Copyright © 1983 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 4, April 1983
Printed in U.S.A.

An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition

By S. E. LEVINSON, L. R. RABINER, and M. M. SONDHI

(Manuscript received September 10, 1982)

In this paper we present several of the salient theoretical and practical issues associated with modeling a speech signal as a probabilistic function of a (hidden) Markov chain. First we give a concise review of the literature with emphasis on the Baum-Welch algorithm. This is followed by a detailed discussion of three issues not treated in the literature: alternatives to the Baum-Welch algorithm; critical facets of the implementation of the algorithms, with emphasis on their numerical properties; and behavior of Markov models on certain artificial but realistic problems. Special attention is given to a particular class of Markov models, which we call "left-to-right" models. This class of models is especially appropriate for isolated word recognition. The results of the application of these methods to an isolated word, speaker-independent speech recognition experiment are given in a companion paper.

I. INTRODUCTION

It is generally agreed that information in the speech signal is encoded in the temporal variation of its short-duration power spectrum. To decode the signal, then, requires techniques for both estimation of power spectra and tracking their changes in time. This paper is concerned with the application of the theory of probabilistic functions of a (hidden) Markov chain to modeling the inherent nonstationarity of the speech signal for the purposes of automatic speech recognition (ASR).

The use of hidden Markov models for ASR was proposed by Baker[1,2] and, independently, by a group at IBM.[3-12] The theory on which their work rests is due to Baum et al.[13-17] Its first appearance in the literature occurred several years before Baker's studies and has since been explored in some detail.[18,19] Our previous work in ASR has used temporal alignment procedures based on dynamic programming techniques,[20] and we hoped that through studying the new (to us) body of material we could improve the performance and/or capabilities of our present ASR systems.

Our initial goal, therefore, was to understand the theory of hidden Markov models sufficiently well to enable us to implement a new ASR system that could be compared directly to our existing ones. We have, in fact, been able to accomplish that goal, and a description and the results of our experiments are reported in a companion paper.[21] In the course of our studies, we have collected and integrated a number of loosely related mathematical techniques pertinent to Markov modeling. We have also modified and adapted these techniques to the specific ASR problems we wished to study. Our purpose in writing this tutorial, then, is to present this synthesis in a way that will be enlightening to those not familiar with the theory of hidden Markov models. We also hope that this treatment will provide for a better understanding of our companion paper. Finally, we hope to make the presentation general enough so that the theory is seen to be applicable to more than the problem of ASR.

We shall proceed as follows. We begin by defining probabilistic functions of a (hidden) Markov[22] chain and then show how they may be used in a natural way to model the speech signal. Once this is done, our task is reduced to solving two specific and well-defined mathematical problems: (i) computing the parameters of a proposed model conditioned on a sequence of observations assumed to have been generated by the model, and (ii) calculating the probability that a given set of observations was produced by a particular model.

First we review the solution to these problems as originally given by Baum,[13] who treated them as problems in statistical estimation. Lest the problems be too narrowly construed, we look at them as problems of classical constrained optimization. This allows us to give a partial geometrical interpretation to the Baum-Welch algorithm and to relate it to other studies of the problem by Baum and Eagon[14] and Baum and Sell.[17] It also makes clear that there are other methods of solution available that may, in certain instances, have advantages over the Baum-Welch algorithm. Finally, we discuss the dynamic programming algorithm of Viterbi[23] as an alternative to the so-called "forward-backward" method of Baum[13] for computing the probability of a sequence of observations conditioned on a specific model.

The treatment of these problems in the existing literature, and as recounted here, can lull a prospective user of the theory into a false sense of security. The equations look innocuous enough but, in reality, they overlook two problems that, though uninteresting from a theoretical standpoint, are of great significance for a robust implementation. We believe it is worthwhile to address, first, a numerical problem arising from the evaluation of certain frequently occurring algebraic expressions, and then an experimental difficulty precipitated by the inescapable reality of finite training-data set.

The numerical problem arises because, regardless of the method of solution chosen, one is required to evaluate a product of stochastic matrices involving a number of factors proportional to the number of observations. In any real computer, this will ultimately result in underflow. Fortunately, the computation can be scaled using a technique that subsequently will be seen to have some very useful properties.

The problem of insufficient training data can be ameliorated by changing the constraints on the optimization problem. This can be simply and directly accomplished in the classical methods. We show that the Baum-Welch algorithm, too, can be modified to produce the same result. Both of these methods appear to be simpler in implementation than the technique proposed by Jelinek and Mercer.[9]

Finally, under the heading of implementational considerations, we discuss techniques for model averaging. These can be used both for block processing of observations in case one is subject to storage limitations, and for increasing model stability under some circumstances.

The speech recognition experiment that we had in mind was on a speaker-independent, isolated word recognition system with a small vocabulary. Oddly enough, this is a simpler task than those to which the theory had already been applied by Baker[1] and the IBM group.[3-12] Perhaps our choice of a manageable problem is responsible for the degree of success reported in the companion paper.[21] We determined that for our ASR task it is advantageous to use a particular kind of hidden Markov model, which we call a left-to-right model. In such a model there are strong temporal constraints on the Markov chain. First, any state, once left, cannot be later revisited. Second, there is a final absorbing state in which all observation sequences are assumed to terminate. These restrictions on the sequences affect both parameter estimation and probability computation procedures. We show how the Baum-Welch algorithm, the Viterbi algorithm, and the classical methods can all be adapted for use with left-to-right models.

We conclude our presentation with several sample solutions to some artificial but nontrivial problems that illustrate concepts treated in the foregoing discussion.

II. A REVIEW OF THE THEORY

A probabilistic function of a (hidden) Markov chain is a stochastic process generated by two interrelated mechanisms, an underlying Markov chain having a finite number of states, and a set of random functions, one of which is associated with each state. At discrete instants of time, the process is assumed to be in some state and an observation is generated by the random function corresponding to the current state. The underlying Markov chain then changes states according to its transition probability matrix. The observer sees only the output of the random functions associated with each state and cannot directly observe the states of the underlying Markov chain; hence the term hidden Markov model.

In principle, the underlying Markov chain may be of any order and the outputs from its states may be multivariate random processes having some continuous joint probability density function. In this discussion, however, we shall restrict ourselves to consideration of Markov chains of order one, i.e., those for which the probability of transition to any state depends only upon that state and its predecessor. We shall also limit the discussion to processes whose observations are drawn from a discrete finite alphabet according to discrete probability distribution functions associated with the states.

It is quite natural to think of the speech signal as being generated by such a process. We can imagine the vocal tract as being in one of a finite number of articulatory configurations or states. In each state a short (in time) signal is produced that has one of a finite number of prototypical spectra depending, of course, on the state. Thus, the power spectra of short intervals of the speech signal are determined solely by the current state of the model, while the variation of the spectral composition of the signal with time is governed predominantly by the probabilistic state transition law of the underlying Markov chain. For speech signals derived from a small vocabulary of isolated words, the model is reasonably faithful. The foregoing is, of course, an oversimplification intended only for the purpose of motivating the following theoretical discussion.

Let us say that the underlying Markov chain has N states q_1, q_2, $\cdots$, q_N and the observations are drawn from an alphabet, V, of M prototypical spectra, $v_1, v_2, \cdots, v_M$. The underlying Markov chain can then be specified in terms of an initial state distribution vector $\pi' = (\pi_1, \pi_2, \cdots, \pi_N)$ and a state transition matrix, $A = [a_{ij}]$ $1 \le i, j \le N$. Here, π_i is the probability of q_i at some arbitrary time, $t = 0$, and a_{ij} is the probability of transiting to state q_j given current state, q_i, that is $a_{ij} = \text{prob}(q_j \text{ at } t + 1 | q_i \text{ at } t)$.

The random processes associated with the states can be collectively represented by another stochastic matrix $B = [b_{jk}]$ in which for $1 \le j \le N$ and $1 \le k \le M$, b_{jk} is the probability of observing symbol v_k given current state q_j. We denote this as $b_{jk} = \text{prob}(v_k \text{ at } t | q_j \text{ at } t)$. Thus a hidden Markov model, **M**, is identified with the parameter set (π, A, B).

To use hidden Markov models to perform speech recognition we must solve two specific problems: observation sequence probability estimation, which will be used for classification of an utterance; and model parameter estimation, which will serve as a procedure for training models for each vocabulary word. Both problems proceed from a sequence, **O**, of observations $O_1O_2 \cdots O_T$ where each O_t for $1 \le t \le T$ is some $v_k \in \mathbf{V}$.

Our particular classification problem is as follows. We wish to recognize utterances known to be selected from some vocabulary, W, of words $w_1, w_2, \cdots, w_V$. We are given an observation sequence, **O**, derived from the utterance of some unknown $w_i \in W$ and a set of V models $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_V$. We must compute $P_i = \text{prob}(\mathbf{O} \mid \mathbf{M}_i)$ for $1 \le i \le V$. We will then classify the unknown utterance as w_i iff $P_i \ge P_j$ for $1 \le j \le V$.

The training problem is simply that of determining the models $\mathbf{M}_i = (\pi_i, A_i, B_i)$ for $1 \le i \le V$ given training sequences $\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \cdots, \mathbf{O}^{(V)}$, where $\mathbf{O}^{(i)}$ is known to have been derived from an utterance of word w_i for $1 \le i \le V$.

One could, in principle, compute $\text{prob}(\mathbf{O} \mid \mathbf{M})$ by computing the joint probability prob $(\mathbf{O}, \mathrm{s} \mid \mathbf{M})$ for each state sequence, s, of length T, and summing over all state sequences. Obviously this is computationally intractable. Fortunately, however, there is an efficient method for computing P. Let us define the function $\alpha_t(i)$ for $1 \le t \le T$ as $\text{prob}(O_1O_2 \cdots O_t$ and q_i at $t \mid \mathbf{M})$. According to the definition $\alpha_1(i) = \pi_i b_i(O_1)$, where $b_i(O_1)$ is understood to mean b_{ik} iff $O_1 \equiv v_k$; then we have the following recursive relationship for the "forward probabilities":

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

Similarly, we define another function, $\beta_t(j) = \text{prob}(O_{t+1}O_{t+2} \cdots O_T \mid q_j$ at t and $\mathbf{M})$. We set $\beta_T(j) = 1 \ \forall \ j$ and then use the backward recursion

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad T-1 \ge t \ge 1 \tag{2}$$

to compute the "backward probabilities."

The two functions can be used to compute P according to

$$P = \text{Prob}(\mathbf{O} \mid \mathbf{M}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{3}$$

for any t such that $1 \le t \le T-1$. Equations (1) to (3) are from Baum[13] and are sometimes referred to as the "forward-backward" algorithm.

Setting $t = T - 1$ in (3) gives

$$P = \sum_{i=1}^{N} \alpha_T(i) \tag{4}$$

so that P can be computed from the forward probabilities alone. A similar formula for P can be obtained from the backward probabilities by setting $t = 1$. These and several other formulas in this section may be compactly written in matrix notation (see Appendix A). For instance,

$$P = \pi' B_1 A B_2 A \cdots A B_T \mathbf{1}, \tag{5}$$

where $\mathbf{1}$ is the N-vector $(1, 1, 1, \cdots, 1)'$ and

$$B_t = \begin{pmatrix} b_1(O_t) & & & \\ & b_2(O_t) & & \mathbf{O} \\ & & \ddots & \\ \mathbf{O} & & & b_N(O_t) \end{pmatrix} \tag{6}$$

for $1 \le t \le T$. From (5) it is clear that P is a homogeneous polynomial in the π_i, a_{ij}, and b_{jk}. Any of eqs. (3) through (5) may be used to solve the classification problem. The forward and backward probabilities will prove to be convenient in other contexts.

When we compute P with the forward-backward algorithm, we are including the probabilities of all possible state sequences that may have generated $\mathbf{O}$. Alternatively, we may define P as the maximum over all state sequences $i = i_0, i_1, \cdots i_T$ of the joint probability $P(\mathbf{O}, i)$. This distinguished state sequence and the corresponding probability of the observation sequence can be simultaneously computed by means of the Viterbi[23] algorithm. This dynamic programming technique proceeds as follows: Let $\phi_1(i) = \pi_i b_i(O_1)$ for $1 \le i \le N$. Then we can perform the following recursion for $2 \le t \le T$ and $1 \le j \le N$

$$\phi_t(j) = \max_{1 \le i \le N} [\phi_{t-1}(i) a_{ij}] b_j(O_t) \tag{7a}$$

and

$$\psi_t(j) = i^*, \tag{7b}$$

where i^* is a choice of an index i that maximizes $\phi_{t-1}(i)$.

The result is that $P = \max\limits_{1 \le i \le N} [\phi_T(i)]$. Also the maximum likelihood state sequence can be recovered from ψ as follows. Let $q_T = i^*$, where i^* maximizes P. Then for $T \ge t \ge 2$, $q_{t-1} = \psi_t(q_t)$. If one only wishes to compute P, the linked list, ψ, need not be maintained as in (7b). Only the recursion (7a) is required.

The problem of training a model, unfortunately, does not have such a simple solution. In fact, given any finite observation sequence as training data, we cannot optimally train the model. We can, however, choose π, A, and B such that prob(**O**|**M**) is locally maximized. For an asymptotic analysis of the training problem the reader should consult Baum and Petrie.[15]

We can use the forward and backward probabilities to formulate a solution to the problem of training by parameter estimation. Given some estimates of the parameter values we can compute, for example, that the expected number of transitions, γ_{ij}, from q_i to q_j, conditioned on the observation sequence is just

$$\gamma_{ij} = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j). \tag{8}$$

Then, the expected number of transitions, γ_i out of q_i, given **O**, is

$$\gamma_i = \sum_{j=1}^{N} \gamma_{ij} = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i), \tag{9}$$

the last step of which is based on (2). The ratio γ_{ij}/γ_i is then an estimate of the probability of state q_j, given that the previous state was q_i. This ratio may be taken as a new estimate, $\bar{a}_{ij}$, of a_{ij}. That is,

$$\bar{a}_{ij} = \frac{\gamma_{ij}}{\gamma_i} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)}. \tag{10}$$

Similarly, we can make a new estimate of b_{jk} as the frequency of occurrence of v_k in q_j relative to the frequency of occurrence of any symbol in state q_j. Stated in terms of the forward and backward probabilities we have

$$\bar{b}_{jk} = \frac{\sum_{t \ni O_t = v_k} \alpha_t(j) \beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)}. \tag{11}$$

Finally, new values of the initial state probabilities may be obtained from

$$\bar{\pi}_i = \frac{1}{P} \alpha_1(i) \beta_1(i). \tag{12}$$

As we shall see in the next section, the reestimates are guaranteed to increase P, except at a critical point.

2.1 Proof of the reestimation formula

The reestimation formulas (10), (11), and (12) are instances of the Baum-Welch algorithm. Although it is not at all obvious, each application of the formulas is guaranteed to increase P except if we are at a critical point of P, in which case the new estimates will be identical to their current values. Several proofs of this rather surprising fact are given in the literature.[13,17] Because we shall need to modify it later to cope with the finite sample size problem, we shall briefly sketch Baum's proof[13] here. The proof is based on the following two lemmas:

Lemma 1: Let u_i, $i = 1, \cdots, S$ be positive real numbers, and let v_i, $i = 1, \cdots, S$ be nonnegative real numbers such that $\sum_i v_i > 0$. Then from the concavity of the log function it follows that

$$\ln\left(\frac{\sum v_i}{\sum u_i}\right) = \ln\left[\sum_i \left(\frac{u_i}{\sum_k u_k}\right)\cdot\frac{v_i}{u_i}\right]$$

$$\geq \sum_i \frac{u_i}{\sum_k u_k} \ln\left(\frac{v_i}{u_i}\right)$$

$$= \frac{1}{\sum_k u_k}\left[\sum_i (u_i \ln v_i - u_i \ln u_i)\right]. \tag{13}$$

Here every summation is from 1 to S.

Lemma 2: If $c_i > 0$ $i = 1, \cdots, N$, then subject to the constraint $\sum_i x_i = 1$, the function

$$F(\mathbf{x}) = \sum_i c_i \ln x_i \tag{14}$$

attains its unique global maximum when

$$x_i = \frac{c_i}{\sum_i c_i}. \tag{15}$$

The proof follows from the observation that by the Lagrange method

$$\frac{\partial}{\partial x_i}\left[F(\mathbf{x}) - \lambda \sum_i x_i\right] = \frac{c_i}{x_i} - \lambda = 0. \tag{16}$$

Multiplying by x_i and summing over i gives $\lambda = \sum_i c_i$, hence the result.

Now in Lemma 1, let S be the number of state sequences of length T. For the ith sequence let u_i be the joint probability

$$u_i = \text{Prob[state sequence } i, \text{ observation } \mathbf{O} \,|\, \text{model } \mathbf{M}]$$
$$= P(i, \mathbf{O}|\mathbf{M}).$$

Let v_i be the same joint probability conditioned on model $\bar{\mathbf{M}}$. Then

$$\sum_i u_i = p(\mathbf{O}|\mathbf{M}) \triangleq P(\mathbf{M})$$

$$\sum_i v_i = p(\mathbf{O}|\bar{\mathbf{M}}) \triangleq P(\bar{\mathbf{M}}) \tag{17}$$

and the lemma gives

$$\ln \frac{P(\bar{\mathbf{M}})}{P(\mathbf{M})} \geq \frac{1}{P(\mathbf{M})} \cdot [Q(\mathbf{M}, \bar{\mathbf{M}}) - Q(\mathbf{M}, \mathbf{M})], \tag{18}$$

where

$$Q(\mathbf{M}, \bar{\mathbf{M}}) \triangleq \sum_i u_i \ln v_i. \tag{19}$$

Thus, if we can find a model $\bar{\mathbf{M}}$ that makes the right-hand side of (18) positive, we have a way of improving the model **M**. Clearly, the largest guaranteed improvement by this method results for $\bar{\mathbf{M}}$, which maximizes $Q(\mathbf{M}, \bar{\mathbf{M}})$ [and hence maximizes the right-hand side of (18)]. The remarkable fact proven in Ref. 13 is that $Q(\mathbf{M}, \bar{\mathbf{M}})$ attains its maximum when $\bar{\mathbf{M}}$ is related to **M** by the reestimation formulas (10) through (12). To show this let the sth-state sequence be $s_0, s_1, \cdots, s_T$, and the given observation sequence be $O_{k_1}, \cdots, O_{k_T}$. Then

$$\ln v_s = \ln p(s, \mathbf{O} | \bar{\mathbf{M}}) = \ln \bar{\pi}_{s_0} + \sum_{t=0}^{T-1} \ln \bar{a}_{s_t s_{t+1}} + \sum_{t=0}^{T-1} \ln \bar{b}_{s_{t+1}}(O_{t+1}). \tag{20}$$

Substituting this in (19) for $Q(\mathbf{M}, \bar{\mathbf{M}})$, and regrouping terms in the summations according to state transitions and observed symbols, it can be seen that

$$Q(\mathbf{M}, \bar{\mathbf{M}}) = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \ln \bar{a}_{ij} + \sum_{j=1}^{N} \sum_{k=1}^{M} d_{jk} \ln \bar{b}_j(k) + \sum_{i=1}^{N} e_i \ln \bar{\pi}_i. \tag{21}$$

Here

$$c_{ij} = \sum_{s=1}^{S} p(s, \mathbf{O} | \mathbf{M}) n_{ij}(s) \tag{22a}$$

$$d_{jk} = \sum_{s=1}^{S} p(s, \mathbf{O} | \mathbf{M}) m_{jk}(s) \tag{22b}$$

$$e_i = \sum_{s=1}^{S} p(s, \mathbf{O} | \mathbf{M}) r_i(s), \tag{22c}$$

and for the sth-state sequence

$n_{ij}(s)$ = number of transitions from state q_i to q_j

$m_{jk}(s)$ = number of times symbol k is generated in state q_j

$r_i(s)$ = 1 if initial state is q_i

= 0 otherwise.

Thus, c_{ij}, d_{jk}, and e_i are the expected values of n_{ij}, m_{jk}, r_i, respectively, based on model **M**.

The expression (21) is now a sum of $2N + 1$ independent expressions of the type maximized in Lemma 2. Hence, $Q(\mathbf{M}, \bar{\mathbf{M}})$ is maximized if

$$\bar{a}_{ij} = \frac{c_{ij}}{\sum_j c_{ij}} \tag{23a}$$

$$\bar{b}_j(k) = \frac{d_{jk}}{\sum_k d_{jk}} \tag{23b}$$

$$\bar{\pi}_i = \frac{e_i}{\sum_i e_i}. \tag{23c}$$

These are recognized as the reestimation formulas.

2.2 Solution by optimization techniques

Lest the reader be led to believe that the reestimation formulas are peculiar to stochastic processes, we shall examine them briefly from several different points of view. Note that the reestimation formulas update the model in such a way that the constraints

$$\sum_{i=1}^{N} \pi_i = 1 \tag{24a}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \text{for} \quad 1 \le i \le N \tag{24b}$$

and

$$\sum_{k=1}^{M} b_{jk} = 1 \quad \text{for} \quad 1 \le j \le N \tag{24c}$$

are automatically satisfied at each iteration. The constraints are, of course, required to make the hidden Markov model well defined. It is thus natural to look at the training problem as a problem of constrained optimization of P and, at least formally, solve it by the classical method of Lagrange multipliers. For simplicity, we shall restrict the discussion to optimization with respect to A. Let Q be the Lagrangian of P with respect to the constraints (24b). We see that

$$Q = P + \sum_{i=1}^{N} \lambda_i \left(\sum_{j=1}^{N} a_{ij} - 1 \right), \tag{25}$$

where the λ_i are the as yet undetermined Lagrange multipliers.

At a critical point of P on the interior of the manifold defined by (24a through c), it will be the case that for $1 \le i, j \le N$

$$\frac{\partial Q}{\partial a_{ij}} = \frac{\partial P}{\partial a_{ij}} + \lambda_i = 0. \tag{26}$$

Multiplying (26) by a_{ij} and summing over j we get

$$\sum_{j=1}^{N} a_{ij} \frac{\partial P}{\partial a_{ij}} = -\left[\sum_{j=1}^{N} a_{ij}\right] \lambda_i = -\lambda_i = \frac{\partial P}{\partial a_{ij}}, \tag{27}$$

where the right-hand side of (27) follows from substituting (24b) for the sum of a_{ij} and then replacing λ_i according to (26). From (27) it may be seen that P is maximized when

$$a_{ij} = \frac{a_{ij} \dfrac{\partial P}{\partial a_{ij}}}{\displaystyle\sum_{k=1}^{N} a_{ik} \frac{\partial P}{\partial a_{ik}}}. \tag{28}$$

A similar argument can be made for the π and B parameters.

While it is true that solving (28) for a_{ij} is analytically intractable, it can be used to provide some useful insights into the Baum-Welch reestimation formulas and alternatives to them for solving the training problem. Let us begin by computing $\partial P/\partial a_{ij}$ by differentiating (3), according to the formula for differentiating a product,

$$\frac{\partial P}{\partial a_{ij}} = \sum_{t=1}^{T-1} \alpha_t(i) b_j(O_{t+1}) \beta_{t+1}(j). \tag{29}$$

Substituting the right-hand side of (29) for $\partial P/\partial a_{ij}$ in (28), we get

$$a_{ij} = \frac{\displaystyle\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\displaystyle\sum_{j=1}^{N} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}. \tag{30}$$

Then changing the order of summation in the denominator of (30) and substituting in the right-hand side of (2) we get

$$a_{ij} = \frac{\displaystyle\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\displaystyle\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)}. \tag{31}$$

The right-hand side of (31) is thus seen to be identical to that of the reestimation formula (10). Thus, at a critical point, the reestimation formula (10) solves the equations (31). Similarly, if we differentiate (3) with respect to π_i and b_{jk} we get

$$\begin{aligned} \frac{\partial P}{\partial \pi_i} &= \sum_{j=1}^{N} b_i(O_1) a_{ij} b_j(O_2) \beta_2(j) \\ &= b_i(O_1) \beta_1(i) \end{aligned} \tag{32}$$

and

$$\frac{\partial P}{\partial b_{jk}} = \sum_{t \ni O_t = v_k} \sum_{i=1}^{N} \alpha_t(i) a_{ij} \beta_{t+1}(j) + \delta(O_1, v_k) \pi_j \beta_1(j), \tag{33}$$

respectively. In (33) δ is understood to be the Kronecker δ function.

By substituting (32) and (33) into their respective analogs of (28), we obtain the reestimation formulas (12) and (11), respectively, at a critical point. Thus it appears that the reestimation formulas may have more general applications than might appear from their statistical motivation.

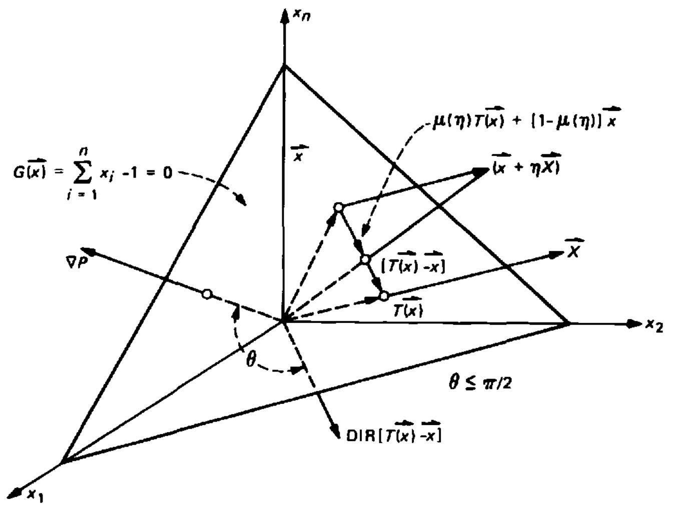
Equation (28) suggests that we define a transformation, **T**, of the parameter space onto itself as

$$\mathbf{T}(x)_{ij} = \frac{x_{ij} \dfrac{\partial P}{\partial x_{ij}}}{\sum_{k=1}^{N} x_{ik} \dfrac{\partial P}{\partial x_{ik}}}, \tag{34}$$

where $\mathbf{T}(\mathbf{x})_{ij}$ is understood to mean the i, jth coordinate of the image of $\mathbf{x}$ under **T**. The parameter space is restricted to be the manifold such that $x_{ij} \geq 0$ for $1 \leq i, j \leq N$ and $\sum_{j=1}^{N} x_{ij} = 1$ for $1 \leq i \leq N$. Thus, the reestimation formulas (10), (11), and (12) are a special case of the transformation (34), with P a particular homogeneous polynomial in the x_{ij} having positive coefficients. Here the x_{ij} include the π_i, the a_{ij}, and the b_{jk}. Baum and Eagon[14] have shown that for any such polynomial $P[\mathbf{T}(\mathbf{x})] > P(\mathbf{x})$ except if $\mathbf{x}$ is a critical point of P. Thus the transformation, **T**, is appropriately called a growth transformation. The conditions under which **T** is a growth transformation were relaxed by Baum and Sell[17] to include all polynomials with positive coefficients. They further proved that P increases monotonically on the segment from $\mathbf{x}$ to $\mathbf{T}(\mathbf{x})$. Specifically, they showed that $P[\eta\mathbf{T}(\mathbf{x}) + (1 - \eta)\mathbf{x}] \geq P(\mathbf{x})$ for $0 \leq \eta \leq 1$. Other properties of the transformation (34) have been explored by Passman[18] and Stebe.[19] There may be still less restrictive general criteria on P for **T** to be a growth transformation.

We can give $\mathbf{T}(\mathbf{x})$ a simple geometric interpretation. For the purposes of this discussion we shall restrict ourselves to $\mathbf{x} \in \mathbf{R}^N$, $x_i \geq 0$ for $1 \leq i \leq N$, and the single constraint $G(\mathbf{x}) = \sum_{i=1}^{N} x_i - 1 = 0$. We do so without loss of generality, since constraints such as those of (24a, b, and c) are disjoint, i.e., no pair of constraints has any common variables. As shown in Fig. 1, given any $\mathbf{x}$ satisfying $G(\mathbf{x}) = 0$, $\mathbf{T}(\mathbf{x})$ is the intersection of the vector $\mathbf{X}$, or its extension, with the hyperplane $\sum_{i=1}^{N} x_i - 1 = 0$, where $\mathbf{X}$ has components $x_i \dfrac{\partial P}{\partial x_i}$ for $1 \leq \text{i} \leq N$.

This may be shown by observing that a line in the direction of $\mathbf{X}$ passing through the origin has the equation $\mathbf{y} = r\mathbf{X}$, where r is a non-



x_n
$G(\vec{x}) = \sum_{i=1}^{n} x_i - 1 = 0$
$\vec{x}$
$\mu(\eta)\overrightarrow{T(x)} + [1-\mu(\eta)]\vec{x}$
$(\vec{x} + \eta\vec{X})$
∇P
$[\overrightarrow{T(x)} - \vec{x}]$
$\vec{X}$
$\overrightarrow{T(x)}$
x_2
θ
$\theta \le \pi/2$
$\mathrm{DIR}[\overrightarrow{T(x)} - \vec{x}]$
x_1

Fig. 1—Geometrical relationship of the quantities involved in the reestimation formulas.

negative scalar. Component-wise this is equivalent to

$$y_i = r x_i \frac{\partial P}{\partial x_i} \quad \text{for} \quad 1 \le i \le N. \tag{35}$$

We can find that r for which $\mathbf{y}$ intersects the hyperplane $G(\mathbf{x}) = 0$ by summing over i. Thus

$$\sum_{i=1}^{N} y_i = r \sum_{i=1}^{N} x_i \frac{\partial P}{\partial x_i} = 1, \tag{36}$$

since $\mathbf{y}$ lies on the hyperplane $G(\mathbf{x}) = 0$. Rearranging (36) we have

$$r = \frac{1}{\sum_{i=1}^{N} x_i \frac{\partial P}{\partial x_i}} \tag{37}$$

and

$$y_i = \frac{x_i \frac{\partial P}{\partial x_i}}{\sum_{j=1}^{N} x_j \frac{\partial P}{\partial x_j}}. \tag{38}$$

Furthermore, as also shown in Fig. 1, the vector $[\mathbf{T}(\mathbf{x}) - \mathbf{x}]$ is the set of intersections of the vector $(\mathbf{x} + \eta\mathbf{X})$ with the hyperplane $G(\mathbf{x}) = 0$ for $0 \le \eta \le +\infty$ with $\mathbf{T}(\mathbf{x})$ corresponding to $\eta = +\infty$ and $\mathbf{x}$ to $\eta = 0$.

Finally, in view of the result of Baum and Sell, quoted above, the vector $\mathbf{T}(\mathbf{x}) - \mathbf{x}$ must also have a positive projection on ∇P. This, too, is easily seen. If P is a polynomial with positive coefficients, then $\partial P/\partial x_i \geq 0$ for $1 \leq i \leq N$. From the definition of $\mathbf{T}$ it is clear that

$$\mathbf{T}(\mathbf{x})_i \geq x_i \quad \textit{iff} \quad \frac{\partial P}{\partial x_i} \geq \sum_{j=1}^{N} x_j \frac{\partial P}{\partial x_j} = r, \tag{39}$$

where r is some constant. Then it must be true that

$$\sum_{i=1}^{N} [\mathbf{T}(\mathbf{x})_i - x_i] \left(\frac{\partial P}{\partial x_i} - r\right) \geq 0 \tag{40}$$

since both factors in each summand are of the same sign. Rearranging (40) we have

$$\sum_{i=1}^{N} [\mathbf{T}(\mathbf{x})_i - x_i] \frac{\partial P}{\partial x_i} \geq r \sum_{i=1}^{N} [\mathbf{T}(\mathbf{x})_i - x_i] = 0. \tag{41}$$

The right-hand side is zero since $\sum_{i=1}^{N} \mathbf{T}(\mathbf{x})_i = \sum_{i=1}^{N} x_i = 1$. Thus $[\mathbf{T}(\mathbf{x}) - \mathbf{x}] \cdot \nabla P \geq 0$, proving that a step of the transformation has a positive projection along the gradient of P.

This merely guarantees that we can move an infinitesimal amount in the direction of $[\mathbf{T}(\mathbf{x}) - \mathbf{x}]$ while increasing P. The theorem of Baum and Eagon, however, guarantees much more, namely that we can take a finite step and be assured of increasing P. We may, in fact, be able to continue past $\mathbf{T}(\mathbf{x})$ while still increasing P. We are unable, at present, to give a geometrical interpretation of this fact.

While the reestimation formulas provide an elegant method for maximizing P, their success depends critically on the constraint set (24a, b, and c). As we will suggest later, in some cases there may be advantages in using classical optimization methods.

The principle of the classical methods is to search along the projection of ∇P on the constraint space, G, for a local maximum. The method of Rosen,[24] for example, uses only ∇P and a crude search strategy. The method of Davidon is one of many quasi-Newton techniques that uses the Fletcher-Powell[25] approximation to the inverse of the Hessian of P and an exact line search with adaptive step size. There are many collections of general purpose subroutines for constrained optimization that can be used to solve the training problem. We have successfully used a version of the Davidon procedure from the Harwell Subroutine Library.[26] However, for the constraints that π, A, and B be stochastic, the computation can be greatly simplified.

We illustrate this by outlining the gradient search algorithm for the case where P is a function of the variables $x_1, \cdots, x_N$ subject to the constraints $x_i \geq 0$ for $1 \leq i \leq N$ and $\sum_{i=1}^{N} x_i - 1 = 0$. For convenience we will call the last constraint G_1, and the inequality constraints on $x_1, \cdots, x_N$ as $G_2, \cdots, G_{N+1}$, respectively.

An initial starting point $\mathbf{x}$ is chosen and the "active" constraints identified. For our case G_1 is always active. For $i > 1$, G_i is active if $x_{i-1} = 0$. Let G_{n_j}, $j = 1, \cdots, \ell$ be the active constraints (with $n_1 = 1$) at the initial point. Let $Q = P + \sum_{j=1}^{\ell} \lambda_j G_{n_j}$. Then according to the Kuhn-Tucker theorem,[27] the Lagrangian multipliers, λ_j, are determined by demanding that ∇Q be orthogonal to ∇G_{n_j} for $1 \le j \le \ell$. Now

$$\nabla Q = \nabla P + \sum_{j=1}^{\ell} \lambda_j \nabla G_{n_j}$$

$$= \nabla P + \Gamma\lambda, \tag{42}$$

where Γ is the $N \times \ell$ matrix with $\Gamma_{ij} = (\nabla G_{n_j})_i = \partial G_{n_j}/\partial x_i$, and λ is the vector with components λ_j for $j = 1, \cdots, \ell$. Thus the Kuhn-Tucker requirement is equivalent to

$$\Gamma' \nabla Q = 0 \tag{43}$$

or, from (42),

$$\lambda = -(\Gamma'\Gamma)^{-1}\Gamma'\nabla P. \tag{44}$$

For our special constraints we have

$$\Gamma_{i1} = 1 \quad \text{for} \quad 1 \le i \le N \tag{45}$$

and, for $j \neq 1$

$$\Gamma_{ij} = \begin{cases} 1 & \text{if} \quad i = n_j - 1 \\ 0 & \text{otherwise.} \end{cases} \tag{46}$$

With Γ defined this way

$$\Gamma'\Gamma = \begin{pmatrix} N & 1 & 1 & \cdots & 1 \\ 1 & 1 & & & \\ 1 & & 1 & \mathbf{O} & \\ \vdots & \mathbf{O} & & \ddots & \\ 1 & & & & 1 \end{pmatrix} \tag{47}$$

and $(\Gamma'\Gamma)^{-1}$ may be shown to be

$$(\Gamma'\Gamma)^{-1} = \frac{1}{N - \ell + 1} \begin{pmatrix} 1 & -1 & -1 & \cdots & -1 \\ -1 & N-\ell & 1 & \cdots & 1 \\ -1 & 1 & N-\ell & 1 & 1 \\ \vdots & & & \ddots & 1 \\ -1 & 1 & 1 & & N-\ell \end{pmatrix}. \tag{48}$$

Substituting (48) into (44) gives λ. When this λ is substituted back into (42), it turns out that the resulting vector ∇Q can be computed by the following simple steps:

(*i*) Compute ∇P and let S be the sum of all components of ∇P *except* $(\nabla P)_{n_j - 1}$, $j = 2, \cdots, \ell$.

(*ii*) Then

$$\begin{aligned}(\nabla Q)_i &= 0 \qquad i = n_j, \qquad j = 2, \cdots, \ell \\ &= (\nabla P)_i - \frac{S}{N - \ell + 1} \qquad \text{otherwise.}\end{aligned}$$

Finally, the values of P are searched along the line

$$\mathbf{x}(\eta) = \mathbf{x} + \eta \frac{\nabla Q}{\|\nabla Q\|} \tag{49}$$

for a maximum with respect to η. The procedure is repeated at this new point.

In applying this technique to the actual training problem, there will be $2N + 1$ stochasticity constraints analogous to G_1 and a corresponding number of positivity constraints analogous to $G_2, G_3, \cdots G_{N+1}$. In this case we have the option of treating all the parameters and their associated constraints together, or we may divide them into disjoint subsets and determine search directions for each subset independently.

Notice that this derivation does not require P to be of any special form. This may prove to be an advantage since the Baum-Welch algorithm is not applicable to all P. Furthermore, the constraints may be changed. Although, as we shall see later, the Baum-Welch algorithm can be somewhat generalized in this respect, it does not generalize to work with arbitrary linear constraints.

III. CONSIDERATIONS FOR IMPLEMENTATION

From the foregoing discussion it might appear that solutions to the problems of hidden Markov modeling can be obtained by straightforward translation of the relevant formulas into computer programs. Unfortunately, for all but the most trivial problems, the naive implementation will not succeed for two principal reasons. First, any of the methods of solution presented here for either the classification or the training problem require evaluation of $\alpha_t(i)$ and $\beta_t(i)$ for $1 \leq t \leq T$ and $1 \leq i \leq N$. From the recursive formulas for these quantities, (1) and (2), it is clear that as $T \rightarrow \infty$, $\alpha_T(i) \rightarrow 0$, and $\beta_1(i) \rightarrow 0$ in exponential fashion. In practice, the number of observations necessary to adequately train a model and/or compute its probability will result in underflow on any real computer if (1) and (2) are evaluated directly. Fortunately, there is a method for scaling these computations that not only solves the underflow problem but also greatly simplifies several other calculations.

The second problem is more serious, more subtle, and admits of a less gratifying, though still effective, solution. Baum and Petrie[15] have shown that the maximum likelihood estimates of the parameters of a hidden Markov process are consistent estimates (converge to the true values as $T \rightarrow \infty$) of the parameters. The practical implication of the theorem is that, in training, one should use as many observations as possible which, as we have noted, make scaling necessary. In reality, of course, the observation sequence will always be finite. Then the following situation can arise. Suppose a given training sequence of length T results in $b_{jk} = 0$. (It is, in fact, possible for a local maximum of P to lie on a boundary of the parameter manifold.) Suppose further that we are subsequently asked to compute the probability that a new observation sequence was generated by our model. Even if the new sequence was actually generated by the model, it can be such that $\alpha_{t-1}(i)a_{ij}$ is nonzero for only one value of j and that $O_t = v_k$, whence $\alpha_t(j) = 0$ and the probability of the observation then becomes zero. This phenomenon is fatal to a classification task; yet, the smaller T is, the more likely is its occurrence. Jelinek and Mercer[9] have dealt with this problem in a slightly different context. Here, we offer the much simpler solution of constraining the parameter values so that $x_{ij} \geq \epsilon_{ij} > 0$.

Finally, in this section we discuss the related problem of model stability. Baum and Eagon[14] note that successive applications of the reestimation formulas converge to a connected component of the local maximum set of P. In case there are only a finite number of such extrema, the point of convergence is unique to within a renaming of the states. The component of the local maximum set to which the iteration converges as well as which of the $N!$ labelings of the states is determined by the initial estimates of the parameters. If we wish to average several models resulting from several different starting points to achieve model stability, we must be able to match the states of models whose states are permuted. We have devised a solution to this problem based on a minimum-weight bipartite matching algorithm.[28]

3.1 Scaling

The principle on which we base our scaling is to multiply $\alpha_t(i)$ by some scaling coefficient independent of i so that it remains within the dynamic range of the computer for $1 \leq t \leq T$. We propose to perform a similar operation on $\beta_t(i)$ and then, at the end of the computation, remove the total effect of the scaling.

We illustrate the procedure for (10), the reestimation formula for the state transition probabilities. Let $\alpha_t(i)$ be computed according to (1) and then be multiplied by a scaling coefficient, c_t, where say,

$$c_t = \left[\sum_{i=1}^{N} \alpha_t(i) \right]^{-1} \tag{50}$$

so that $\sum_{i=1}^{N} c_t\alpha_t(i) = 1$ for $1 \leq t \leq T$. Then, as we compute $\beta_t(i)$ from (2), we form the product $c_t\beta_t(i)$ for $T \geq t \geq 1$ and $1 \leq i \leq N$. In terms of the scaled forward and backward probabilities, the right-hand side of (10) becomes

$$\frac{\sum_{t=1}^{T-1} C_t\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)D_{t+1}}{\sum_{t=1}^{T-1}\sum_{\ell=1}^{N} C_t\alpha_t(i)a_{i\ell}b_\ell(O_{t+1})\beta_{t+1}(\ell)D_{t+1}}, \tag{51}$$

where

$$C_t = \prod_{\tau=1}^{t} c_\tau \tag{52}$$

and

$$D_t = \prod_{\tau=t}^{T} c_\tau .$$

This results from the individual scale factors being multiplied together as we perform the recursions of (1) and (2).

Now note that each summand in both the numerator and the denominator has the coefficient $C_tD_{t+1} = \prod_{\tau=1}^{T} c_\tau$. These coefficients can be factored out and canceled so that (51) has the correct value $\bar{a}_{ij}$ as specified by (10). The reader can verify that this technique may be equally well applied to the reestimation formulas (11) and (12). It should also be obvious that, in practice, the scaling operation need not be performed at every observation time. One can use any scaling interval for which underflow does not occur. In this case, the scale factors corresponding to values of t within any interval are set to unity.

While the above described scaling technique leaves the reestimation formulas invariant, (3) and (4) are still useless for computing P. However, log P can be recovered from the scale factors as follows. Assume that we compute c_t according to (50) for $t = 1, 2, \cdots T$. Then

$$C_T \sum_{i=1}^{N} \alpha_T(i) = 1 \tag{53}$$

and from (53) it is obvious that $C_T = 1/P$. Thus, from (52) we have

$$\prod_{t=1}^{T} c_t = \frac{1}{P}. \tag{54}$$

The product of the individual scale factors cannot be evaluated but we can compute

$$\log P = -\sum_{t=1}^{T} \log c_t. \tag{55}$$

If one chooses to use the Viterbi algorithm for classification, then log P can be computed directly from π, A, and B without regard for the scale factors. Initially, we let $\phi_1(i) = \log[\pi_i b_i(O_1)]$ and then modify (7a) so that

$$\phi_t(j) = \max_{1\le i\le N}[\phi_{t-1}(i) + \log a_{ij}] + \log[b_j(O_t)]. \tag{56}$$

In this case $\log P = \max_{1\le i\le N}[\phi_T(i)]$.

If the parameters of the model are to be computed by means of classical optimization techniques, we can make the computation better conditioned numerically by maximizing $\log P$ rather than P. The scaling method of (50) makes this straightforward.

First note that if we are to maximize $\log P$, then we will need the partial derivatives of $\log P$ with respect to the parameters of the model. So, for example, we will need

$$\frac{\partial}{\partial a_{ij}}(\log P) = \frac{1}{P}\frac{\partial P}{\partial a_{ij}} = C_T\frac{\partial P}{\partial a_{ij}}. \tag{57}$$

Substituting the right-hand side of (29) for $\partial P/\partial a_{ij}$ in the right-hand side of (57) yields

$$\begin{aligned}\frac{\partial}{\partial a_{ij}}(\log P) &= C_T\sum_{t=1}^{T-1}\alpha_t(i)b_j(O_{t+1})\beta_{t+1}(j)\\ &= \sum_{t=1}^{T-1}C_t\alpha_t(i)b_j(O_{t+1})\beta_{t+1}(j)D_{t+1}\\ &= \sum_{t=1}^{T-1}\left(\prod_{\tau=1}^{t}c_\tau\right)\alpha_t(i)b_j(O_{t+1})\beta_{t+1}(j)\left(\prod_{\tau=t+1}^{T}c_\tau\right).\end{aligned} \tag{58}$$

So that if we evaluate (29) formally, using not the true values of the forward and backward probabilities but the scaled values, then we will have the correct value of the partial derivatives of $\log P$ with respect to the transition probabilities. A similar argument can be made for the other parameters of the model and, thus, the scaling method of (50) provides a means for the direct evaluation of $\nabla(\log P)$, which is required for the classical optimization algorithms. Later we shall see that the combination of maximizing $\log P$ and this scaling technique simplifies the solution of the left-to-right Markov modeling problem as well.

3.2 Finite training sets

We now turn our attention to solving the problems created by finite-training-set size. As we noted earlier, the effect of this problem is that observation sequences generated by a putative model will have zero probability conditioned on the model parameters. Since the cause of the difficulty is the assignment of zero to some parameters, usually one or more symbol probabilities, it is reasonable to try to solve the problem by constraining the parameters to be positive.

We can maximize P subject to the new constraints $a_{ij} \geq \epsilon > 0$; $b_{jk} \geq \epsilon > 0$, most easily using the classical methods. In fact, the algorithm described earlier based on the Kuhn-Tucker theorem is unchanged except that the procedure for determining the active constraints is based on ϵ rather than zero.

While the Lagrangian methods are perfectly adequate, it is also possible to build the new constraints into the Baum-Welch algorithm. We can show how this is done by making a slight modification to the proof of the algorithm given earlier (Section 2.1). Recall that the proof of the Baum-Welch algorithm was based on maximization of $2N + 1$ expressions of the type maximized in Lemma 2, eq. (14). Since these expressions involve disjoint sets of variables chosen from A, B, π, it suffices to consider any one of the maximizations. In fact, it suffices to show how Lemma 2 gets modified. Thus we wish now to maximize

$$F(\mathbf{x}) = \sum_i c_i \ln x_i \tag{59}$$

subject to the constraints

$$\sum_i x_i = 1 \tag{60a}$$

and

$$x_i \geq \epsilon, \qquad i = 1, \cdots N. \tag{60b}$$

(From the following discussion it will be obvious that a trivial generalization allows ϵ to depend on i.)

Now without the inequality constraints (60b), Lemma 2 showed that $F(\mathbf{x})$ attains its unique global maximum when $x_i = c_i/\sum_i c_i$. Suppose now that this global maximum occurs outside the region specified by the inequality constraints (60b). Specifically, let

$$\bar{x}_i = \frac{c_i}{\sum_{j=1}^{N} c_j} \geq \epsilon \quad \text{for} \quad i = 1, \cdots, N - \ell \tag{61a}$$

$$< \epsilon \quad \text{for} \quad i = N - \ell + 1, \cdots, N. \tag{61b}$$

From the concavity of $F(\mathbf{x})$ it follows that the maximum, subject to the inequality constraints, must occur somewhere on the boundary specified by the violated constraints (61b). Now it is easily shown that if $\bar{x}_i$ for some $i > N - \ell$ is replaced by ϵ, then the global maximum over the rest of the variables occurs at values *lower* than those given above. From this we conclude that we must set

$$\bar{x}_i = \epsilon \quad \text{for} \quad i > N - \ell \tag{62}$$

and maximize

$$\tilde{F}(\mathbf{x}) = \sum_{i=1}^{N-\ell} c_i \ln x_i \tag{63}$$

subject to the constraint $\sum_{i=1}^{N-\ell} x_i = 1 - \ell\epsilon$. But this, analogously to Lemma 2, occurs when

$$\bar{x}_i = (1 - \ell\epsilon) \frac{c_i}{\sum_{j=1}^{N-\ell} c_j} \qquad i \le N - \ell. \tag{64}$$

If these new values of $\bar{x}_i$ satisfy the constraints, we are done. If one or more become lower than ϵ, they too must be set equal to ϵ, and ℓ augmented appropriately.

Thus the modified Baum-Welch algorithm is as follows. Suppose we wish to constrain $b_{jk} \ge \epsilon$ for $1 \le j \le N$ and $1 \le k \le M$. We first evaluate B using the reestimation formulas. Assume that some set of the parameters in the jth row of B violates the constraint so that $b_{jk_i} < \epsilon$ for $1 \le i \le \ell$. Then set $\bar{b}_{jk_i} = \epsilon$ for $1 \le i \le \ell$ and readjust the remaining parameters according to (64) so that

$$\bar{b}_{jk} = (1 - \ell\epsilon) \frac{b_{jk}}{\sum_{i=1}^{N-\ell} b_{ji}} \qquad \forall k \notin \{k_i \,|\, 1 \le i \le \ell\}. \tag{65}$$

After performing the operation of (65) for each row of B, the resulting $\bar{B}$ is the optimal update with respect to the desired constraints. The method can be extended to include the state transition matrix if so desired. There is no advantage to treating π in the same manner since, for any single observation sequence, $\bar{\pi}$ will always be a unit vector with exactly one nonzero component. In any case, (65) may be applied at each iteration of the reestimation formulas, or once as a post-processing stage after the Baum-Welch algorithm has converged.

3.3 Combining models

The final implementational issue that we shall consider in this section is that of combining models for improved stability. There are several circumstances under which it may be desirable to combine several models into one. In spectral estimation, for example, to compute a long-term average spectrum of a stationary signal, it may be convenient to average a number of spectra computed over shorter intervals. It seems quite natural to apply similar block-processing techniques to the Markov modeling problem if the source is assumed to be ergodic. We may, for example, divide a long sequence of observations into contiguous subsequences, estimate model parameters for each subsequence, and combine the results.

Whether or not the source is ergodic, we may still attempt to increase the robustness of our model by averaging the parameter estimates derived from multiple initial values and/or independent observation sequences.

In any case, the difficulty that will be encountered is that even if there are finitely many isolated local maxima of P, they are only unique to within a renaming of the states. For two different observation sequences, q_i and q_j may be topologically equivalent, but $i \neq j$. We might try to avoid this problem by using the final parameter values for one observation sequence as the initial values for the next in hopes that this will restrict the search to a neighborhood of a single local maximum. This method, unfortunately, is not reliable. A better approach is that of finding a renaming of the states that minimizes, in some sense, the difference between two models.

Suppose $\mathbf{b}_j$ and $\tilde{\mathbf{b}}_j$, $1 \le j \le N$ are, respectively, the rows of two estimates of B. Let $p(j)$ be a permutation of the state index, j, and let $d(\cdot,\cdot)$ be some distance metric. Then we seek the permutation, p, of $(q_1, q_2, \cdots, q_N)$ such that

$$D = \sum_{j=1}^{N} d[\mathbf{b}_j, \tilde{\mathbf{b}}_{p(j)}] \tag{66}$$

is minimized. The naive solution is to try all possible $N!$ permutations and select the best one in the sense of (66). However, for $N > 10$ the computation becomes intractable. The problem can be brought within reach, however, by transforming it into a minimum-weight bipartite-graph-matching problem on $2N$ vertices. In the literature on combinatorial optimization (see, e.g., Ref. 28), several algorithms are available for accomplishing such a match in a number of operations that grow as N^3. In Appendix B, we describe one such algorithm based on an outline provided to us by R. E. Tarjan.

IV. LEFT-TO-RIGHT HIDDEN MARKOV MODELS

For the purposes of isolated word recognition, it is useful to consider a special class of absorbing Markov chains that leads to what we call left-to-right models. These models have the following properties:

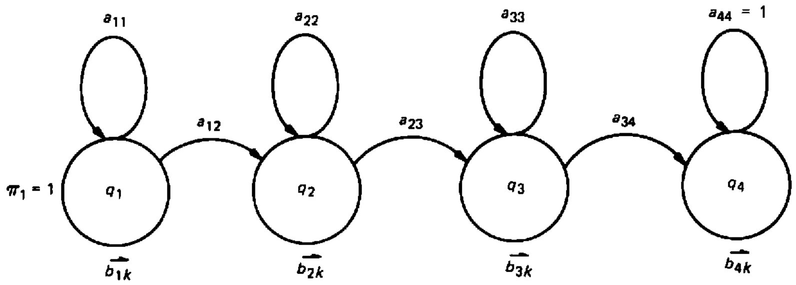
(*i*) The first observation is produced while the Markov chain is in a distinguished state called the starting state, designated q_1.

(*ii*) The last observation is generated while the Markov chain is in a distinguished state called the final or absorbing state, designated q_N.

(*iii*) Once the Markov chain leaves a state, that state cannot be revisited at a later time.

The simplest form of a left-to-right model is shown in Fig. 2, from which the origin of the term left-to-right becomes clear.

In this section we shall consider two problems associated with these



a_{11}
a_{22}
a_{33}
$a_{44} = 1$
a_{12}
a_{23}
a_{34}
$\pi_1 = 1$
q_1
q_2
q_3
q_4
$\overrightarrow{b_{1k}}$
$\overrightarrow{b_{2k}}$
$\overrightarrow{b_{3k}}$
$\overrightarrow{b_{4k}}$

Fig. 2—The simplest form of left-to-right model.

special hidden Markov models. Note that a single, long-observation sequence is useless for training such models, because once the state q_N is reached, the rest of the sequence provides no further information about earlier states. The appropriate training data for such a model are a *set* of observation sequences obtained by several starts in state q_1. In the case of isolated word recognition, for instance, several independent utterances of the same word provide such a set. We wish, therefore to modify the training algorithm to handle such training data. We also wish to compute the probability that a single given observation sequence, $O_1, O_2 \cdots, O_T$, was produced by the model, with the assumption that O_1 was produced in state q_1 and O_T in state q_N. The three conditions mentioned above can be satisfied as follows:

Condition (i) will be satisfied if we set $\pi = (1, 0, \cdots, 0)$ and do not reestimate it. Condition (ii) can be imposed by setting

$$\beta_T(j) = \begin{cases} 1 & \text{for } j = N \\ 0 & \text{otherwise.} \end{cases} \tag{67}$$

Condition (iii) can be guaranteed in the Baum-Welch algorithm by initially setting $a_{ij} = 0$ for $j < i$ (and in fact for any other combination of indices that specify transitions to be disallowed). It is clear from (34) that any parameter once set to zero will remain zero. For the gradient methods the appropriate a_{ij}'s are just set to zero and only the remaining parameters are adjusted.

The modification of the training procedure is as follows: Let us denote by $\mathbf{O} \equiv [\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \cdots \mathbf{O}^{(K)}]$ the set of observation sequences, where $\mathbf{O}^{(k)} = O_1^{(k)} O_2^{(k)} \cdots O_{T_k}^{(k)}$ is the kth sequence. We treat the observation sequences as independent of each other and then we adjust the parameters of the model $\mathbf{M}$ to maximize

$$P = \prod_{k=1}^{K} \text{Prob}(\mathbf{O}^{(k)} | \mathbf{M}).$$
$$= \prod_{k=1}^{K} P_k. \tag{68}$$

Since the Baum-Welch algorithm computes the frequency of occurrence of various events, all we need to do is to compute these frequencies of occurrence in each sequence separately and add them together. Thus the new reestimation formulas may be written as

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(O_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \tag{69}$$

and

$$\bar{b}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t \ni O_t(k)=v_j} \alpha_t^k(i) \beta_t^k(i)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i)}. \tag{70}$$

As noted above, π is not reestimated.

Scaling these computations requires some care since the scale factors for each individual set of forward and backward probabilities will be different. One way of circumventing the problem is to remove the scale factors from each summand before adding. We can accomplish this by returning the $1/P$ factor [which appears in (8) and (9) and was cancelled to obtain (10)] to the reestimation formula. Using the reestimation formula for the transition probabilities as an example, (69) becomes

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(O_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}. \tag{71}$$

If the right-hand side of (71) is evaluated using the scaled values of the forward and backward probabilities, then each term in the inner summation will be scaled by $C_t^k D_{t+1}^k$, which will then be cancelled by the same factor which multiplies P_k. Thus, using the scaled values in computing (69) results in an unscaled $\bar{a}_{ij}$. The procedure is easily extended to computation of the symbol probabilities. Also note that for the purposes of classification only one subsequence is to be considered so that either (55) or (56) may be used unaltered to compute P.

To apply Lagrangian techniques to left-to-right models we note that upon taking logarithms of (68) we have

$$\log P = \sum_{k=1}^{K} \log P_k. \tag{72}$$

The derivatives needed to maximize $\log P$ in (72) can be obtained by evaluating expressions for the derivatives of each individual subsequence and summing. For example, for a_{ij} we have [cf. (57) and (58)]

$$\frac{\partial}{\partial a_{ij}}(\log P) = \sum_{k=1}^{K} \frac{\partial}{\partial a_{ij}}(\log P) = C_T^k \sum_{t=1}^{T_k-1} \alpha_t^k(i) b_j(O_{t+1}^{(k)}) \beta_{t+1}^k(j). \quad (73)$$

As in all previous cases an analogous formula may be derived for the other parameters.

In practice, A and B for left-to-right models are especially sparse. Some of the zero values are so by design but others are dependent on **O**. Parameters of this type will be found one at a time by standard line search strategies. We have found that the convergence of the Lagrangian techniques can be substantially accelerated by taking large enough steps so that several positivity constraints become binding. The corresponding variables are then clamped and (65) is applied before beginning the next iteration.

V. NUMERICAL EXAMPLES

In this section, we give some instructive examples of the behavior of several of the algorithms discussed above. The algorithms were all coded in FORTRAN 77 on a Data General MV-8000, which uses a 32-bit floating point word. The data used in the tests came from either a Monte Carlo simulation of a hidden Markov chain or from a portion of a newspaper text that was edited to include only the 26 characters of the English alphabet and a special character denoting an interword space. The simulations have the valuable property that the model is known a priori, so that simple models may be used for checking program correctness while the more complicated ones can elicit some subtle and important numerical and methodological characteristics of the algorithms.

In our experiments we used the following procedure to generate observation sequences by means of a random number generator whose output is uniform on [0, 1] and specified values of π, A, B, T, and V:

(*i*) Partition the unit interval proportionally to the components of π. Generate a random number and select a start state, q_i, according to the subinterval in which the number falls. Set $t = 1$.

(*ii*) Partition the unit interval proportionally to the components of the ith row of B. Generate a random number and select a symbol, v_k, according to the subinterval in which the number falls. Set $O_t = v_k$.

(*iii*) Partition the unit interval proportionally to the components of the ith row of A. Generate a random number and select the next state, q_j, according to the subinterval in which the number falls.

(*iv*) Increment t. If $t \leq T$ set $q_i = q_j$ and repeat (*ii*) through (*iv*); otherwise stop.

Using this observation generator, several two- and three-state Markov models were simulated. These simulations were used to verify that the parameter estimation algorithms were working correctly and to study the effects of the scaling interval on the accuracy of the algorithms. In this study we found that all scaling intervals that were sufficiently short to prevent underflow yielded numerically identical results. Thus one can, at one extreme, scale the forward and backward probabilities after each observation or, at the other, wait until a threshold signaling that underflow is imminent is exceeded and only then perform the scaling operation.

We next proceeded to study a pair of four-state ($N = 4$), four-symbol ($M = 4$) models shown below and referred to as SRC44 and SRC45.

SRC44:

$$A = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0.5 \end{bmatrix}$$

$$\pi = [0.25 \quad 0.25 \quad 0.25 \quad 0.25]$$

and

SRC45:

$$A = \begin{bmatrix} 0 & 0 & 0.25 & 0.75 \\ 0.15 & 0 & 0 & 0.85 \\ 0.2 & 0.8 & 0 & 0 \\ 0 & 0.22 & 0.78 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0.25 & 0.75 & 0 & 0 \\ 0 & 0.15 & 0.85 & 0 \\ 0 & 0 & 0.1 & 0.9 \\ 0.2 & 0 & 0 & 0.8 \end{bmatrix}$$

$$\pi = [0.25 \quad 0.25 \quad 0.25 \quad 0.25].$$

The state transition diagrams for these models are shown in Fig. 3.

Model SRC44 is a balanced model in the sense that all permissible transitions and symbols are equally likely, whereas model SRC45 is skewed in that it distinctly favors some transitions and symbols over others.

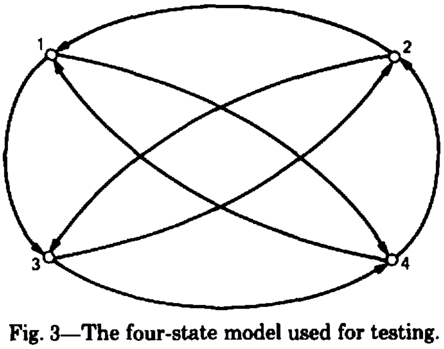


Fig. 3—The four-state model used for testing.

For each of these sources we processed observation sequences ranging in length from 100 to 4000 with the Baum-Welch algorithm. Initial values for A, B, and π were chosen at random and the algorithm was terminated in one of two ways; either when the change in log P from one iteration to the next fell below an arbitrary threshold, or when the number of iterations exceeded a specified maximum value. The maximum number of iterations was varied from 100 to 1000. For each estimate, $\hat{B}$, of the source matrix B, a measure of estimation error

$$\|\hat{B} - B\| = \left\{\frac{1}{M \cdot N} \sum_{j=1}^{N} \sum_{k=1}^{M} \left[\hat{b}_{jk} - b_{p(j)k}\right]^2\right\}^{1/2} \tag{74}$$

was computed, where $p(j)$ is the state permutation that minimizes the

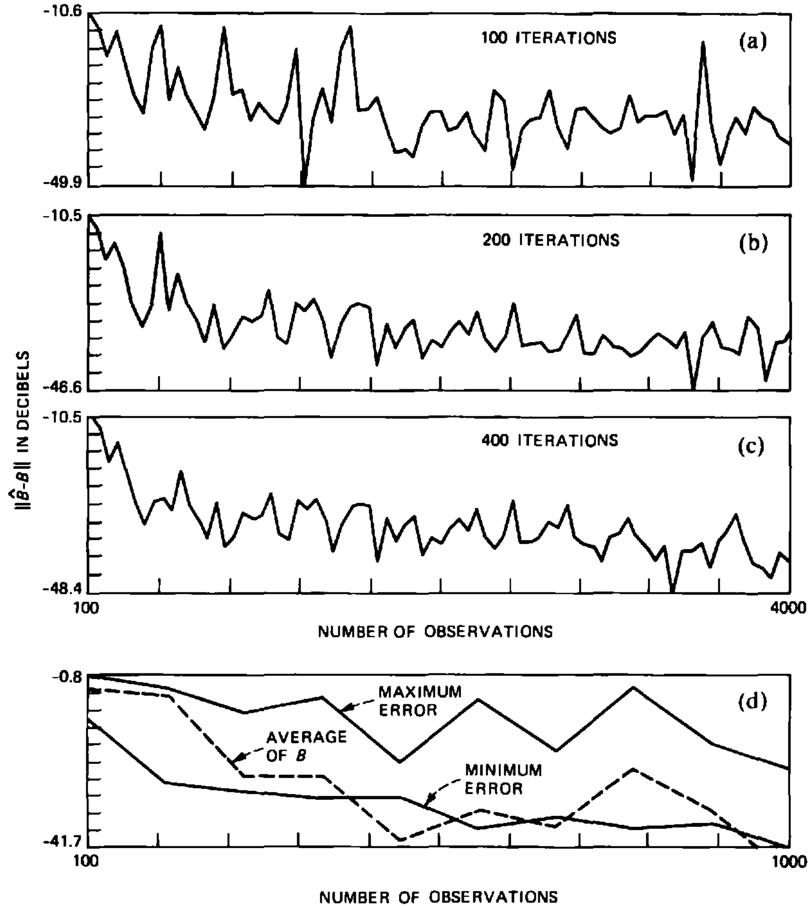


Fig. 4—Estimation error as a function of number of observations for source SRC44 for: (a) 100 iterations maximum, (b) 200 iterations, (c) 400 iterations, and (d) 10 random initial starts with 200 iterations maximum.

estimation error. The technique of minimum bipartite matching (see Appendix B) was used to determine the optimum state permutation.

Plots of the quantity $\|\hat{B} - \mathrm{B}\|$ (on a log scale) versus T, the number of observations, are given in Fig. 4 for source SRC44. Separate results are shown for 100 iterations maximum (part a) of the BW reestimation procedure, 200 iterations (part b), and 400 iterations (part c). Also shown in Fig. 4d are the results of using 10 random initial starts with 200 iterations maximum. Shown in Fig. 4d are the maximum and minimum estimation errors for each T and the estimation error for the average of all 10 B matrices. (The reader should note that T goes to 4000 in parts a through c, but only to 1000 in part d of Fig. 4.)

The curves given in Fig. 4 show several very interesting properties of the reestimation procedure. First we see that as the number of observations increases, a slow decrease in the average estimation error is obtained. However, we can see that statistical fluctuations (owing to different initial guesses for the model parameters) are often of larger magnitude than the slowly decreasing components of the curve. As the maximum number of iterations increases, the magnitude of the statistical fluctuations decreases, especially for larger values of T.

The curves of Fig. 4d show that although there is a wide range in the value of the estimation error for multiple starting choices, averaging the B matrices (after appropriate state alignment) leads to estimation errors comparable with the best single estimates.

Figure 5 shows a similar set of results for the Markov source SRC45. Figure 5a shows a curve of estimation error versus number of observations for a maximum of 400 iterations, Fig. 5b shows the same curve with a maximum of 1000 iterations, Fig. 5c shows the curve when the initial estimates of both A and B are set to the source values exactly, and Fig. 5d shows maximum and minimum estimation errors for 10 random starting points.

Although the general trends of the data in Fig. 5 are similar to those of Fig. 4, there are several key differences. From Fig. 5b it can be seen that even for 1000 iterations, the variation in model estimates is enormous (36-dB variations). This result suggests that it is significantly more difficult to estimate parameters of a skewed Markov model than those of a fairly uniform model. The curves of Fig. 5c, in which the initial conditions were set to the source generator values, show that extremely good solutions could be obtained if the reestimation procedure could start in the neighborhood of the "exact" solution. Obviously this situation (i.e., starting near the correct parameter values) is not enforceable for real data.

The curves of Fig. 5d, in which multiple estimates of the Markov model are averaged, show that averaging the individual parameter estimates does not lead to a low error estimate for SRC45. This is

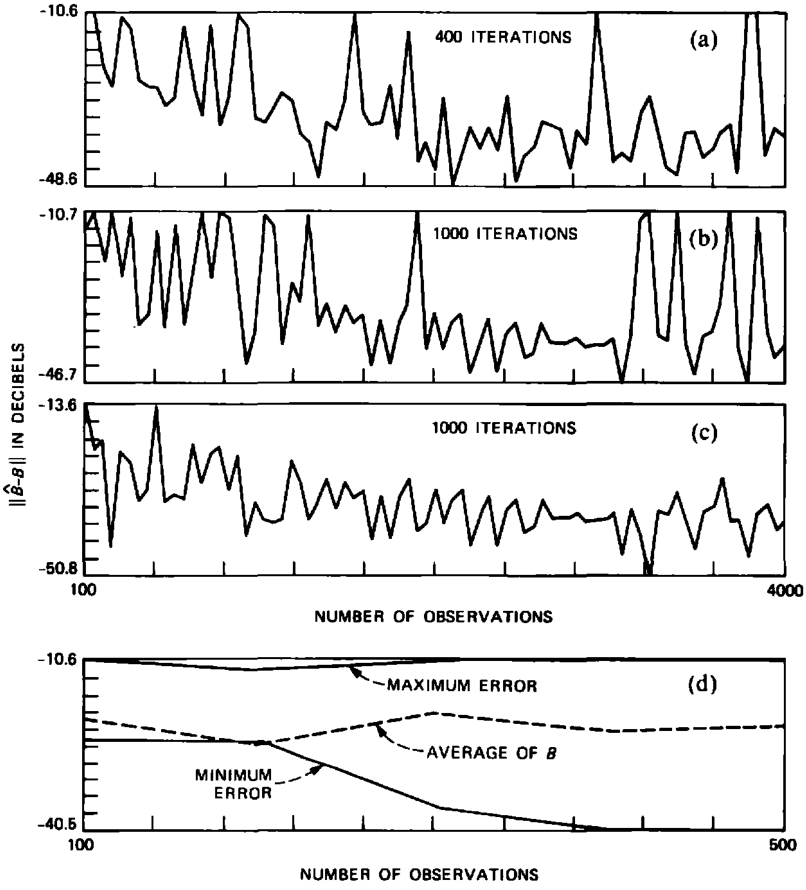


Fig. 5—Estimation error as a function of number of observations for source SRC45 for: (a) 400 iterations maximum, (b) 1000 iterations maximum, (c) initial estimates of A and B set to source values, and (d) maximum and minimum estimation errors for 10 random starting points.

undoubtedly because of the parameter estimates with high errors that occur and which have an undue influence on the average.

5.1 Left-to-right Markov source estimation

The second series of experiments dealt with the left-to-right Markov models, as would be appropriate for our intended application to isolated word recognition. Figure 6 shows four such models. For each of these models (denoted as SRC195, SRC295, SRC395, and SRC495 in Fig. 6), the specifications were:

$$N = 5, M = 9, \pi = \{1, 0, 0, 0, 0\}$$

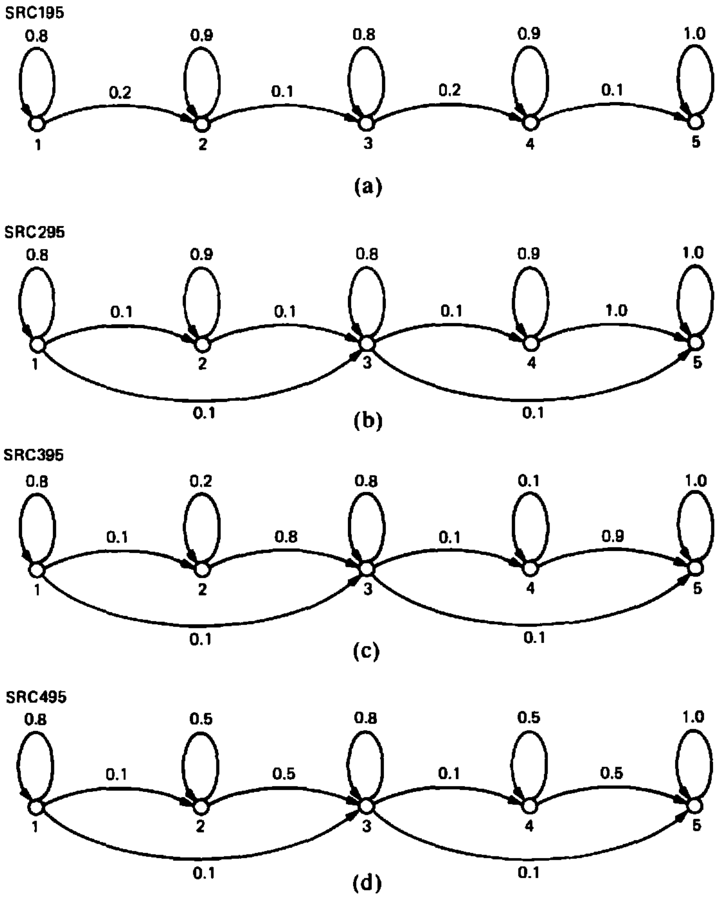


Fig. 6—Left-to-right models used for testing for: (a) SRC195, (b) SRC295, (c) SRC395, and (d) SRC495.

and

$$B = \begin{bmatrix} 0.7 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \end{bmatrix}.$$

The state transition probabilities were those shown in Fig. 6. The SRC195 model is a left-to-right model. The SRC295 model allows a transition between states 1 and 3 and states 3 and 5, as well as transitions between sequentially numbered states. Both the SRC395 and SRC495 models include states whose self-transition probabilities

(a_{ii}) are very small. We will see below that the average occupancy of such states is only 1 to 2 observations. For these non-ergodic models the concept of occupancy of the transient (i.e., non-absorbing) states is important.

If we denote the probability of a transition from a state to itself as p, then the probability of a transition out of that state at time $T + 1$ (assuming the state was entered at time $t = 0$) is

$$\text{Prob}(q_i \text{ at } t = 0, 1, 2, \cdots, T \quad \text{and} \quad q_j \neq q_i \text{ at } T + 1) = p^T(1 - p)$$

for models of the form of SRC195. Hence the average occupancy of a state is given by

$$\bar{d} = \sum_{t=1}^{\infty} (t + 1)p^t(1 - p)$$

$$= \frac{1}{1 - p}. \tag{75}$$

For $p = 0.9$ we get $\bar{d} = 10$, for $p = 0.8$ we get $\bar{d} = 5$, for $p = 0.5$ we get $\bar{d} = 2$, and for $p = 0.1$ we get $\bar{d} = 1.1$. Standard formulas are available for computing average state occupancies for arbitrary transition matrices. We will not consider them here; however, it is clear that states 2 and 4 in models SRC395 and SRC495 are of low occupancy.

To test the reestimation procedure on the Markov sources of Fig. 6, a set of K sequences were generated for each model, where K was the set (10, 25, 50, 100). Each sequence was generated using the Markov sequence generator described earlier, modified slightly to ensure that each sequence terminated in state 5 (the final state) and stayed there for 5 observations. The sequences were, however, of variable duration, depending on the exact sequence of state transitions that occurred.

The results showed that for sequences generated from model SRC195, the correct model parameters (to within small estimation errors) were obtained for all values of K from 10 to 100. For sequences generated from model SRC295 only the 10 observation training sequence yielded grossly incorrect model parameter estimates. All other sequences ($K = 25, 50, 100$) yielded the correct parameter values.

For both source models SRC395 and SRC495, however, no completely correct parameter estimates were obtained. In particular, the states whose expected occupancy was small (i.e., states 2 and 4 in both models) were merged with either the preceding or the following state (or both), while other states whose expected occupancy was larger were often split into 2 states, each with the same set of output symbols. These experiments indicate that it is difficult to reliably estimate parameters of a state, in a left-to-right model, whose average occupancy is very much smaller than that of the states to which it is connected.

5.2 Tests on a non-Markov source

In the experiments described above, the observations were in fact generated by a known probabilistic function of a Markov chain. A more difficult test of these techniques was performed in which the observations were 5000 characters from a newspaper article edited to contain only the letters of the alphabet and spaces. We used the observation sequence to train a four-state, 27-symbol model. Of course, the "true" underlying model is not known, as it was in the tests previously described. Nor is it likely that the four-state model is complex enough to model the richness of structure of written English. Even if it were, it is unlikely that 5000 characters is sufficient to capture the structure. Unfortunately, these are exactly the limitations with which the experimenter will be faced in trying to model "real" processes. Our hope was that the text analysis problem would reveal some of the ambiguities that will be encountered in making hidden Markov models of natural phenomena.

The text was first analyzed using the Baum-Welch reestimation formulas with a randomly chosen starting point. The algorithm converged in 310 iterations with $\log P = -1.317 \times 10^4$. For purposes of comparison, we analyzed the same data with a quasi-Newton optimization routine, VE01A, from the Harwell Library.[26] It required 125 iterations to obtain a maximum value of $\log P$ of -1.356×10^4. In this case some care was required with the parameter values. The finite precision arithmetic occasionally results in a parameter value of -10^{-7}, which appears to satisfy the positivity constraints. Such a value is fatal to the computation of $\log P$ since it will result in an attempt to take a logarithm of a negative quantity. Fortunately, this condition is readily detected in the scaling routine and corrected by setting the offending parameter to zero.

Finally, we applied the Lagrangian technique described earlier to the same observation sequence but with a different set of initial parameter values. After 136 iterations, a still different model with $\log P = -1.327 \times 10^4$ is obtained. These results illustrate some important features of hidden Markov modeling. The computational methods used to obtain the models are roughly equivalent. All of the resulting models capture some of the structure of the data being analyzed. There are many different possible models with very little evidence for selecting the "best" one. For even very simple models, the likelihood function is too complicated to attribute the selection of one model or another by one algorithm or another to its properties. Finally, we note that all of the algorithms tested make large improvements to P during the early iterations and only slight incremental improvements later. In fact, the last half of the iterations provides no significant change to the model. We have used a convergence criterion of 10^{-7} on the relative increment between iterations. This may be relaxed substantially, resulting in many fewer iterations with no attendant degradation in the model.

VI. CONCLUSION

We have presented some of the salient theoretical and practical issues associated with modeling data by probabilistic functions of a Markov chain. In our presentation we have concentrated on three issues: alternatives to the Baum-Welch reestimation algorithm, critical facets of implementation, and behavior of Markov models on certain artificial but realistic signals.

We have observed that, while most of the discussion of parameter estimation for Markov models in the open literature is devoted to the Baum-Welch algorithm, classical optimization techniques are not only a viable alternative but may even be preferable in some cases. In particular, classical techniques are virtually unrestricted by the forms of either the likelihood function or the constraints. The reestimation formulas may be growth transformations for a wider class of functions and constraints than has heretofore been proven; however, it is not likely that a universal reestimation formula exists. For applications to continuous density functions (c.f. Liporace[29]), the classical techniques may have still other advantages.

The open literature has provided only a perfunctory, if any, discussion of some crucial numerical and implementational problems associated with Markov modeling. We have given details of methods of dealing with floating point loss of significance, finite-training-set size, and model stability. Wherever possible we have made our techniques formal and algorithmic.

Finally, we have given several examples of the behavior of Markov modeling techniques on some reasonably realistic data. The most important lesson that can be drawn from these experiments is that even under ideal conditions (i.e., when the data are associated with a known hidden Markov process) and all the more so under realistic conditions, the computed models may contain artifacts and may not faithfully represent the inherent structure of the data. Thus, great caution and empirical validation is required in using these techniques.

Despite this caveat, hidden Markov models may be beneficial in studying many diverse problems. In our companion paper[21] we recount a successful application of this body of theory to a problem in automatic speech recognition.

REFERENCES

1. J. K. Baker, "The DRAGON System—An Overview," IEEE Trans. Acoustics, Speech and Signal Processing, *ASSP-23* (February 1975), pp. 24–9.

2. J. K. Baker, "Stochastic Modeling for Automatic Speech Understanding," in *Speech Recognition*, R. Reddy, ed., New York: Academic Press, 1975.
3. L. R. Bahl and F. Jelinek, "Decoding for Channels with Insertions, Deletions and Substitutions, with Applications to Speech Recognition," IEEE Trans. Information Theory, *IT-21* (July 1975), pp. 404–11.
4. L. R. Bahl, J. K. Baker, P. S. Cohen, A. G. Cole, F. Jelinek, B. L. Lewis, and R. L. Mercer, "Automatic Recognition of Continuously Spoken Sentences from a Finite State Grammar," Proc. ICASSP (April 1978), pp. 418–21.
5. L. R. Bahl, J. K. Baker, P. S. Cohen, N. R. Dixon, F. Jelinek, R. L. Mercer, and H. F. Silverman, "Preliminary Results on the Performance of a System for the Automatic Recognition of Continuous Speech," Proc. ICASSP (April 1976), pp. 425–9.
6. L. R. Bahl, J. K. Baker, P. S. Cohen, F. Jelinek, B. L. Lewis, and R. L. Mercer, "Recognition of a Continuously Read Natural Corpus," Proc. ICASSP (April 1978), pp. 422–4.
7. L. R. Bahl, R. Bakis, P. S. Cohen, A. G. Cole, F. Jelinek, B. L. Lewis, and R. L. Mercer, "Further Results on the Recognition of a Continuously Read Natural Corpus," Proc. ICASSP (April 1980), pp. 872–5.
8. L. R. Bahl, R. Bakis, P. S. Cohen, A. G. Cole, F. Jelinek, B. L. Lewis, and R. L. Mercer, "Recognition Results with Several Experimental Acoustic Processors," Proc. ICASSP (April 1979), pp. 249–51.
9. F. Jelinek and R. L. Mercer, "Interpolated Estimation of Markov Source Parameters from Sparse Data," in *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal, eds., Amsterdam, The Netherlands: North Holland Publishing Company, 1980.
10. F. Jelinek, "Fast Sequential Decoding Algorithm Using a Stack," IBM J. Res. Develop. (1969), pp. 675–85.
11. F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," IEEE Trans. Information Theory, *IT-21* (May 1975), pp. 250–6.
12. F. Jelinek, "Continuous Speech Recognition by Statistical Methods," Proc. IEEE, *64* (April 1976), pp. 532–56.
13. L. E. Baum, "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process," Inequalities, *3* (1972), pp. 1–8.
14. L. E. Baum and J. A. Eagon, "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology," Bull. AMS, *73* (1967), pp. 360–3.
15. L. E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," Ann. Math. Stat., *37* (1966), pp. 1559–63.
16. 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., *41* (1970), pp. 164–71.
17. L. E. Baum and G. R. Sell, "Growth Transformations for Functions on Manifolds," Pac. J. Math., *27* (1968), pp. 211–27.
18. D. S. Passman, "The Jacobian of a Growth Transformation," Pac. J. Math., *44* (1973), pp. 281–90.
19. P. Stebe, "Invariant Functions of an Iterative Process for Maximization of a Polynomial," Pac. J. Math., *43* (1972), pp. 765–83.
20. L. R. Rabiner and S. E. Levinson, "Isolated and Connected Word Recognition—Theory and Selected Applications," IEEE Trans. Commun., *COM-29* (May 1981), pp. 621–59.
21. 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," B.S.T.J., this issue.
22. A. A. Markov, "An Example of Statistical Investigation in the Text of 'Eugene Onyegin' Illustrating Coupling of Tests in Chains," Proc. Acad. Sci. St. Petersburg, *7* (1913), pp. 153–62.
23. A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm," IEEE Trans. Information Theory, *IT-13* (April 1967), pp. 260–9.
24. J. B. Rosen, "The Gradient Projection Method for Nonlinear Programming—Part I: Linear Constraints," J. Soc. Indust. Appl. Math., *8* (1960), pp. 181–217.
25. R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer J., *6* (1963), pp. 163–8.

26. M. J. Hopper, ed., *Harwell Subroutine Library: A Catalog of Subroutines, 55,* Oxfordshire, England: A.E.R.E. Harwell, 1979.
27. M. R. Hestenes, *Optimization Theory,* New York: Wiley, 1975.
28. E. Lawler, *Combinatorial Optimization,* New York: Holt Rinehart and Winston, 1976.
29. L. R. Liporace, "Maximum Likelihood Estimation for Multivariate Observations of Markov Sources," IEEE Trans. Information Theory, *IT-28* (September 1982), pp. 729–34.

APPENDIX A

Several of the formulas derived in the text are much more compact in matrix notation. Let $'$ denote matrix transposition, as usual, and let the column vectors π and **1**, and the matrices A, B_t, $t = 1, \cdots, T$ be defined as in the text. Also let α_t and β_t be column vectors with components $\alpha_t(i)$, $i = 1, \cdots, N$ and $\beta_t(i)$, $i = 1, \cdots, N$, respectively. Then the recursion for α_t is

$$\alpha_{t+1} = B_{t+1}A'\alpha_t, \qquad t = 1, \cdots, T-1. \tag{76}$$

The recursion for β_t is

$$\beta_t = A B_{t+1}\beta_{t+1} \qquad t = T-1, \cdots, 1. \tag{77}$$

The starting values are

$$\begin{aligned}\alpha_1 &= B_1\pi \\ \beta_T &= \mathbf{1}.\end{aligned} \tag{78}$$

The probability P is given by

$$P = \beta_t'\alpha_t \qquad \text{for any } t \text{ in } (1, T). \tag{79}$$

The special cases $t = 1$ and $t = T$ give

$$P = \pi' B_1\beta_1 \tag{80}$$

and

$$\begin{aligned}P &= \mathbf{1}'\alpha_T \\ &= \mathbf{1}'B_TA'B_{T-1} \cdots A'B_1\pi.\end{aligned} \tag{81}$$

In each of these formulas P can be regarded as the trace of a 1×1 matrix, which [as expanded in (81)], is a product of several matrices. The fact that the trace of a product of matrices is invariant to a cyclic permutation of the matrices can be used to advantage in finding the gradient of P. Define $\nabla_A P$ as the matrix whose ijth component is $\partial P/\partial a_{ij}$. Similarly, define $\nabla_B P$ and $\nabla_\pi P$. Then it is straightforward to show that

$$\begin{aligned}\nabla_\pi P &= B_1\beta_1 \\ \nabla_A P &= \sum_{t=1}^{T-1} \alpha_t\beta_{t+1}'B_{t+1} \\ (\nabla_B P)_{jk} &= \sum_{t \ni O_t = k} (A'\alpha_{t-1})_j(\beta_t)_j.\end{aligned} \tag{82}$$

In the last equation, if $O_1 = v_k$ then the corresponding term in the sum is just π'/β_1.

APPENDIX B

As we mentioned in Section 3.3, it is frequently necessary to compare (or average) two different estimates B and $\tilde{B}$ of the symbol matrix. The optimization procedures, in general, relabel the states; therefore the rows of $\tilde{B}$ may, in general, be permuted relative to those of B. Before comparison, therefore, the optimum permutation must be found. This is defined as one that minimizes the distance D defined in eq. (66). The problem of finding this permutation can be converted to a network optimization problem called "bipartite weighted matching." To this end define w_{ij} as the distance of the ith row of B from the jth row of $\tilde{B}$. As we have done in Fig. 7, draw two sets P and Q of N vertices each. For $1 \le i, j \le N$, draw an edge from the ith vertex in P to the jth vertex in Q, and label this edge with the weight w_{ij}. The resulting graph is a complete weighted bipartite graph, and the problem is to find an N-match (i.e., one to one matching with N edges) such that it has minimum weight.

Suppose that $\mathbf{Z}_k$ is a k-match ($k \le N$). With respect to this match make the following definitions:

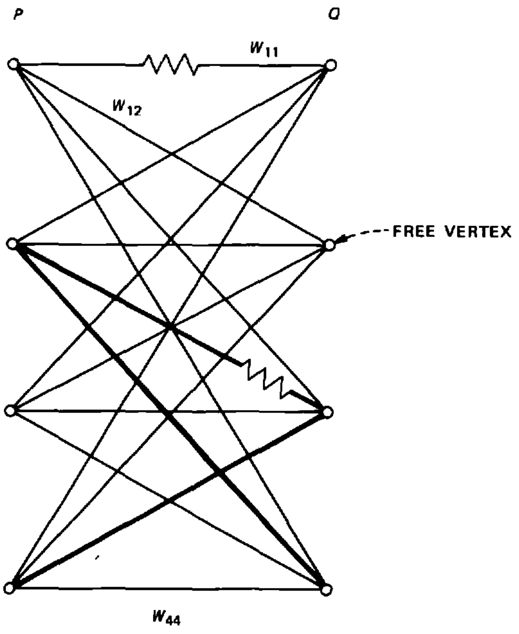


Fig. 7—Complete bipartite weighted graph ($N = 4$) and a 2-match (matched edge $\rightarrow$). Path shown in heavy lines is an alternating path for the 2-match. It also happens to be an augmenting path.

(*i*) A matched edge is an edge in $\mathbf{Z}_k$.

(*ii*) A free vertex is one that is not on a matched edge.

(*iii*) An alternating path is a path along edges that alternately belong to $\mathbf{Z}_k$ and do not belong to $\mathbf{Z}_k$. [N.B. the number of matched edges, m, on an alternating path may have any value $m \leq k$. The degenerate case $m = 0$ is also valid.]

(*iv*) An augmenting path is an alternating path between two free vertices. Again note that a single edge connecting two free vertices is a valid augmenting path.

An augmenting path has the structure

$$p_1 U q_1 M p_2 U q_2 \cdots U q_j. \tag{83}$$

Here U represents an unmatched edge, M represents a matched edge, and p_1 and q_j are the only free vertices on the path. (Here p_i is the ith P-vertex along the path. The q_i are similarly defined.) Note that the number of U's on an augmenting path is exactly one more than the number of M's. Hence the total number of edges in an augmenting path is always odd.

Suppose we are given a k-match $\mathbf{Z}_k$ and an augmenting path **ap** of length $2j + 1$. Then we can obtain a $(k + 1)$-match $\mathbf{Z}_{k+1}$ by a complementary labeling of the edges of **ap** (i.e., by changing every U to M and vice versa). If $w_1, w_2, \cdots, w_{2j+1}$ are the weights along **ap**, then the weight $\mathbf{Z}_{k+1}$ exceeds that of $\mathbf{Z}_k$ by the amount $w_1 - w_2 + w_3 - \cdots + w_{2j+1}$.

This method of obtaining a $(k + 1)$-match from a k-match has the following key property (proof given below): Suppose $\mathbf{M}_k$ is a minimum-weight k-match. Let **apm** be an augmenting path for $\mathbf{M}_k$ with minimum incremental weight. Then the match $\mathbf{M}_{k+1}$ obtained from $\mathbf{M}_k$ and **apm** is a minimum weight $(k + 1)$-match.

Assuming this property for the moment, a minimum-weight N-match can be determined by the following N-step algorithm:

For $k = 1, 2, \cdots, N$ generate $\mathbf{M}_k$ by finding an optimum augmenting path for $\mathbf{M}_{k-1}$. (Note that $\mathbf{M}_0$ is an empty set.)

We now show that finding an optimum augmenting path is equivalent to a shortest path problem. With reference to Fig. 8 let us generate a directed graph by directing each edge in $\mathbf{M}_k$ to the left and each unmatched edge to the right. Also let us multiply the weights of all matched edges by -1. Finally, add two vertices labeled **s** and **t**. Connect **s** to all *free* vertices in P by right-going edges of zero weight. Similarly, connect all *free* vertices of Q to **t** by right-going edges of zero weight. In this directed graph any path from **s** to **t** is an augmenting path for the matching $\mathbf{M}_{k-1}$. Hence, if we interpret the weights as lengths, it is clear than an optimum augmenting path is a shortest path from **s** to **t**.

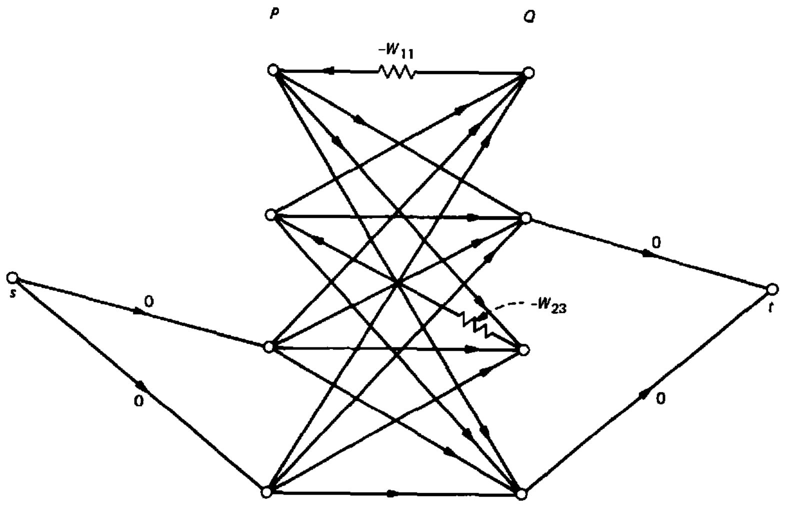


Fig. 8—Directed graph obtained from Fig. 7. Every path from s to t is an augmenting path of the 2-match (except for the dummy edges of 0 weight from *s* and to *t*). Interpreting weights as lengths, the length of a path from s to t is the incremental weight of the corresponding augmenting path.

A shortest path problem can be solved in polynomial time. However, the problem can be solved much more easily and efficiently if the path lengths are all nonnegative. In that case the problem can be solved in N^2 time by Dijkstra's method.[28]

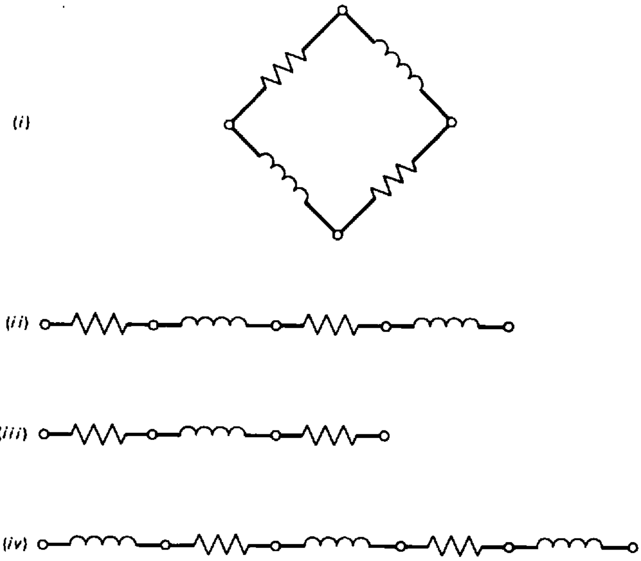
It is possible to avoid negative distances in our problem by the method of assigning a "potential" $f(v)$ to each vertex. Suppose that at step k the graph has nonnegative weights, and the edges corresponding to $\mathbf{M}_{k-1}$ have zero weight. Then use Dijkstra's method to find shortest paths to all vertices (including **t**) from **s**. Define $f(v)$ for the vertex **v** as its shortest distance from **s**. Next, modify the weight w_{ij} of the edge from i to j to

$$w'_{ij} = w_{ij} + f(i) - f(j). \tag{84}$$

It is easily seen that this procedure leaves all weights nonnegative, does not alter shortest paths, and all shortest paths have weight 0. Reversing a shortest path from **s** to **t** gives us a matching $\mathbf{M}_k$, and the new graph has nonnegative weights and zero weights for the matched edges. Now $\mathbf{M}_0$ trivially has the postulated properties. Therefore, at every step the graph will have these properties.

An important property of the above procedure is that if a vertex p (or q) is on some matched edge in $\mathbf{M}_k$, then it will be on some matched edge in $\mathbf{M}_{k+1}$ also. In writing the actual computer program, this property simplifies the bookkeeping considerably.

Another important property concerns the vertex potentials. Suppose



$\in \mathbf{M}_k$

Fig. 9—Types of paths in the symmetric difference between $\mathbf{M}_k$ and $\mathbf{N}_{k+1}$. $\in \mathbf{N}_{k+1}$. There must be exactly one more path of type (*iv*) than of type (*iii*).

$F(v)$ is the sum of the potentials assigned to vertex v at each of the N steps. Suppose w_{ij} and d_{ij} are the original and final weights, respectively, of the edge connecting vertex i to vertex j. Then

$$d_{ij} = w_{ij} + F(i) - F(j) \geq 0, \tag{85}$$

and $d_{ij} = 0$ for every edge in the final N-match. The numbers $F(v)$ thus provide a simple proof that the final match is indeed an optimum match.

We turn now to a proof of the key property mentioned above. Let $\mathbf{M}_k$ be an optimum k-match and let N_{k+1} be *any optimum* k + 1-match. Then we will show that there is a k + 1-match obtained from an augmenting path of $\mathbf{M}_k$ such that its weight is equal to that of N_{k+1}. For this purpose define $\mathbf{S}$ as the set of edges in the symmetric difference of $\mathbf{M}_k$ and $\mathbf{N}_{k+1}$. (Recall that the symmetric difference of two sets, $\mathbf{A}$ and $\mathbf{B}$, is the set of elements that belong either to $\mathbf{A}$ or to $\mathbf{B}$ but not to both.)

From the geometry of a bipartite graph three properties of the set $\mathbf{S}$ are obvious:

(*i*) The number of edges in $\mathbf{S}$ which belong to $\mathbf{N}_{k+1}$ must exceed the number that belongs to $\mathbf{M}_k$ by exactly 1.

(*ii*) The edges on any path in $\mathbf{S}$ must alternate between $\mathbf{M}_k$ and $\mathbf{N}_{k+1}$.

(*iii*) A vertex on an edge in **S** cannot be shared with any edge in $\mathbf{M}_k$ or $\mathbf{N}_{k+1}$ that does not belong to **S**.

From the first two of these properties it follows that there can be four types of paths in **S** (see Fig. 9).

(*i*) Circuits, each consisting of an even number of edges;

(*ii*) Open paths, each with an even number of edges;

(*iii*) Open paths, each with an odd number of edges beginning with an edge in $\mathbf{M}_k$;

(*iv*) Open paths, each with an odd number of edges beginning with an edge in $\mathbf{N}_{k+1}$. The number of such paths must be exactly one more than the number of paths of type (*iii*).

It is easily seen that the incremental weight of every path of type (*i*) or (*ii*) must be exactly zero. (If it is negative then the weight of $\mathbf{M}_k$ can be decreased by a complementary labeling of the path; if positive then the weight of $\mathbf{N}_{k+1}$ can be decreased in the same way. But this contradicts the hypothesis that $\mathbf{M}_k$ and $\mathbf{N}_{k+1}$ are optimum matches.)

In view of this and property (3), $\mathbf{N}_{k+1}$ can be modified by replacing its edges on all paths of types (*i*) and (*ii*) by the corresponding edges of $\mathbf{M}_k$. The modified k + 1-match has exactly the same weight as that of N_{k+1}; however, the symmetric difference between $\mathbf{M}_k$ and the modified k + 1-match has no paths of type (*i*) or (*ii*).

The same argument applies to pairs of paths, one path each of types (*iii*) and (*iv*). Thus a final modified k + 1 match is obtained whose symmetric difference with $\mathbf{M}_k$ is exactly one path of type (*iv*). The weight of this final modified k + 1-match is exactly the same as that of the original $\mathbf{N}_{k+1}$, and is obtained from an augmenting path of $\mathbf{M}_k$.

We have written a subroutine that implements the above procedure. The timing, from a number of test runs, is approximately $0.063N^3$ ms central processing unit (CPU) time on the MV-8000. Thus for $N = 40$ about 4 seconds of CPU time is needed.