstral parameters show the need for more than a single Gaussian component to provide good fits to the empirical data. Many of the other parameters appear to be well fitted by a single Gaussian; in some cases, however, even $M = 5$ mixtures do not provide a sufficiently good fit.

Another experimentally verified fact about the HMM is that it is important to limit some of the parameter estimates in order to prevent them from becoming too small. For example, for the discrete symbol models, the constraint that $b_j(k)$ be greater than or equal to some minimum value ϵ is necessary to insure that even when the kth symbol never occurred in some state j in the training observation set, there is always a finite probability of its occurrence when scoring an unknown observation set. To illustrate this point, Fig. 17

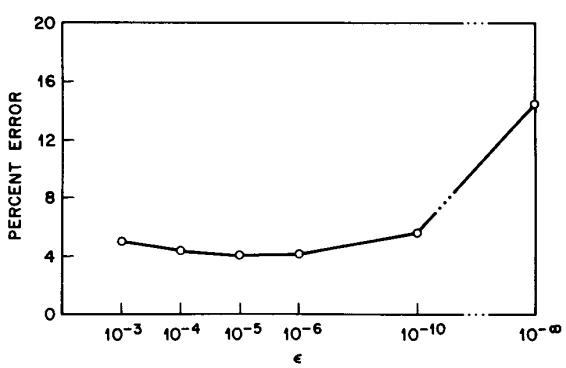

Fig. 17. Average word error rate as a function of the minimum discrete density value ϵ.

shows a curve of average word error rate versus the parameter ϵ (on a log scale) for a standard word recognition experiment. It can be seen that over a very broad range ($10^{-10} \leq \epsilon \leq 10^{-3}$) the average error rate remains at about a constant value; however, when ϵ is set to 0 (i.e., $10^{-\infty}$), then the error rate increases sharply. Similarly, for continuous densities it is important to constrain the mixture gains c_{jm} as well as the diagonal covariance coefficients $U_{jm}(r, r)$ to be greater than or equal to some minimum values (we use 10^{-4} in all cases).

F. Segmental k-Means Segmentation into States [42]

We stated earlier that good initial estimates of the parameters of the $b_j(O_t)$ densities were essential for rapid and proper convergence of the reestimation formulas. Hence a procedure for providing good initial estimates of these parameters was devised and is shown in Fig. 18. The training

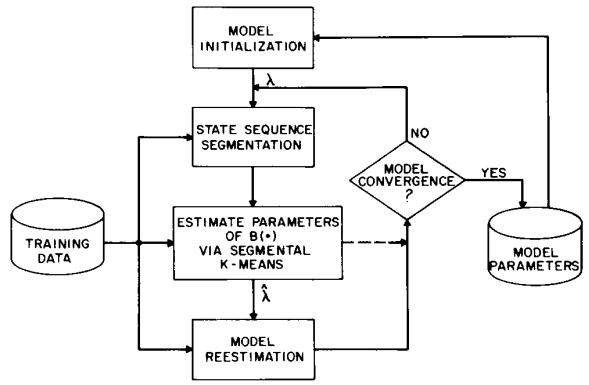

Fig. 18. The segmental k-means training procedure used to estimate parameter values for the optimal continuous mixture density fit to a finite number of observation sequences.

procedure is a variant on the well-known K-means iterative procedure for clustering data.

We assume we have a training set of observations (the same as is required for parameter reestimation), and an initial estimate of all model parameters. However, unlike the one required for reestimation, the initial model estimate can be chosen randomly, or on the basis of any available model which is appropriate to the data.

Following model initialization, the set of training observation sequences is segmented into states, based on the current model λ.[13] This segmentation is achieved by finding the optimum state sequence, via the Viterbi algorithm, and then backtracking along the optimal path. This procedure is illustrated in Fig. 19 which shows a log-energy plot, an accumulated log-likelihood plot, and a state segmentation for one occurrence of the word "six." It can be seen in Fig. 19 that the states correspond roughly to the sounds in the spoken word "six."

The result of segmenting each of the training sequences is, for each of the N states, a maximum likelihood estimate of the set of the observations that occur within each state S_j according to the current model. In the case where we are using discrete symbol densities, each of the observation vectors within a state is coded using the M-codeword codebook, and the updated estimate of the $b_j(k)$ parameters is

$\hat{b}_j(k)$ = number of vectors with codebook index k in state j divided by the number of vectors in state j.

[13]The current or initial model could be one created from another set of talkers, or it could be one created from a uniform segmentation of each word into states.

Amazon / Zentian Limited
Exhibit 1029
Page 23

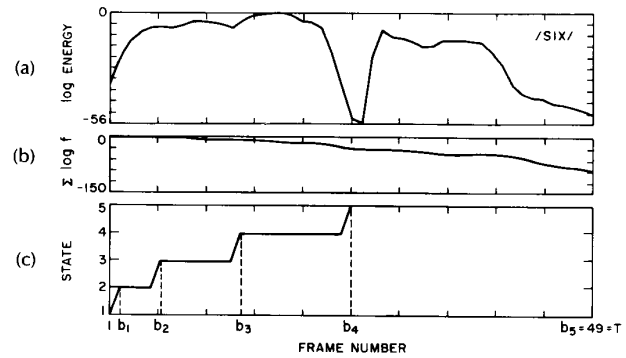

Fig. 19. Plots of: (a) log energy; (b) accumulated log likelihood; and (c) state assignment for one occurrence of the word "six."

In the case where we are using continuous observation densities, a segmental K-means procedure is used to cluster the observation vectors within each state S_j into a set of M clusters (using a Euclidean distortion measure), where each cluster represents one of the M mixtures of the $b_j(O_t)$ density. From the clustering, an updated set of model parameters is derived as follows:

$\hat{c}_{jm}$ = number of vectors classified in cluster m of state j divided by the number of vectors in state j

$\hat{\mu}_{jm}$ = sample mean of the vectors classified in cluster m of state j

$\hat{U}_{jm}$ = sample covariance matrix of the vectors classified in cluster m of state j.

Based on this state segmentation, updated estimates of the a_{ij} coefficients can be obtained by counting the number of transitions from state i to j and dividing it by the number of transitions from state i to any state (including itself).

An updated model $\hat{\lambda}$ is obtained from the new model parameters and the formal reestimation procedure is used to reestimate all model parameters. The resulting model is then compared to the previous model (by computing a distance score that reflects the statistical similarity of the HMMs). If the model distance score exceeds a threshold, then the old model λ is replaced by the new (reestimated) model $\bar{\lambda}$, and the overall training loop is repeated. If the model distance score falls below the threshold, then model convergence is assumed and the final model parameters are saved.

G. Incorporation of State Duration into the HMM

In Section IV-C we discussed the theoretically correct method of incorporating state duration information into the mechanics of the HMM. We also showed that the cost of including duration density was rather high; namely a D^2-fold increase in computation and a D-fold increase in storage. Using a value of $D = 25$ (as is required for word recognition), the cost of the increased computation tended to make the techniques not worth using. Thus the following alternative procedure was formulated for incorporating state duration information into the HMM.

For this alternative procedure, the state duration probability $p_j(d)$ was measured directly from the segmented training sequences used in the segmental K-means procedure of the previous section. Hence the estimates of $p_j(d)$ are strictly heuristic ones. A typical set of histograms of $p_j(d)$ for a 5-state model of the word "six" is shown in Fig. 20. (In this figure the histograms are plotted versus normalized duration (d/T), rather than absolute duration d.) It can be

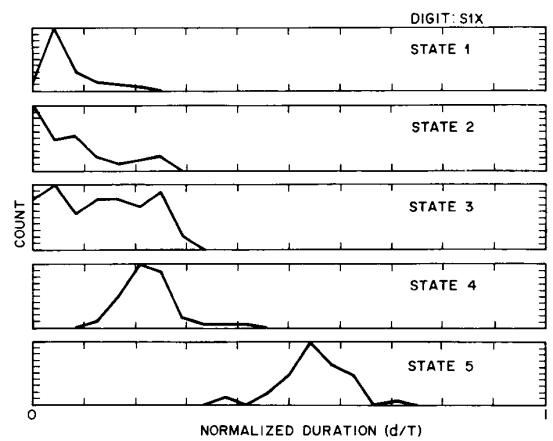

Fig. 20. Histograms of the normalized duration density for the five states of the digit "six."

seen from Fig. 20 that the first two states account for the initial /s/ in "six"; the third state accounts for the transition to the vowel /i/; the fourth state accounts for the vowel; and the fifth state accounts for the stop and the final /s/ sound.

The way in which the heuristic duration densities were used in the recognizer was as follows. First the normal Viterbi algorithm is used to give the best segmentation of the observation sequence of the unknown word into states via a backtracking procedure. The duration of each state is then measured from the state segmentation. A postprocessor then increments the log-likelihood score of the Viterbi algorithm, by the quantity

$$\log \hat{P}(q, O|\lambda) = \log P(q, O|\lambda) + \alpha_d \sum_{j=1}^{N} \log [p_j(d_j)] \quad (119)$$

where α_d is a scaling multiplier on the state duration scores, and d_j is the duration of state j along the optimal path as determined by the Viterbi algorithm. The incremental cost of the postprocessor for duration is essentially negligible, and experience has shown that recognition performance

Amazon / Zentian Limited
Exhibit 1029
Page 24

is essentially as good as that obtained using the theoretically correct duration model.

H. HMM Performance on Isolated Word Recognition

We conclude this section on isolated word recognition using HMMs by giving a set of performance results (in terms of average word error rate) on the task of recognizing isolated digits in a speaker independent manner. For this task, a training set consisting of 100 occurrences of each digit by 100 talkers (i.e., a single occurrence of each digit per talker) was used. Half the talkers were male; half female. For testing the algorithm, we used the initial training set, as well as three other independent test sets with the following characteristics:

- TS2: the same 100 talkers as were used in the training; 100 occurrences of each digit
- TS3: a new set of 100 talkers (50 male, 50 female); 100 occurrences of each digit
- TS4: another new set of 100 talkers (50 male, 50 female); 100 occurrences of each digit

The results of the recognition tests are given in Table 1. The recognizers are the following:

- LPC/DTW: Conventional template-based recognizer using dynamic time warping (DTW) alignment
- LPC/DTW/VQ: Conventional recognizer with vector quantization of the feature vectors (M = 64)
- HMM/VQ: HMM recognizer with M = 64 codebook
- HMM/CD: HMM recognizer using continuous density model with M = 5 mixtures per state
- HMM/AR: HMM recognizer using autoregressive observation density

Table 1 Average Digit Error Rates for Several Recognizers and Evaluation Sets

Recognizer Type	Evaluation Set: Original Training	TS2	TS3	TS4
LPC/DTW	0.1	0.2	2.0	1.1
LPC/DTW/VQ	—	3.5	—	—
HMM/VQ	—	3.7	—	—
HMM/CD	0	0.2	1.3	1.8
HMM/AR	0.3	1.8	3.4	4.1

It can be seen that, when using a VQ, the performance of the isolated word recognizer degrades in both the conventional and HMM modes. It can also be seen that the performances of the conventional template-based recognizer, and the HMM recognizer with a continuous density model are comparable. Finally Table 1 shows that the autoregressive density HMM gives poorer performance than the standard mixture density model.

VII. Connected Word Recognition Using HMMs [59]–[63]

A somewhat more complicated problem of speech recognition, to which HMMs have been successfully applied, is the problem of connected word recognition. The basic premise of connected word recognition is that the recognition is based on individual word models (as opposed to models of speech units smaller than words). The recognition problem (once the appropriate word models have been derived) is to find the optimum sequence (concatenation) of word models that best matches (in a maximum likelihood sense) an unknown connected word string. In this section we discuss one method (called the level building approach) for solving for such optimum sequences of word models. An alternative method for obtaining the optimum sequence of words is a frame (time) synchronous Viterbi search [31]. There are several practical advantages of the frame synchronous search (e.g., ease of real-time hardware implementation, ease of path pruning, etc.) but these do not affect the optimality of the two methods. For convenience, we restrict our discussion to the recognition of strings of connected digits.

A. Connected Digit Recognition from Word HMMs Using Level Building

A block diagram of the overall level building connected digit recognizer is given in Fig. 21. There are essentially three steps in the recognition process:

1) Spectral Analysis: The speech signal $s(n)$ is converted to either a set of LPC vectors or a set of cepstral and delta

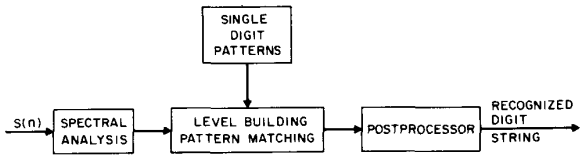

Fig. 21. Block diagram of level building, connected digit recognizer.

cepstral vectors. This defines the observation sequence $\boldsymbol{O}$ of the unknown connected digit string.

2) Level Building[14] *Pattern Matching:* The sequence of spectral vectors (the observations) of the unknown connected digit string is matched against the single word HMMs using a Viterbi scoring algorithm. The output of this process is a set of candidate digit strings, generally of different lengths (i.e., different number of digits per string), ordered by log probability scores.

3) Postprocessor: The candidate digit strings are subjected to further validity tests (e.g., duration), to eliminate unreasonable (unlikely) candidates. The postprocessor chooses the most likely digit string from the remaining (valid) candidate strings.

Individual digits are each characterized by an HMM of the type shown in Fig. 22. (Transitions between words are handled by a switch mode from the last state of one word model, to the first state of another word model, in the level building implementation.) The parameters of the HMMs used for characterizing digits are the following:

1) N = 5 or 8 states for digit models trained from observations of a single talker, and N = 8 or 10 states, for

[14]A level is a word position in a string. Hence a 5 digit string would have at least 5 level outputs, one for each digit in the string.

Amazon / Zentian Limited
Exhibit 1029
Page 25

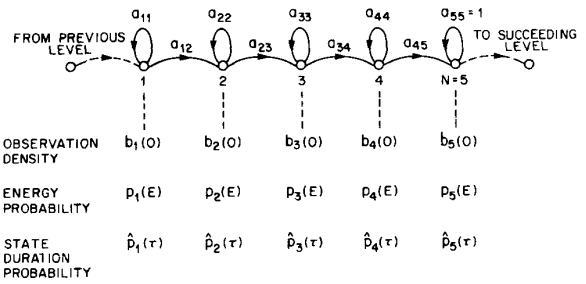

Fig. 22. HMM characterization of individual digits for connected digit recognition.

digit models trained from observations of more than a single talker.

2) Continuous observation mixture densities with $M = 3$ or 5 mixtures per state for single talker models and $M = 9$ mixtures per state for multiple talker models.
3) Energy probability $p_j(\epsilon)$ where ϵ_t is the dynamically normalized log energy of the frame of speech used to give observation vector $\boldsymbol{O}_t$, and $p_j(\cdot)$ is a discrete density of log energy values in state j. The density is derived empirically from the training data.
4) State duration density $p_j(d)$, $1 \le d \le D = 25$.

In addition to the observation density, log energy probability, and state duration density, each word HMM λ^v is also characterized by an overall word duration density $p_v(D)$ of the form

$$p_v(D) = \mathfrak{N}(\overline{D}_v, \sigma_v^2) \tag{120}$$

where $\overline{D}_v$ is the average duration for word v, σ_v^2 is the variance in duration for word v, and $\mathfrak{N}$ is the normal density.

B. Level Building on HMMs

The way in which level building is used on HMMs is illustrated in Fig. 23. If we denote the set of V word HMMs as $\lambda^v, 1 \le v \le V$, then to find the optimum sequence of HMMs that match $\boldsymbol{O}$ (i.e., maximize the likelihood), a sequence of Viterbi matches is performed. For each HMM λ^v, and at each level ℓ, we do a Viterbi match against $\boldsymbol{O}$, starting at frame (observation interval) 1 on level 1, and retain for each possible frame t the following:

1) $P_\ell^v(t)$, $1 \le t \le T$, the accumulated log probability to frame t, at level ℓ, for reference model λ^v, along the best path.
2) $F_\ell^v(t)$, $1 \le t \le T$, a backpointer indicating where the path started at the beginning of the level.

To compute $P_\ell^v(t)$, we need a local measure for the probability that observation $\boldsymbol{O}_t$, with log energy ϵ_t, occurred in state j of model λ^v. We use, as the observation density, the function

$$\hat{b}_j^v(\boldsymbol{O}_t) = b_j^v(\boldsymbol{O}_t) \cdot [p_j^v(\epsilon_t)]^{\gamma_\epsilon} \cdot K_1 \tag{121}$$

where γ_ϵ (set to 0.375) is a log energy scaling coefficient and K_1 is a normalization constant. The state transition coefficients enter the calculation of $P_\ell^v(t)$ via the dynamic programming optimization in determining the Viterbi path.

At the end of each level ℓ (where the level corresponds to word position within the string), a maximization over v

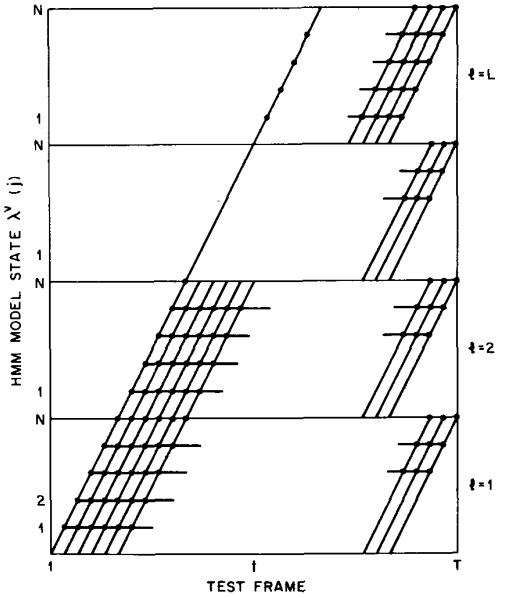

Fig. 23. Illustration of how HMMs are applied in the level building algorithm.

is performed to get the best model at each frame t as follows:

$$P_\ell^B(t) = \max_{1 \le v \le V} P_\ell^v(t), \qquad 1 \le t \le T \tag{122a}$$

$$W_\ell^B(t) = \operatorname*{argmax}_{1 \le v \le V} P_\ell^v(t), \qquad 1 \le t \le T \tag{122b}$$

$$F_\ell^B(t) = F_\ell^{W_\ell^B(t)}(t), \qquad 1 \le t \le T \tag{122c}$$

where $W_\ell^B(t)$ records the number of the word model which gave the best score at frame t, level ℓ, and $F_\ell^B(t)$ records the backpointer of the best word model.

Each new level begins with the initial best probability at the preceding frame on the preceding level and increments the Viterbi score by matching the word models beginning at the new initial frame. This process is repeated through a number of levels equivalent to the maximum expected number of digits in any string (e.g., typically 7).

At the end of each level, a best string of size ℓ words ($1 \le t \le L$) with probability $P_\ell^B(T)$ is obtained by backtracking using the backpointer array $F_\ell^B(t)$ to give the words in the string. The overall best string is the maximum of $P_\ell^B(T)$ over all possible levels ℓ.

C. Training the Word Models [59], [61]

The key to success in connected word recognition is to derive word models from representative connected word strings. We have found that although the formal reestimation procedures developed in this paper work well, they are costly in terms of computation, and equivalently good parameter estimates can be obtained using a segmental K-means procedure of the type discussed in Section VI. The only difference in the procedure, from the one discussed earlier, is that the training connected word strings are first segmented into individual digits, via a Viterbi alignment procedure, then each set of digits is segmented into states, and the vectors within each state are clustered into the best

Amazon / Zentian Limited
Exhibit 1029
Page 26

M cluster solution. The segmental K-means reestimation of the HMM parameters is about an order of magnitude faster than the Baum–Welch reestimation procedure, and all our experimentation indicates that the resulting parameter estimates are essentially identical in that the resulting HMMs have essentially the same likelihood values. As such, the segmental K-means procedure was used to give all the results presented later in this section.

D. Duration Modeling for Connected Digits

There are two forms of durational information used in scoring connected digit sequences, namely word duration and state duration. The way in which word duration information is incorporated into the model scoring is as follows. At the end of each level, for each frame t, the accumulated probability $P_\ell^B(t)$ is modified by determining the word duration $\tau_v(t)$ as

$$\tau_v(t) = t - F_\ell^B(t) + 1 \tag{123}$$

and then multiplying the accumulated probability by the word duration probability, i.e.,

$$\hat{P}_\ell^v(t) = P_\ell^v(t) \cdot [\mathfrak{N}(\tau_v(t), \overline{D}_v, \sigma_v^2)]^{\gamma_{wD}} \cdot K_2 \tag{124}$$

where γ_{wD} (set to 3.0) is a weighting factor on word durations, and K_2 is a normalization constant.

State duration probabilities are incorporated in a postprocessor. The level building recognizer provides multiple candidates at each level (by tracking multiple best scores at each frame of each level). Hence overall probability scores are obtained for R^L strings of length L digits, where R is the number of candidates per level (typically $R = 2$). Each of the R^L strings is backtracked to give both individual words and individual states within the words. For an L-word string, if we denote the duration of state j at level ℓ as $\Delta_\ell(j)$, then, for each possible string, the postprocessor multiplies the overall accumulated probability $P_L^B(T)$ by the state duration probabilities, giving

$$\hat{P}_L^B(T) = P_L^B(T) \cdot \prod_{\ell=1}^{L} \prod_{j=1}^{N} [p_j^{w(\ell)}(\Delta_\ell(j))]^{\gamma_{sD}} \cdot K_3 \tag{125}$$

where γ_{sD} (set to 0.75) is a weighting factor on state durations, $w(\ell)$ is the word at level ℓ, and K_3 is a normalization constant. The computation of (125) is performed on all R^L strings, and a reordered list of best strings is obtained. The incremental cost of the postprocessor computation is negligible compared to the computation to give $P_L^B(T)$, and its performance has been shown to be comparable to the performance of the internal duration models.

E. Performance of the Connected Digit HMM Recognizer

The HMM-based connected digit recognizer has been trained and tested in 3 modes:

1) Speaker trained using 50 talkers (25 male, 25 female) each of whom provided a training set of about 500 connected digit strings and an independent testing set of 500 digit strings.
2) Multispeaker in which the training sets from the 50 talkers above were merged into a single large training set, and the testing sets were similarly merged. In this case a set of 6 HMMs per digit was used, where each HMM was derived from a subset of the training utterances.
3) Speaker independent based on the TI training and testing databases. Both the training and testing sets had about 113 talkers (different ones were used in each set) and the talkers were divided into 22 dialectal groups. In this case a set of 4 HMMs per digit was used.

In each of the above databases there were variable length digit strings with from 1 to 7 digits per string.

The performance of the HMM connected digit recognizer, in these modes, is given in Table 2, where the entries in the table are average string error rates for cases in which the string length was unknown apriori (UL), and for cases in which the string length was known apriori (KL). Results are given both for the training set (from which the word models were derived), and for the independent test set.

Table 2 Performance of the HMM Connected Digit Recognizer in Three Modes

Mode	Training Set		Testing Set	
	UL	KL	UL	KL
Speaker trained (50 talkers)	0.39	0.16	0.78	0.35
Multispeaker (50 talkers)	1.74	0.98	2.85	1.65
Speaker independent (112/113 talkers)	1.24	0.36	2.94	1.75

VIII. HMMs for Large Vocabulary Speech Recognition [6]–[13], [31], [37], [38], [51], [64]–[66]

Although HMMs have been successfully applied to problems in isolated and connected word recognition, the anticipated payoff of the theory, to problems in speech recognition, is in its application to large vocabulary speech recognition in which the recognition of speech is performed from basic speech units smaller than words. The research in this area far outweights the research in any other area of speech processing and is far too extensive to discuss here. Instead, in this section we briefly outline the ideas of how HMMs have been applied to this problem.

In the most advanced systems (e.g., comparable to those under investigation at IBM, BBN, CMU and other places), the theory of HMMs has been applied to the representation of phoneme-like sub-words as HMMs; representation of words as HMMs; and representation of syntax as an HMM. To solve the speech recognition problem, a triply embedded network of HMMs must be used. This leads to an expanded network with an astronomical number of equivalent states; hence an alternative to the complete, exhaustive search procedure is required. Among the alternatives are the stack algorithm [7] and various forms of Viterbi beam searches [31]. These procedures have been shown to be capable of handling such large networks (e.g., 5000 words with an average word branching factor of 100) in an efficient and reliable manner. Details of these approaches are beyond the scope of this paper.

In another attempt to apply HMMs to continuous speech recognition, an ergodic HMM was used in which each state represented an acoustic-phonetic unit [47]. Hence about 40–50 states are required to represent all sounds of English. The model incorporated the variable duration feature in each state to account for the fact that vowel-like sounds

Amazon / Zentian Limited
Exhibit 1029
Page 27

have vastly different durational characteristics than consonant-like sounds. In this approach, lexical access was used in conjunction with a standard pronouncing dictionary to determine the best matching word sequence from the output of the sub-word HMM. Again the details of this recognition system are beyond the scope of this paper. The purpose of this brief discussion is to point out the vast potential of HMMs for characterizing the basic processes of speech production; hence their applicability to problems in large vocabulary speech recognition.

A. Limitations of HMMs

Although use of HMM technology has contributed greatly to recent advances in speech recognition, there are some inherent limitations of this type of statistical model for speech. A major limitation is the assumption that successive observations (frames of speech) are independent, and therefore the probability of a sequence of observations $P(O, O_2 \cdots O_T)$ can be written as a product of probabilities of individual observations, i.e.,

$$P(O, O_2 \cdots O_T) = \prod_{i=1}^{T} P(O_i).$$

Another limitation is the assumption that the distributions of individual observation parameters can be well represented as a mixture of Gaussian or autoregressive densities. Finally the Markov assumption itself, i.e., that the probability of being in a given state at time t only depends on the state at time $t-1$, is clearly inappropriate for speech sounds where dependencies often extend through several states. However, in spite of these limitations this type of statistical model has worked extremely well for certain types of speech recognition problems.

IX. Summary

In this paper we have attempted to present the theory of hidden Markov models from the simplest concepts (discrete Markov chains) to the most sophisticated models (variable duration, continuous density models). It has been our purpose to focus on physical explanations of the basic mathematics; hence we have avoided long, drawn out proofs and/or derivations of the key results, and concentrated primarily on trying to interpret the meaning of the math, and how it could be implemented in practice in real world systems. We have also attempted to illustrate some applications of the theory of HMMs to simple problems in speech recognition, and pointed out how the techniques could be (and have been) applied to more advanced speech recognition problems.

Acknowledgment

The author gratefully acknowledges the major contributions of several colleagues to the theory of HMMs in general, and to the presentation of this paper, in particular. A great debt is owed to Dr. J. Ferguson, Dr. A. Poritz, Dr. L. Liporace, Dr. A. Richter, and to Dr. F. Jelinek and the various members of the IBM group for introducing the speech world to the ideas behind HMMs. In addition Dr. S. Levinson, Dr. M. Sondhi, Dr. F. Juang, Dr. A. Dembo, and Dr. Y. Ephraim have contributed significantly to both the theory of HMMs as well as the author's perspective and knowledge as to how the theory is best applied to problems of speech recognition.

References

[1] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, vol. 37, pp. 1554–1563, 1966.

[2] L. E. Baum and J. A. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bull. Amer. Meteorol. Soc.*, vol. 73, pp. 360–363, 1967.

[3] L. E. Baum and G. R. Sell, "Growth functions for transformations on manifolds," *Pac. J. Math.*, vol. 27, no. 2, pp. 211–227, 1968.

[4] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164–171, 1970.

[5] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.

[6] J. K. Baker, "The dragon system—An overview," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-23, no. 1, pp. 24–29, Feb. 1975.

[7] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, pp. 675–685, 1969.

[8] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Trans. Informat. Theory*, vol. IT-21, pp. 404–411, 1975.

[9] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Informat. Theory*, vol. IT-21, pp. 250–256, 1975.

[10] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, pp. 532–536, Apr. 1976.

[11] R. Bakis, "Continuous speech word recognition via centisecond acoustic states," in *Proc. ASA Meeting* (Washington, DC), Apr. 1976.

[12] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Continuous speech recognition: Statistical methods," in *Handbook of Statistics, II*, P. R. Krishnaiad, Ed. Amsterdam, The Netherlands: North-Holland, 1982.

[13] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 179–190, 1983.

[14] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1035–1074, Apr. 1983.

[15] B. H. Juang, "On the hidden Markov model and dynamic time warping for speech recognition—A unified view," *AT&T Tech. J.*, vol. 63, no. 7, pp. 1213–1243, Sept. 1984.

[16] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, 1986.

[17] J. S. Bridle, "Stochastic models and template matching: Some important relationships between two apparently different techniques for automatic speech recognition," in *Proc. Inst. of Acoustics*, Autum Conf., pp. 1–8, Nov. 1984.

[18] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, no. 11, pp. 1551–1588, Nov. 1985.

[19] S. E. Levinson, "Structural methods in automatic speech recognition," *Proc. IEEE*, vol. 73, no. 11, pp. 1625–1650, Nov. 1985.

[20] A. W. Drake, "Discrete—state Markov processes," Chapter 5 in *Fundamentals of Applied Probability Theory*. New York, NY: McGraw-Hill, 1967.

[21] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Trans. Informat. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[22] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum like-

Amazon / Zentian Limited
Exhibit 1029
Page 28

lihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc.*, vol. 39, no. 1, pp. 1-38, 1977.

[24] L. A. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Informat. Theory*, vol. IT-28, no. 5, pp. 729-734, 1982.

[25] B. H. Juang, "Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains," *AT&T Tech. J.*, vol. 64, no. 6, pp. 1235-1249, July-Aug. 1985.

[26] B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Trans. Informat. Theory*, vol. IT-32, no. 2, pp. 307-309, Mar. 1986.

[27] A. B. Poritz, "Linear predictive hidden Markov models and the speech signal," in *Proc. ICASSP '82* (Paris, France), pp. 1291-1294, May 1982.

[28] B. H. Juang and L. R. Rabiner, "Mixture autoregressive hidden Markov models for speech signals," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-33, no. 6, pp. 1404-1413, Dec. 1985.

[29] M. J. Russell and R. K. Moore, "Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition," in *Proc. ICASSP '85* (Tampa, FL), pp. 5-8, Mar. 1985.

[30] S. E. Levinson, "Continuously variable duration hidden Markov models for automatic speech recognition," *Computer, Speech and Language*, vol. 1, no. 1, pp. 29-45, Mar. 1986.

[31] B. Lowerre and R. Reddy, "The HARPY speech understanding system," in *Trends in Speech Recognition*, W. Lea, Editor. Englewood Cliffs, NJ: Prentice-Hall, 1980, pp. 340-346.

[32] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. ICASSP '86* (Tokyo, Japan), pp. 49-52, Apr. 1986.

[33] Y. Ephraim, A. Dembo, and L. R. Rabiner, "A minimum discrimination information approach for hidden Markov modeling," in *Proc. ICASSP '87* (Dallas, TX), Apr. 1987.

[34] B. H. Juang and L. R. Rabiner, "A probabilistic distance measure for hidden Markov models," *AT&T Tech. J.*, vol. 64, no. 2, pp. 391-408, Feb. 1985.

[35] L. R. Rabiner, B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Some properties of continuous hidden Markov model representations," *AT&T Tech. J.*, vol. 64, no. 6, pp. 1251-1270, July-Aug. 1985.

[36] F. Jelinek and R. L. Mercer, "Interpolated estimation of Markov source parameters from sparse data," in *Pattern Recognition in Practice*, E. S. Gelesma and L. N. Kanal, Eds. Amsterdam, The Netherlands: North-Holland, 1980, pp. 381-397.

[37] R. Schwartz *et al.*, "Context-dependent modeling for acoustic-phonetic recognition of continuous speech," in *Conf. Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 1205-1208, Apr. 1985.

[38] K. F. Lee and H. W. Hon, "Large-vocabulary speaker-independent continuous speech recognition," in *Conf. Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 123-126, Apr. 1988.

[39] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1075-1105, Apr. 1983.

[40] ——, "On the use of hidden Markov models for speaker-independent recognition of isolated words from a medium-size vocabulary," *AT&T Tech. J.*, vol. 63, no. 4, pp. 627-642, Apr. 1984.

[41] R. Billi, "Vector quantization and Markov source models applied to speech recognition," in *Proc. ICASSP '82* (Paris, France), pp. 574-577, May 1982.

[42] L. R. Rabiner, B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Recognition of isolated digits using hidden Markov models with continuous mixture densities," *AT&T Tech. J.*, vol. 64, no. 6, pp. 1211-1222, July-Aug. 1986.

[43] A. B. Poritz and A. G. Richter, "Isolated word recognition," in *Proc. ICASSP '86* (Tokyo, Japan), pp. 705-708, Apr. 1986.

[44] R. P. Lippmann, E. A. Martin, and D. B. Paul, "Multistyle training for robust isolated word speech recognition," in *Proc. ICASSP '87* (Dallas, TX), pp. 705-708, Apr. 1987.

[45] D. B. Paul, "A speaker stress resistant HMM isolated word recognizer," in *Proc. ICASSP '87* (Dallas, TX), pp. 713-716, Apr. 1987.

[46] V. N. Gupta, M. Lennig and P. Mermelstein, "Integration of acoustic information in a large vocabulary word recognizer," in *Conf. Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 697-700, Apr. 1987.

[47] S. E. Levinson, "Continuous speech recognition by means of acoustic-phonetic classification obtained from a hidden Markov model," in *Proc. ICASSP '87* (Dallas TX), Apr. 1987.

[48] J. G. Wilpon, L. R. Rabiner, and T. Martin, "An improved word detection algorithm for telephone quality speech incorporating both syntactic and semantic constraints," *AT&T Bell Labs Tech. J.*, vol. 63, no. 3, pp. 479-498, Mar. 1984.

[49] J. G. Wilpon and L. R. Rabiner, "Application of hidden Markov models to automatic speech endpoint detection," *Computer Speech and Language*, vol. 2, no. 3/4, pp. 321-341, Sept./Dec. 1987.

[50] A. Averbuch *et al.*, "Experiments with the TANGORA 20,000 word speech recognizer," in *Conf. Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 701-704, Apr. 1987.

[51] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Am.*, vol. 50, pp. 637-655, 1971.

[52] F. I. Itakura and S. Saito, "Analysis-synthesis telephony based upon the maximum likelihood method," in *Proc. 6th Int. Congress on Acoustics* (Tokyo, Japan), pp. C17-20, 1968.

[53] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561-580, 1975.

[54] J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*. New York, NY: Springer-Verlag, 1976.

[55] Y. Tokhura, "A weighted cepstral distance measure for speech recognition," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-35, no. 10, pp. 1414-1422, Oct. 1987.

[56] B. H. Juang, L. R. Rabiner, and J. G. Wilpon, "On the use of bandpass liftering in speech recognition," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-35, no. 7, pp. 947-954, July 1987.

[57] S. Furui, "Speaker independent isolated word recognition based on dynamics emphasized cepstrum," *Trans. IECE of Japan*, vol. 69, no. 12, pp. 1310-1317, Dec. 1986.

[58] F. K. Soong and A. E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition," in *Proc. ICASSP '86* (Tokyo, Japan), pp. 877-880, Apr. 1986.

[59] L. R. Rabiner, J. G. Wilpon, and B. H. Juang, "A segmental *k*-means training procedure for connected word recognition," *AT&T Tech. J.*, vol. 65, no. 3, pp. 21-31, May-June 1986.

[60] L. R. Rabiner and S. E. Levinson, "A speaker-independent, syntax-directed, connected word recognition system based on hidden Markov models and level building," *IEEE Trans. Acoust. Speech Signal Processing*, vol. ASSP-33, no. 3, pp. 561-573, June 1985.

[61] L. R. Rabiner, J. G. Wilpon, and B. H. Juang, "A model-based connected digit recognition system using either hidden Markov models or templates," *Computer, Speech, and Language*, vol. 1, no. 2, pp. 167-197, Dec. 1986.

[62] H. Bourlard, Y. Kamp, H. Ney, and C. J. Wellekens, "Speaker-dependent connected speech recognition via dynamic programming and statistical methods," in *Speech and Speaker Recognition*, M. R. Schroeder, Ed. Basel, Switzerland: Karger, 1985, pp. 115-148.

[63] C. J. Wellekens, "Global connected digit recognition using Baum-Welch algorithm," in *Proc. ICASSP '86* (Tokyo, Japan), pp. 1081-1084, Apr. 1986.

[64] A. M. Derouault, "Context dependent phonetic Markov models for large vocabulary speech recognition," in *Proc. ICASSP '87* (Dallas, TX), Paper 10.1.1, pp. 360-363, Apr. 1987.

[65] B. Merialdo, "Speech recognition with very large size dictionary," in *Proc. ICASSP '87* (Dallas, TX), Paper 10.2.2, pp. 364-367, Apr. 1987.

[66] Y. L. Chow *et al.*, "BYBLOS: The BBN continuous speech recognition system," in *Proc. ICASSP '87* (Dallas, TX), Paper 3.7.1, pp. 89-92, Apr. 1987.

Amazon / Zentian Limited
Exhibit 1029
Page 29

Lawrence R. Rabiner (Fellow, IEEE) was born in Brooklyn, NY, on September 28, 1943. He received the S.B. and S.M. degrees, both in 1964, and the Ph.D. degree in electrical engineering, in 1967, all from the Massachusetts Institute of Technology, Cambridge, MA.

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 the Acoustical Society of America.

Amazon / Zentian Limited
Exhibit 1029
Page 30