

A PROBABILISTIC STATE MACHINE FOR SPEECH RECOGNITION

António J. Serralheiro*, Luis B. Almeida*

ABSTRACT

This paper proposes a recognition structure designed for handling continuous speech in a natural and computationally efficient way, without the need for a higher level algorithm (like, e.g., level building). This structure is based on a probabilistic state machine (PSM), but unlike Hidden Markov Models, the transition probabilities at each time frame depend on the observation made on the input speech signal, in that frame. Some of the states of the PSM are associated to the various words to be recognized, such that a high probability in one of those states at a given time is interpreted as a high probability that the corresponding word to that state has been found, at that time, in the input signal. This model is highly efficient, requiring only one vector-matrix multiplication per input observation. The theoretical formulations of the recognition and training algorithms are presented, together with some very preliminary experimental results.

INTRODUCTION

One of the most widely used classes of speech recognition algorithms is based on Hidden Markov Models (HMM) (ref 4). This class of algorithms was designed, originally, for isolated words. When used for connected words or continuous speech, it requires some higher level scheme (e.g. level building (ref 3)), with a significant increase in computational requirements (the same applies also to Dynamic Time Warping algorithms (ref 5)). In this paper, we describe a recognition structure designed directly for connected words and continuous speech. This direct approach yields a system which has no loss of computational efficiency in the continuous speech case, relative to the case of isolated words. Like the earlier versions of HMM, the present formulation of this new structure is based, for the time being, on a sequence of discrete-valued observations of the input signal. The discrete observations, also designated input symbols, can be obtained, for example, through a vector quantizer (VQ) (ref 2) operation. Like HMM the new structure is based on a set of states, with probabilistic transitions between them. However, the probabilities of the transitions made upon observation of a given input symbol will now depend on the specific symbol that was observed. This structure can thus be seen as a probabilistic state machine (PSM) which is continuously "driven" by the stream of input symbols. For recognition purposes, some of the states will be assigned to correspond, on a one-to-one basis, with the words to be recognized. The remaining states will be left "free", in order to increase the flexibility of the model. The probabilities of the PSM being at one of the assigned states at a given time will be interpreted as the probability, given by the machine, to having just recognized the word that corresponds to that state. For example, if a machine has states corresponding to the words ONE, TWO and THREE, we would expect that, after suitable training, the temporal behavior of the probabilities of these states would be of the form shown in fig.1, for the input utterance ONETWO (pronounced in a connected manner). Hence, the PSM operates as a continuously running word spotter, which has always available the probabilities of having just recognized each of the words in the vocabulary. As will be seen, the computational demands of this new structure amount to just one vector-matrix multiplication per input symbol, irrespective of whether the recognition is being performed on isolated words or on continuous speech, where the system retains its full computational efficiency. Some preliminary isolated word recognition results will also be presented.

* INESC - Instituto de Engenharia de Sistemas e Computadores Rua Alves Redol, 9-2.
1000 Lisboa PORTUGAL

THE PSM MODEL. RECOGNITION ALGORITHM

We shall assume that we have a probabilistic state machine with N states. Let us designate by s_i^t the probability of being in state i at time t . We define the state vector at time t as

$$S^t = [s_i^t] \quad i=1, \dots, N \quad (1)$$

(we will consider S^t to be a row vector, for notational simplicity). Being probabilities, the s_i^t must satisfy the following constraints

$$s_i^t \geq 0 \quad \text{for all } t, \text{ and for } i=1, \dots, N \quad \text{and} \quad \sum_{i=1}^N s_i^t = 1 \quad \text{for all } t \quad (2)$$

Let K be the size of the observation alphabet, i.e., the number of different symbols that can be observed (e.g. if a VQ is used, K will be the codebook size). Assuming that these symbols are numbered from 1 to K , we shall denote by $k(t)$ the number of the symbol observed at time t . As previously mentioned, the transition probabilities at each instant will depend on the symbol that was observed at that instant. Thus, for each symbol k ($k=1, \dots, K$) we shall have a transition matrix P^k such that

$$P^k = [p_{ij}^k] \quad i, j=1, \dots, N \quad \text{and} \quad k=1, \dots, K \quad (3)$$

where

$$p_{ij}^k = \text{prob}(\text{state } j \text{ at time } t, \text{ given state } i \text{ at time } t-1 \text{ and symbol } k \text{ at time } t) \quad (4)$$

Once again, the elements of P^k are probabilities and thus they must satisfy

$$p_{ij}^k \geq 0 \quad \text{for } i, j=1, \dots, N \quad \text{and} \quad k=1, \dots, K \quad \text{and} \quad \sum_{j=1}^N p_{ij}^k = 1 \quad \text{for } i=1, \dots, N \quad \text{and} \quad k=1, \dots, K \quad (5)$$

The relationship between the state vector at time t , after observation of symbol $k(t)$, and the state vector at time $t-1$, before that observation, is

$$S^t = S^{t-1} P^{k(t)} \quad (6)$$

Assuming that observations have started at $t=1$, and denoting by S^0 the initial state vector,

$$S^t = S^0 \prod_{\tau=1}^t P^{k(\tau)} \quad (7)$$

As previously described, a subset of the machine states will be put in a one-to-one correspondence with the words to be recognized. The probability s_i^t of being in such a state i at time t will be interpreted as the probability assigned by the machine to having just recognized the word that corresponds to state i . Thus, recognition of continuous speech will be performed as follows: input symbols $k(t)$ are observed for each time t , and the machine is allowed to evolve according to eq.(6). Whenever the probability s_i^t of a state i , that has been assigned to some word, goes above a certain threshold, the corresponding word is said to have been recognized at time t . Eq. (6) shows that the processing during recognition amounts to just one vector-matrix multiplication per input symbol, regardless of whether the framework is of isolated words or continuous speech. A consequence of the fact that the PSM "ends" in different states for different words in a continuous input stream, is that the model has some inherent capability of handling coarticulation effects: the transition probabilities from states corresponding to distinct words can be quite different from each other, for the same input symbol.

TRAINING ALGORITHM

The main parameters to be estimated in the PSM model, are the transition matrices P^k and the initial state vector S^0 . One should note that in a "continuously running" machine operating on continuous speech, the influence of S^0 on S^t is expected to fade away quite rapidly as t grows, and thus training of S^0 is not crucial. For isolated words, however, the utterances are sufficiently short that S^0 may have a significant influence on the final state vector, and there may be some advantage to also training S^0 . For simplicity, in what follows, we will first present the training algorithm for isolated words, and only afterwards shall we discuss the case of continuous speech.

ISOLATED WORDS. Consider an utterance from the training set, and let m be the state that corresponds to the word pronounced in that utterance. We would wish that, after inputting the whole sequence of observations from this utterance, to the PSM, the final state vector would be

$$S^T = \hat{S} = [\hat{s}_i] \quad (8)$$

where

$$\hat{s}_i = \begin{cases} 0 & i \neq m \\ 1 & i = m \end{cases} \text{ for } i=1, \dots, N \quad (9)$$

and T is the utterance length. This result would mean that the word had been perfectly recognized. We will thus define, for each utterance, a difference vector

$$D = S^T - \hat{S} \quad (10)$$

and an error vector

$$E = f(D) \quad (11)$$

where f is an element-by-element function, i.e.,

$$e_i = f(d_i) \quad i=1, \dots, N \quad (12)$$

whose purpose is to allow us to nonlinearly weight the errors. We shall now define the scalar error

$$\epsilon_l = E_l \Lambda E_l' \quad (13)$$

where the index l indicates the dependency on the specific utterance being considered (we have dropped this dependency in eqs. 8-12 for simplicity), E' denotes E transpose, and Λ is an $N \times N$ diagonal matrix which allows us to give different weights to the error components from different states. This weighting is useful since we want to give more importance to the error components from the words assigned to them, than to those from unassigned states (the latter may even be assigned zero weight). The total error, that we want to minimize, is given by

$$\epsilon = \sum_{l=1}^L \epsilon_l \quad (14)$$

where L is the number of utterances in the training set. The steepest descent method we shall use is a simple gradient descent algorithm, and thus the parameter update equations are

$$\mathbf{P}_{n+1}^k = \mathbf{P}_n^k - \alpha \sum_{l=1}^L \nabla_{\mathbf{P}^k}(\epsilon_l)_n \quad (15)$$

and

$$\mathbf{S}_{n+1}^0 = \mathbf{S}_n^0 - \alpha \sum_{l=1}^L \nabla_{\mathbf{S}^0}(\epsilon_l)_n \quad (16)$$

where ∇_X denotes the gradient relative to the components of X , α is a parameter controlling the descent step size, and n is the optimization iteration number. Before proceeding, one should note that the new matrix \mathbf{P}_{n+1}^k obtained from eq. (17) generally will not satisfy conditions (5), and thus it must be replaced by the closest element from the subspace of legal transition matrices. The same comments also apply to \mathbf{S}_{n+1}^0 and conditions (2). Consider now the symbol $k(t)$, occurring at a specific time t in utterance l , and assume for the time being, that it is not repeated at any other time in the same utterance (the dependency of $k(t)$ on l has been dropped for notational simplicity). Then, it is possible to show that

$$\nabla_{\mathbf{P}^k(t)}(\epsilon_l)_n = 2(\mathbf{S}_l^{t-1})' [\mathbf{E}_l \Lambda \mathbf{F}_l \prod_{\tau=T}^{t+1} (\mathbf{P}^k(\tau))'] \quad (17)$$

where \mathbf{S}^{t-1} is the state vector obtained in utterance l at time $t-1$, the prime again denotes transposition, and \mathbf{F}_l is an $N \times N$ diagonal matrix with elements

$$f_{ii} = \dot{f}(d_i) \quad i=1, \dots, N \quad (18)$$

\dot{f} being the derivative of f (cf. eq. 11). All of the factors in the right hand side of eq. (19) should be computed using the matrices \mathbf{P}^k and the vector \mathbf{S}^0 from the n -th iteration. Note that $\nabla_{\mathbf{P}^k}$ is a square matrix, obtained from the product of a column and a row vectors. If symbol k occurs more than once in utterance l , the corresponding gradient is the sum of the results obtained by applying eq. (19) for all instants t in which that symbol occurs. One can also show that

$$\nabla_{\mathbf{S}^0}(\epsilon_l)_n = 2\mathbf{E}_l \Lambda \mathbf{F}_l \prod_{\tau=T}^1 (\mathbf{P}^k(\tau))' \quad (19)$$

CONTINUOUS SPEECH. In the continuous speech case, each utterance will need to have its word boundaries hand marked, and the desired temporal evolution of the probabilities will be of the type suggested by fig. 1. In this case, one cannot train only at the end of each word, because if during recognition a high probability would occur in one of the assigned states somewhere in the middle of an input word, this would be interpreted as a(n) (erroneous) word recognition. Thus, errors ϵ will be computed for all observation instants within each word, and their sum will be minimized relative to all matrices \mathbf{P}^k that have occurred in that word. Adaptation of the previous equations to this case is quite straightforward.

VARIATIONS OF THE MODEL

The PSM, as previously described, is in its more general form, and it may involve a very large number of parameters. In fact, each matrix \mathbf{P}^k has $N \times N$ elements and there are K different matrices. As N must be somewhat larger than the vocabulary size, the total number of parameters can indeed be very large, raising problems of training data sufficiency, and storage size. Placing restrictions on the allowable transitions can make the number of parameters depend linearly, instead of quadratically, on the vocabulary size. A linear dependency can also be obtained in a different way, by departing from the previous formulation of a single machine for recognizing all the words in the vocabulary, and using

one PSM for each word instead. Note that in this case each PSM should be trained to recognize its corresponding word, and not to recognize other words, i.e., it will be trained to discriminate between the word it must recognize and other words. The one-PSM-per-word formulation is, however less appealing in theoretical terms, than the single PSM one, since it has no interaction between the recognizers of different words; it may even happen that, at some instant, the recognition probabilities of two or more words are high, what is impossible in the single-PSM formulation.

EXPERIMENTAL RESULTS

Only a few preliminary recognition results are presently available. These are results of tests whose main purpose was to better understand the operation of the recognition and training algorithms. They were all performed on isolated words, a framework which was deemed more appropriate for preliminary tests, since it is simpler and easier to control. Two tests (designated by A and B) were performed. Test A consisted of 5 utterances of each of the 10 English digits, spoken by a male speaker. Test B consisted of 5 utterances by each of two speakers, a male and a female, of each of the 10 digits. A coarse vector quantization (64 codewords) of the LPC shape (no energy information) was used to obtain the input symbols. In test A a single 15-state PSM was used for the ten digits, while in test B, ten 5-state PSM were used, one for each digit. Both tests report recognition scores on the training and testing data sets. In test A, two different training procedures were tested, one (denominated *final*) as previously described; the other (*smooth*), postulates a smooth variation of the recognition probabilities, starting at 0 in the beginning of the word, and reaching a value of 1 by the end of the word. In this case, training was based on the whole sequence of state vectors, as previously discussed, in an attempt to simulate some of the conditions of the training for continuous speech recognition. Training and test sets, for test A, were similar to, but disjunct from each other. In test B, the training set consisted in 5 utterances of the word that the PSM should recognize, and one utterance of each of the other words. The test set consisted in 5 utterances of each of the 10 digits. The recognition criteria in both tests involved some form of smoothing (averaging) of the last 50% (approx.) of the state vectors in each utterance. Table 1 summarizes the results of these tests. The results of table 1 show scores compatible with the coarse quantization used, and suggest that some form of smoothing may be beneficial.

CONCLUSIONS

We have presented a model for speech recognition, that has the potential to handle continuous speech in a natural and computationally efficient way, and to deal with coarticulation effects. This model is based on a probabilistic state machine (PSM) continuously driven by the observation sequence. Preliminary tests on isolated words are indicative of a good recognition performance. However, much more testing, both on isolated words and on continuous speech, must be performed, to confirm in practice the model validity.

BIBLIOGRAPHY

1. Bahl, L. R., Jelinek, F. *Decoding for Channels with Insertions, Deletions, and Substitutions with Applications to Speech Recognition*. IEEE Trans. Inform. Theory, pp 404-411, July 1975.
2. Buzo, A., Gray, A., Gray, R., Markel, J. *Speech Coding Based upon Vector Quantization*. IEEE Trans. ASSP., pp 562-574, October 1980.
3. Levinson, S. E. *Structural Methods in Automatic Speech Recognition*. Proc. IEEE, pp 1625-1650, November 1985.
4. Levinson, S. E. Rabiner, L. R., Sondhi, M. M. *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition*. BSTJ, pp 1035-1073, April 1983.

5. Rabiner, L. R., Juang, B. H. *An Introduction to Hidden Markov Models*. IEEE ASSP Magazine, pp 4-16, January 1986.
6. Rabiner, L. R., Levinson, S. E. *Isolated and Connected Word Recognition - Theory and Selected Applications*. IEEE Trans. Comm. vol COM-29, pp 621-659, 1981.

Recognition score (%)		
descr.	A	B
train, final	100	100
train, smooth	100	-
test, final	100	90
test, smooth	92	-

Table 1- Recognition results.

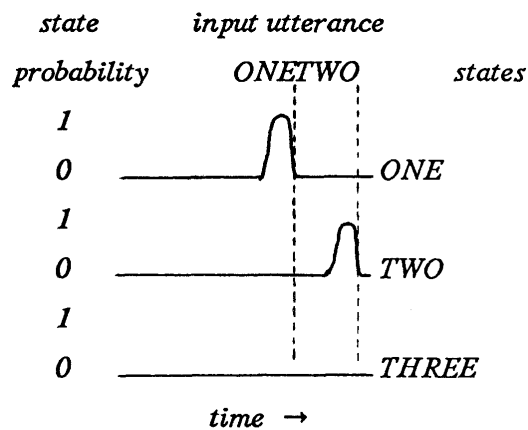


Fig. 1- Idealized example of word recognition by a PSM