Links Between Markov Models and Multilayer Perceptrons

H. Bourlard[1,2] and C. J. Wellekens[2]

TR-88-008

November 10, 1988

ABSTRACT

Hidden Markov models are widely used for automatic speech recognition. They inherently incorporate the sequential character of the speech signal and are statistically trained. However, the a priori choice of a model topology limits the flexibility of the HMM's. Another drawback of these models is their weak discriminating power.

Multilayer perceptrons are now promising tools in the connectionist approach for classification problems and have already been sucessfully tested on speech recognition problems. However, the sequential nature of the speech signal remains difficult to handle in that kind of machine.

In this paper, a discriminant hidden Markov model is defined and it is shown how a particular multilayer perceptron with contextual and extra feedback input units can be considered as a general form of such Markov models. Relations with other recurrent networks commonly used in speech recognition are also pointed out.

[1]International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, California 94704-1105, USA.

[2]Philips Research Laboratory Brussels, Av. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium.

Links between Markov Models and Multilayer Perceptrons

H. Bourlard [†,‡] & C.J. Wellekens [†]

([†])Philips Research Laboratory Brussels
Av. Van Becelaere 2, Box 8,
B-1170 Brussels, Belgium.

([‡])International Computer Science Institute
1947 Center Street, Suite 600
Berkeley,
CA 94704 USA.

1 Introduction

Hidden Markov models (HMM) [1],[3],[14] are widely used for automatic isolated and connected speech recognition. Their main advantages lie in the ability to take account of the time sequential order and variability of speech signals. However, the a priori choice of a model topology (number of states, probability distributions and transition rules) limits the flexibility of the *HMM*'s, in particular speech contextual information is difficult to incorporate. Another drawback of these models is their weak discriminating power. This fact is clearly illustrated in [6],[32] and several solutions have recently been proposed in [2],[4],[6],[8],[20],[21].
In Section 2, the discriminant training is placed in relation to the standard algorithms as *Maximum Likelihood Estimation* and *Viterbi* . A modified *HMM* which satisfies discrimination requirements, is defined as a target model in Section 2.4.

The *multilayer perceptron (MLP)* is now a familiar and promising tool in connectionist approach to classification problems [15],[16],[27] and has already been widely tested on speech recognition problems [5],[6],[9],[32],[34]. However, the sequential nature of the speech signal remains difficult to handle with *MLP*. Section 3 shows how an *MLP* with extra feedback input units can be considered as a form of discriminant *HMM* defined in Section 2 with particular emission and transition probabilities. In the *HMM* formalism, the speech signal is supposed to be produced by a (first order) Markov source for which the probability of reaching a particular state depends entirely on the previous state(s) and on the

observed acoustic vectors associated with the speech time slots. This probability plays a role similar to the local distance (with associated weights) between an acoustic vector of the test utterance and that of a reference template. In the sequel, it will be occasionally referred to as *local contribution* in contrast with the global probability which measures the dissimilarity between the whole observed utterance and the model.
In Section 3.2.1, it is shown that the same local contributions could be generated by an *MLP* feeding back the output values associated with the previous input frames and thus allowing a straightforward generalization to high order Markov models. It is also shown how a new definition of the input, as in a NETtalk machine [28], permits an easy capture of the contextual information of speech (Section 3.2.2). The generated local contributions can be used in a dynamic programming algorithm to achieve discriminant recognition (Section 3.2.3).
Relations with other recurrent *MLP* [10],[34] are also briefly pointed out in Section 3.3. An advantage of this *MLP* with output feedback, over and above the link with *HMM*, is the possibility during the training to supply the feedback input units with the correct information. It is also shown how context-sensitive networks [32] approximate recurrent networks over a finite time interval.

2 Stochastic models

2.1 Training criteria

Stochastic speech recognition is based on the comparison of an utterance to be recognized with a particular probabilistic finite state machine known as *hidden Markov model (HMM)*. This model is generally the concatenation of sub-unit models: a sentence model is the concatenation of word models (or even of phoneme models if the goal of the processing is phonemic labeling) and a word model is the concatenation of phoneme models. The sub-unit models are trained on a segmented (this is not required in case of embedded training) and labeled speech data base. In this training phase, the probability $P(W_i|X)$ that model W_i has produced the associated utterance X must be maximized but the parameter space which this optimization is performed over makes the difference between independently trained models and *discriminant* ones.
Indeed, the probability $P(W_i|X)$ can be written as

$$P(W_i|X) = \frac{P(X|W_i).P(W_i)}{P(X)} . \qquad (1)$$

In a recognition phase, $P(X)$ may be considered as a constant since the model parameters are fixed but, in a training phase, this probability depends on the parameters of all possible models. This can be made explicit by taking into

account the fact that the models are mutually exclusive so that

$$P(X) = \sum_k P(X|W_k)P(W_k) \tag{2}$$

where the summation extends over all models (all possible word or phoneme sequences). Substituting (2) in (1) gives

$$P(W_i|X) = \frac{P(X|W_i)P(W_i)}{P(X|W_i)P(W_i) + \sum_{k \neq i} P(X|W_k)P(W_k)}, \tag{3}$$

Maximization of $P(W_i|X)$ as given by (3) is usually simplified by restricting it to the subspace of the W_i parameters. This restriction leads to the *Maximum Likelihood Estimators (MLE)*. Indeed, maximization of $P(X|W_i)$ implies that of its bilinear map (3) since the summation term in the denominator is constant over the parameter space of W_i. A language model provides the value of $P(W_i)$ independently of the acoustic decoding [14]. This model by model optimization allows important simplifications in the training algorithms by avoiding the computations of all rival sequences but at the price of a loss in discriminating power.

On the other hand, maximization of $P(W_i|X)$ with respect to the whole parameter space (i.e. the parameters of *all* models $W_1, W_2, \ldots$) leads to discriminative models since it implies that the contribution of $P(X|W_i)P(W_i)$ should be enhanced while that of the rival models, represented by $\sum_{k \neq i} P(X|W_k)P(W_k)$, should be reduced. This maximization with respect to the whole parameter space has been shown equivalent to the maximization of *Mutual Information (MMI)* between a model and a vector sequence [2],[8],[21]. This justification still holds in case of embedded training (compulsory for the t raining of phonemic models) where discrimination is increased between the various word or sentence models W_k (and no longer between sub-unit models) by adjusting the parameters of the constituting sub-unit models which may appear several times and in several different W_k.

2.2 Training algorithms

Two different algorithms are used for the training of *HMM*'s [1],[3]: the *Baum-Welch* or the *Viterbi* algorithms. In the first one, the total probability to produce an acoustic vector sequence $X = \{x_1, \ldots, x_N\}$ on a given model is maximized while the *Viterbi* criterion focuses only on the best path through the model.

The *Baum-Welch* algorithm which iteratively provides parameter estimators based on partial path probabilities computed by forward and backward recurrences, is used to maximize the *MLE* algorithm. Convergence (at least to local optima) is then guaranteed.

This advantage is unfortunately lost if the *MMI* criterion is used [2] and a gradient method is generally preferred for the optimization; the forward and

backward recurrences can still be used to compute the gradient [8]. If phoneme models are trained, a *looped phonetic model* (as already described in [35] for a recognition task) may generate all possible phoneme sequences and provides thus the value of the probability of producing all rival sequences while the numerator of (3) is obtained in a second step via a serial model [21]. This method could in principle also apply to word model training but would require an excessive computation time for a large lexicon due to the size of the looped word model. The *MMI* criterion is thus particularly unsuitable for embedded training of word models and severe hypotheses must be accepted to cope with the complexity [8].

The *Viterbi* algorithm, which is the main tool for the recognition task, is also often used in the training phase. The parameters are updated so as to increase the probability of the most probable path and $P(X|W)$ is thus not actually maximized. It is much faster than the *Baum-Welch* algorithm and convergence (at least to local optima) is also guaranteed. It is well adapted to a simplified form of *MLE* optimization (considering the best path only) but not at all to *MMI* maximization. Indeed, this criterion requires to take account of all paths (not only the best one) between phonemes to generate all possible rival sequences.

2.3 Standard hidden Markov models

In the classical (i.e. non discriminant) discrete *HMM*, the acoustic vectors are quantized in a front-end processor and each one is replaced by the closest (according to an Euclidean norm) prototype vector y_i selected in a predetermined finite set $\mathcal{Y}$ of cardinality I. Let $\mathcal{Q}$ be a set of K different states $q(k)$ with $k = 1, \ldots, K$. Sub-unit models are constituted by the association of some states and speech utterances are synchronized with paths from state to state in the model by using local properties as transition or emission probabilities defined hereunder. Models must be optimized with respect to these local parameters according to a *Viterbi* or *MLE* criterion which basically relies on the maximization of $P(X|W)$ where X is a training sequence of quantized acoustic vectors $x_n \in \mathcal{Y}$, with $n = 1, \ldots, N$ and W is its associated Markov model (concatenation of sub-unit models) made up of L states $q_\ell \in \mathcal{Q}$ with $\ell = 1, \ldots, L$. The same state may occur several times and with different indices ℓ, so that $L \neq K$. Let us denote the presence on state q_ℓ at a given time $n \in [1, N]$ by q_ℓ^n. Since events q_ℓ^n are mutually exclusive, probability $P(X|W)$ can be written for any arbitrary n:

$$P(X|W) = \sum_{\ell=1}^{L} P(q_\ell^n, X|W) \,. \tag{4}$$

In (4), $P(q_\ell^n, X)$ denotes the probability that X is produced by W while visiting state q_ℓ at time n and can also be written as:

$$P(q_\ell^n, X|W) = P(q_\ell^n, X_1^n|W).P(X_{n+1}^N|q_\ell^n, X_1^n, W)\,,$$

where X_m^n denotes the vector sequence $x_m, x_{m+1}, ..., x_n$. Trivial manipulations show that recurrence (5) holds:

$$P(q_\ell^n, X_1^n|W) = \sum_{k=1}^{L} P(q_k^{n-1}, X_1^{n-1}|W).p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, W) \qquad (5)$$

which is the forward recurrence of the *Baum-Welch* algorithm. The conditional probability $p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, W)$ in (5) is the *local contribution*. The role of the conditional events are usually limited by some relaxing hypotheses. For example, the dependence is restricted to the last emitted vector; this could be achieved by concatenating two successive vectors in [12] and in [18],[19], while an explicit formulation is given in [36]. In classical *HMM* [3],[14], the local contribution is assumed independent of the whole previously emitted vector sequence X_1^{n-1} and reduces thus to $p(q_\ell^n, x_n|q_k^{n-1}, W)$. It represents the probability to make a transition from state q_k to q_ℓ while emitting the vector x_n. The set of all sub-unit Markov models are thus characterized by $I \times K^2$ parameters

$$p[q(\ell), y_i|q^{(-)}(k), W]\,,$$

for $i = 1, ..., I$ and $k, \ell = 1, \ldots, K$. Notations $q^{(-)}(k)$ and $q(\ell)$ denote states $\in \mathcal{Q}$ observed at two consecutive instants.

The update formulas for the *Baum-Welch* algorithm have been described elsewhere [3],[14] a.o. and are thus not recalled here.

The *Viterbi* criterion can be viewed as a simplified version of the *MLE* criterion where, instead of taking account of all possible state sequences in W capable of producing X, one merely considers the most probable one. To make apparent all possible paths, (4) can also be rewritten as

$$P(X|W) = \sum_{\ell_1=1}^{L} \ldots \sum_{\ell_N=1}^{L} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, X|W)\,.$$

An explicit formulation of the *Viterbi* criterion is obtained by replacing all summations by a " *max* " operator. Probability (4) is then approximated by:

$$\bar{P}(X|W) = \max_{\ell_1,\ldots,\ell_N} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, X|W)\,. \qquad (6)$$

The same modification in (5) provides the basic formula of the *dynamic time warping (DTW)* process [3]:

$$\bar{P}(q_\ell^n, X_1^n|W) = \max_k \left[\bar{P}(q_k^{n-1}, X_1^{n-1}|W).p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, W)\right]\,. \qquad (7)$$

The global probability $\bar{P}(q_\ell^N, X|W)$ for all ℓ is computed by using (7) recursively. Then, the optimal final state is pointed out by that particular ℓ which maximizes the global probability (also referred to as matching score) and the associated best path can be recovered by classical backtracking. Each training vector is then unequivocally associated with only one particular transition. Let $n_{ik\ell}$ denote the number of times each prototype vector y_i has been associated with a transition $\{q(k) \rightarrow q(\ell)\}$ between two states $\in \mathcal{Q}$ during the training sequence X; clearly, with this definition, $n_{ik\ell}$ sums up all transitions wherever they appear in the model of X. The estimators of the probabilities $p[q(\ell), y_i|q^{(-)}(k), W]$ effectively used in the classical *HMM* and which guarantee the convergence of the *Viterbi* training are simply given by:

$$\hat{p}[q(\ell), y_i|q^{(-)}(k), W] = \frac{n_{ik\ell}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jkm}}, \quad \forall i \in [1, I], \quad \forall k, \ell \in [1, K], \quad (8)$$

and thus:

$$\sum_{i=1}^{I}\sum_{\ell=1}^{K} \hat{p}[q(\ell), y_i|q^{(-)}(k), W] = 1, \quad \forall k \in [1, K].$$

The local probability (8) is often split into a product of a transition probability and an emission probability (transition emitting models) [14] of which estimators are:

$$\hat{p}[q(\ell)|q^{(-)}(k), W] = \frac{\sum_{j=1}^{I} n_{jk\ell}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jkm}}, \quad (9)$$

and

$$\hat{p}[y_i|q(\ell), q^{(-)}(k), W] = \frac{n_{ik\ell}}{\sum_{j=1}^{I} n_{jk\ell}}. \quad (10)$$

A further and usual simplification is to assume that the emission probabilities only depend on the current state $q(\ell)$ (state emitting models). Its estimator is then

$$\hat{p}[y_i|q(\ell), W] = \frac{\sum_{m=1}^{K} n_{im\ell}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jm\ell}}. \quad (11)$$

and the product of (9) by (11) is different from (8) due to the additional assumption on the emission probability.

If the models are trained by using this formulation of the *Viterbi* algorithm, no discrimination is taken into account. For instance, it is interesting to observe that the local probability (8) is not the suitable measure to obtain a correct labeling of a prototype vector y_i or, in other words, to find the most probable associated state given a specified previous state. Indeed, the decision must ideally be based on the Bayes rule [11] or more explicitly, the correct state will be $q(\ell_{opt})$ such that

$$\ell_{opt} = \underset{\ell}{\operatorname{argmax}} \; p[q(\ell)|y_i, q^{(-)}(k)], \quad (12)$$

and not

$$\ell_{opt} = \underset{\ell}{\text{argmax}} \; p[q(\ell), y_i | q^{(-)}(k)] .$$

In particular, it is thus necessary that $\sum_{\ell=1}^{K} p[q(\ell)|y_i, q^{(-)}(k)] = 1$. As in classical *HMM*, these probabilities are related to local contributions. Indeed, this discriminant local probability can be written

$$p[q(\ell)|y_i, q^{(-)}(k)] = \frac{p[y_i, q(\ell)|q^{(-)}(k)]}{p[y_i|q^{(-)}(k)]} . \tag{13}$$

Summing (8) on ℓ yields an estimator of $p[y_i|q^{(-)}(k)]$

$$\hat{p}[y_i|q^{(-)}(k)] = \frac{\sum_{m=1}^{K} n_{ikm}}{\sum_{j=1}^{I} \sum_{m=1}^{K} n_{jkm}} . \tag{14}$$

By (8),(13) and (14), an estimator of the discriminant local probability is :

$$\hat{p}[q(\ell)|y_i, q^{(-)}(k)] = \frac{n_{ik\ell}}{\sum_{m=1}^{K} n_{ikm}} , \tag{15}$$

and sums up to unity as required. This probability was already used in the ERIS system described in [18],[19] and it will be shown in Section 3.2.1, that the optimal output values of a *Multilayer Perceptron (MLP)* are the estimates of these discriminant local probabilities.

2.4 Discriminant HMM

For vector quantized speech inputs and *Viterbi* criterion, an alternative *HMM* using discriminating local probabilities can also be described on the basis of the arguments described here above. Indeed, the actual objective in recognition is to find the model W_i which maximizes the probability (1) when observing X. Comparing with (6), the "Viterbi formulation" of this probability is

$$\bar{P}(W_i|X) = \max_{\ell_1,\ldots,\ell_N} P(q^1_{\ell_1}, \ldots, q^N_{\ell_N}, W_i|X) . \tag{16}$$

Expression (16) clearly puts the best path into evidence. The right hand side factorizes into

$$P(q^1_{\ell_1}, \ldots, q^N_{\ell_N}, W_i|X) = P(q^1_{\ell_1}, \ldots, q^N_{\ell_N}|X).P(W_i|X, q^1_{\ell_1}, \ldots, q^N_{\ell_N}) .$$

and suggests two separate steps for the recognition. The first factor represents the acoustic decoding in which the acoustic vector sequence is converted into a sequence of states. Then, the second factor represents a phonological and lexical step: once the sequence of states is known, the model W_i associated with X can be found from the state sequence without an explicit dependence on X so that

$$P(W_i|X, q^1_{\ell_1}, \ldots, q^N_{\ell_N}) = P(W_i|q^1_{\ell_1}, \ldots, q^N_{\ell_N}) .$$

For example, if the states represent phonemes, this probability must be estimated from phonological knowledge of the vocabulary once for all in a separate process without any reference to the input vector sequence. On the contrary, $P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X)$ is immediately related to the discriminant local probabilities and may be factorized in

$$P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X) = p(q_{\ell_1}^1|X).p(q_{\ell_2}^2|X, q_{\ell_1}^1)\ldots p(q_{\ell_N}^N|X, q_{\ell_1}^1, \ldots, q_{\ell_{N-1}}^{N-1}). \quad (17)$$

Now, each factor of (17) may be simplified by relaxing the conditional constraints; specifically, the factors of (17) are assumed dependent on the previous state only and on a signal window of length $2p+1$. The current expression of these local contributions becomes

$$p(q_{\ell_i}^i|X, q_{\ell_1}^1, \ldots, q_{\ell_{i-1}}^{i-1}) = p(q_{\ell_i}^i|X_{i-p}^{i+p}, q_{\ell_{i-1}}^{i-1}) , \quad (18)$$

where *input contextual information* is taken into account. In Section 3.2.1, a parallelism between these local contributions and the optimal outputs of a multilayer perceptron with contextual input and output feedback will be pointed out.

If input contextual information is neglected ($p = 0$), equation (18) represents nothing else but the discriminant local probability (15) and is the base of a *discriminant discrete HMM*. Each observed acoustic vector is in relation with a transition so that no separate emission and transition probabilities as (9) and (10) or (11) are defined. Moreover, when comparing with the classical transition emitting *HMM*, the main difference lies in the normalization of the local contribution which is performed on the set of states and not on the prototype vector space as done for (10) and (11).

A phoneme or a word model built up from several states must have a final state deprived of selfloops as already pointed out by [14]; assuming this to be done, any concatenation of models is possible by assimilating this final state with the initial state of the next model. In that case, the one stage dynamic programming [7],[22] applies exactly as in the classical case except that discriminant local probabilities are used. Inside a sub-unit model, the following dynamic programming recurrence holds

$$\bar{P}(q_\ell|X_1^n) = \max_k \left(\bar{P}(q_k|X_1^{n-1}).p(q_\ell|x_n, q_k)\right) , \quad (19)$$

where parameter k runs over all possible states preceding q_ℓ and $\bar{P}(q_\ell|X_1^n)$ denotes the cumulated best path probability of reaching state q_ℓ and having emitted the partial sequence X_1^n .

Besides the advantage of forcing discrimination, numerical problems which plague the classical *HMM* are avoided when using discriminant models: namely, the lack of balance between the transition probability values (9) which only depend on the topology of the model and the emission probability values which

decrease with the number of prototype vectors I. This effect worsens if context dependence is introduced by using $2p+1$ appended consecutive vectors as features. Indeed, the number of possible prototype combinations (and thus the number of discrete emission probabilities for each state) grows as I^{2p+1} thereby decreasing the values of the emission probabilities.
Unfortunately, even with discriminant models, the exponential increase of the number of parameters with the width $2p+1$ of the window is not avoided, resulting in a need of huge storage capacity and requiring an excessive size of training data to obtain statistically significant parameters. Moreover, transitions unobserved in the training set will yield zero probabilities despite the fact that they may actually occur in a recognition phase. To cope with insufficiently large data sets, interpolation techniques can be used [1] and a lower bound can be imposed to the local probabilities. Anyway, a waste of required memory still occurs since a lot of explicitly stored discriminant local probabilities reach the lower bound. Indeed, the likelihood of actually observing most of the I^{2p+1} possible input vectors is extremely weak.

In Section 3, it is shown how all of these drawbacks are circumvented or attenuated by taking advantage of the generalization properties of the *multilayer perceptron (MLP)* when used as a discriminant local probability generator.

3 MLP and discriminant local probabilities

In this section, links are set up between the discriminant discrete *HMM* defined in Section 2.4 and the *MLP* approach. To this aim, a *MLP* with output feedback is described in Section 3.2. Moreover, the easy extension of the *MLP* input makes possible the use of contextual information what is a supplementary advantage of *MLP* versus that kind of *HMM*.
In the sequel, the concept of states is replaced by that of classes.

3.1 Sequential processing architectures with context-sensitive inputs

Let $q(k)$, with $k = 1, \ldots, K$, be the output units of an *MLP* associated with different classes (each of them corresponding to a particular word, phoneme or *HMM* state) and v_i be a binary representation of the index i of prototype vector y_i with $i = 1, \ldots, I$. In the sequel, the representation v_i of a prototype y_i is assumed to be an index I-vector with all zero components but the i-th one equal to 1. In the case of contextual input, vector v_i can also be obtained by concatenating several representations of prototype vectors belonging to a given contextual window centered on a current y_i. Contextual inputs were already used in the ERIS system [18],[19] where the classification of acoustic vectors was based on discriminant principles (responsible "demons" for the recognition of one particular class and the rejection of all other stimuli as "non-words")

applied on *"state-pair vectors"*. For larger contexts, a NETtalk-like *MLP*, initially described in [28] for mapping written texts to phoneme strings, has also be proved successful in performing the classification of 10 ms acoustic vector strings into phoneme strings, where each current vector was classified by taking account of its surrounding vectors [6]. Figure 1 shows the schematic arrangement of that system which is characterized by two layers of perceptrons (hidden and output layers) computing the classification of the input field. The input field (input units) is constituted by several groups (5 in Fig.1) of units, each group representing a prototype vector. Thus, if $2p+1$ is the width of the contextual window, there are $2p+1$ groups of I units in the input layer. During the training, the desired output of the network is the correct phoneme associated with the center or "current" prototype vector in a particular left and right context. The prototype vectors are stepped through the contextual window time slot by time slot and, at each step, the matrix parameters W_{10} and W_{20} (including weights and biases) are adjusted by the classical *error back-propagation (EBP)* algorithm [15],[27].
The use of contextual information may be generalized to several hidden layers as described in [32].

However, since each acoustic vector is classified independently of the preceding classifications in such feedforward architectures, there is no representation of the sequential character of the speech signal. The system has thus no short-term memory from one classification to the next one and successive classifications can be contradictory. This phenomenon does not appear in *HMM* since the preceding classifications are effectively used and since only some state sequences are permitted; *HMM* lead to a "global" classification of the vector string. Section 3.2 shows how to circumvent this problem by supplying to the input field some information about the preceding classification. This leads to a particular case of the recurrent networks used in speech recognition and for which several other alternative structures are known in the literature; these are described and compared in Section 3.3. Relations between recurrent networks and context-sensitive networks are also discussed there.

3.2 A recurrent context-sensitive MLP

In this section, a modified *MLP* able to modelize the sequence of decisions is described. A second improvement is to make the input field context-sensitive (Section 3.2.2). Finally, in Section 3.2.3, the use of this machine as a local probability generator is described for a connected speech application based on the *DTW* algorithm.

3.2.1 MLP with output feedback

Sequential classification must rely on the previous decisions but the final goal remains the association of the current input vectors with their own classes.

A MLP achieving this task will generate, for each current input vector v_i, K output values $g(i,k,\ell)$ with $\ell \in [1,\ldots,K]$ depending on the class $q(\ell)$ it belongs to and on the class $q(k)$ in which the preceding input vector has been classified. In order to focus on the recurrent structure, the input of the MLP is assumed context independent ($p = 0$) in this section; simultaneous context dependency and recurrence will be considered in Section 3.2.2.
Supervised training is performed on a sequence of N quantized vector representations $\{v_{i_1},\ldots,v_{i_N}\}$ where i_n is the index of the prototype vector at time n. Supervision comes from the a priori knowledge of the classification of each v_{i_n}. Classically, the training of the MLP parameters is based on the minimization of a *mean square criterion (LMSE)* [27] which, with our requirements, takes the form:

$$E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{\ell=1}^{K}[g(i_n,k,\ell) - d(i_n,\ell)]^2 \ , \tag{20}$$

where $d(i,\ell)$ represents the target value of the ℓ-th output associated with the input vector v_i. Since the purpose is to associate each input vector with a single class, the target outputs, for a vector $v_i \in q(\ell)$, are:

$$\begin{aligned} d(i,\ell) &= 1, \\ d(i,m) &= 0, \quad \forall m \neq \ell \ , \end{aligned}$$

which can also be expressed, for each particular $v_i \in q(\ell)$ as: $d(i,m) = \delta_{m\ell}$. The target outputs $d(i,\ell)$ only depend on the current input vector v_i and the considered output unit, and not on the classification of the previous one. The difference between criterion (20) and that of a memoryless machine is the additional summation in k which takes account of the previous decision. Collecting all terms depending on the same prototypes, (20) can thus be rewritten as:

$$E = \frac{1}{2}\sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{\ell=1}^{K}\sum_{m=1}^{K} n_{ik\ell} \cdot [g(i,k,m) - d(i,m)]^2 \tag{21}$$

where $n_{ik\ell}$ represents the number of times v_i must be classified in $q(\ell)$ while the previous vector was known to belong to class $q(k)$. Thus, whatever the MLP topology, i.e. the number of its hidden layers and of units per layer, the optimal output values $g_{opt}(i,k,m)$ are obtained by canceling the partial derivative of E versus $g(i,k,m)$:

$$\frac{\partial E}{\partial g(i,k,m)} = \sum_{\ell=1}^{K} n_{ik\ell}.[g(i,k,m) - d(i,m)] = 0 \ .$$

The optimal values of the outputs are then

$$g_{opt}(i,k,m) = \frac{\sum_{\ell=1}^{K} n_{ik\ell}.d(i,m)}{\sum_{\ell=1}^{K} n_{ik\ell}} = \frac{\sum_{\ell=1}^{K} n_{ik\ell}.\delta_{\ell m}}{\sum_{\ell=1}^{K} n_{ik\ell}} \ ,$$

and, finally:

$$g_{opt}(i,k,m) = \frac{n_{ikm}}{\sum_{\ell=1}^{K} n_{ik\ell}} . \tag{22}$$

Thus, the optimal $g(i,k,m)$'s obtained from the minimization of the *MLP* criterion are in fact the estimates of the Bayes probabilities [11] or, equivalently, the discriminant local probabilities defined by (15). It is easily verified that they sum up to unity:

$$\sum_{m=1}^{K} g_{opt}(i,k,m) = 1 .$$

It is important to keep in mind that the optimal values can be reached only provided the *MLP* contains enough parameters.

A convenient way to generate the $g(i,k,\ell)$ is to modify its input as follows. For each v_{i_n}, an extended vector $V_{i_n} = (v_{i_n}^+, v_{i_n})$ is formed where $v_{i_n}^+$ is an extra input vector containing the information on the decision taken at time $n-1$. Since output information is fed back in the input field, such an *MLP* has a *recurrent* topology. However, as shown in Section 3.3, the main advantage of this topology, when compared with other recurrent models proposed in speech processing, over and above the possible interpretation in terms of *HMM*, is the control of the information fed back during the training. Indeed, since the training data consists of consecutive labeled speech frames, the correct sequence of output states is knownand the training is supervised by providing the correct information.

Replacing in (21) $d(i,m)$ by the optimal values (22) provides a new criterion where the target outputs depend now on the current vector, the considered output and the classification of the previous vector:

$$E^* = \frac{1}{2} \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \sum_{m=1}^{K} n_{ik\ell} \cdot \left[g(i,k,m) - \frac{n_{ikm}}{\sum_{\ell=1}^{L} n_{ik\ell}} \right]^2 , \tag{23}$$

and it is clear (by canceling the partial derivative of E^* versus $g(i,k,m)$) that the lower bound for E^* is reached for the same optimal outputs as (22) and is now equal to zero, thus providing a very useful control parameter during the training phase. Moreover, it was suggested earlier that "training of an adaptive machine is best done *not* with the *correct* output (1 or 0) being supplied, but with an expert's estimate of the *probabilities* of the possible output states" [17].

It is evident that these results directly follow from the minimized criterion and *not from the topology of the model.* In that way, it is interesting to note that the same optimal values (22) may result from other criteria, for instance the entropy [13] or relative entropy [29] of the targets with respect to outputs. Indeed, in the case of relative entropy, e.g., criterion criterion (20) is changed

in:

$$E_e = \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \left[d(i_n,\ell).\ln \frac{d(i_n,\ell)}{g(i_n,k,\ell)} + (1-d(i_n,\ell)).\ln\left(\frac{1-d(i_n,\ell)}{1-g(i_n,k,\ell)}\right)\right], \tag{24}$$

and canceling its partial derivative versus $g(i,k,m)$ yields the optimal values (22). In that case, the optimal outputs effectively correspond now to $E_{e,min} = 0$.

Of course, since these results are independent of the topology of the models, they remain also valid for linear discriminant functions [4] but, in that case, it is not guaranteed that the optimal values (22) can be reached. However, it has to be noted that in some particular cases, even for not linearly separable classes, these optimal values are already obtained with linear discriminant functions (and thus with a one layered perceptron trained according to an *LMS* criterion). For instance, if n_i is the number of input units and if the training vectors are only constituted by repetitions of n_i linearly independent vectors, it can easily be proved that the Bayes probabilities can be generated by linear functions.

Since it is now proved that the considered *MLP* is able to estimate the local probabilities used in the discriminant *HMM* defined in Section 2.4, it is clear that if the training acoustic vectors are not labeled (no explicit segmentation), the training of the *MLP* can be embedded in a *DTW* process using the output values $g(i,k,\ell)$ as local probabilities and iteratively improving an initial segmentation. The convergence of this process, proved for Markov models, remains thus valid for particular *HMM*'s where the local emission probabilities are generated by *MLP*. Proof of this assertion was already presented in [4],[6].

It is also important to point out that a same kind of recurrent *MLP* could also be used to estimate local probabilities of higher order Markov models where the local contribution in (17) are no longer assumed dependent on the previous state only but also on several preceding ones. This is easily implemented by extending the input field to the information related to these preceding classifications.

3.2.2 Incorporation of context-sensitivity

The results presented in Section 3.2.1 still hold with a modified input taking the context into account as described in Section 3.1. The proposed architecture represented on figure 2, is then a NETtalk-like *MLP* as in figure 1, but supplied with a direct feedback of the outputs associated with the previous input frames. The feedback is implemented by adding extra input units which reflect the delayed output values (or a binary representation of them). Matrices W_{10}, W_{11}, W_{20} are the parameters (including weights and biases) to be trained. Since the training data consist of consecutive labeled speech frames, the correct sequence of output states is known and training with the correct feedback values is thus possible.

The outputs of this *MLP with contextual input and output feedback* are then estimators of (18). Of course, the number of weights increases with the width of

the contextual window and thus a huge volume of training data is again required for significant training. However, the generalization properties of the *MLP* already justified in [5],[6] by the generation of crossproducts by the nonlinearities of the hidden units play the role of the interpolation required by *HMM* (Section 2.4). In particular, contrary to a discrete *HMM*, an *MLP* will not generate zero outputs when it is stimulated by an input vector never observed in the training set.

3.2.3 Application to connected speech recognition

The outputs of the *MLP* (or their logarithms) can be used in a classical one-stage *DTW* [7],[22] for connected speech recognition in the same way as the local contributions (18) of the discrete discriminant *HMM* (Section 2.4). If the *MLP* has been trained for phonemic labeling (each output of the *MLP* is associated with a phoneme or a state of a phonemic *HMM*), "word models" can be build along the vertical axis of a *DTW* table by concatenating the outputs of their constituting phonemes, the horizontal axis corresponding to the time ordering of the acoustic vectors. The *DTW* table defines thus a set of grid point (i, ℓ, w) associated with time slot i of the speech signal ($i = 1, ..., N$) and state $q(j_\ell(w))$ with $\ell = 1, ..., L_w$ and where $j_\ell(w)$ is the index of the ℓ-th state in word w.

The local contributions are computed by using the *MLP* described in Section 3.2.2 along two methods.

In the first one, they are explicitly associated with transitions from one grid point of the *DTW* table to another. The feedback part v_i^+ of the *MLP* input vector V_i corresponding to time slot i is successively modified to represent the originating state of each transition ending at $q(j_\ell(w))$ while the remaining part v_i, which is the binary representation of the current vector at time i and its left and right contexts, remains unchanged. Outputs $g(i, k, j_\ell(w))$ provide the local contributions at grid point (i, ℓ, w) for each transition originating from q_k. In this method, a forward computation of the *MLP* outputs is required for each time slot *and* for each possible originating state. Vectors v_i^+ are index vectors (all components are zeroes but one).

In the second method, the local contribution is associated with the grid point itself and not with transitions. It is provided for all states by a single computation of the *MLP* outputs per time slot. For time i, the components of vector v_i^+ are the output values of the *MLP* at time $(i - 1)$ and are thus no longer binary but real-valued in the range $[0, 1]$. Full advantage is taken here from the recurrent *MLP* for reproducing the learned sequential nature of the speech signal; this justifies that in the *DTW* table the local contribution is no longer associated with transitions but only with the ending state (or grid point).

Then, using the technique of dynamic programming, we seek, among the paths starting from grid points $(1, 1, w)$ and arriving at (N, L_w, w), $\forall w = 1, ..., S$ (S = size of the lexicon), that path which provides the least cumulated "distance" $G(N, L_w, w)$. That "distance", referred to as matching score, is associ-

ated with the best path and defines the optimal word sequence.
These cumulated distances are obtained by the following recurrences:

$$G(i,\ell,w) = \min_{\{k\}} \{G(i-1,k,w) - \ln[g(i,k,j_\ell(w))]\} \quad (25)$$

within a word model, and:

$$G(i,1,w) = \min_{\{w'\}} \{G(i-1,L_{w'},w') - \ln[g(i,j_{L_{w'}}(w'),j_1(w))]\} \quad (26)$$

between word models, where $\{k\}$ and $\{w'\}$ respectively stand for the set of possible predecessor states of $q_{j_\ell(w)}$ and possible predecessor words of w. Recurrence (25) is obtained by taking minus logarithm of both sides of (19).
The best path is recovered by backtracking through the *DTW* table.

3.3 Comparison with existing architectures

This section gives proper credit to the pioneering work of several authors in view to capture the sequential information of a signal in a neural net. The use of recurrent *MLP* for dealing with sequential inputs was already introduced in [27] and was shown to be useful on a simple example of sequence completion. In that approach, recurrent *MLP* were approximated over a finite time period, by the associated "unfolded" feedforward network. The *EBP* algorithm may still be used but with the constraint that all unfolded versions of a synaptic link are identically weighted. In [23],[24] the *EBP* algorithm used for the training of *MLP* was generalized to recurrent networks and *MLP*'s can now be trained with any kind of feedback.
Several recurrent networks have already been proposed for speech recognition. In [25], a particular Boltzmann machine for dealing with sequential inputs was defined where some of the hidden units, called "carry units", were supplied as extra inputs with the purpose of generating time dynamic. However, since feedback must be applied at the "thermal equilibrium", this architecture leads to an excessive time consumption which is inherent to the Boltzmann machines. In [34], sequential processing is obtained with the "temporal flow model" (figure 3). Selfloops with delay are added to each hidden unit (a single layer of hidden units is considered) and to each output unit of an *MLP*. A training procedure based on a generalized *EBP* is developed [33]. No contextual information is explicitly used at the input but of course this architecture could easily put up with an extended input as in a NETtalk machine. It can easily be observed that the system described in [10] (fig.4) is an alternative implementation of this network where the output selfloops are eliminated and where the delayed hidden unit values are fed back as supplementary input units. Feedback is indeed easily implemented by extending the input vectors as described in Section 3.2.1 with a vector v_i^+ containing the hidden unit values generated by the preceding input frame. Moreover, the input is also extended to incorporate contextual

information. The associated *MLP* architecture is then represented in figure 4, where W_{ij} represents the matrix of parameters (including weights and biases) and can be interpretated in terms of state space equations of the control theory [26]. For linear units, this network has a rational input-output transfer function and, in view of the analogy with digital filters, it may be called "Infinite Impulse Response *(IIR)* Dynamic Net".

Recurrent networks can also be approximated over a finite time period (say D time slots) by a feedforward network where the loops are replaced by the explicit use of several preceding activation values. In that case, the activations in a particular layer are computed from the current and multiple delayed values of the preceding layer. That approach has been used for speech recognition in [32] and the corresponding architecture (in the particular case of only one hidden layer) is represented in figure 5. A similar approach using delayed input units but without hidden units has also been presented in [30],[31] in an analog input signal application.
By analogy with filter theory, that kind of network may be called "Finite Impulse Response *(FIR)* Dynamic Net". In the linear case, *IIR* and *FIR* dynamic nets may be compared to *IIR* and *FIR* filters where the first one (Fig.4) generates an infinite impulse response of the form

$$\frac{1}{1-az^{-1}} \simeq \sum_{i=0}^{\infty} a^i z^{-i} \quad ,$$

where a is related to the loop weight matrix W_{11}; the impulse response of the second one (Fig.5) is finite and is of the type

$$\sum_{i=0}^{D} a_i z^{-i}$$

for an approximation over D time slots and where a_i are related to W_{11} and W_{12}, are now independent contrary to what happens with an actual loop.

As already pointed out in [17], a common disadvantage of all these approaches is the impossibility to supervise the information fed back and to insure its usefulness. The *MLP* with output feedback (Section 3.2) avoids this deficiency. However, when comparing with the methods proposed in [10],[34], a drawback of the present approach is the impossibility for the system to learn to feed back other useful information not included in our output representation. Indeed, by (22) it is clear that, even in an *MLP* with a sufficiently large number of hidden units, the error (20) can be reduced to zero only if a same extended input V_i (i.e. same vector v_i appearing after the same output decision) never appears in the training set as a member of different classes. This restriction is eliminated with Elman's machine where feedback is applied from hidden units to the input (Fig.4). Let us first observe that in any *MLP*, if the number of hidden units is higher than the number of output ones, different hidden vectors

may generate the same output. Now, suppose that in the training set, the same input vector appears several times with the same previous output decision but associated with different current decisions. For each presentation, the previous hidden vectors may be different although they generated the same output and so are consequently the extended input vectors. Error (20) can thus vanish for a sufficiently large number of hidden units. The feedback is unsupervised and internal representation of the speech sequential information is built up by the training.

4 Conclusions

Discrimination is an essential requirement in speech recognition and is not incorporated in the standard *Hidden Markov Model*. A *discriminant HMM* has been described. Links between this new model and a *recurrent Multilayer Perceptron* are pointed out. Recurrence permits to implicitly modelize the sequential information in the output sequence, i.e. the phonetic labeling, in our case. The local probabilities of the *discriminant HMM* may be then computed by using that *MLP* as a local contribution generator in a *DTW* process. Since the feedback in the *MLP* is applied from the output units to the input field, it is easily supervised during the training. Moreover, input contextual information is also easily captured by extending the input of the *MLP* as in a NETtalk machine.

References

[1] L.R. Bahl, F. Jelinek and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", *IEEE Trans. on PAMI*, vol. 5, No. 2, pp. 179-190, 1983.

[2] L.R. Bahl, P.F. Brown, P.V. de Souza and R.L. Mercer, "Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition", *Proc. ICASSP-86*, Tokyo, pp.49-52, 1986.

[3] H. Bourlard, Y. Kamp, H. Ney and C.J. Wellekens, "Speaker-Dependent Connected Speech Recognition via Dynamic Programming and Statistical Methods", *Speech and Speaker Recognition*, Ed. M.R. Schroeder, KARGER, 1985.

[4] H. Bourlard and C.J. Wellekens, " Discriminant Functions for Connected Speech Recognition", *Proceedings of the EUSIPCO-86*, The Hague (The Netherlands), pp. 507-510, Ed. I.T. Young, J. Biemond, R.P.W. Duin & J.J. Gerbrands, 1986.

[5] H. Bourlard and C.J. Wellekens, "Multilayer Perceptrons and Automatic Speech Recognition", *Proceedings of the First International Conference on Neural Networks*, IV-407-416, San Diego, CA, 1987.

[6] H. Bourlard and C.J. Wellekens," Speech Pattern Discrimination and Multilayer Perceptrons", to appear in *Computer, Speech and Language*, 1988.

[7] J.S. Bridle, M.D. Brown and R.M. Chamberlain, "An algorithm for connected word recognition", *Proc. ICASSP-82*, Paris, pp.899-902, 1982.

[8] P. Brown, *The Acoustic-Modeling Problem in Automatic Speech Recognition*, Ph.D. thesis, Comp.Sc.Dep., Carnegie-Mellon University, 1987.

[9] D.J. Burr, "Speech Recognition Experiments with Perceptrons", *AIP Conference Proceedings*, Neural Information Processing Systems, Denver, CO, 1987.

[10] J.L. Elman, "Finding Structure in Time", *CRL Tech. Report 8801*, University of California, San Diego, 1988.

[11] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, NY, 1972.

[12] S. Furui, "Speaker Independent Isolated Word Recognizer Using Dynamic Features of Speech Spectrum", *IEEE Trans. ASSP-34*,pp. 52-59, 1986.

[13] G.E. Hinton, "Connectionist Learning Procedures", *Technical Report CMU-CS-87-115*, Carnegie Mellon University, 1987.

[14] F. Jelinek, "Continuous Recognition by Statistical Methods", *Proceedings IEEE,* vol. 64, no.4, pp. 532-555, 1976.

[15] Y. Le Cun, *Modeles Connexionistes de l' Apprentissage*, These de doctorat a l'Universite de Paris VI, 1987.

[16] R.P. Lippmann, "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, vol. 4, pp. 4-22, 1987.

[17] D.J. MacKay, "A Method of Increasing the Contextual Input to Adaptive Pattern Recognition Systems", *Technical Report* no. RIPRREP/1000/14/87, Royal Signals and Radar Establishment, Malvern, U.K.,1987.

[18] S.M. Marcus," ERIS-context sensitive coding in speech perception", *Journal of Phonetics*, 9, pp. 197-220, 1981.

[19] S.M. Marcus," Associative Models and the Time Course of Speech", *Speech and Speaker Recognition*, Ed. M.R. Schroeder, KARGER, 1985.

[20] E.A. Martin, R.P. Lippmann and D.B. Paul, " Two-Stage Discriminant Analysis for Improved Isolated Word Recognition", *Proc. ICASSP-87*, Dallas, pp.17-5-1,4, 1987.

[21] B. Merialdo, "Phonetic Recognition Using Hidden Markov Models and Maximum Mutual Information Training", *Proc. ICASSP-88*, New York, pp. 111-114, 1988.

[22] H. Ney, "The use of a one-stage dynamic programming algorithm for connected word recognition", *IEEE Trans. ASSP* vol. 32, pp.263-271, 1984.

[23] F.J. Pineda, "Generalization of backpropagation to recurrent neural networks", *Phys. Rev. Lett.*, 18, pp. 2229-2232, 1987.

[24] F.J. Pineda, "Dynamics and Architecture in Neural Computation", to appear in *Journal of Complexity*, special issue on Neural Networks, September 1988,

[25] R.W. Prager, T.D. Harrison and F. Fallside, "Boltzmann machines for speech recognition", *Computer, Speech and Language*, vol. 1, pp. 3-27, 1986.

[26] A.J. Robinson, F. Fallside, "The Utility Driven Dynamic Error Propagation Network", *Tech. Report CUED/F-INFENG/TR.1*, Cambridge University, U.K., 1987.

[27] D.E. Rumelhart, G.E. Hinton and R.J. Williams, " Learning Internal Representations by Error Propagation", *Parallel Distributed Processing. Exploration of the Microstructure of Cognition. vol. 1: Foundations*, Ed. D.E.Rumelhart and J.L.McClelland, MIT Press, 1986.

[28] T.J. Sejnowski and C.R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", *Complex Systems*, vol. 1, pp. 145-168, 1987.

[29] S.A. Solla, E. Levin and M. Fleisher M., "Accelerated Learning in Layered Neural Networks", *AT&T Bell Labs. Manuscript*, 1988.

[30] D.W. Tank and J.J. Hopfield, "Concentrating Information in Time: Analog Neural Networks with Applications to Speech Recognition Problems",*Proceedings of the First International Conference on Neural Networks*, IV -455-468, San Diego, CA, 1987.

[31] K.P. Unnikrishnan, J.J. Hopfield and D.W. Tank, "Learning Time-delayed Connections in a Speech Recognition Circuit", *Neural Network for Computing*, Snowbird, UT, 1988.

[32] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano and K. Lang, " Phoneme Recognition Using Time-Delay Neural Networks, *Proc. ICASSP-88*, New York, 1988.

[33] R.L. Watrous and L. Shastri, "Learning phonetic features using connectionist networks: an experiment in speech recognition", *Technical Report MS-CIS-86-78*, University of Pennsylvania, 1986.

[34] R.L. Watrous and L. Shastri, "Learning phonetic features using connectionist networks: an experiment in speech recognition", *Proceedings of the First International Conference on Neural Networks*, IV -381-388, San Diego, CA, 1987.

[35] C.J. Wellekens, "Global Connected Digit Recognition Using Baum-Welch Algorithm", *Proc. ICASSP-86*, Tokyo, pp. 21.5.1-21.5.4, 1986.

[36] C.J. Wellekens, "Explicit Time Correlation in Hidden Markov Models for Speech Recognition", *Proc. ICASSP-87*, Dallas, TX, pp. 10.7.1-10.7.3, 1987.

Figure captions

Figure 1 NETtalk architecture.
Figure 2 Recurrent and context-sensitive MLP.
Figure 3 Watrous and Shastri temporal flow model.
Figure 4 Elman MLP architecture.
Figure 5 Waibel context-sensitive model.

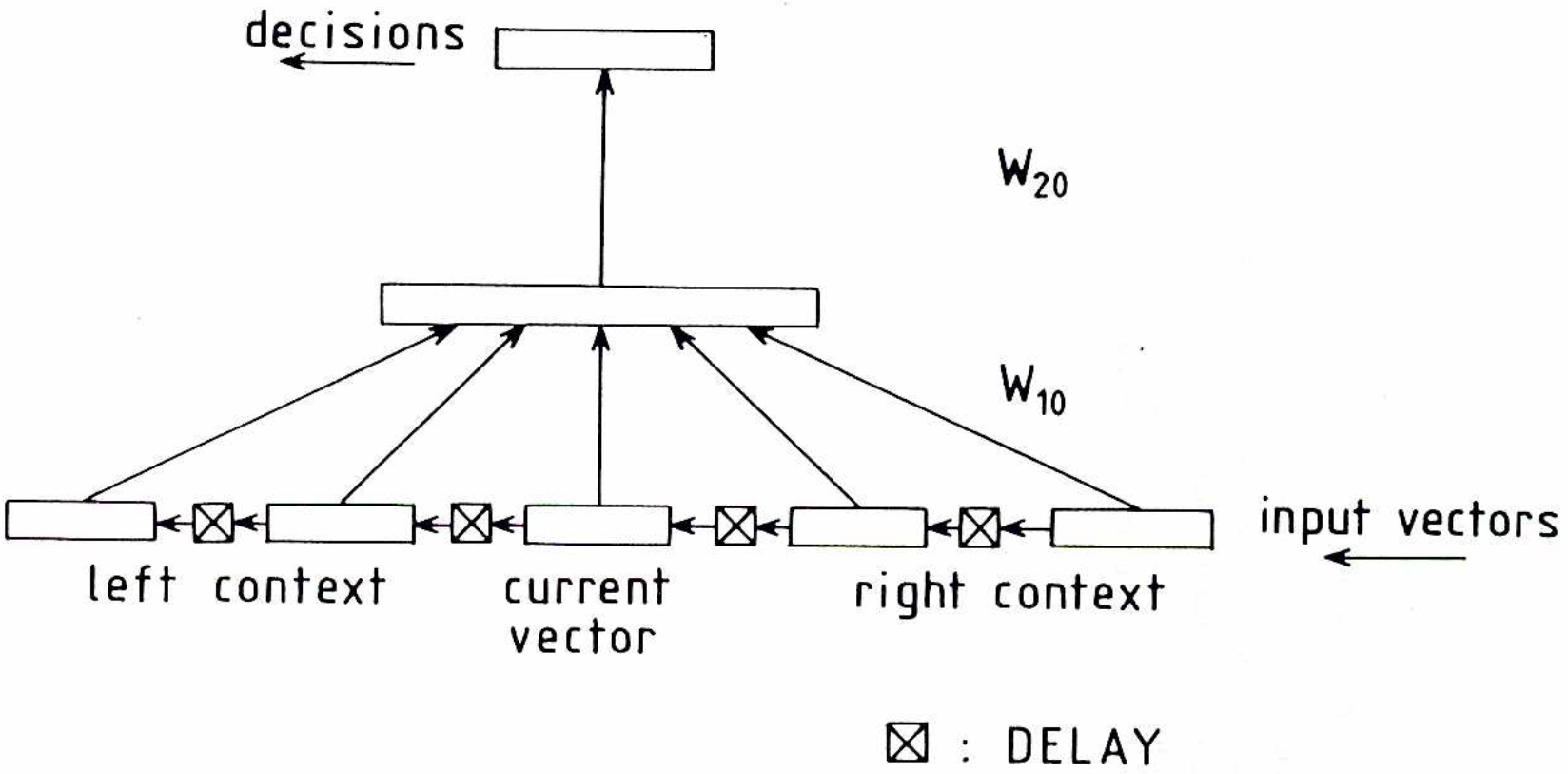
decisions
W_{20}
W_{10}
input vectors
left context
current vector
right context
⊠ : DELAY

FIG. 1

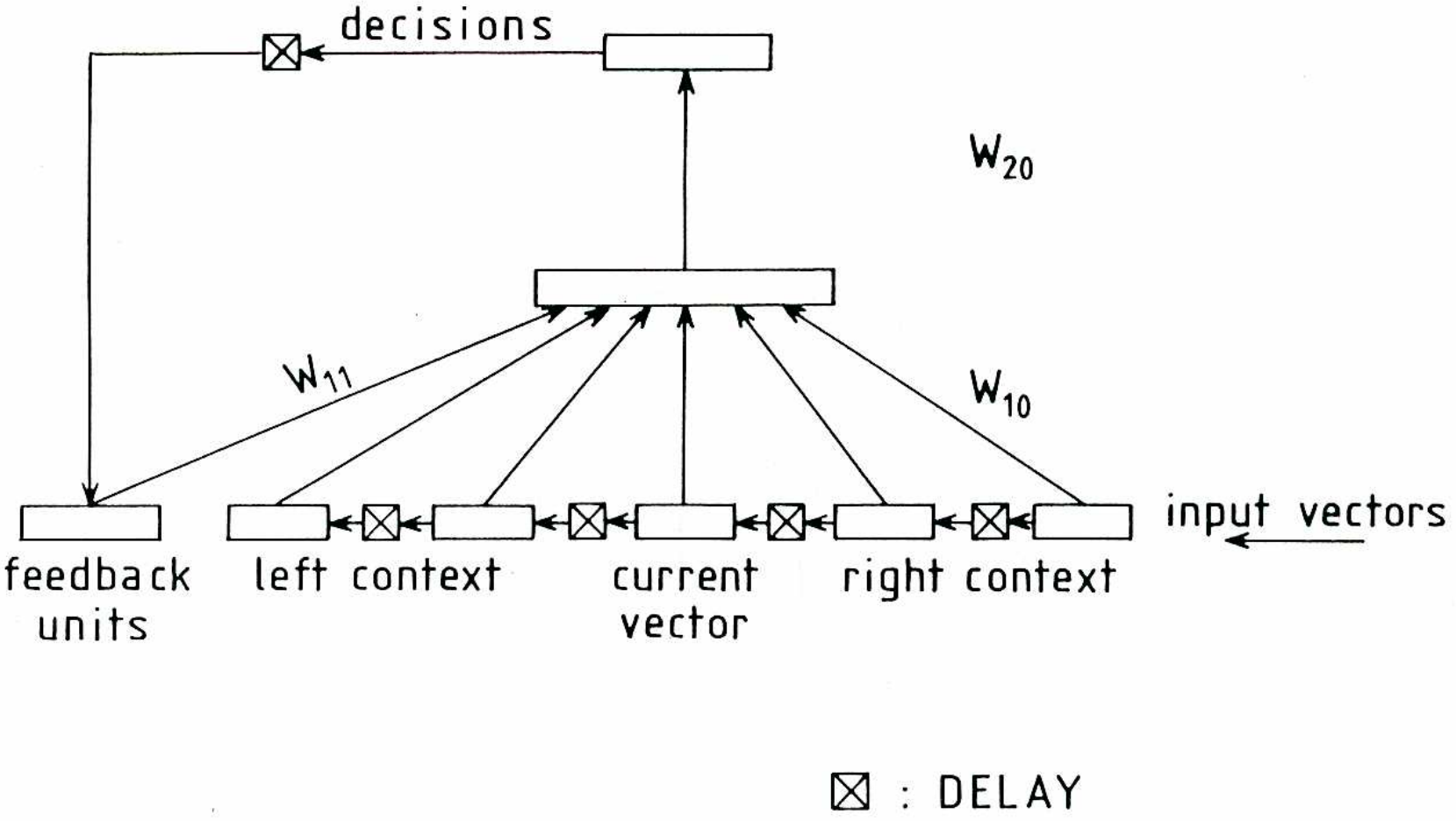
decisions
W_{20}
W_{11}
W_{10}
input vectors
feedback units
left context
current vector
right context
⊠ : DELAY

FIG. 2

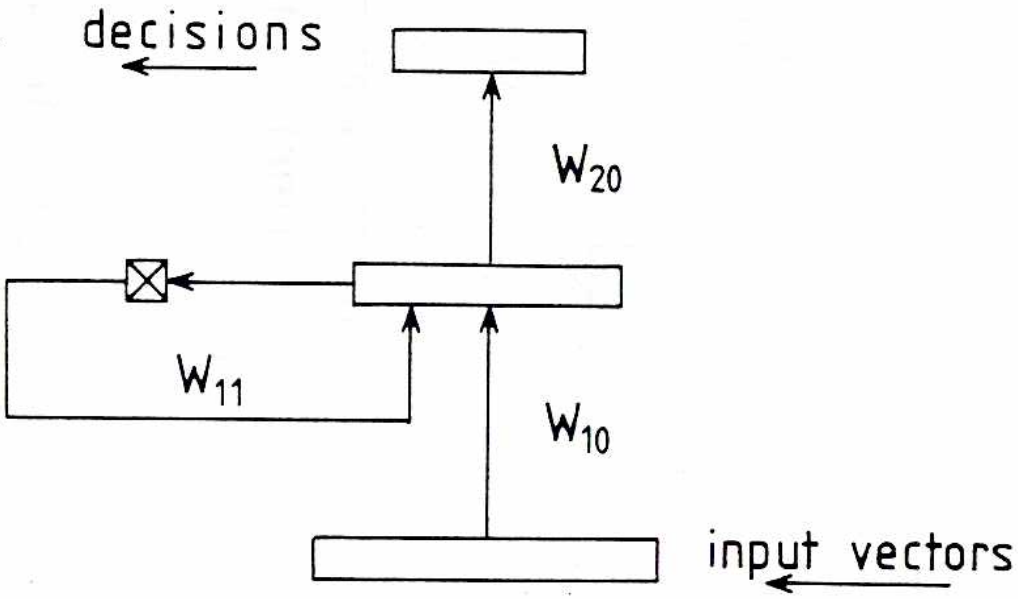

FIG. 3

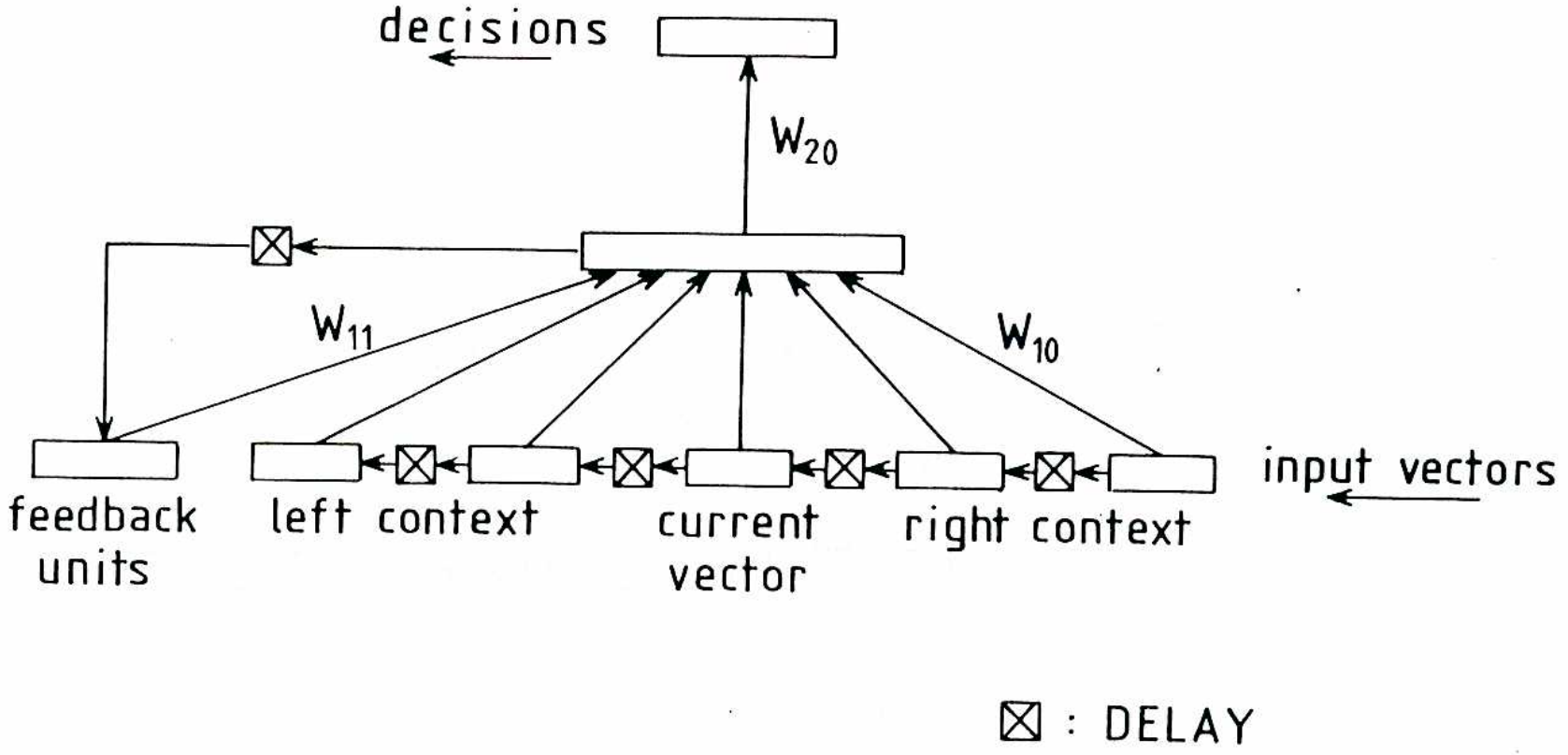
decisions
W20
W11
W10
feedback units
left context
current vector
right context
input vectors
⊠ : DELAY

FIG. 4

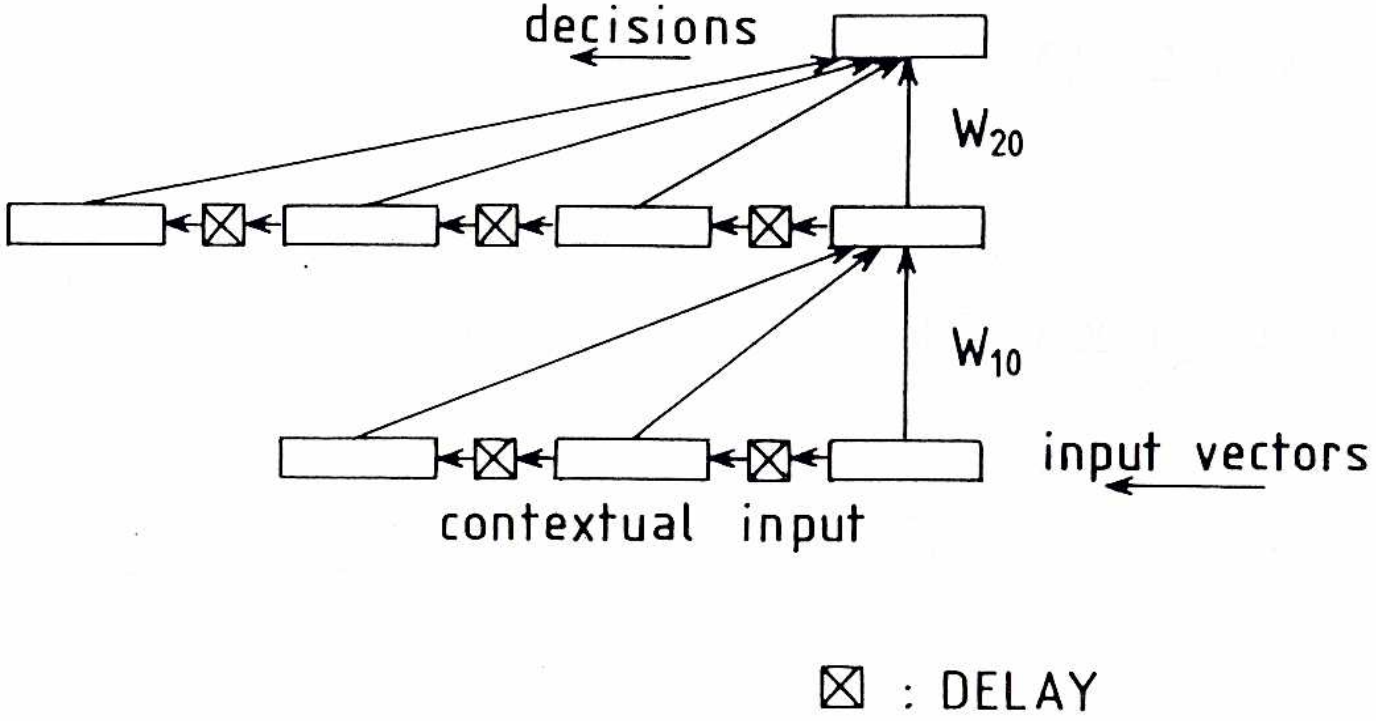

FIG. 5