HIDDEN MARKOV MODELS FOR AUTOMATIC SPEECH RECOGNITION: THEORY AND APPLICATION

S J Cox

ABSTRACT

Hidden Markov modelling is currently the most widely used and successful method for automatic recognition of spoken utterances. This paper reviews the mathematical theory of hidden Markov models and describes how they are applied to automatic speech recognition.

1. Introduction

Machine transcription of fluent speech - the so-called 'phonetic typewriter' - remains a challenging research goal. A machine capable of performing this task would require an enormous amount of knowledge about the real world working in conjunction with a mechanism for decoding acoustic signals. However, automatic speech recognition (ASR) has reached the point where, given some restrictions on speakers, vocabulary and environmental noise, reliable recognition of isolated words or short phrases is possible.

A distinction is generally made in ASR between recognition of utterances from a speaker who has previously 'enrolled' his voice (speaker dependent recognition) and a speaker whose voice the recogniser has never 'heard' previously (speaker independent recognition). To some extent, it is possible to trade vocabulary size for speaker dependence, so that it is currently feasible to build either a speaker independent ASR device that recognises a small vocabulary (5-20 words or phrases), or a speaker dependent device that recognises a large vocabulary (≈10 000 words). For telephony applications, enrolment free ASR is clearly crucial and it turns out that many useful applications are possible with a small vocabulary.

Broadly speaking, attempts at ASR fall into two categories - a knowledge-based approach, in which knowledge about speech from the domains of linguistics and phonetics is used to construct a set of rules which is in turn used to interpret the acoustic input signal, and a 'pattern -matching' approach in which *a priori* knowledge about speech is largely ignored and techniques of pattern classification are applied to the input signal. The knowledge-based approach is able to exploit a body of knowledge about speech and, in particular, the relationship between features extracted from the speech signal and higher level linguistic representations (e.g. phonemes, syllables). However, this relationship, which is very complex, is still far from being understood, largely

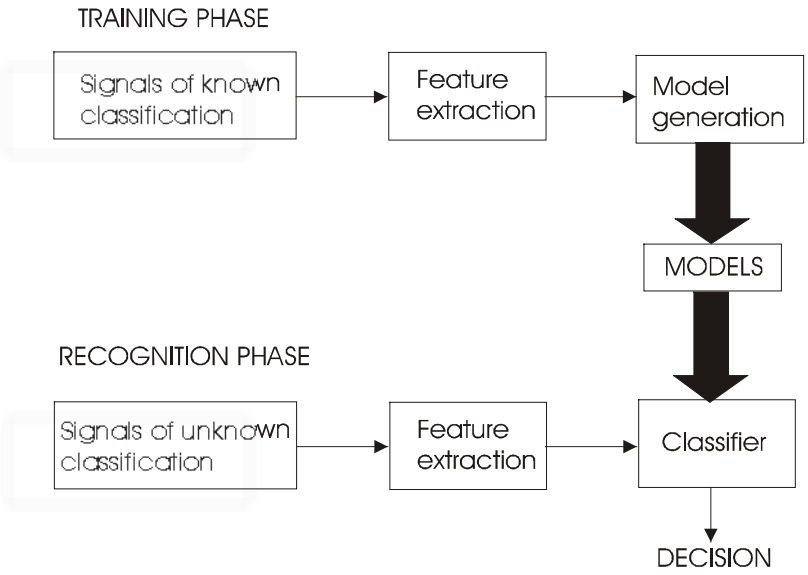
because of the enormous variability of signals interpreted by the brain as representing the same linguistic units. Of the knowledge that is available, it is not yet clear how best to represent or use it in a computational framework.

The pattern-matching approach makes no attempt to use this kind of knowledge, but gathers its 'knowledge' of the speech signal in a statistical form by being shown examples of speech patterns. As such, it would work equally well on any acoustic patterns - birdsong', machinery noise, seismic signals etc. However, powerful mathematical techniques are available which are guaranteed to optimise the technique and these ensure that the approach is surprisingly successful, even though almost all knowledge of speech production, perception and the speech signal is ignored.* Furthermore, these techniques are applicable to patterns at any level and hence can be used to optimally decode other representations of speech signals (e.g. phonetic segments, words), thus providing a coherent framework for speech recognition and understanding.

This paper focuses exclusively on the pattern matching approach which is used by all commercial ASR devices. In particular, two techniques have been found to be especially appropriate to ASR - dynamic time warping (DTW) and hidden Markov modelling. DTW is a special case of hidden Markov modelling and is discussed in Appendix A. A third *connectionist* approach is now evolving, based on adaptive parallel distributed processing networks [1]

2. Pattern classification

Figure 1 shows a block diagram of a pattern classifier in training and recognition modes.



*** Fig 1: A pattern classifier in training and recognition modes**

The two approaches to ASR have been pithily summarised by one researcher as 'doing the right thing wrong or the wrong thing right'.

In training mode, many examples of each class are used to build a model for the class, and these models are subsequently stored. In recognition mode, a pattern of unknown class is compared with each model and classified according to the model to which it is 'closest'. When performing recognition of isolated words, each different word is regarded as a class.

The feature extraction shown in Fig 1 is necessary for two reasons. Firstly, it enables focusing on information within the signal which is important for discriminating between patterns of different classes; a good set of features '...enhances within-class similarity and between-class dissimilarity' [2]. Secondly, it enables data reduction so that manipulation of patterns becomes computationally feasible. (A telephone speech signal has a typical data-rate of 64 000 bits/s, too high for practical computation.)

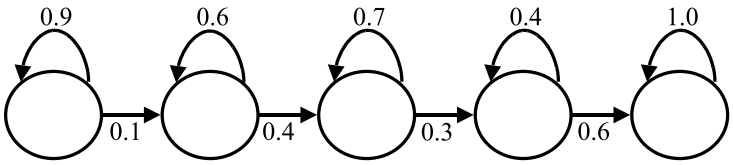
Features for speech recognition are generally either related to the instantaneous spectrum of the speech signal or to the instantaneous shape of the vocal tract; clearly, there is a large overlap of information in these representations. Feature selection is of great importance in speech recognition, as accuracy is highly dependent on the type and number of features used [3]. Because of the sluggishness of the speech articulators, an adequate representation of the speech pattern can be made by measuring features at regular intervals of approximately 1/100 s. Each sample is a d dimensional vector of features, so that an utterance of length t seconds is reduced to a sequence of $T = 100 * t$ d-dimensional analysis vectors. In this paper, utterances are considered to be isolated words and so the approach to recognition is sometimes called whole word pattern matching [4].

3. Speech pattern classification using hidden Markov models

Speech differs from most signals dealt with by conventional pattern classifiers in that information is conveyed by the temporal order of speech sounds. A stochastic process provides a way of dealing with both this temporal structure and the variability within speech patterns representing the same perceived sounds. Stochastic processes '...develop in time or space in accordance with probabilistic laws' [5]; a stochastic process that has been found to be particularly useful in ASR is a hidden Markov model (HMM).

3.1 Description of a hidden Markov model

Figure 2 shows an example HMM.



$$A = [a_{ij}] = \begin{bmatrix} 0.7 & 0.3 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.2 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.65 & 0.35 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.9 & 0.1 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$$B = [b_{jk}] = \begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{bmatrix} 0.0 & 0.2 & 0.8 & 0.0 \\ 0.3 & 0.7 & 0.0 & 0.0 \\ 0.65 & 0.0 & 0.35 & 0.0 \\ 0.0 & 0.9 & 0.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \end{array}$$

$$\pi = \begin{bmatrix} 0.5 & 0.5 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Fig 2 A 5-state left-right hidden Markov model with 4 output symbols.

The five circles represent the states of the model and, at a discrete time instant t, the model occupies one of the states and emits an observation. At instant $t+1$, the model either moves to a new state or stays in the same state and emits another observation, and so on. This continues until a final terminating state is reached at time T. An important characteristic of this process is that the state occupied at time instant $t+1$ is determined probabilistically and depends only on the state occupied time t - this is the Markov property. The probabilities of moving from state to state are tabulated in a $N \times N$ state transition matrix, $A = [a_{ij}]$ - an example A is shown in Fig 2. The i, j th entry of A is the probability of occupying state s_j at time $t+1$ given state s_i at time t. Since the total probability of making a transition from a state must be 1.0, each row of A sums to 1.0. Notice also that A is upper triangular, because this particular HMM is a left-to-right model in which no 'backwards' jumps are allowed and the model progresses through the states in a left-right manner. In a generalised HMM, a transition is possible from any state to any other state, but there is clearly no point in using anything but a left-to-right HMM for ASR as only left-to-right models will effectively model the temporal ordering of speech sounds. The topology of the model shown in Fig 2 is rather simple, and richer topologies in which states may be skipped are used to advantage in ASR.

An observation emitted at a time instant t by the HMM shown in Fig 2 can be one of only 4 symbols, A, B, C or Z. In general, a model can emit any of a finite alphabet of M symbols $(v_1, v_2, ..., v_M)$ from each state, and the probability of emitting symbol v_k from state s_j is given by the j, k th entry of a $N \times M$ matrix $B = [b_{jk}]$ (an example B is

shown in Fig 2). A final parameter needed to set the model in motion is a vector π whose ith component $\pi(i)$ is the probability of occupying state i at $t = 1$ - again, an example is given.

An observation sequence is produced by the model of Fig 2 as follows.

- Use the vector π and a random number generator (RNG) to determine which state the model starts in - assume this is state s_i. Set $t = 1$.

- Use the probabilities in row i of B and the RNG to select a symbol v_k to output.

- Use the probabilities in row i of A and the RNG to determine which state si to occupy next. Set $t = t + 1$.

- Repeat the second and third steps until a terminating state is reached. (State 5 in the model of Fig 2 is such a state, because it has a self transition probability of 1.0 and only outputs the symbol Z. Hence the observation sequence would be an infinite succession of Z s once state 5 is reached.)

A typical observation sequence produced by this process acting on the model of Fig 2 might be: CCBCCBMACAAACACBBBZ. Notice that the states that produced each symbol are hidden in the sense that, given an observation sequence, it is in general impossible to say what the state sequence that produced these observations was[1]. An algorithm which finds the most likely state sequence given an observation sequence and a model is introduced later.

[1] If a one-to-one correspondence between states and symbols exists, the process is known as a Markov chain.

3.2 Relationship of HMMs to speech production and recognition.

The above description of an HMM was deliberately kept abstract, but the reader may already have an inkling of how the process is related to speech production. Let us make the following premise about the production of an utterance of a particular word: *an utterance is produced by the articulators passing through an ordered sequence of 'stationary states' of different duration; the 'outputs from each state (i.e. the observations) can be regarded as probabilistic functions of the state.* This is a crude and drastically simplified model of the complexities of speech; speech is a smooth and continuous process and does not jump from one articulatory position to another. However, the success of HMMs in speech recognition demonstrates that if the model is correctly optimised on speech data, it captures enough of the underlying mechanism to be powerful.

The observation sequence of the HMM is thus the sequence of analysis vectors described in section 2. The model described in section 3.1 can output only a finite alphabet of symbols but the analysis vectors are continuously valued - hence each analysis vector must be mapped to a symbol by the process of vector quantisation [6]. This process causes an inevitable loss of accuracy and it will later be seen how HMMs can be extended to handle continuous distributions rather than discrete symbols.

The correspondence of HMM states to articulatory states is by no means clear cut because it is difficult to define what one means by a 'state' in a smoothly produced utterance. The closest physical correspondence to a state might be the articulatory position in a long vowel sound. Although this correspondence is of considerable theoretical interest for future work in modelling speech signals, it is not necessary to attempt to make it for the purposes of performing speech recognition, and it is not of present concern.

There are two problems in the application of HMMs to isolated word speech recognition.

- Given a set of utterances of a vocabulary of words (the training set), construct one or more HMMs for each word (training).

- Given a set of HMMs, one or more for each word in the vocabulary, classify an unknown utterance (recognition).

The recognition problem is solved by computing the likelihood of each model emitting the observation sequence corresponding to the unknown utterance and assigning the unknown utterance to the class of the model which produced the greatest likelihood. The training problem is more difficult, but a powerful algorithm exists (the Baum-Welch algorithm) which guarantees to find a locally optimal model. These two problems are considered in detail in sections 4.1 and 4.2.

4. The recognition problem

The recognition problem is tackled first because it is the more straightforward of the two and the training problem builds on some of the definitions it introduces. Firstly, a more formal definition of some of the notation already used is given below: [2]

S_j	=	j th state of the HMM	$j = 1,2,\ldots,N$
v_k	=	k th output symbol in the alphabet	$k = 1,2,\ldots,M$
a_{ij}	=	$Pr(\text{state } s_j \text{ @ } t+1 \mid \text{state } s_i \text{ @ } t)$	$i = 1,2,\ldots,N$
			$j = 1,2,\ldots,N$
b_{jk}	=	$Pr(\text{outputting symbol } v_k \text{ from state } s_j)$	$j = 1,2,\ldots,N$
			$k = 1,2,\ldots,M$
O_t	=	tth observation (analysis vector)	$t = 1,2,\ldots,T$
$b_j(O_t)$	=	$Pr(\text{outputting observation } O_t \text{ from state } s_j)$	
	=	b_{jk} when $O_t \rightarrow v_k$	
w_i	=	the ith word in the regocnition vocabulary	$i = 1,2,\ldots,W$

The hidden Markov model M is fully defined by the parameter set $[\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}]$. We are given W such models, $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_W$, one for each word in the vocabulary, and an unknown utterance $\boldsymbol{O}$, which consists of a sequence of T observations $O_1, O_{2,\ldots}, O_T$; each O_i is one of the symbols $v_1, v_2, ..., v_M$. Recognition is achieved by computing the likelihood of each model having produced $\boldsymbol{O}$, i.e. by computing $Pr_i(\boldsymbol{O} \mid \boldsymbol{M}_i), i = 1,2,\ldots W$ and assigning $\boldsymbol{O}$ to class k where $P_k = \max_{i=1,2,\ldots W} Pr_i(\boldsymbol{O}|\boldsymbol{M}_i)$

4.1 Baum- Welch recognition

The most obvious way of computing $Pr(\boldsymbol{O}|\boldsymbol{M})$ is to consider every possible sequence of states that could have generated the observation sequence and find the one which produces the highest $Pr(\boldsymbol{O}|\boldsymbol{M})$. However, it is easy to see that this is unrealistic, as in general there are N^T possible sequences. This number is considerably reduced if A is sparse, but it is still impossibly large for practical purposes ($N^T \cong 9 \times 10^{20}$ for $N = 5, T = 30$). Fortunately, a recursive algorithm exists to calculate $Pr(\boldsymbol{O}|\boldsymbol{M})$. The algorithm depends upon calculating the so-called forward probabilities, which are the probabilities of the joint event of emitting the partial observation sequence $O_1, O_{2,\ldots}, O_t$ and

[2] The notation used here generally follows that of [7] and [8].

occupying state s_j at time t. These probabilities are later used in the Baum-Welch training algorithm and so the associated $Pr(\boldsymbol{O}|\boldsymbol{M})$ is denoted by P^{BW}. Let the forward probabilities be denoted by $\alpha_t(j)$ i.e.

$$\alpha_t(j) = Pr(O_1, O_{2,\ldots}, O_t, state\ s_j\ @\ time\ t \mid \boldsymbol{M}) \quad j = 1,2,\ldots N \qquad \ldots.(1)$$

It should then be clear that the required probability is:

$$P^{BW} = \sum_{j=1}^{N} \alpha_T(j) \qquad \ldots.(2)$$

since $\alpha_T(j)$ is the probability of emitting $\boldsymbol{O}$ and ending in state s_j. The α_ts can be computed recursively, as follows. Suppose $\alpha_t(i), i = 1,2,\ldots N$, has been computed at some instant t, then

$$Pr(O_1, O_2, \ldots O_t, state\ s_j\ @\ time\ t+1 \mid state\ s_i\ @ time\ t, \boldsymbol{M}) = \alpha_t(i)a_{ij}$$

Hence the probability of occupying state s_j at time $t+1$ is:

$$Pr(O_1, O_2, \ldots O_t, state\ s_j\ @\ time\ t+1 \mid \boldsymbol{M}) = \sum_{i=1}^{N} \alpha_t(i)a_{ij}$$

Finally, accounting for observation O_{t+1} from state s_j gives:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(O_{t+1}) \qquad t = 1,2,\ldots T-1 \qquad \ldots.(3)$$

Figure 3 illustrates these steps by showing the computation of $\alpha_{t+1}(4)$ for a six state HMM.

Since $\alpha_1(j) = Pr(O_1, state\ s_j\ @ t = 1 \mid \boldsymbol{M})$, the recursion in equation (3) is initialised by setting $\alpha_1(j) = \pi_1 b_j(O_1)$.

4.2 Viterbi recognition

In the above calculation of P^{BW} each α_{t+1}, is calculated by summing contributions of α_t from all states and hence P^{BW} is the likelihood of emitting $\boldsymbol{O}$, summed over all state sequences. The Viterbi algorithm computes the likelihood P^v of the most likely state sequence emitting $\boldsymbol{O}$; by backtracking, this sequence can also be recovered.

This likelihood P^v, and the associated most likely state sequence $\boldsymbol{S}_T$, are found by computing $\phi_t(j)$, where:

$$\phi_{t+1}(j) = \left[\max_{i=1,2,\ldots N} \phi_t(i) a_{ij} \right] b_j(O_{t+1}) \qquad t = 1,2,\ldots T-1 \qquad \ldots.(4)$$

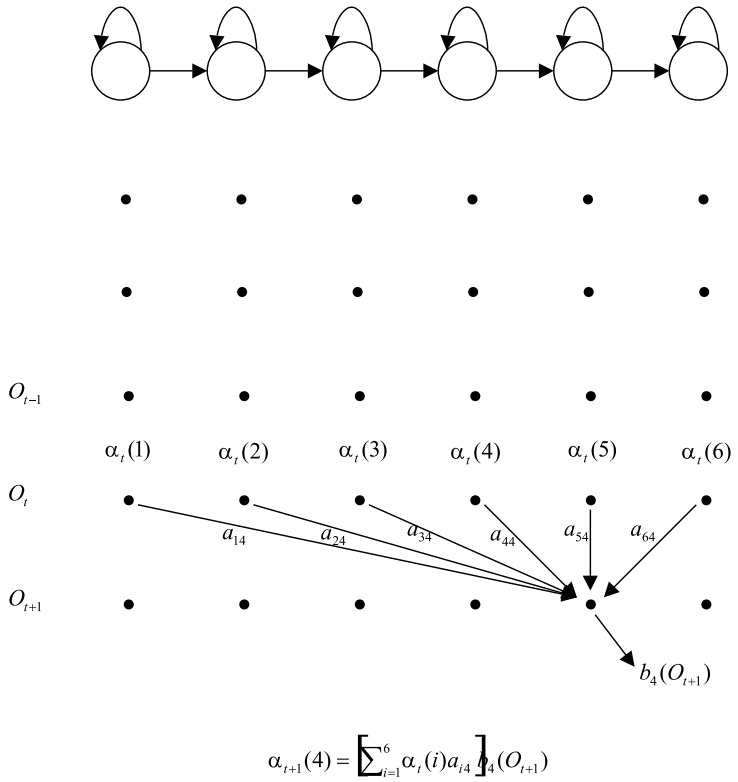


Fig 3 Computation of $\alpha_{t+1}(4)$ for a six state HMM.

Equation (4) is identical to equation (3) except the summation has been replaced by the max operator. The probability of outputting $\boldsymbol{O}$ is then given by:

$$P^v = \max_{j=1,2,\ldots N} \phi_T(j)$$

Compare equation (5) with equation (2). If it is desired to recover the most likely state sequence, $\psi_t(j)$ is also recorded, where $\psi_t(j)$ is the most likely state at time $t-1$ given state s_j at time t. Hence, $\psi_t(j) = i^*$ where i^* is the number of the state which maximises the RHS of equation (4).

Having calculated all the ϕs and ψs for $j = 1,2,\ldots N$ and $t = 1,2,\ldots,T$ the backtracking proceeds as follows: the most likely state at time T is state s_k, where k maximises the RHS

of equation (5); hence $\psi_t(k)$ gives the most likely state at time $T-1$, from which the most likely state at time $T-2$ is found and so on until the most likely state at $t=1$ is recovered. Figure 4 gives an example of the most likely state sequence for a 20 frame utterance and a six state HMM.

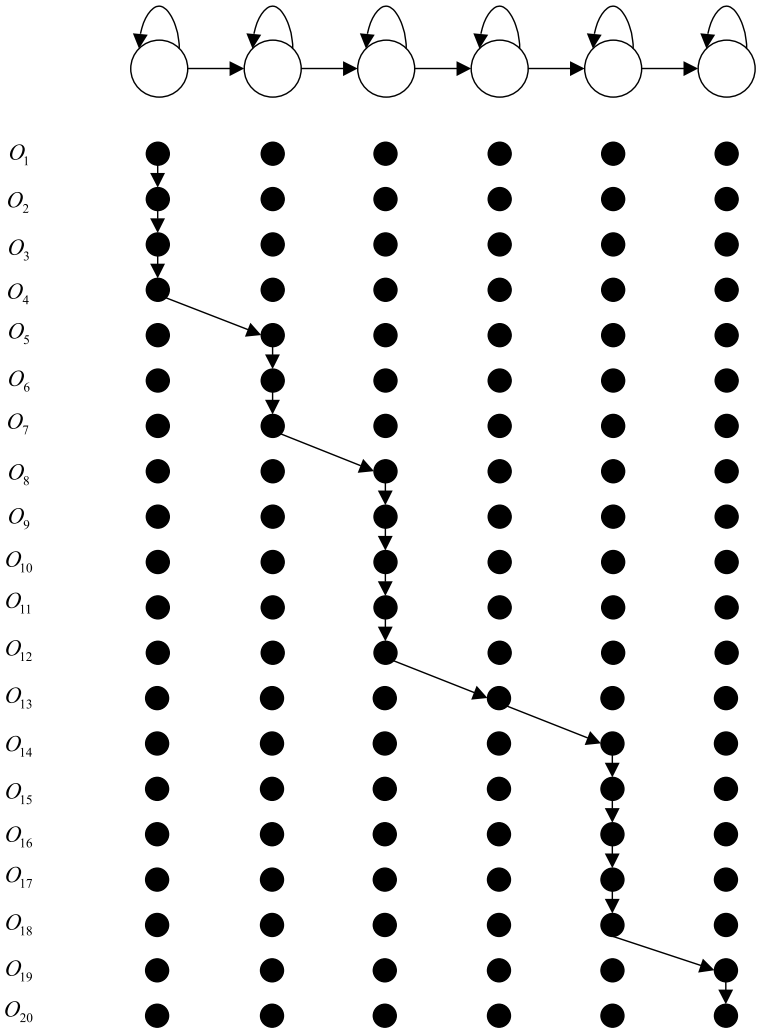


Fig 4 A Viterbi state sequence for an observation sequence of length 20 and a six state hidden Markov model.

Notice that $P^v \leq P^{BW}$, with the equality satisfied only when the state-sequence producing the observations is unique.

5. The training problem

The principal reason for using HMMs in isolated word recognition is to capture statistical information about the variability in patterns representing the same word, and so a training technique that optimises the model for a large number of utterances (observation sequences) is required. However, assume for the moment a single observation sequence $\boldsymbol{O}$ for each word; the extension to multiple observation sequences is straightforward (section 5.2). The essential steps in the training algorithm for a given word are as follows: an initial estimate of $\boldsymbol{M}$ is made; the re-estimation algorithm and $\boldsymbol{O}$ are used to generate a new model $\boldsymbol{M'}$, with the property that $Pr(\boldsymbol{O} \mid \boldsymbol{M'}) \geq Pr(\boldsymbol{O} \mid \boldsymbol{M})$; $\boldsymbol{M'}$ then plays the rôle of $\boldsymbol{M}$ and a new estimate is determined. This process iterates until the inequality in the above expression is arbitarily small. Two reestimation algorithms, the Baum-Welch and Viterbi algorithms, are discussed here.

5.1 The Baum- Welch (forward- backward) algorithm

The underlying idea behind this algorithm is that, given some estimate $\boldsymbol{M}$ of the model and an observation sequence $\boldsymbol{O}$, the best estimates of the parameters of a new model $\boldsymbol{M'}$ are:

$$a'_{ij} = \frac{Pr(\text{transition from } s_i \text{ to state } s_j \mid \boldsymbol{M})}{Pr(\text{transition from } s_i \text{ to any state} \mid \boldsymbol{M})} \qquad \ldots.(6)$$

$$b'_{jk} = \frac{Pr(\text{emitting symbol } \mathrm{v_k} \text{ from state } s_j \mid \boldsymbol{M}}{Pr(\text{emitting any symbol from state } s_j \mid \boldsymbol{M}} \qquad \ldots.(7)$$

$$\pi'_i = Pr(\text{observation sequence begins in state } s_j \mid \boldsymbol{M} \qquad \ldots.(8)$$

The remarkable property of the Baum-Welch algorithm is that the above re-estimates of $\boldsymbol{A}$, $\boldsymbol{B}$ and π are guaranteed to increase $Pr(\boldsymbol{O} \mid \boldsymbol{M})$ until a critical point is reached, at which point the parameters do not change.

To compute these quantities, the forward probabilities $\alpha_t(j)$ are complemented, by defining backward probabilities $\beta_t(i)$ as follows:

$$\beta_t(i) = Pr(O_{t+1}, O_{t+2}, \ldots, O_T, state\ s_i\ @ time\ t \mid \boldsymbol{M})$$

In other words, $\beta_t(i)$ is the probability of starting in state s_i at time t and then completing the observation-sequence (N.B. beginning with observation O_{t+1}, not O_t). The reader should have no difficulty in convincing himself that, using similar reasoning to the calculation of equation (3), $\beta_t(i)$ can be calculated by the following backward recursion:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad t = T-1, T-2, \ldots 1 \qquad \ldots.(9)$$

With $\beta_T(i) = 1, i = 1,2,\ldots,N$. The numerator of equation (6) is then given by:
$Pr(\text{transition from state } s_i \text{ to state } s_j \mid \boldsymbol{M}) =$

$$\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

This is built up as follows: $\alpha_t(i)$ gives the probability of being in state S_i at time t; $a_{ij} b_j(O_{t+1})$ accounts for moving to another state s_j and outputting symbol O_{t+1} from this state; $\beta_{t+1}(j)$ accounts for occupying state s_j at time $t+1$ and then completing the sequence. This probability is summed over all times at which it is possible to make a transition i.e. from 1 to $T-1$ (N.B. not T). To calculate the denominator of equation (6) recall that:

$$\alpha_t(i) = Pr(O_1, O_{2,\ldots}, O_t, state\ s_i\ @\ time\ t \mid \boldsymbol{M})$$
$$\beta_t(i) = Pr(O_{t+1}, O_{t+2}, \ldots, O_T, state\ s_i\ @ time\ t \mid \boldsymbol{M})$$

Hence $\alpha_t(i)\beta_t(i)$ is the probability of occupying state s_i at time t given $\boldsymbol{M}$. So the denominator of equation (6) is:

$$Pr(\text{transition from state } s_i \text{ to any state} \mid \boldsymbol{M}) = \sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)$$

and the complete equation is expressed in terms of the forward and backward probabilities as:

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)} \qquad \ldots.(10)$$

A similar reasoning leads to expressing equation (7) as:

$$b'_{jk} = \frac{\sum_{t \ni O_t = v_k} \alpha_t(j)\beta_t(j)}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)} \qquad \ldots.(11)$$

The summation of the numerator of the above equation is read: 'Sum over all ts at which O_t is symbol V_k'. Finally, the re-estimation of π is shown to be:

$$\pi = \frac{1}{P^{BW}} \alpha_1(i)\beta_1(i) \qquad \ldots.(12)$$

Although equations (6), (7) and (8) seem a plausible way of updating the model parameters, no proof has been given that they are guaranteed to increase $Pr(\boldsymbol{O} \mid \boldsymbol{M})$. The reader interested in rigorous proofs will find them in Liporace [9].

5.2 Training on multiple observation sequences

Equations (10), (11) and (12) are extended to permit re-estimation on many observation sequences by simply considering the definitions in equations (6), (7) and (8) to act over all the observation sequences. However, given a certain model, each observation sequence will have a different P^{BW} and the probabilities from sequences having low P^{BW} will give a disproportionately low contribution in equations (6), (7) and (8) - the result is that the model is optimised only for utterances having high P^{BW}. The solution is to weight all probabilities by $1/P^{BW}$. Denoting equation (10) as $a_{ij}^{'} = N/D$, the change is:

$$a_{ij}^{'} = \frac{\sum_{l=1}^{U} \frac{1}{P_l^{BW}} N_l}{\sum_{l=1}^{U} \frac{1}{P_l^{BW}} D_l} \qquad \ldots.(13)$$

N_l and D_l, are the numerators and denominators formed when processing observation sequence (utterance) O_l, P_l^{BW} is $Pr(\boldsymbol{O}^l \mid \boldsymbol{M})$ and U is the total number of utterances in the training set. Similar alterations-apply to equations (11) and (12). When applied to multiple observation sequences, the Baum-Welch algorithm is guaranteed to increase $\sum_{l=1}^{U} P_l^{BW}$

5.3 Viterbi training

The Viterbi algorithm can be used for model re-estimation as well as for recognition. In this case, the backward probabilities (βs) are not required and there is a significant computational saving. Conceptually, the Viterbi training procedure is simple: the Viterbi algorithm is used to segment each utterance according to the current model,

as in Fig 4. The new values of $[\pi, \boldsymbol{A}, \boldsymbol{B}]$ are then derived directly by examining the numbers of transitions to and from each state and the symbols output by each state. The procedure is as follows.

- Make an initial estimate of the model, $\boldsymbol{M} = \boldsymbol{M}^0$.
- Using model, $\boldsymbol{M}$ execute the Viterbi algorithm on each of the observations $\boldsymbol{O}^1, \boldsymbol{O}^2, \ldots \boldsymbol{O}^U$. Store the set of most likely state sequences produced, $\boldsymbol{S}^1, \boldsymbol{S}^2, \ldots, \boldsymbol{S}^U$ and set

  $$L = \sum_{l=1}^{U} P^V(\boldsymbol{O}^l \mid \boldsymbol{M}).$$

- Use the Viterbi re-estimation equations ((14) (15) and (16) below) to generate a new $\boldsymbol{M}$.
- Iterate the second and third steps until the increase in L upon each iteration is arbitarily small.

The re-estimates are given by considering all the sequences $\boldsymbol{S}^1, \boldsymbol{S}^2, \ldots, \boldsymbol{S}^U$ and setting:

$$\overline{a_{ij}} = \frac{(\text{No of transitions from state } s_i \text{ to state } s_j \mid \boldsymbol{M})}{(\text{Total number of transitions out of state } s_i \mid \boldsymbol{M})} \quad \ldots.(14)$$

$$\overline{b_{jk}} = \frac{(\text{No of emissions of symbol } v_k \text{ from state } s_j \mid \boldsymbol{M})}{(\text{Total number of symbols emitted from state } s_j \mid \boldsymbol{M})} \quad \ldots.(15)$$

$$\overline{\pi_i} = \frac{(\text{No observation sequences begining in state } s_i \mid \boldsymbol{M})}{U} \quad \ldots.(16)$$

Note that weighting by $1/P$ is not required here since numbers of transitions are considered rather than probabilities of transitions. As with the Baum-Welch algorithm, this procedure is guaranteed to increase $\sum_{l=1}^{U} P_l^v$ at each iteration.

6. Extension to continuous probability density functions

So far, it has been assumed that the HMM must output one of a finite alphabet of M symbols. This means that the observation vectors from the speech signal must be quantized, with an inherent loss of accuracy. If it is attempted to minimise this quantisation distortion by using a large number of quantisation symbols, a large number of probabilities in the matrix $\boldsymbol{B}$ must then be re-estimated (for instance, with 128 symbols and 10 states in the HMM, it is necessary to re-estimate 1280 probabilities on each

iteration of the training algorithm). There is rarely enough training data available to estimate such a large number of parameters.

If it can be assumed that the observation vectors from a particular state are drawn from some underlying continuous probability distribution, the information in the corresponding row of $\boldsymbol{B}$ can be replaced by the parameters of this distribution. This is equivalent to making a Normal approximation to some histogram data and replacing the individual probabilities in the histogram by the mean and variance of the Normal distribution.

The question is then immediately raised of how well the observed data fits the parametric distribution. Experimentally, the analysis vectors in a particular state tend to cluster rather than produce a smooth distribution [8]. However, it has been shown that an arbitrary multivariate continuous probability density function (PDF) can be approximated, to any desired accuracy, by mixtures (weighted sums) of multivariate Gaussian PDFs. It is therefore appropriate to model the probability distribution of each state as a Gaussian mixture (although the re-estimation formulae have been proved for any log-concave distribution [9]). The probability $b_j(O_t)$ of emitting an analysis vector O_t from state s_j is then:

$$b_j(O_t) = \sum_{m=1}^{X} c_{jm} N[O_t, \mu_{jm,} \boldsymbol{U}_{jm}] \qquad j = 1,2,\ldots,N$$

$$N[O_t, \mu_{jm,} \boldsymbol{U}_{jm}]$$

where X is the number of mixtures, c_{jm} is the weight of mixture m in state s_j and $N[O_t, \mu_{jm,} \boldsymbol{U}_{jm}]$ is the probability of drawing the vector O_t from a multivariate Normal distribution with mean-vector μ_{jm} and covariance matrix $\boldsymbol{U}_{jm}$ [3].

Clearly, $\boldsymbol{O} \le c_{jm} \le 1$ and $\sum_{m=1}^{X} c_{jm} = 1$.

6.1 Re-estimation with continuous mixture distributions

How does the change from discrete to continuous distributions affect the equations built up in the previous sections? The only change in the equations in sections 4.1-5.2

is that $b_j(O_t)$ is now defined as equation (17), so that once $b_j(O_t)$ has been calculated for each O_t, the recognition algorithms (Baum-Welch or Viterbi) are unaffected.

[3] For readers unfamiliar with multivariate Normal distribution, a brief review is given in Appendix A.

The re-estimation formulae (both Baum-Welch and Viterbi) for $a_{ij}^{'}$ and $\pi_i^{'}$ are similarly only affected by the change in the definition of $b_j(O_t)$. However, the training procedure must now re-estimate $c_{jm}, \mu_{jm}, \boldsymbol{U}_{jm}, m = 1,2,\ldots,X$ for each state s_j.

To cope with a number of mixtures in each state, a third function augments the forward and backward probabilities:

$$\rho_t(j,m) = Pr(O_1, O_2, \ldots, O_t, state\ s_j, mixture\ m @ time\ t \mid \boldsymbol{M}) \qquad \ldots.(18)$$

Comparing equations (1) and (18), it is clear that:

$$\sum_{m=1}^{X} \rho(j,m) = \alpha_t(j)$$

The mixture-weights for each state are then calculated as:

$$c_{mj}^{'} = \frac{Pr\ (\text{mixture} \quad m, \text{state} \quad s_j \mid \boldsymbol{M})}{Pr\ (\text{state} \quad s_j \mid \boldsymbol{M})}$$

$$= \frac{\sum_{t=1}^{T-1} \rho_t(j,m)\beta_t(j)}{\sum_{t=1}^{T-1} \alpha_t(j)\beta_t(j)}$$

The re-estimate of the mean vector of mixture m in state s_j is:

$$\mu_m^{'} = \frac{\sum_{t=1}^{T} \rho_t(j,m)\beta_t(j)O_t}{\sum_{t=1}^{T} \rho_t(j,m)\beta_t O_t} \qquad \ldots.(19)$$

Notice that if $X = 1$ (i.e. a single Gaussian distribution per state), equation (19) becomes:

$$\mu_j^{'} = \frac{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)O_t}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)} \qquad \ldots.(20)$$

$$= \frac{\sum_{t=1}^{T} Pr(\text{state}\ s_j @ t \mid \boldsymbol{M}).O_t}{Pr(\text{state}\ s_j \mid \boldsymbol{M})} \qquad \ldots.(21)$$

The re-estimate of the mean vector of a state is thus an average of each of the T analysis vectors, weighted by the probability of occupying the state at time t. The elements of the covariance matrix are found in the usual way by considering the mean vector μ_{jm} and the T analysis vectors. In practice, a diagonal covariance matrix is usually used because of the difficulty in estimating $d(d+1)/2$ components for each mixture and state with limited training data. Proofs of all the re-estimation equations for the case of a single mixture density per state are given in Liporace [9]. The Viterbi algorithm can also be extended to re-estimate the parameters of continuous mixture distributions.

7. Some considerations of implementation

Sections 2-6 have presented the mathematical theory of HMMs. This section discusses some of the practical issues raised in using HMMs for automatic speech recognition.

7.1 Preventing underflow

It is apparent that many of the equations in sections 4 and 5 will underflow quite rapidly on real computers as they involve recursive products of small probabilities. One solution is to introduce scaling terms at certain points in the recursion [7]. A seemingly more direct method is simply to use logarithms throughout. However, it is necessary to add terms in many of the equations, so a function returning $\log(a+b)$ given $\log a$ and $\log b$ is required. This function requires an exponentiation and a logarithm so that computationally, there is little to choose between these solutions.

7.2 Zero symbol probabilities

When using discrete HMMs with finite training data, it sometimes happens that the probability of observing a certain symbol v_k from state s_j is 0. If this occurs, any α, β or ϕ of an equation in section 5 will be 0, which will be fatal to both recognition and training algorithms. The obvious, although somewhat inelegant solution, is to set the offending probability to some small but non-zero value, ε. This situation has been analysed theoretically by Levinson et al [7] who also ran some experiments on the effect on recognition accuracy of the value of ε [10]. Their conclusions were that the performance was almost identical using values in the range $10^{-10} \le t \le 10^{-3}$, for $M = 128$.

7.3 Initial model estimate

The Baum-Welch re-estimation algorithm is a hill-climbing algorithm which converges to a locally optimal model; hence the final model will depend on the initial model. The question then arises of how to make a good initial estimate of $\boldsymbol{A}$ and the output parameters, $\boldsymbol{B}$ (in the discrete case) or $c_{jm}, \mu_{jm}, \boldsymbol{U}_{jm}$ (in the continuous case). The initial

estimate of $\boldsymbol{A}$ is chiefly determined by the topology chosen for the model, but the initial estimate of the output parameters can be based upon the training data. Rabiner, Levinson et al have shown that continuous HMMs are very sensitive to poor initial estimates of means [11]; it therefore seems wise to base the initial estimate of the output parameters on the available training data.

Two sensible methods of obtaining initial estimates of the output parameters are as follows.

- **Uniform segmentation:** each utterance in the training set is partitioned into N segments of equal length and the analysis vectors in each segment are pooled. In the case of discrete distributions, they are then quantized and an estimate Of b_{jk} is made:

  $$b_{jk} = \frac{\text{No of vectors of symbol } v_k \text{ in segment } j}{\text{Total no of vectors in segment } j}$$

  and covariance matrices are made by applying a clustering algorithm to the vectors of segment j (state s_j) to produce X clusters. μ_{jm} and $\boldsymbol{U}_{jm}$ are then respectively the mean vector and covariance matrix of cluster m in the state.

- **Optimal segmentation:** each utterance in the training set is partitioned into N segments according to the algorithm of Bridle and Sedgwick [12]. This algorithm finds the optimal segmentation of the utterance into N segments in the sense that, if the mean vector of each segment is computed, the sum of the intra-segment variances is minimised. The vectors in each segment are pooled and the estimates are made as described above.
  All elements of $\boldsymbol{A}$ are set initially equal to $1/K_i$ where k_i is the number of allowed transitions (non-zero elements) in row i, and similarly, the first l elements of π may be set to $1/l$.

8. Summary

Hidden Markov models provide a coherent framework for dealing with variability in speech patterns. Although the assumptions about speech made by HMMs are crude, they are offset to a great extent by the availability of powerful optimisation techniques. An attractive feature of HMMs is that it is possible to extend and improve them whilst retaining their mathematical rigour. This can be done in such a way as to utilise some knowledge about the speech signal (see, for instance, Russell and Cook [13]) and this offers promise for future speech recognition algorithms.

HMMs need large amounts of training data to work effectively on large speaker populations, leading to a considerable computational requirement for building the models. However, this is done 'off- line' and recognition of a small vocabulary using

HMMs can be implemented to work in real time. Using some of the techniques described in this paper, a speaker independent recognition algorithm has achieved an accuracy of 98.1 % on a vocabulary of the digits.

Acknowledgements

My thanks are due to Dr M J Russell of the Speech Research Unit, RSRE, for his helpful comments and discussions during the preparation of this paper, and to Dr R K Moore (also of the Speech Research Unit) for permission to reproduce Fig 5.

Appendix A

Dynamic time warping and hidden Markov models

An earlier and still much used pattern matching technique for ASR is dynamic time warping (DTW). It was the desire to extend and generalise DTW that lead to the use of HMMs for ASR and it was quickly realised that DTW is a special case of hidden Markov modelling [14,15].

The science fiction sound of the name of this technique refers to the way in which an utterance is non-linearly 'stretched' or 'compressed' in time to align it with another utterance. Figure 5 shows an example of this process.

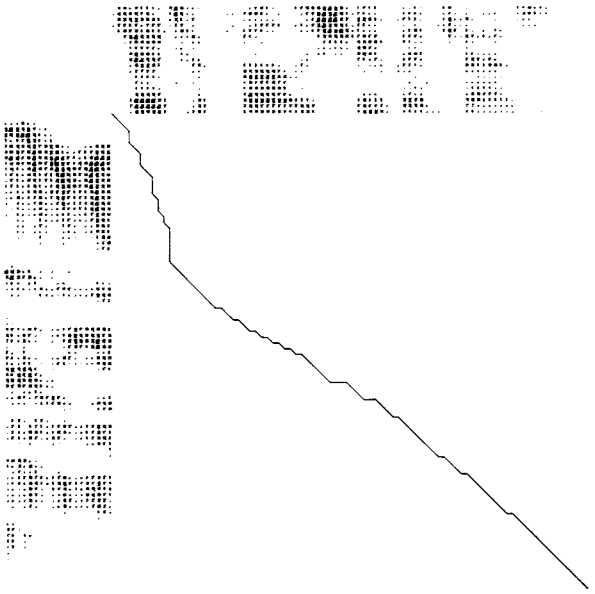


Fig 5 Dynamic time warping alignment of two utterances.

The analysis vectors (frames) of the two utterances U_1, and U_2, of lengths T_1 and T_2 respectively, are positioned with their first frames in the top left corner of the figure and subsequent vectors following in a rightwards and downwards direction respectively. In this case, each frame is a spectrum of the instantaneous speech signal, formed by sampling the outputs of 19 bandpass filters along the audio spectrum. The partially

enclosed rectangle can be thought of as a $T_1 \times T_2$ grid whose i, jth element, $M_{i,j}$, is the distance between frame i of U_1, and frame j of U_2. Commonly used distance metrics are the Euclidean and the City-Block (Manhattan) metric.

The zig-zag path through the grid is known as the optimal time registration path and maps every frame of U_2 to a frame of U_1. If a constraint is imposed on the mapping that the beginnings and the ends of the utterances must coincide, the optimal mapping is a path through the grid which starts in the top left square, ends in the bottom right square, and minimises the total cumulative 'distance' en route. This minimisation is performed by using the technique of Dynamic Programming. Imagine a second grid whose i, jth element $N_{i,j}$ holds the lowest cumulative distance to that position. This element may be calculated as follows:

$$N(i,j) = M(i,j) + \min[(N(i-1,j), N(i-1,j-1), N(i-1,j-2)] \qquad \ldots.(22)$$

In other words, the cumulative distance to element i, j is found by minimising over three possible 'routes' to the element, as shown in Fig 6.

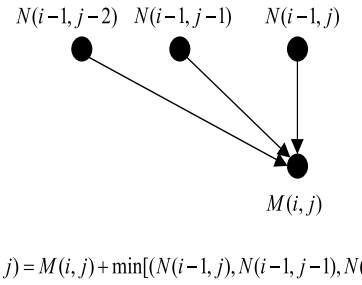


Fig 6 Calculation of the cumulative distance to element (i, j).

The cumulative distance in the bottom right square, $N(T_2, T_1)$, is regarded as a measure of the dissimilarity of the two utterances.

The similarity to the calculation of the Viterbi coefficients (section 4.2) should be obvious. If each frame of U_1, is regarded as an HMM 'state', then the DTW 'path' is equivalent to the 'most likely state sequence' of Fig 4. Furthermore, if the Euclidean metric is used to measure the distance between two frames, this is equivalent in HMM terms to the assumptions of a multivariate Normal distribution with identity covariance matrix for each 'state' (see Appendix B). Notice that in the DTW algorithm, the 'state transition probabilities' are equal and in the algorithm of equation (22), a skip of only one state is allowed. Historically, attempts were made to refine the DTW algorithm by the addition of 'penalties' (which can be interpreted in terms of altering HMM state transition probabilities) and incorporating more complex 'routes' to an element (skipping states in HMM terms).

The chief drawback of DTW is that it has no mechanism for optimising models on large amounts of data, although the technique of 'clustering' [16] goes some way in the direction of HMMs. However, DTW is especially attractive in applications where only limited training data is available, because a single utterance can function as training data for a class.

Appendix B

The multivariate normal distribution

The equation of the PDF of the multivariate Normal distribution is:

$$f(x) = \frac{1}{(2\pi)^{d/2} \mid \boldsymbol{U} \mid^{1/2}} \exp^{\left[-\frac{1}{2}(x-\mu)^T \boldsymbol{U}^{-1}(x-\mu)\right]} \qquad \ldots.(23)$$

Here, d is the dimensionality of the vectors x and μ, and the matrix $\boldsymbol{U}$ (the covariance matrix) is a $d \times d$ symmetrical matrix. The superscript T denotes vector transposition.
Notice that when $d = 1$, the equation collapses to the familiar univariate Normal probability density function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp^{\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]}$$

Notice also that the term $\left[-\frac{1}{2}(x-\mu)^T U^{-1}(x-\mu)\right]$ in equation (23) is a scalar, since it is a $[1 \times d]$ vector $\times$ a $(d \times d)$ matrix $(d \times 1)$ a (d x 1) vector. Computation of this term is particularly simplified if the covariance matrix $\boldsymbol{U}$ is diagonal, when it becomes:

$$-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i - \mu_i)^2}{\sigma_i^{\,2}} \qquad \ldots.(24$$

where $\sigma_i^{\,2}$ is the variance in dimension i. Furthermore, if $\sigma_i^{\,2} = 1$, equation (24) reduces to squared Euclidean distance between x and μ.

In Appendix A, it was stated that if the Euclidean distance was used as a metric between two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, it was equivalent to calculating the probability of observing $\boldsymbol{A}$ from an HMM state with a multivariate Normal distribution, of mean vector $\boldsymbol{B}$ and identity covariance matrix. Because the term in equation (24) is exponentiated in the calculation of the probability, there is, in fact a non-linear relationship between the Euclidean distance and the probability. However, as the exponential is a monotonic function, this relationship is monotonic and hence classification is not affected.

References

1 Lipprnann R P and Gold B: 'Neural net classifiers useful for speech recognition', In Proc 1st Int Conf on Neural Networks, San Diego (June 1987).

2 Devijver P and Kittler J: 'Pattern recognition - a statistical approach', Prentice-Hall International Inc (1982).

3 Dautrich B A, Rabiner L R and Martin T B: 'On the effects of varying filter bank parameters on isolated word recognition', lEEE Transactions on Acoustics, Speech and Signal Processing, 31 , pp 793-807 (August 1983).

4 Russell M J, Moore R K and Tomlinson M J: 'Dynamic programming and statistical modelling in automatic speech recognition', J Opl Res Soc 37(1) , pp 21-30 (1986).

4 Cox D R and Miller H D: 'The theory of stochastic processes', Meuthuen and Co Ltd (1965).

6 Makhoul J, Roucos S and Gish H: 'Vector quantization in speech coding', Proc of the lEEE, 73 pp 1551-1588 (1985).

7 Levinson S E, Rabiner L R and Sondhi M M: 'An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition', The Bell System Technical Journal, 62 , pp 1035-1074 (1983).

8 Rabiner L R and Juang B H: 'An introduction to hidden Markov models', lEEE ASSP Magazine (January 1986).

9 Liporace L A: 'Maximum likelihood estimation for multivariate observations of Markov sources', IEEE Transactions on Information Theory, 28 , pp 729-734 (1982).

10 Levinson S E, Rabiner L R and Sondhi M M: 'On the application of vector quantization and hidden Markov models to speaker-independent, isolated word recognition', The Bell System Technical Journal, 62 , pp 1075-1105 (1983).

11 Rabiner L R, Juang B H, Levinson S E and Sondhi M M: 'Some properties of continuous hidden Markov model representations', AT and T Technical Journal, 64 , pp 1251-1270 (1985).

12 Bridle J S and Sedgwick N S: 'A method for segmenting acoustic patterns, with applications to automatic speech recognition', In Proc lEEE Conf on Acoustics, Speech and Signal-processing, pp 656-659 (1977).

13 Russell M J and Cook A E: 'Experimental evaluation of durational modelling techniques for automatic speech recognition', In Proc lEEE Conf on Acoustics, Speech and Signal-processing, pp 2376-2379 (1987)

14 Bridle J S: 'Stochastic models and template matching: some important relationships between two apparently different techniques for automatic speech recognition', In Proc The Institute of Acoustics (1984).

15 Juang B H: 'On the hidden Markov model and dynamic time warping for speech recognition - a unified view', AT&T Bell Laboratories Technical Journal, 63 , pp 1213-1243 (1984).

16 Wilpon J G and Rabiner L R: 'A modified k-means clustering algorithm for use in isolated word recognition', lEEE Transactions on Acoustics, Speech and Signal Processing, 33 , pp 587-594 (June 1985).