Stochastic Recurrent Networks: Prediction and Classification of Time Series

Athanasios Kehagias
Division of Applied Mathematics
Brown University
Providence, RI 02912

e-mail: st401843@brownvm.brown.edu

March 4, 1991

Abstract

We use *Stochastic Recurrent Networks* of the type introduced in [Keh91a] as models of finite-alphabet time series. We develop the *Maximum Likelihood Prediction Algorithm* and the *Maximum A Posteriori Classification Algorithm* (which can both be implemented in recurrent PDP form). The prediction problem is: given the output up to the present time: $Y^1, ..., Y^t$ and the input up to the immediate future: $U^1, ..., U^{t+1}$, predict with Maximum Likelihood the output Y^{t+1} that the SRN will produce in the immediate future. The classification problem is: given the output up to the present time: $Y^1, ..., Y^t$ and the input up to the present time: $U^1, ..., U^t$, as well as a number of candidate SRN's: $\mathcal{M}_1, \mathcal{M}_2, .., \mathcal{M}_K$, find the network that has Maximum Posterior Probability of producing $Y^1, ..., Y^t$. We apply our algorithms to prediction and classification of speech waveforms.

Contents

NOTATION

Given a finite set A, we denote the number of elements in A by $|A|$. E.g., for $A = \{a_1, ..., a_N\}$, $|A| = N$. The **alphabet** A of a stochastic process $\{Z^t\}_{t=1}^{\infty}$ is the set of all possible values that Z^t can take for any t. E.g. we could have a *binary* stochastic process $\{Z^t\}_{t=1}^{\infty}$ where Z^t equals either 0 or 1 for every t. In that case the alphabet is $A = \{0, 1\}$. Or, we could have a *vector-binary* process $\{Z^t\}_{t=1}^{\infty}$, where $Z^t = [Z_1^t ... Z_N^t]$ and Z_n^t is either 0 or 1 for every t, $n = 1, ..., N$. In that case the alphabet is $A^N \doteq \{[a_1...a_N] : a_n \in A, n = 1, ..., N\}$.

We use capital letters X, Y, Z etc. for stochastic processes and small letters x, y, z for the values of the processes (characters of the alphabet). For instance we write $Prob(X^t = x)$ for the probability that X^t equals the character $x \in A$; we write $Prob(X^{t+1} = x^1, ..., X^{t+\tau} = x^\tau)$ for the probability that $X^{t+1}...X^{t+\tau}$ equals $x^1...x^\tau \in A^\tau$. Say $x = [x_1...x_m]$, $y = [y_1...y_n]$; then the **concatenation** of x, y is $xy = [x_1...x_m y_1...y_n]$.

We will often consider probabilities that depend on the value of a certain parameter, say $\mathcal{P}$. Then we write, for instance, $Prob(X^{t+1} = x^1, ..., X^{t+\tau} = x^\tau; \mathcal{P})$. Also, some times we denote a SRN by a letter such as $\mathcal{M}$ and we want to talk about probabilities of events related this SRN. Then we use notation such as $Prob(X^{t+1} = x^1, ..., X^{t+\tau} = x^\tau; \mathcal{M})$.

We often consider a set $S = \{1, ..., N\}$ and vector $x = [x_1...x_N]$. We sometimes write x_S in place of x. Similarly, for a set $R = \{r_1, ..., r_M\} \subset S$ we write x_R in place of $[x_{r_1}...x_{r_M}]$. Obviously, if $x_s \in A$ for $s \in S$, then $x_S \in A^{|S|}$.

1 Introduction

In [Keh91a] we introduced **Stochastic Recurrent Networks** (henceforth, SRN) as a model for finite-alphabet stochastic processes. In this paper we develop **prediction** and **classification** methods for such SRN's. In what follows we use the term **Time Series** as a synonym for **Stochastic Process**. We will develop algorithms for prediction and classification of finite alphabet stochastic processes and will apply them to speech data problems.

The subject of time series **prediction** has received considerable attention in the connectionist literature [LF87, MD88, Sut88, WZ88]. All of these works deal with continuously valued time series. Here we deal with time series that can take values in a finite alphabet; however, using quantization of continuous-valued waveforms we can solve approximately the continuous-valued case as well.

Roughly, the prediction problem is: given the past history $Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t$ of a time series (and, when appropriate, the input history up to present $U^1 = u^1, ..., U^{t+1} = u^{t+1}$) compute a value $\hat{Y}^{t+1}$ which is *close* to the actual value Y^{t+1}. The problem will be described in a mathematically precise form in Section 3.

Classification of static patterns is a standard problem of Connectionism. However, classification of time series has not received similarly wide attention. There is one significant exception to this statement: Speech Recognition. In particular, phoneme recogntion is obviously a case of dynamic patterns classification. This problem has been attacked using both feedforward, static classifiers [W+89a, W+89b] and feedback, dynamic classifiers [BW88, BW89, Rob88, Rob89, Be+90a, Be+90b]. The general classification problem is, roughly, the following: given several candidate SRN models: $\mathcal{M}_1, \mathcal{M}_2, .., \mathcal{M}_N$, find the SRN that (given the input up to the present time: $U^1 = u^1, ..., U^t = u^t$) is most likely to have produced $Y^1 = y^1, ..., Y^t = y^t$. Once again, we postpone a mathematically precise formulation to Section 5.

Our point of view is the following. For prediction, we assume the sample $y^1, y^2, ...$ is produced by a known SRN $\mathcal{M} = ((S, \mathcal{N}), \mathcal{P})$ with input $u_1, u_2, ...$ (this is the modelling step, considered in [Keh91a]). Then we proceed to compute the conditional probability of $Y^{t+1} = a$ given $Y^1 = y^1, ..., Y^t = y^t$ $U^1 = u^1, ..., U^{t+1} = u^{t+1}$:

$$Prob(Y^{t+1} = a \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^{t+1} = u^{t+1}; \mathcal{M}) \qquad a \in A. \tag{1}$$

Our prediction of Y^{t+1} is obtained by maximizing (1):

$$\hat{Y}^{t+1} \doteq \arg\max_a Prob(Y^{t+1} = a \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^{t+1} = u^{t+1}; \mathcal{M}) \qquad a \in A. \tag{2}$$

This computation is updated for every t and can be implemented in recursive form by a recurrent PDP network.

Similarly, for the classification problem, we assume N models $\mathcal{M}_1, ..., \mathcal{M}_N$ with known parameters and we compute $Prob(\mathcal{M}_n \mid Y^1, ..., Y^t)$ for $n = 1, 2, ..., N$. We classify the signal as being produced by model $\mathcal{M}_{\hat{N}^t}$, where $\hat{N}^t$ is defined by

$$\hat{N}^t \doteq \arg\max_n Prob(\mathcal{M}_n \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^t = u^t) \tag{3}$$

Once again, this computation is updated for every t and can be implemented in recursive form by a recurrent PDP network.

The motivation for our methods comes from stochastic control. As pointed out in [Keh91a], any SRN can be implemented as a network of input/output units, with hidden unit state vector X^t and output unit state vector Y^t, $t = 1, 2, ...$ The equations these vectors satisfy are:

$$X^t = f(X^{t-1}, U^t, V^t), \tag{4}$$

$$Y^t = f(X^t, U^t, W^t); \tag{5}$$

where U^t is the input vector, and V^t, W^t are white noise inputs. The formal similarity of (4), (5) to the equations of control systems [May82] suggests that we apply modified versions of standard control/estimation algorithms such as Kalman filtering for prediction [Kal60] and the Lainiotis Partition algorithm algorithm for classification [Lain69,Lain71]. The treatment of continuous state time series as stochastic dynamical systems is standard; see for instance [Aok87, BD87].

We should point out that all the algorithms presented in the following sections apply equally well to *global* Hidden Markov Models, of which SRN are a special case (see also [Keh91b]). Both the prediction and classification algorithm require the specification of global transition and emission probabilities. In the SRN context these are readily computed in trems of the local conditionals (see [Keh91a]); in the HMM context they are *given* to us (the HMM is specified in terms of the transition and emission matrices).

2 The SRN Model

In this section we briefly review the Stochastic Recurrent Network (SRN) model introduced in [Keh91a]. A more detailed presentation can be found in [Keh91a]. Consider a SRN with M_i

input units, M_h hidden units and M_o output units, all of them taking states from a finite set A. Call U^t the M_i-long state vector of input units at time t, X^t the M_h-long state vector of hidden units at time t, Y^t the M_o-long state vector of output units at time t. The components of these vectors all come from the same finite set $A = \{0, 1, ..., K-1\}$. The sequences $\{U^t\}_{t=1}^{\infty}$, $\{X^t\}_{t=1}^{\infty}$ and $\{Y^t\}_{t=1}^{\infty}$ are the input, hidden and output stochastic processes, respectively.

We want a specification of a SRN which, together with the initial state X^0 and the input process $\{U^t\}_{t=1}^{\infty}$, will describe the hidden process $\{X^t\}_{t=1}^{\infty}$ and the output process $\{Y^t\}_{t=1}^{\infty}$. The processes $\{U^t\}_{t=1}^{\infty}$, $\{X^t\}_{t=1}^{\infty}$, $\{Y^t\}_{t=1}^{\infty}$ all take values in finite alphabets (A^{M_i}, A^{M_h}, A^{M_o} respectively). Hence they are fully described by their **probability functions**:

$$p_U(u^1...u^m) \doteq Prob(U^1 = u^1...U_m = u^m) \qquad \forall m, \forall u^1, ..., u^m \in A^{M_i},$$

$$p_X(x^1...x^m) \doteq Prob(X^1 = x^1...X_m = x^m) \qquad \forall m, \forall x^1, ..., x^m \in A^{M_h},$$

$$p_Y(y^1...y^m) \doteq Prob(Y_1 = y^1...Y_m = y^m) \qquad \forall m, \forall y^1, ..., y^m \in A^{M_o}.$$

The SRN will be specified in terms of a *directed graph* $\mathcal{G}$ and a set of *local conditional probabilities* $\mathcal{P}$. Thus, a stochastic recurrent network is a pair $(\mathcal{G}, \mathcal{P})$; given $(\mathcal{G}, \mathcal{P})$, p_U and an initial condition X^0, we can compute $p_X(x^1, ..., x^m)$, $p_Y(y^1, ..., y^m)$ for all m, x^1, ..., x^m, y^1, ..., y^m.

The directed graph $\mathcal{G}$ is itself a pair $\mathcal{G} = (S, \mathcal{N})$, where $S = \{s_1, ..., s_M\}$ is the collection of units (or nodes, to use the graph theoretic term). The unit set S is partitioned into three mutually exclusive sets: $S = S_i \cup S_h \cup S_o$, where S_i is the set of input units, S_h is the set of hidden units, S_o is the set of output units. We have $|S_i| = M_i$, $|S_h| = M_h$, $|S_o| = M_o$. Note that $M = M_i + M_h + M_o$.

Take any two units $r, s \in S$. If s reads the state of r before changing its own state then there is a directed edge from unit r to unit s. In such a case we say that r is a **parent** of s. A unit $s \in S$ can have none, one or many parents and even be a parent of itself. The set of s's parents is indicated by $N(s)$ and the class of all parent sets is denoted by $\mathcal{N} \doteq \{N(s), s \in S\}$.

$(S, \mathcal{N})$ is a complete description of the **topology** of the net. We assume the SRN topology satisfies the following restriction. The parent set of every unit can be partitioned as follows:

$$\forall s \in S_i \qquad N(s) = \emptyset,$$

$$\forall s \in S_h \qquad N(s) = N_i(s) \cup N_h(s) \text{ where } N_i(s) \subset S_i \text{ and } N_h(s) \subset S_h,$$

$$\forall s \in S_o \qquad N(s) = N_i(s) \cup N_h(s) \text{ where } N_i(s) \subset S_i \text{ and } N_h(s) \subset S_h.$$

In words, this means that input units receive no input, while hidden and output units receive input only from input and hidden units. This completes the description of the topology of the network.

The probabilistic state update mechanism is described by $\mathcal{P}$, which is the set of *local conditional probabilities.* The state update takes place synchronously and locally for every unit. Mathematically, this is reflected in the Markovian factorization of joint probabilities:

$$Prob(X^t = x^0 | X^{t-1} = x^{-1}, X^{t-2} = x^{-2}, ..., U^t = u^0, U^{t-1} = u^{-1}, U^{t-2} = u^{-2}...) =$$

$$\prod_{s\in S_h} Prob(X_s^t = x_s^0 | X_{N_h(s)}^{t-1} = x_{N_h(s)}^{-1}, U_{N_i(s)}^t = u_{N_i(s)}^0). \tag{6}$$

Similarly,

$$Prob(Y^t = y^0 | ..., X^{t+1} = x^1, X^t = x^0, X^{t-1} = x^{-1}, ..., U^{t+1} = u^1, U^t = u^0, U^{t-1} = u^{-1}, ...) =$$

$$\prod_{s\in S_o} Prob(Y_s^t = y_s^0 | X_{N_h(s)}^t = x_{N_h(s)}^0, U_{N_i(s)}^t = u_{N_i(s)}^0). \tag{7}$$

The process $\{X^t\}_{t=1}^{\infty}$ is obviously Markov, not only in time, but also *locally* within the network. Therefore, if we define the **local conditional probabilities** for all $s \in S_h \cup S_i$, $a \in A$, $b \in A^{|N_h(s)|}$, $c \in A^{|N_i(s)|}$

$$p_s(a|b,c) \doteq Prob(X_s^t = a | X_{N_h(s)}^{t-1} = b, U_{N_i(s)}^t = c)$$

we can compute the probability $Prob(X^t|X^{t-1}, U^t)$ in terms of the local conditionals:

$$Prob(X^t = x^0 | X^{t-1} = x^{-1}, X^{t-2} = x^{-2}, ..., U^t = u^0, U^{t-1} = u^{-1}, ...) =$$

$$\prod_{s\in S_h} p_s(x_s^0 | x_{N_h(s)}^{-1}, u_{N_i(s)}^0).$$

Similarly we can compute

$$Prob(Y^t = y^0 | ..., X^{t+1} = x^1, X^t = x^0, X^{t-1} = x^{-1}, ..., U^{t+1} = u^1, U^t = u^0, U^{t-1} = u^{-1}, ...) =$$

$$\prod_{s\in S_o} p_s(y_s^0 | x_{N_h(s)}^0, u_{N_i(s)}^0).$$

The set $\mathcal{P}$ is the set of all the local conditionals:

$$\mathcal{P} \doteq \left\{ p_s(a|b,c), s \in S_h \cup S_o, a \in A, b \in A^{|N_h(s)|}, c \in A^{|N_o(s)|} \right\}.$$

This model can be fully implemented by a network of nonlinear input/output units with additional white noise input:

$$X^t = f(X^{t-1}, U^t, V^t),$$
$$Y^t = g(X^t, U^t, W^t).$$

This is proven in [Keh91a]. Deterministic behavior can be obtained as a special case, when the noise has zero variance. There is a formal similarity of the equations above with the equations of a continuously valued stochastic control system. This formal similarity suggests that we apply stochastic control methods to solve connectionist problems such as prediction and classification. This approach has already been introduced in [Dre90, Keh90, Ruc89, Sin89].

3 Prediction

The problem considered here is the following: given a sample sequence of inputs $U^1 = u^1, U^2 = u^2, ..., U^{t+1} = u^{t+1}$ and outputs $Y^1 = y^1, Y^2 = y^2, ..., Y^t = y^t$ generated by a SRN $\mathcal{M} = ((S, \mathcal{N}), \mathcal{P})$, find a "reasonable" prediction of Y^{t+1}. Repeat for $t = 1, 2, ...$ The prediction is written as

$$\hat{Y}^{t+1}(y^1, ..., y^t, u^1, ..., u^{t+1})$$

to stress the dependence on the past samples. For the rest of the discussion assume the y's, the u's and the SRN $\mathcal{M}$ to be fixed.

We choose our "reasonable" prediction such that it maximizes the Likelihood function. For $t = 1, 2, ...$ and $y \in A^{M_o}$, define

$$\gamma_t(y) \doteq Prob(Y^{t+1} = y \mid Y^1...Y^t = y^1...y^t; U^1...U^{t+1} = u^1...u^{t+1}; \mathcal{M}).$$

(The dependence of γ_t on $y^1, ..., y^t$, $u^1, ..., u^{t+1}$ is suppressed from the notation because the u's and y's are fixed for the rest of this discussion.) Now the Maximum Likelihood Prediction is defined by:

$$\hat{Y}^{t+1}(y^1, ..., y^t, u^1, ..., u^{t+1}) \doteq \arg \max_{y \in A^{M_o}} \gamma_t(y).$$

For fixed sample $y^1, ..., y^t$, $u^1, ..., u^{t+1}$ and fixed $\mathcal{M}$, we have $Prob(Y^1...Y^t = y^1...y^t, U^1...U^{t+1} = u^1...u^{t+1}; \mathcal{M})$ is fixed. Then:

$$\begin{aligned}
&\arg \max_{y \in A^{M_o}} \left(Prob(Y^{t+1} = y \mid Y^1...Y^t = y^1...y^t, U^1...U^{t+1} = u^1...u^{t+1}; \mathcal{M}) \right) = \\
&\arg \max_{y \in A^{M_o}} \left(Prob(Y^{t+1} = y \mid Y^1...Y^t = y^1...y^t, U^1..U^{t+1} = u^1..u^{t+1}; \mathcal{M}) \cdot \right. \\
&\qquad \left. Prob(Y^1...Y^t = y^1...y^t \mid U^1...U^{t+1} = u^1...u^{t+1}; \mathcal{M}) \right) = \\
&\arg \max_{y \in A^{M_o}} \left(Prob(Y^1...Y^t Y^{t+1} = y^1...y^t y \mid U^1...U^{t+1} = u^1...u^{t+1}; \mathcal{M}) \right). \qquad (8)
\end{aligned}$$

Define for $t = 0, 1, 2, ..., y \in A^{M_o}$

$$\delta_{t+1}(y) \doteq Prob(Y^1...Y^t Y^{t+1} = y^1...y^t y \mid U^1...U^{t+1} = u^1...u^{t+1}; \mathcal{M}).$$

The prediction problem is now reduced to finding an efficient way to compute $\delta_t(y)$ for all $y \in A^{M_o}$. As soon as that is accomplished, we can find the value of y that maximizes $\delta_t(y)$ by exhaustive search. This is the same y that maximizes $\gamma_t(y)$ and hence we have obtained the ML prediction. We will now develop a recursive algorithm to compute the δ's.

The following definitions are from [Keh91a]: The **transition matrix** is defined for all $z, x \in A^{M_h}$, for $t = 1, 2, ...$:

$$P_{zx}(t) \doteq Prob(X^t = x \mid X^{t-1} = z, U^t = u^t; \mathcal{M}).$$

This can be computed in terms of the p's:

$$P_{zx}(t) = \prod_{s \in S_h} p_s(x_s \mid z_{N_h(s)}, u^t_{N_i(s)}). \qquad (9)$$

The **emission matrix** is defined for all $x \in A^{M_h}$, $y \in A^{M_o}$, for $t = 1, 2, ..., T$:

$$Q_{xy}(t) \doteq Prob(Y^t = y \mid X^t = x, U^t = u^t; \mathcal{M}). \tag{10}$$

This can also be computed in terms of the p's:

$$Q_{xy}(t) = \prod_{s \in S_o} p_s(y_s \mid x_{N_h(s)}, u^t_{N_i(s)}).$$

Next we define the **forward probabilities** for all $x \in A^{M_h}$, $t = 1, 2, ...$ [1]

$$\alpha_t(x) \doteq Prob(Y^1 ... Y^t = y^1 ... y^t, X^t = x | U^1 .. U^t = u^1 .. u^t; \mathcal{M}),$$

Now we will obtain an evolution equation for the forward probabilities. This evolution equation in most cases is only approximately true. It is exactly true under the assumption that the input stochastic process $\{U^t\}_{t=1}^{\infty}$ is a sequence of independent random variables. In that case the forward probabilities obey the forward evolution equation for all $x \in A^{M_h}$, $t = 0, 1, 2, ...$:

$$\alpha_{t+1}(x) = \sum_{z \in A^{M_h}} \alpha_t(z) P_{zx}(t+1) Q_{xy^{t+1}}(t+1); \tag{11}$$

Further, if we assume all initial states to be equally likely, we have initial condition $\alpha_0(x) \doteq Prob(X^0 = x) = 1/|A|^{M_h}$ for all $x \in A^{M_h}$.

As already mentioned, the evolution equation holds true only if $U^1, U^2, ...$ are independent of each other (or, trivially, if there is no input process). In many cases we have no reason to expect the input process to be independent but we also have no information about the nature of correlation across time. In that case the assumption of independence is a natural one and in most cases we can expect the evolution equation to hold up to a reasonably good approximation.

Given α_t we can easily compute δ_t:

$$\delta_{t+1}(y) = \sum_{x,z} \alpha_t(x) P_{xz}(t+1) Q_{zy}(t+1) \qquad \forall y \in A^{M_o}$$

This completes the solution of the prediction problem. Putting all the pieces together we get:

Maximum Likelihood Prediction Algorithm

Given a sample sequence of inputs $u^1, u^2, ..., u^{t+1}$ and outputs $y^1, y^2, ..., y^t$ generated by a SRN $\mathcal{M} = ((S, \mathcal{N}), \mathcal{P})$, find the Maximum Likelihood Prediction defined by:

$$\hat{Y}^{t+1}(y^1, ..., y^t, u^1, ..., u^{t+1}) \doteq \arg\max_{y \in A^{M_o}} \gamma_t(y)$$

[1] Note that this definition is slightly different from the one given in [Keh91a] (as well as in the Hidden Markov Models literature [Jel+83]) in that the terminal time t is not fixed but variable; also there is no conditioning on the initial state X^0.

where $\forall y \in A^{M_o}$

$$\gamma_t(y) \doteq Prob(Y^{t+1} = y \mid Y^1 ... Y^t = y^1 ... y^t; U^1 ... U^{t+1} = u^1 ... u^{t+1}; \mathcal{M}).$$

To find the ML prediction, first set

$$\alpha_0(x) = 1/|A|^{M_h} \qquad \forall x \in A^{M_h}.$$

Then for every $t = 1, 2, ..., x, z \in A^{M_h}, y \in A^{M_o}$ compute

$$P_{zx}(t) = \prod_{s \in S_h} p_s(x_s \mid z_{N_h(s)}, u^t_{N_i(s)}).$$

$$Q_{xy}(t) = \prod_{s \in S_o} p_s(y_s \mid x_{N_h(s)}, u^t_{N_i(s)}).$$

$$\alpha_{t+1}(x) = \sum_{z \in A^{M_h}} \alpha_t(z) P_{zx}(t+1) Q_{xy^{t+1}}(t+1).$$

Finally, for all $y \in A^{M_h}$, $t = 1, 2, ...,$ compute

$$\delta_{t+1}(y) = \sum_{x,z} \alpha_t(x) P_{xz}(t+1) Q_{zy}(t+1)$$

and find the maximizing y by exhaustive enumeration; $\delta_{t+1}(y)$ and $\gamma_{t+1}(y)$ are maximized at the same value of y.

This completes the description of the recursive prediction algorithm. A few remarks are in order:

Remark: The procedure we just outlined is a recursive procedure for computing the maximum likelihood estimate of Y^{t+1} given $Y^1, ..., Y^t$, $U^1, ..., U^{t+1}$. As such, it is completely analogous to Kalman filtering [Kal60] of continuous valued stochastic processes.

Remark: Following Kalman's classification of Estimation problems, we note that the prediction problem discussed here is only one of several possible problems. A general class of problems is the following. Given $Y^1, ..., Y^t$, $U^1, ..., U^{t+1}$, compute the Maximum Likelihood estimate of $Z^{t+\tau}$ (where $\{Z^t\}_{t=1}^{\infty}$ is some stochastic process that is (Y, U)-measurable).

1. When $\tau > 0$ we have a problem of prediction.
2. When $\tau = 0$ we have a problem of filtering.
3. When $\tau < 0$ we have a problem of smoothing.

In our case $\{Z^t\}_{t=1}^{\infty}$ is the process $\{Y^t\}_{t=1}^{\infty}$ itself so problems (2) and (3) are trivial, because $\{Y^t\}_{t=1}^{\infty}$ is fully observable. An interesting case for which (1)-(3) are nontrivial is *state* estimation, where $\{Z^t\}_{t=1}^{\infty}$ is the hidden process $\{X^t\}_{t=1}^{\infty}$. The problem can be solved in exactly the same way as the Y^t prediction problem.

Fig.1 about here

Figure 1: Plot of Quantized Speech Waveform - [ah]

Remark: The ML prediction algorithm applies equally well to the prediction of global HMM's. In that case we need not compute the transition and emission matrices (P and Q) as they are given to us directly. The corresponding computation in the algorithm is omitted; the rest of the algorithm is applied in exactly the same manner.

Remark: Essentially the same technique that we apply here has been proposed by Bucy [Bu69, Bu71] for the estimation of continuous valued, nonlinear stochastic systems. The problem in that case is the so called *curse of dimensionality*: the computation of the forward probabilities has to be performed for a continuous set of values; of course this can only be done by an approximation of the state-space by a discrete grid. However, say the hidden process is a K-dimensional vector; then even a gross discretization implies a computational load that is exponential in K. Hence the method is untenable for continuous valued stochastic processes of high dimensionality.

4 An Example of Prediction

In this section we present an example application of the prediction algorithm of the previous section. We will predict future values of a speech waveform. The waveform we will use is a preprocessed steady state segment from an utterance of the phoneme [ah] in the word "one". So as to be able to apply our methods we have to convert this waveform to a finite alphabet stochastic process. We do so by using a 4-level quantized version of the original waveform; this quantized waveform is plotted in Fig.1. We repeat the exact same experiment with a similar waveform, extracted from the steady state segment from an utterance of the phoneme [ou] in the word "one". The 4-level quantized version is plotted in Fig.2.

In Figs.3 and 4 we plot the results of the application of the prediction algorithm; namely we plot the predicted and actual value at every time step t. Let us note that: (a) the SRN model of the phoneme [ou] has been obtained in accordance to the methods of [Keh91a], using the local BF algorithm and (b) in this case we have only an output stochastic processes, but no input; this can be easily accommodated by the prediction algorithm by dropping all u dependence in

Fig.2 about here

Figure 2: Plot of Quantized Speech Waveform - [ou]

Fig.3 about here

Figure 3: Plot of Predicted vs. Actual Speech Waveform

the prediction algorithm.

We see that the predicted values lie quite close to the actual ones and hence the algorithm is quite accurate. It is also very fast as it can be implemented in a Parallel Distributed Processing form; for more details see Section 7.

5 Classification

The classification problem is as follows. Suppose we have N SRN's, $\mathcal{M}_1 = (\mathcal{G}_1, \mathcal{P}_1)$, ..., $\mathcal{M}_N = (\mathcal{G}_N, \mathcal{P}_N)$. We also have a sequence of input/output data $u^1, u^2, ..., u^t$, $y^1, y^2, ..., y^t$ which we know to have been generated by one of the SRN's, but we do not know whether it was $\mathcal{M}_1$, $\mathcal{M}_2$, ... or $\mathcal{M}_N$. We must form a "reasonable" guess as to which SRN model actually produced these observations. This guess, which may be changing with time, is described by a stochastic

Fig.4 about here

Figure 4: Plot of Predicted vs. Actual Speech Waveform

process $\hat{N}^t$ which takes values in $\{1, 2, ..., N\}$. For instance, if $\hat{N}^t = n$ then our guess (at time t) is that the y observations have been produced by $\mathcal{M}_n$ with input u.

We formulate the problem as follows: introduce a new variable Z which takes values in $\{1, 2, ..., N\}$. The data have been produced by $\mathcal{M}_Z$; e.g. if $Z = n$, then the sequence $u^1, ..., u^t$ $y^1, ..., y^t$ was produced by SRN $\mathcal{M}_n$. Notice the difference between Z, which is fixed for all time, reflecting the fact that the observed data $u^1, u^2, ...,\ y^1, y^2, ...$ are indeed produced by a fixed SRN, and $\hat{N}_t$ which changes in time as more data is collected, and reflects our *opinion* as to which is the "true" SRN model. This opinion may change over time, so the value $\hat{N}^\tau$ may be different from that of $\hat{N}^t$ for $t \neq \tau$.

Now we adopt a *Bayesian* point of view. Z is a random variable. Once it takes a value, this value will remain fixed. However, our *prior* knowledge as to what the value of Z is, is not complete; it is described in terms of a *prior probability distribution* $p_n^0 \doteq Prob(Z = n)$, $n = 1, 2, ..., N$. As we collect more data, our knowledge about the value of Z will change and this will be reflected on the posterior distribution $p_n^t(y^1, ..., y^t, u^1, ..., u^t)$ (sometimes denoted simply by p_n^t) which is defined by:

$$p_n^t(y^1, ..., y^t, u^1, ..., u^n) \doteq Prob(Z = n \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^t = u^t).$$

Given $p_n^t(y^1, ..., y^t, u^1, ..., u^t)$, we set

$$\hat{N}^t(y^1, ..., y^t, u^1, ..., u^t) \doteq \arg \max_{1 \leq n \leq N} p_n^t(y^1, ..., y^t, u^1, ..., u^t).$$

That is, at time t we claim that the data $y^1, ..., y^t, u^1, .., u^t$ was produced by $\hat{N}^t$, which maximizes the posterior probability. This is a very reasonable choice, usually referred to as *Maximum A Posteriori* estimate. The issue then is to find an efficient way to compute p_n^t, $t = 1, 2, ...$, $n = 1, 2, ..., N$. We now present a recursive algorithm to do this. Note that

$$p_n^{t+1} = Prob(Z = n \mid Y^1 = y^1, ..., Y^{t+1} = y^{t+1}, U^1 = u^1, ..., U^{t+1} = u^{t+1}) =$$

$$\frac{Prob(Y^{t+1}=y^{t+1},Z=n \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1})}{Prob(Y^{t+1}=y^{t+1} \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1})}=$$

$$\frac{Prob(Y^{t+1}=y^{t+1},Z=n \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1})}{\sum_{m=1}^{N} Prob(Y^{t+1}=y^{t+1},Z=m \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1})}. \quad (12)$$

Also note that

$$Prob(Y^{t+1}=y^{t+1},Z=n \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1})=$$

$$Prob(Y^{t+1}=y^{t+1} \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1},Z=n)\cdot$$

$$Prob(Z=n \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1})=$$

$$Prob(Y^{t+1}=y^{t+1} \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1},Z=n)\cdot p_n^t. \quad (13)$$

Now (12), (13) imply the recursion:

$$p_n^{t+1}=\frac{Prob(Y^{t+1}=y^{t+1} \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1},Z=n)\cdot p_n^t}{\sum_{m=1}^{N} Prob(Y^{t+1}=y^{t+1} \mid Y^1=y^1,...,Y^t=y^t,U^1=u^1,...,U^{t+1}=u^{t+1},Z=m)\cdot p_m^t}.$$

What remains to be done is finding a recursive way to compute the quantity $Prob(y^t \mid y^1, ..., y^{t-1}, u^1, ..., u^{t-1}, u^t; Z=n)$ for $n=1,2,...,N$, $t=1,2,...$. Note however that if $Z=n$ then the *true* model is $\mathcal{M}_n$ and the probabilities in all the equations above can be computed by using the parameter set $\mathcal{P}_n$.

$$Prob(y^{t+1} \mid y^1,...,y^t,u^1,...,u^t,u^{t+1},Z=n)=$$

$$Prob(y^{t+1} \mid y^1,...,y^t,u^1,...,u^t,u^{t+1};\mathcal{P}_n)=$$

$$\frac{Prob(y^{t+1},y^t,...,y^1 \mid u^{t+1},...,u^1;\mathcal{P}_n)}{Prob(y^t,...,y^1 \mid u^t,...,u^1;\mathcal{P}_n)}=\frac{\sum_x \alpha_{t+1}^n(x)}{\sum_x \alpha_t^n(x)}.$$

We compute the α probabilities using the following equations for $n=1,2,...,N$, $t=1,2,...$, $x,z\in A^{M_h}$, $y\in A^{M_o}$

$$P_{zx}^n(t)=\prod_{s\in S_h} p_s^n(x_s \mid z_{N_h(s)},u_{N_i(s)}^t).$$

$$Q_{x,y}^n(t)=\prod_{s\in S_o} p_s^n(y_s \mid x_{N_h(s)},u_{N_i(s)}^t).$$

These are exactly the same as (9), 10) in Section 3; the n superscript implies that we compute them using the local conditionals $\mathcal{P}_n$. The same holds true for the α^n probabilities which are computed according to (11), using the parameter set (local conditionals) $\mathcal{P}_n$. This completes the description of the classification algorithm. Putting all the pieces together we get:

Maximum A Posteriori Classification Algorithm
(Lainiotis Classification Algorithm)

Given a sample sequence of inputs $u^1, u^2, ..., u^{t+1}$ and outputs $y^1, y^2, ..., y^t$ and a set of SRN's $\mathcal{M}_1 = ((S,\mathcal{N})_1, \mathcal{P}_1)$, ..., $\mathcal{M}_N = ((S,\mathcal{N})_N, \mathcal{P}_N)$, we know that the data sequence has been produced by the SRN $\mathcal{M}_Z$, where Z is a random variable with probability distribution $p_n^0 = Prob(Z = n)$, $n = 1, 2, ..., N$. The Maximum A Posteriori Classification of the data sequence $y^1, ..., y^t, u^1, ..., u^t$ is $\mathcal{M}_{\hat{N}^t}$. Here $\hat{N}^t$ is given by

$$\hat{N}^t \doteq \arg \max_{1 \le n \le N} p_n^t(y^1, ..., y^t, u^1, ..., u^t)$$

and p_n^t for $n = 1, ..., N$, $t = 1, 2, ...$ is defined by

$$p_n^t(y^1, ..., y^t, u^1, ..., u^n) \doteq Prob(Z = n \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^t = u^t).$$

To obtain the Maximum A Posteriori classification we need to compute p_n^t for $t = 1, 2, ..$, $n = 1, 2, ..., N$. To do this, first set

$$\alpha_0^n(x) = 1/|A|^{M_h} \qquad \forall x \in A^{M_h}, \qquad n = 1, 2, ..., N$$

then for every $t = 1, 2, ..., n = 1, 2, ..., N$, $x, z \in A^{M_h}$, $y \in A^{M_o}$, compute

$$P_{zx}^n(t) = \prod_{s \in S_h} p_s^n(x_s \mid z_{N_h(s)}, u_{N_i(s)}^t).$$

$$Q_{x,y}^n(t) = \prod_{s \in S_o} p_s^n(y_s \mid x_{N_h(s)}, u_{N_i(s)}^t).$$

$$\alpha_{t+1}^n(x) = \sum_{z \in A^{M_h}} \alpha_t^n(z) P_{zx}^n(t) Q_{xy^{t+1}}^n;$$

$$Prob(y^{t+1} \mid y^1, ..., y^t, u^1, ..., u^t, u^{t+1}, Z = n) = \frac{\sum_x \alpha_{t+1}^n(x)}{\sum_x \alpha_t^n(x)};$$

$$p_n^{t+1} = \frac{Prob(Y^{t+1} = y^{t+1} \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^{t+1} = u^{t+1}, Z = n) \cdot p_n^t}{\sum_{m=1}^N Prob(Y^{t+1} = y^{t+1} \mid Y^1 = y^1, ..., Y^t = y^t, U^1 = u^1, ..., U^{t+1} = u^{t+1}, Z = m) \cdot p_m^t}.$$

Remark: This completes the description of the MAP classification algorithm. *This is the adaptation of the Lainiotis Partition algorithm [Lai69, Lai71] to finite state systems.* Similar methods have been used by Nowlan [Now90a, Now90b, Now90c] for static problems.
Remark: The MAP classification algorithm applies equally well to the classification of global HMM's. In that case we need not compute the transition and emission matrices (P and Q) as they are given to us directly. The corresponding computation in the algorithm is omitted; the rest of the algorithm is applied in exactly the same manner.

Fig.5 about here

Figure 5: Evolution of p_1^t, p_2^t

6 An Example of Classification

For the classification problem we use the two quantized waveforms of Section 4 (Figs. 2 and 4). So we have two candidate models ($N = 2$) with known SRN models (these models were computed using the local BF algorithm as described in [Keh91a]; in particular no input process was used).

The first classification experiment involves using as $y^1, ..., y^{180}$ the waveform of Fig.1 (phoneme [ah]). Things work out exactly as expected: the algorithm picks up the right model $\mathcal{M}_1$ and assigns to it a posterior probability which rapidly rises to practically 1. Consequently model $\mathcal{M}_2$ is assigned probability 0. The evolution of the p_1^t, p_2^t probabilities is plotted in Fig. 5.

Exactly similar results obtain in the second experiment, which is identical to the first one, except that the sequence $y^1, ..., y^{180}$ is now the waveform of Fig.2 (phoneme [ou]). The algorithm picks up the right model $\mathcal{M}_2$ and assigns to it a posterior probability which rapidly rises to practically 1. Consequently model $\mathcal{M}_1$ is assigned probability 0. The evolution of the p_1^t, p_2^t probabilities is plotted in Fig. 6.

Finally we apply the algorithm to a trickier problem. Namely, we use a waveform which is a concatenation of the waveforms in Fig.2 and Fig.4. The composite waveform is plotted in Fig. 7.

Now, the derivation of our Classification algorithm was based on the assumption that the waveform we have been using was produced by a single SRN. As the assumption is obviously violated in this case, there is no theoretical guarantee that the algorithm should work. Still, the sample sequence does come from a fixed SRN over long periods of time. If we could introduce some type of forgetting factor in the algorithm, we would expect that shortly after every model switching, the algorithm should readjust the probability values, just because it forgets the old values of the data sequence.

The method we use to introduce forgetting of old parts of the sequence is entirely *ad hoc* and we have no theoretical justification for it. On the other hand, as we will presently show, it works satisfactorily, so there is some merit to it.

Fig.6 about here

Figure 6: Evolution of p_1^t, p_2^t

Fig.7 about here

Figure 7: Composite Waveform

Fig.8 about here

Figure 8: Evolution of p_1^t, p_2^t superimposed on original waveform

The trick we use is pretty simple. The exact update for the forward probabilities is given by:

$$\alpha_{t+1}^n(x) = \sum_{z \in A^{M_h}} \alpha_t^n(z) P_{zx}^n(t) Q_{xy^{t+1}}^n; \tag{14}$$

Instead of using this update, we use:

$$\alpha_{t+1}^n(x) = \sum_{z \in A^{M_h}} \alpha_t^n(z) P_{zx}^n(t) Q_{xy^{t+1}}^n + \epsilon \tag{15}$$

where ϵ is small positive constant. The result of this modification is the following: probabilities never become extremely small. For models which, under the observed data, have relatively high probability the small ϵ does not make a big difference and (14), (15) yield practically the same results. On the other hand, when under the observed data a model $\mathcal{M}_n$ has very low probability, ϵ is a lower bound for α^n. This forces the algorithm to forget that the past observations essentially rule out model $\mathcal{M}_n$. If at a later time the data-producing mechanism switches to a new model, the algorithm will quickly forget the model's incompatibility with past observations and will concentrate on more recent values. Hence the posterior probability will soon rebound.

Indeed, when we run the algorithm using the trick described above, we obtain the following evolution of the conditional probabilities, plotted in Fig.9. The same plot is superimposed to the original waveform in Fig.8, for illustrative purposes. This is exactly the behavior we described in the previous paragraph.

7 Recurrent Parallel Distributed Processing

We have proposed prediction and classification algorithms for finite alphabet time series. The derivation of these algorithms was based on the assumption that the time series was produced by a Stochastic Recurrent Network. First of all, let us point out once again, that these algorithms

Fig.9 about here

Figure 9: Evolution of p_1^t, p_2^t

would apply equally well to time series that are produced by a global model, say a Hidden Markov Model. The local character of the SRN models comes into play only at the initial stage of our algorithms, where we use the local conditionals parametrization to compute the global transition and emission matrix. If these matrices were given (which would be the case for a Hidden Markov Model), we omit the initial stage of the algorithm and proceed as usual.

The SRN representation has certain advantages when compared to the global HMM representation (see [Keh91a]), particularly with connection to parallel computation and the statistical principle of parsimonious modelling.

It is a matter of some interest to note that starting with a stochastic recurrent network, we obtain another network which computes probabilities associcated with the original SRN. Call the new network a *meta-network* (as it computes statistics of the original network) and note a certain duality: the SRN is stochastic but the meta-network is deterministic; the SRN is finite state but the meta-network is continuous state. Finally they both are dynamic, recurrent networks.

As soon as we fix the model which we assume to be generating the observed data, all the computations of the prediction and classification algorithms can be performed on the meta-network of parallel distributed processors. For instance, the evolution of the α probabilities described by the (11) equations in Section 3 can be implemented on a recurrent network of A^{M_h} units, with summations and a nonlinear function of the input/output observations. The taking of maxima implicit in (8) can be implemented by a softmax unit which is connected to the output of all these processors etc. That parallel algorithms can be programmed on connectionist networks has been reported by several authors [Keh90, Brid89, Kung89, Vlo89]. These networks are definitely of a different flavor than the more classical, Back-Propagation trained ones. In our case the training is limited to the modelling phase [Keh91a]; the meta-networks are handcrfated to implement the appropriate algorithms. Yet the networks are indeed recurrent, parallel distributed processing networks and it is not unlikely that they have advantages as parallel implementations over sequential/global versiosn of the same algorithms. This is a question which cannot be answered without further research.

8 Conclusions

We have presented the ML prediction algorithm and the Lainiotis MAP classification algorithm which apply to SRN's as well as to HMM's. These algorithms can be implemented on recurrent PDP meta-networks. There is a certain duality between the SRN and the meta-network; the SRN is finite state and stochastic, whereas the meta-network is continuous state and deterministic. Both are dynamic networks.

Our algorithms exhibit fast and robust behavior in practical applications and perform their assigned tasks very succesfully. There is no training issue here - all the training is performed at the SRN level, whereas the meta-network is "handcrafted". In the future we would like to apply our algorithms to practical tasks, especially to speech recogntion. It appears that phoneme classification could be performed succesfully by the Lainiotis algorithm.

References

[Aok87] M. Aoki. *State Space Modelling of Time Series.* Springer, Berlin, 1987.

[Be+90a] Y. Bengio et al. A hybrid coder for Hidden Markov Models using a recurrent neural network. *Proceedings of the International conference on Acoustics, Speech and Signal Processing*, Albuquerque, NM, April 1990, pp.537-540.

[Be+90b] Y. Bengio et al. Global optimization of neural network-HMM hybrid. Technical Report TR-SOCS-90.22. McGill University, School of Computer Science. December 1990.

[BW88] H. Bourlard and C.J. Wellekens. Links between Markov models and multilayer perceptrons. Technical Report, Phillips Research Lab, 1988.

[BW89] H. Bourlard and C.J. Wellekens. Speech dynamics and recurrent neural nets. In *Proc. of the ICASSP.* IEEE, 1989.

[BD87] P.J. Brockwell and R.A. Davis. *Time Series: Theory and Methods.* Springer, New York, 1987.

[Bri89] J. S. Bridle. Alpha-nets: A recurrent neural network architecture with a hidden Markov model interpretation. Technical Report SP4 Research Note 104, Royal Signals and Radar Establishment, October 1989.

[Bu69] R.S. Bucy. Bayes Theorem and Digital Realizations for Nonlinear Filters. *J. Aust. Sci.*, 17(2):80-94, Sept.-Oct. 1969.

[Bu71] R.S. Bucy and K.D. Senne. Digital Synthesis of Nonlinear Filters. *Automatica*, 7:287-298, 1971.

[Dre90] S.E. Dreyfus. Artificial neural networks, backpropagation and the Kelly-Bryson gradient procedure. *J. Guid. Cont. Dyn.*, 13(5):926–928, Sept.-Oct. 1990.

[Jel83] F. Jelinek and others A Maximum Likelihood approach to continuous speech recognition. IEEE PAMI-5(2), pp.179-190.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *ASME J. Bas. Eng.*, pages 35–45, 1960.

[Keh90] A. Kehagias. Optimal control for training: the missing link between Hidden Markov Models and connectionist networks. *Math. Comp. Mod.*, Vol. 14, pages 284-289, 1990.

[Keh91a] A. Kehagias. Stochastic Recurrent Networks Training by the Local Backward-Forward Algorithm. Technical Report, Division of Applied Mathematics, Brown Un., Providence, Rhode Island, – In preparation.

[Keh91b] A. Kehagias. *Approximation and Estimation of Stochastic Processes by Hidden Markov Models*. PhD thesis, Division of Applied Mathematics, Brown Un., Providence, Rhode Island, May 1991.

[KH89a] S.Y. Kung and J.N. Hwang. A unified modelling of connectionist neural networks. *IEEE ASSP*, 32(1):1–10, 1989.

[KH89b] S.Y. Kung and J.N. Hwang. A unified systolic architecture for artificial neural networks. In *ICASSP'89*, Glasgow - England, May 1989. IEEE.

[KH89c] S.Y. Kung and J.N. Hwang. A unifying viewpoint of multi-layer perceptrons and hidden Markov models. In *Int. Symposium on Circuits and Systems*, Portland - Oregon, 1989. IEEE.

[SL69] F.L. Sims and D.G. Lainiotis. Recursive algorithm for the calculation of the adaptive Kalman filter coefficients. *IEEE AC*, 14(2):215–218, 1969.

[Lai71] D.G. Lainiotis. Optimal adaptive estimation: Structure and pattern adaptation. *IEEE AC*, 16(2):160–170, 1971.

[LF87] A. Lapedes and R. Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical Report LA UR 87-?, Los Alamos National Lab, 1987.

[May82] P.S. Maybeck. *Stochastic Models Estimation and Control*. Mathematics in Science and Engineering. Academic, New York, 1982.

[MD88] J. Moody and C. Darken. Learning with localized receptor fields. In *Connectionist Models Summer School*. Carnegie Mellon University, 1988.

[Now90a] S.J. Nowlan Maximum Likelihood Competition in RBF Networks. Technical Report CRG-TR-90-2, Dept. of Computer Science, Un. of Toronto, February 1990.

[Now90b] S.J. Nowlan. Competing Experts: An Experimental Investigation of Associative Mixture Models. Technical Report CRG-TR-90-5, Dept. of Computer Science, Un. of Toronto, September 1990.

[Now90c] S.J. Nowlan. Maximum Likelihood Competitive Learning. In *Advances in Neural Information Processing Systems 2*, ed. D. Touretzky. Morgan Kauffman, 1990.

[Rob88] Anthony J. Robinson. A dynamic connectionist model for phoneme recognition. Technical report, Cambridge University Engineering Department, Cambridge, England, 1988.

[Rob89] Anthony J. Robinson. *Dynamic Error Propagation Networks*. PhD thesis, Cambridge University Engineering Department, Cambridge, England, 1989.

[Ruc89] D.W. Ruck et al. Back Propagation: A Degenerate Kalman Filter. Submitted, *IEEE PAMI*.

[Sin89] S. Singal and L. Wu. Training Multilayer Perceptrons with the extended Kalman algorithm. In *Advances in Neural Network Information Processing*, D.S. Touretzky, ed.. Morgan Kauffman, 1989.

[Sut88] R.S. Sutton. Learning to predict by the methods of temporal difference. *Machine Learning*, 3:9–44, 1988.

[Vlo89] J.A. Vlontzos et al. A Systolic Neural Network Architecture for Hidden Markov Models. *IEEE ASSP*, 37(12):1967-1979, Dec. 1989.

[W+89a] A. Waibel et al. Modularity and scaling in large phonemic neural networks. *IEEE ASSP*, ASSP-37(12):1888–1898, December 1989.

[W+89b] A. Waibel et al. Phoneme recognition using time-delay neural networks. *IEEE ASSP*, 37(3):328–339, March 1989.

[WZ88] R.J. Williams and D. Zipser. A learning algorithm for continually running fully connected recurrent neural networks. Technical Report ICS-8805, Un. of California at San Diego, 1988.