

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

May 31, 1997

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, \dots, Y^t and the input up to the immediate future: U^1, \dots, 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, \dots, Y^t and the input up to the present time: U^1, \dots, U^t , as well as a number of candidate SRN's: $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_K$, find the network that has Maximum Posterior Probability of producing Y^1, \dots, Y^t . We apply our algorithms to prediction and classification of speech waveforms.

Contents

1	Introduction	3
2	The SRN Model	5
3	Prediction	7
4	An Example of Prediction	11
5	Classification	13
6	An Example of Classification	16
7	Recurrent Parallel Distributed Processing	20
8	Conclusions	21

NOTATION

Given a finite set A , we denote the number of elements in A by $|A|$. E.g., for $A = \{a_1, \dots, a_N\}$, $|A| = N$. 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 $Z^t = [Z_1^t \dots Z_N^t]$ and Z_n^t is either 0 or 1 for every $t, n = 1, \dots, N$. In that case the alphabet is $A^N \doteq \{[a_1 \dots a_N] : a_n \in A, n = 1, \dots, N\}$.

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 $Prob(X^{t+1} = x^1, \dots, X^{t+\tau} = x^\tau)$ for the probability that $X^{t+1} \dots X^{t+\tau}$ equals $x^1 \dots x^\tau \in A^\tau$. Say $x = [x_1 \dots x_m]$, $y = [y_1 \dots y_n]$; then the **concatenation** of x, y is $xy = [x_1 \dots x_m y_1 \dots y_n]$.

We will often consider probabilities that depend on the value of a certain parameter, say \mathcal{P} . Then we write, for instance, $Prob(X^{t+1} = x^1, \dots, X^{t+\tau} = x^\tau; \mathcal{P})$. 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 $Prob(X^{t+1} = x^1, \dots, X^{t+\tau} = x^\tau; \mathcal{M})$.

We often consider a set $S = \{1, \dots, N\}$ and vector $x = [x_1 \dots x_N]$. We sometimes write x_S in place of x . Similarly, for a set $R = \{r_1, \dots, r_M\} \subset S$ we write x_R in place of $[x_{r_1} \dots x_{r_M}]$. 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 $Y^1 = y^1, Y^2 = y^2, \dots, Y^t = y^t$ of a time series (and, when appropriate, the input history up to present $U^1 = u^1, \dots, U^{t+1} = u^{t+1}$) 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 recognition 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: $\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_N$, find the SRN that (given the input up to the present time: $U^1 = u^1, \dots, U^t = u^t$) is most likely to have produced $Y^1 = y^1, \dots, Y^t = y^t$. 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, \dots is produced by a known SRN $\mathcal{M} = ((S, \mathcal{N}), \mathcal{P})$ with input u_1, u_2, \dots (this is the modelling step, considered in [Keh91a]). Then we proceed to compute the conditional probability of $Y^{t+1} = a$ given $Y^1 = y^1, \dots, Y^t = y^t$ $U^1 = u^1, \dots, U^{t+1} = u^{t+1}$:

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

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

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

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

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

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 , $t = 1, 2, \dots$. The equations these vectors satisfy are:

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

$$Y^t = f(X^t, U^t, W^t); \quad (5)$$

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 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 terms 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 $A = \{0, 1, \dots, K-1\}$. 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**:

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

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

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

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 $p_X(x^1, \dots, x^m)$, $p_Y(y^1, \dots, y^m)$ for all m , $x^1, \dots, x^m, y^1, \dots, y^m$.

The directed graph \mathcal{G} is itself a pair $\mathcal{G} = (S, \mathcal{N})$, where $S = \{s_1, \dots, s_M\}$ 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:

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

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

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

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:

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

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

Similarly,

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

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

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:

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

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

Similarly we can compute

$$Prob(Y^t = y^0 | \dots, X^{t+1} = x^1, X^t = x^0, X^{t-1} = x^{-1}, \dots, U^{t+1} = u^1, U^t = u^0, U^{t-1} = u^{-1}, \dots) = \prod_{s \in S_o} p_s(y_s^0 | x_{N_h(s)}^0, u_{N_i(s)}^0).$$

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 $U^1 = u^1, U^2 = u^2, \dots, U^{t+1} = u^{t+1}$ and outputs $Y^1 = y^1, Y^2 = y^2, \dots, Y^t = y^t$ generated by a SRN $\mathcal{M} = ((S, \mathcal{N}), \mathcal{P})$, find a “reasonable” prediction of Y^{t+1} . Repeat for $t = 1, 2, \dots$. The prediction is written as

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

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 $t = 1, 2, \dots$ and $y \in A^{M_o}$, define

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

(The dependence of γ_t on $y^1, \dots, y^t, u^1, \dots, u^{t+1}$ 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:

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

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

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

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

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

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 $t = 1, 2, \dots$:

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

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

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

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

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

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

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

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

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

¹Note 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 .

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 $x \in A^{M_h}$, $t = 0, 1, 2, \dots$:

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

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

As already mentioned, the evolution equation holds true only if U^1, U^2, \dots 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 :

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

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

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

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

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

To find the ML prediction, first set

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

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

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

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

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

Finally, for all $y \in A^{M_h}$, $t = 1, 2, \dots$, 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 $Y^1, \dots, Y^t, U^1, \dots, U^{t+1}$. 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 $Y^1, \dots, Y^t, U^1, \dots, U^{t+1}$, 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.

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,

Fig.1 about here

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

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 the prediction algorithm.

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

Fig.4 about here

Figure 4: Plot of Predicted vs. Actual Speech Waveform

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, $\mathcal{M}_1 = (\mathcal{G}_1, \mathcal{P}_1), \dots, \mathcal{M}_N = (\mathcal{G}_N, \mathcal{P}_N)$. We also have a sequence of input/output data $u^1, u^2, \dots, u^t, y^1, y^2, \dots, y^t$ which we know to have been generated by one of the SRN's, but we do not know whether it was $\mathcal{M}_1, \mathcal{M}_2, \dots$ 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 process \hat{N}^t which takes values in $\{1, 2, \dots, N\}$. 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 $\{1, 2, \dots, N\}$. The data have been produced by \mathcal{M}_Z ; e.g. if $Z = n$, then the sequence $u^1, \dots, u^t, y^1, \dots, y^t$ 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 $u^1, u^2, \dots, y^1, y^2, \dots$ 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}^τ 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 $p_n^0 \doteq \text{Prob}(Z = n)$, $n = 1, 2, \dots, N$. As we collect more data, our knowledge about the value of Z will change and this will be reflected on the posterior distribution $p_n^t(y^1, \dots, y^t, u^1, \dots, u^t)$ (sometimes denoted simply by p_n^t) which is defined by:

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

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

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

That is, at time t we claim that the data $y^1, \dots, y^t, u^1, \dots, u^t$ 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 , $t = 1, 2, \dots$, $n = 1, 2, \dots, N$. We now present a recursive algorithm to do this. Note that

$$\begin{aligned} p_n^{t+1} &= \text{Prob}(Z = n \mid Y^1 = y^1, \dots, Y^{t+1} = y^{t+1}, U^1 = u^1, \dots, U^{t+1} = u^{t+1}) = \\ &= \frac{\text{Prob}(Y^{t+1} = y^{t+1}, Z = n \mid Y^1 = y^1, \dots, Y^t = y^t, U^1 = u^1, \dots, U^{t+1} = u^{t+1})}{\text{Prob}(Y^{t+1} = y^{t+1} \mid Y^1 = y^1, \dots, Y^t = y^t, U^1 = u^1, \dots, U^{t+1} = u^{t+1})} = \\ &= \frac{\text{Prob}(Y^{t+1} = y^{t+1}, Z = n \mid Y^1 = y^1, \dots, Y^t = y^t, U^1 = u^1, \dots, U^{t+1} = u^{t+1})}{\sum_{m=1}^N \text{Prob}(Y^{t+1} = y^{t+1}, Z = m \mid Y^1 = y^1, \dots, Y^t = y^t, U^1 = u^1, \dots, U^{t+1} = u^{t+1})}. \end{aligned} \quad (12)$$

Also note that

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

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

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

What remains to be done is finding a recursive way to compute the quantity $\text{Prob}(y^t \mid y^1, \dots, y^{t-1}, u^1, \dots, u^{t-1}, u^t; Z = n)$ for $n = 1, 2, \dots, N$, $t = 1, 2, \dots$. 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 .

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\begin{aligned}
Q_{x,y}^n(t) &= \prod_{s \in S_o} p_s^n(y_s \mid x_{N_h(s)}, u_{N_i(s)}^t). \\
\alpha_{t+1}^n(x) &= \sum_{z \in A^{M_h}} \alpha_t^n(z) P_{zx}^n(t) Q_{xy^{t+1}}^n; \\
\text{Prob}(y^{t+1} \mid y^1, \dots, y^t, u^1, \dots, u^t, u^{t+1}, Z = n) &= \frac{\sum_x \alpha_{t+1}^n(x)}{\sum_x \alpha_t^n(x)}; \\
p_n^{t+1} &= \frac{\text{Prob}(Y^{t+1} = y^{t+1} \mid Y^1 = y^1, \dots, Y^t = y^t, U^1 = u^1, \dots, U^{t+1} = u^{t+1}, Z = n) \cdot p_n^t}{\sum_{m=1}^N \text{Prob}(Y^{t+1} = y^{t+1} \mid Y^1 = y^1, \dots, Y^t = y^t, U^1 = u^1, \dots, U^{t+1} = u^{t+1}, Z = m) \cdot p_m^t}.
\end{aligned}$$

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.

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, \dots, 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, \dots, 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.

Fig.5 about here

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

Fig.6 about here

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

Fig.7 about here

Figure 7: Composite Waveform

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.

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

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

Instead of using this update, we use:

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

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

Fig.8 about here

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

Fig.9 about here

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

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 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 associated 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 handcrafted 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 versions 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 successfully. 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 recognition. It appears that phoneme classification could be performed successfully 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.

Fig.1 [ah] phoneme

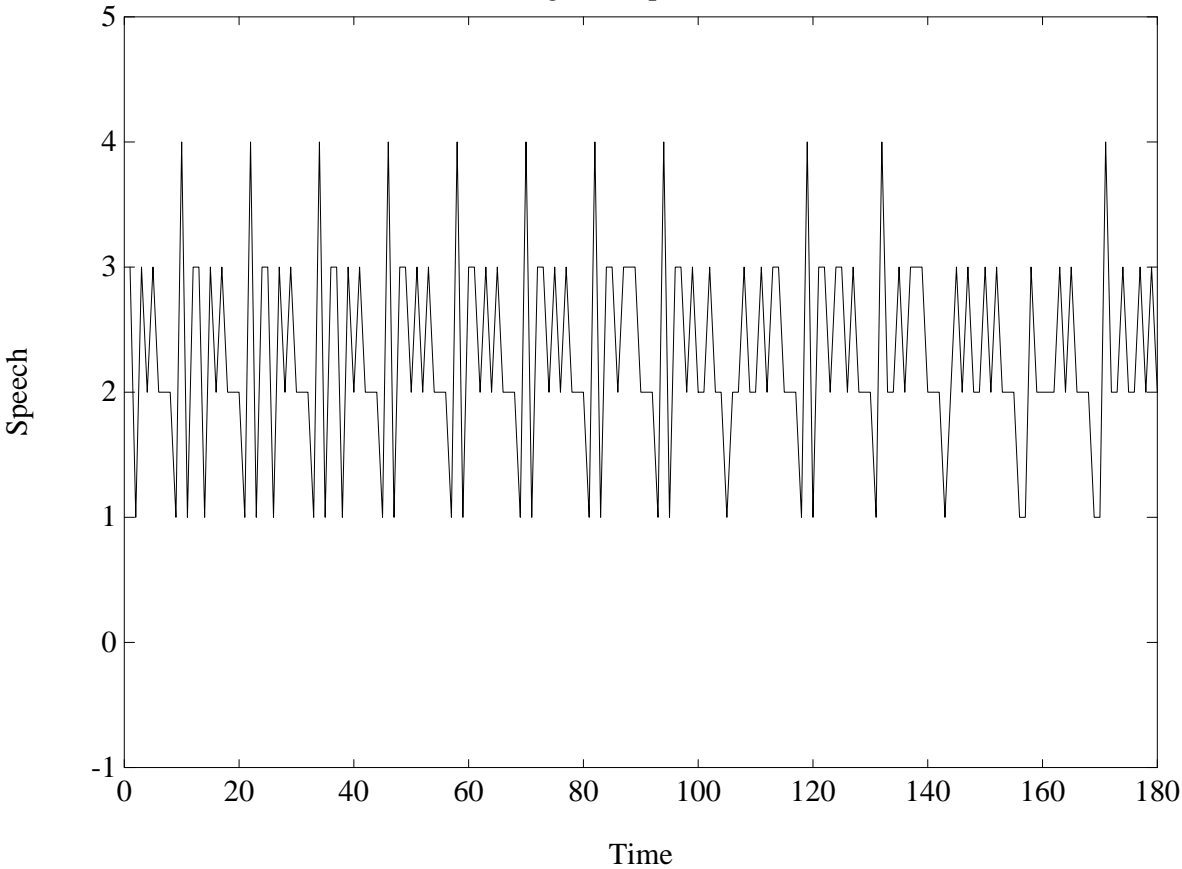


Fig.2 [ah] phoneme

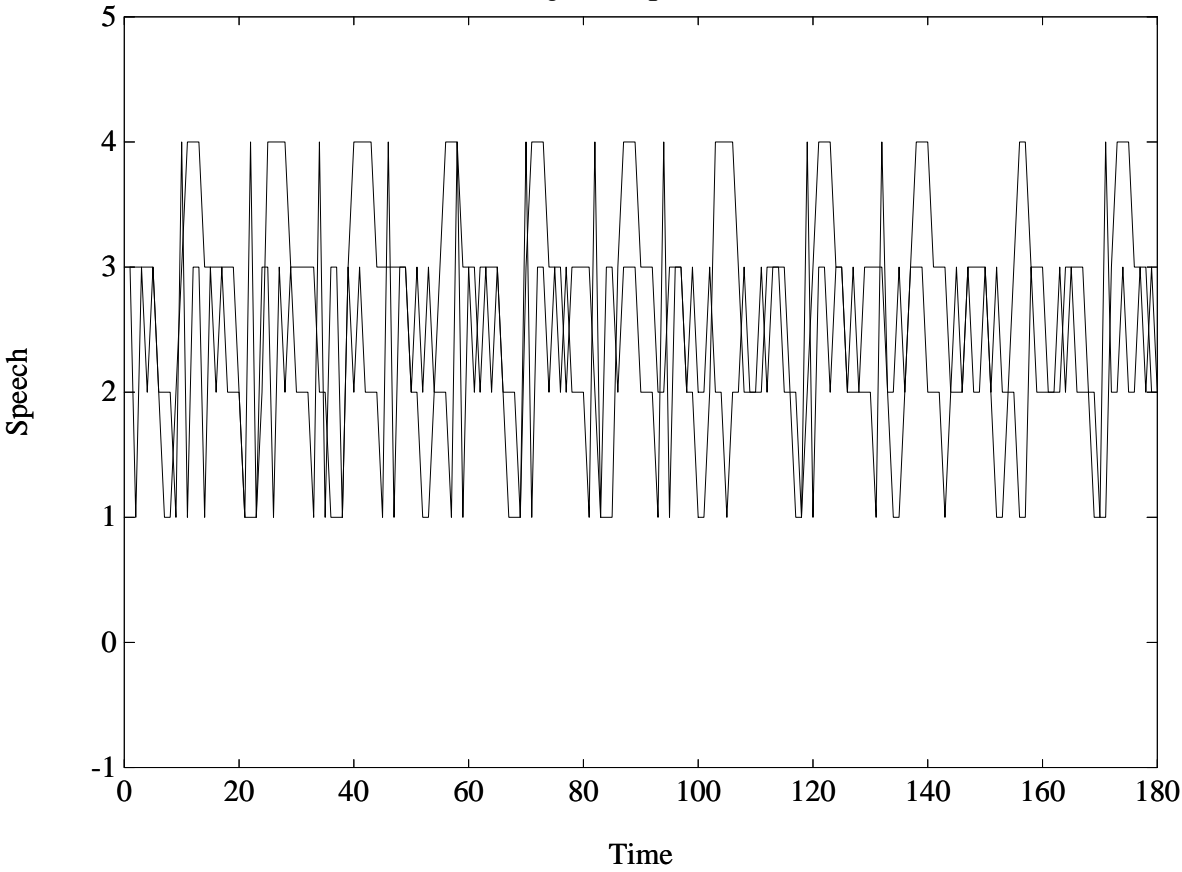


Fig.3: Actual Fig.1 Phoneme Speech Waveform

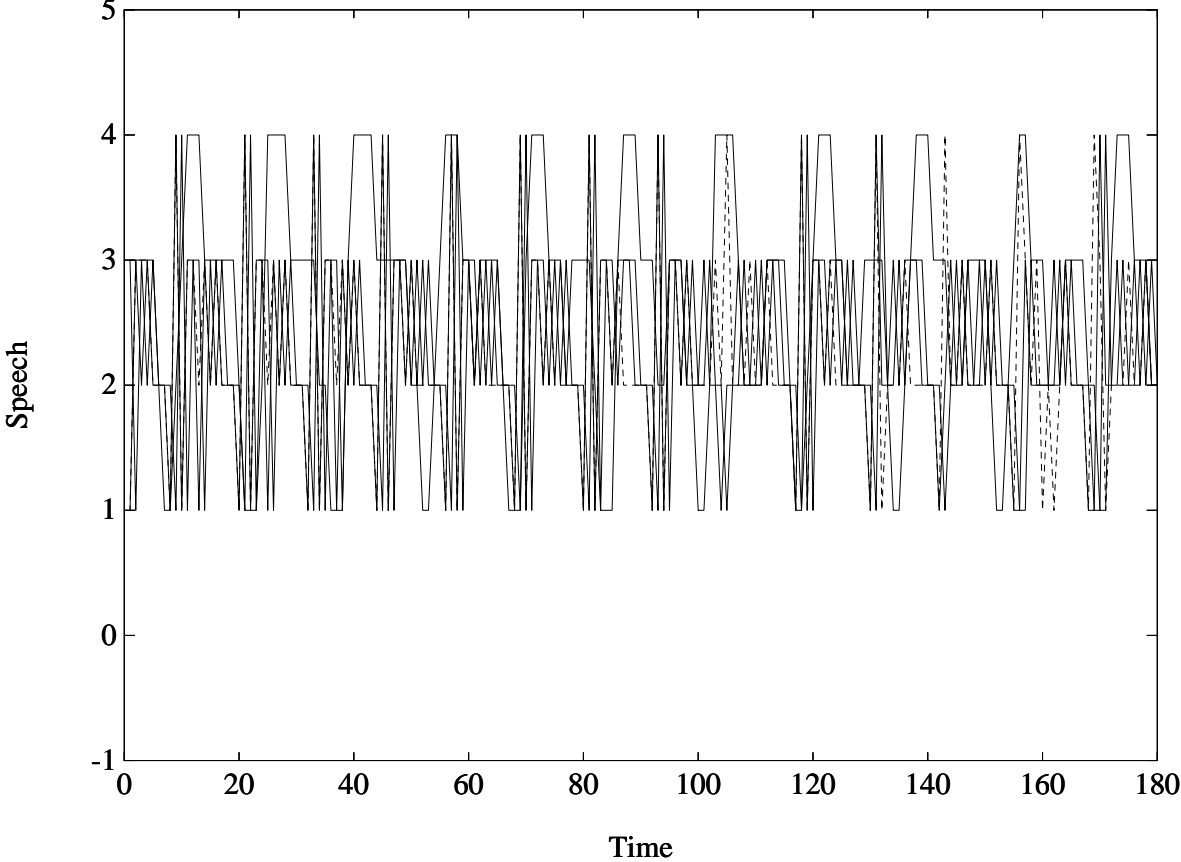


Fig.3: Actual and Predicted Speech Waveform

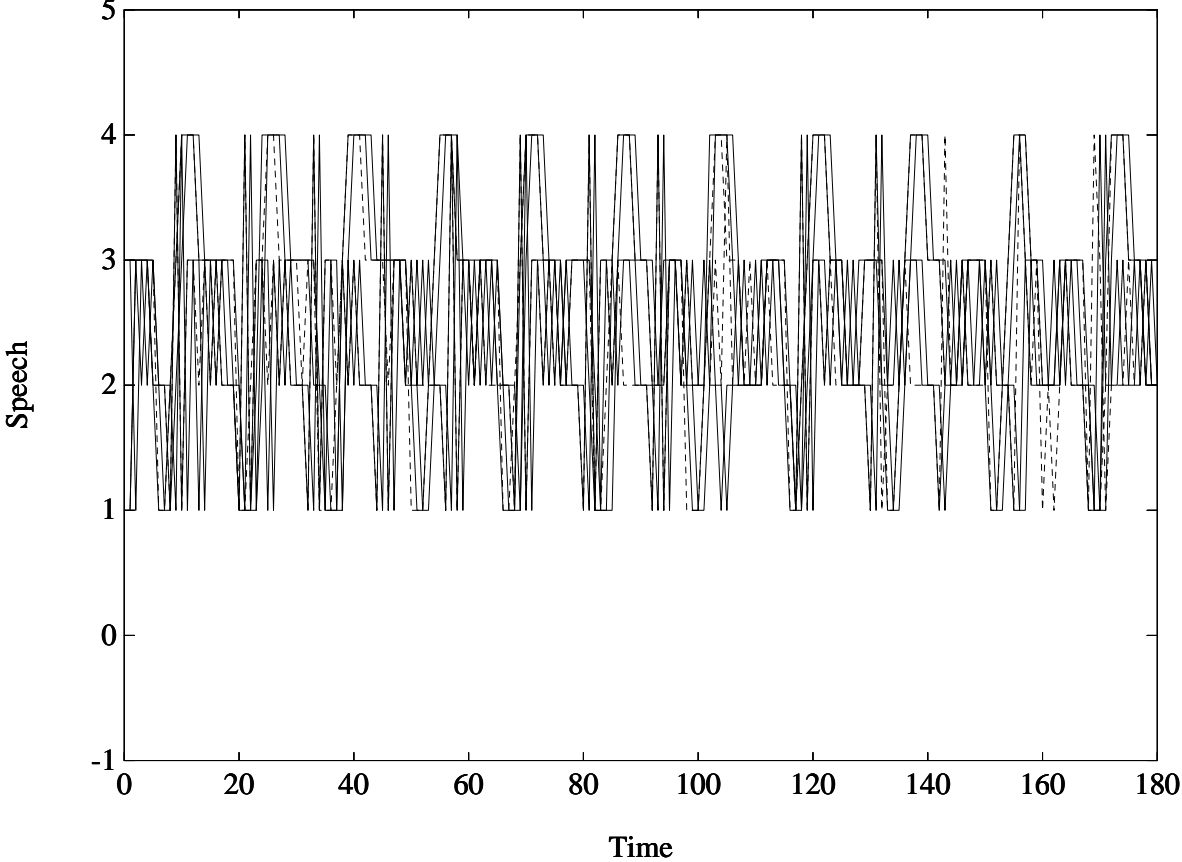


Fig.5 Fig.3 Actual High Frequency Speech Waveform and 2

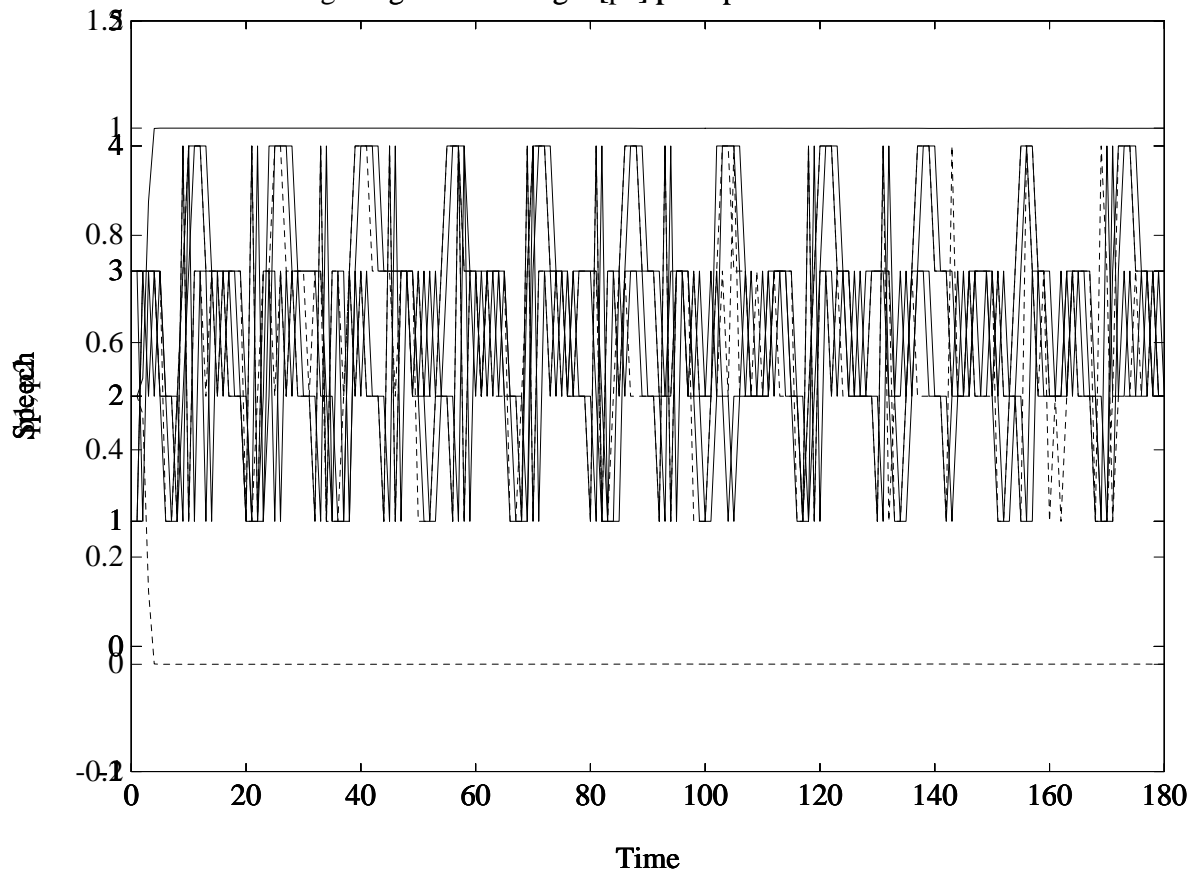


Fig. 5. Fig. 3. Actual and Predicted Speech for Word 2

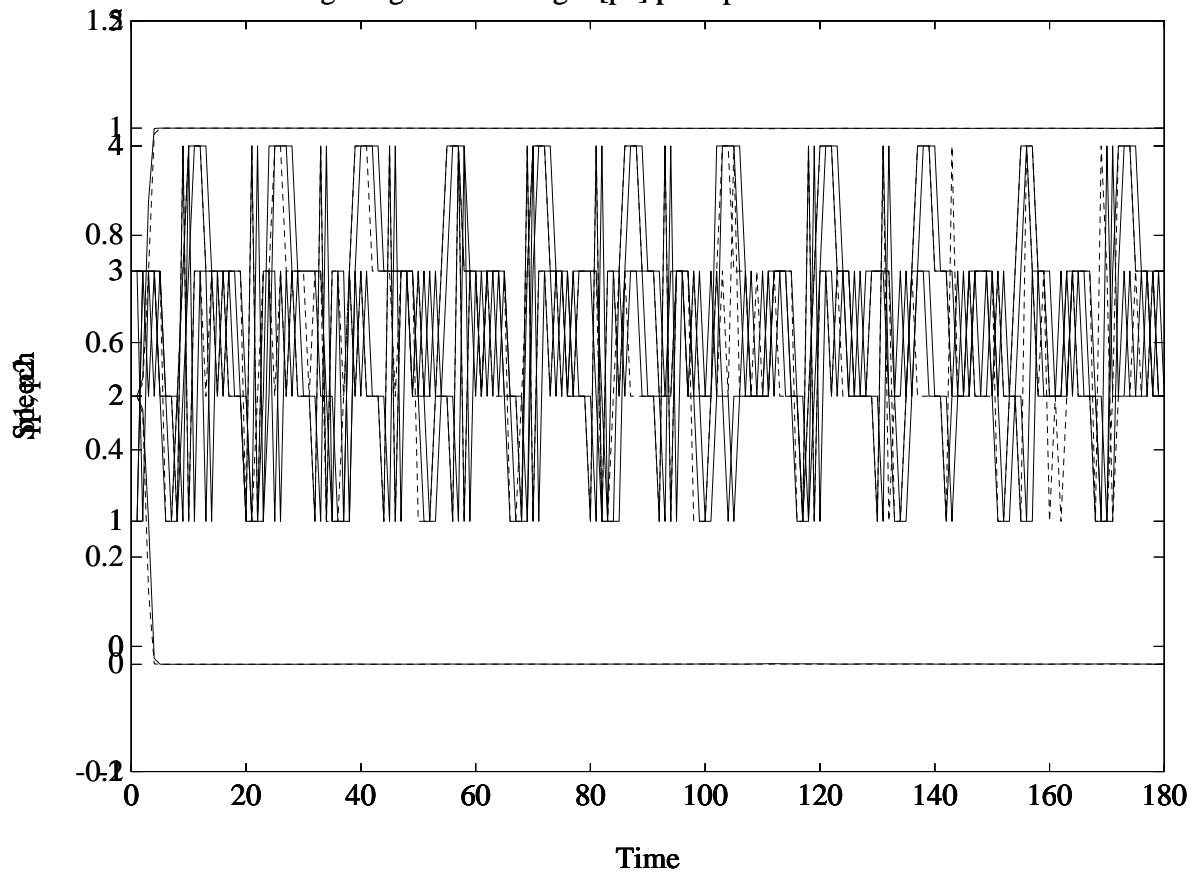
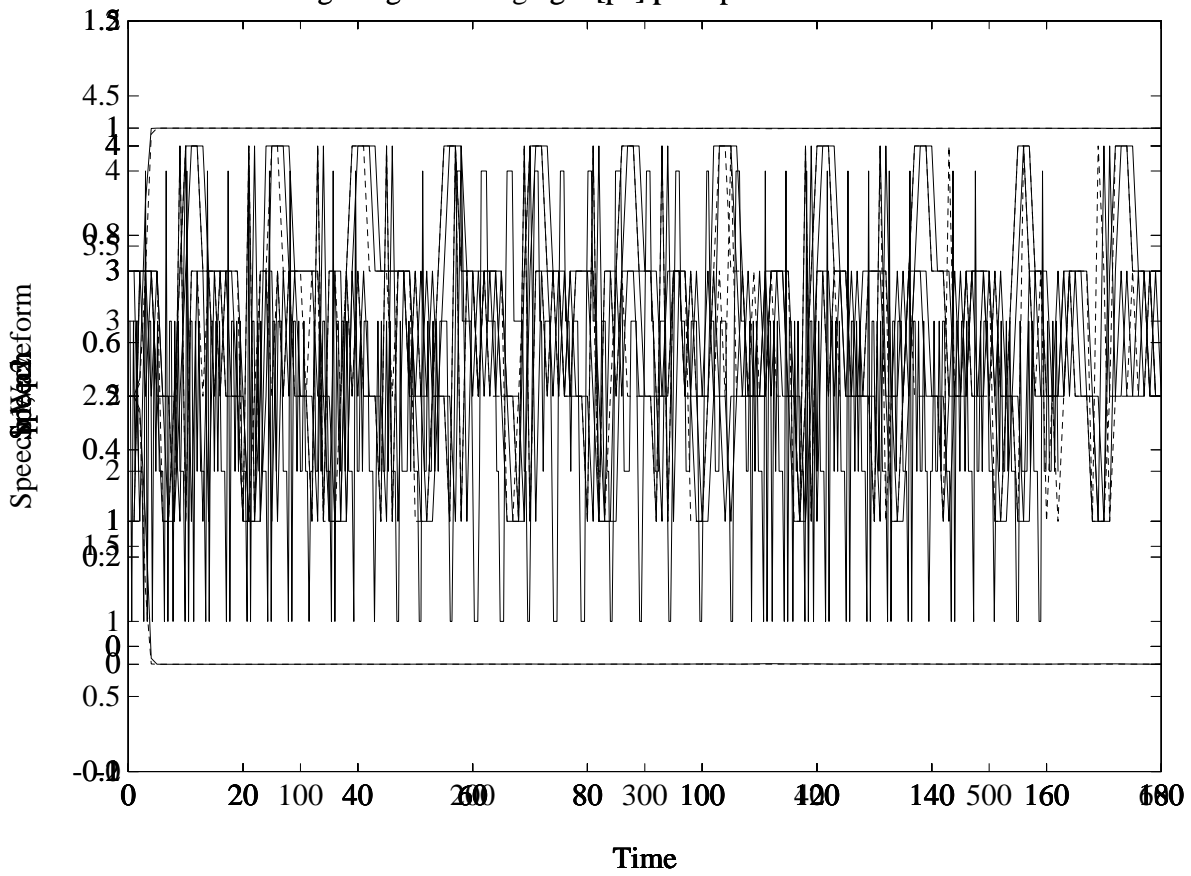


Fig. 5. Fig. 3. Actual High-Speed Signal for Woodchuck and 2



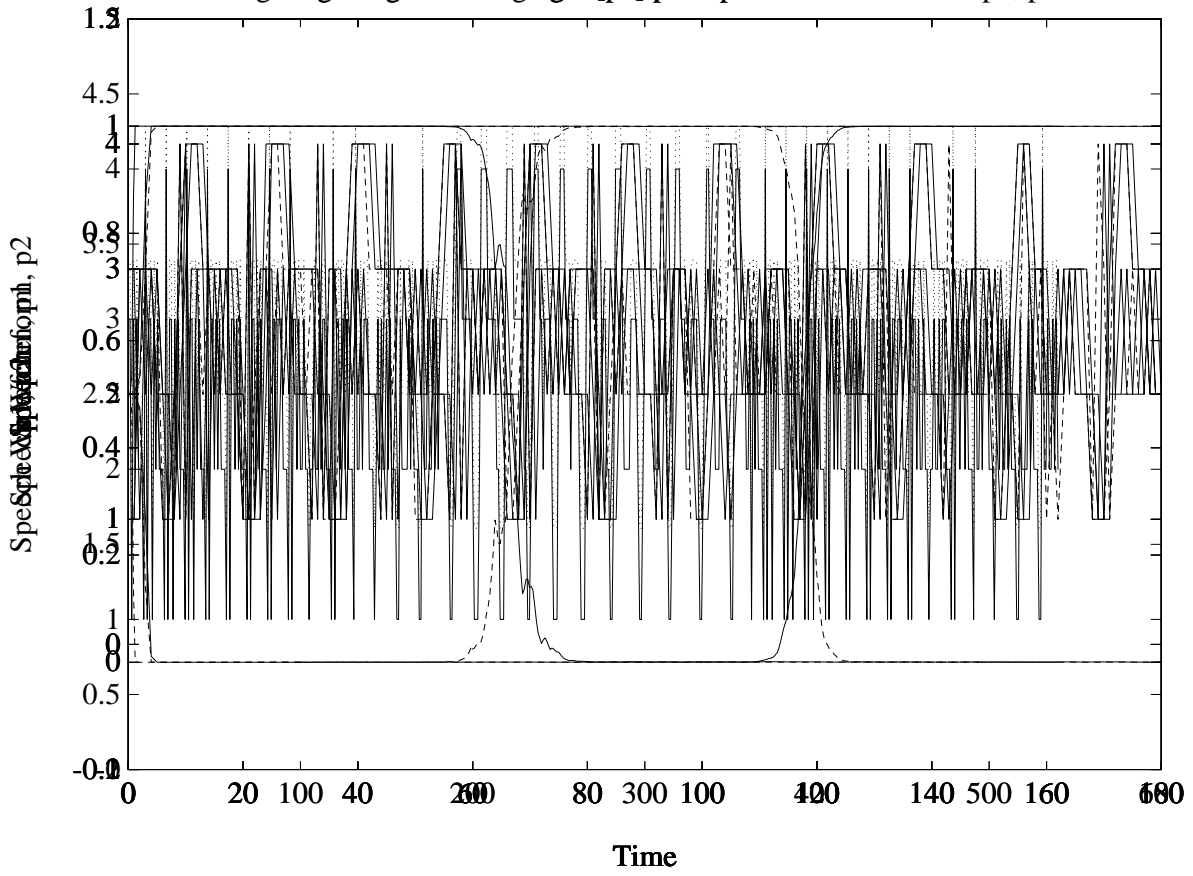
[illegible]

Fig. 8 Fig. 9 Fig. 10 Fig. 11 Fig. 12 Fig. 13 Fig. 14 Fig. 15 Fig. 16 Fig. 17 Fig. 18 Fig. 19 Fig. 20 Fig. 21 Fig. 22 Fig. 23 Fig. 24 Fig. 25 Fig. 26 Fig. 27 Fig. 28 Fig. 29 Fig. 30 Fig. 31 Fig. 32 Fig. 33 Fig. 34 Fig. 35 Fig. 36 Fig. 37 Fig. 38 Fig. 39 Fig. 40 Fig. 41 Fig. 42 Fig. 43 Fig. 44 Fig. 45 Fig. 46 Fig. 47 Fig. 48 Fig. 49 Fig. 50 Fig. 51 Fig. 52 Fig. 53 Fig. 54 Fig. 55 Fig. 56 Fig. 57 Fig. 58 Fig. 59 Fig. 60 Fig. 61 Fig. 62 Fig. 63 Fig. 64 Fig. 65 Fig. 66 Fig. 67 Fig. 68 Fig. 69 Fig. 70 Fig. 71 Fig. 72 Fig. 73 Fig. 74 Fig. 75 Fig. 76 Fig. 77 Fig. 78 Fig. 79 Fig. 80 Fig. 81 Fig. 82 Fig. 83 Fig. 84 Fig. 85 Fig. 86 Fig. 87 Fig. 88 Fig. 89 Fig. 90 Fig. 91 Fig. 92 Fig. 93 Fig. 94 Fig. 95 Fig. 96 Fig. 97 Fig. 98 Fig. 99 Fig. 100 Fig. 101 Fig. 102 Fig. 103 Fig. 104 Fig. 105 Fig. 106 Fig. 107 Fig. 108 Fig. 109 Fig. 110 Fig. 111 Fig. 112 Fig. 113 Fig. 114 Fig. 115 Fig. 116 Fig. 117 Fig. 118 Fig. 119 Fig. 120 Fig. 121 Fig. 122 Fig. 123 Fig. 124 Fig. 125 Fig. 126 Fig. 127 Fig. 128 Fig. 129 Fig. 130 Fig. 131 Fig. 132 Fig. 133 Fig. 134 Fig. 135 Fig. 136 Fig. 137 Fig. 138 Fig. 139 Fig. 140 Fig. 141 Fig. 142 Fig. 143 Fig. 144 Fig. 145 Fig. 146 Fig. 147 Fig. 148 Fig. 149 Fig. 150 Fig. 151 Fig. 152 Fig. 153 Fig. 154 Fig. 155 Fig. 156 Fig. 157 Fig. 158 Fig. 159 Fig. 160 Fig. 161 Fig. 162 Fig. 163 Fig. 164 Fig. 165 Fig. 166 Fig. 167 Fig. 168 Fig. 169 Fig. 170 Fig. 171 Fig. 172 Fig. 173 Fig. 174 Fig. 175 Fig. 176 Fig. 177 Fig. 178 Fig. 179 Fig. 180 Fig. 181 Fig. 182 Fig. 183 Fig. 184 Fig. 185 Fig. 186 Fig. 187 Fig. 188 Fig. 189 Fig. 190 Fig. 191 Fig. 192 Fig. 193 Fig. 194 Fig. 195 Fig. 196 Fig. 197 Fig. 198 Fig. 199 Fig. 200 Fig. 201 Fig. 202 Fig. 203 Fig. 204 Fig. 205 Fig. 206 Fig. 207 Fig. 208 Fig. 209 Fig. 210 Fig. 211 Fig. 212 Fig. 213 Fig. 214 Fig. 215 Fig. 216 Fig. 217 Fig. 218 Fig. 219 Fig. 220 Fig. 221 Fig. 222 Fig. 223 Fig. 224 Fig. 225 Fig. 226 Fig. 227 Fig. 228 Fig. 229 Fig. 230 Fig. 231 Fig. 232 Fig. 233 Fig. 234 Fig. 235 Fig. 236 Fig. 237 Fig. 238 Fig. 239 Fig. 240 Fig. 241 Fig. 242 Fig. 243 Fig. 244 Fig. 245 Fig. 246 Fig. 247 Fig. 248 Fig. 249 Fig. 250 Fig. 251 Fig. 252 Fig. 253 Fig. 254 Fig. 255 Fig. 256 Fig. 257 Fig. 258 Fig. 259 Fig. 260 Fig. 261 Fig. 262 Fig. 263 Fig. 264 Fig. 265 Fig. 266 Fig. 267 Fig. 268 Fig. 269 Fig. 270 Fig. 271 Fig. 272 Fig. 273 Fig. 274 Fig. 275 Fig. 276 Fig. 277 Fig. 278 Fig. 279 Fig. 280 Fig. 281 Fig. 282 Fig. 283 Fig. 284 Fig. 285 Fig. 286 Fig. 287 Fig. 288 Fig. 289 Fig. 290 Fig. 291 Fig. 292 Fig. 293 Fig. 294 Fig. 295 Fig. 296 Fig. 297 Fig. 298 Fig. 299 Fig. 300 Fig. 301 Fig. 302 Fig. 303 Fig. 304 Fig. 305 Fig. 306 Fig. 307 Fig. 308 Fig. 309 Fig. 310 Fig. 311 Fig. 312 Fig. 313 Fig. 314 Fig. 315 Fig. 316 Fig. 317 Fig. 318 Fig. 319 Fig. 320 Fig. 321 Fig. 322 Fig. 323 Fig. 324 Fig. 325 Fig. 326 Fig. 327 Fig. 328 Fig. 329 Fig. 330 Fig. 331 Fig. 332 Fig. 333 Fig. 334 Fig. 335 Fig. 336 Fig. 337 Fig. 338 Fig. 339 Fig. 340 Fig. 341 Fig. 342 Fig. 343 Fig. 344 Fig. 345 Fig. 346 Fig. 347 Fig. 348 Fig. 349 Fig. 350 Fig. 351 Fig. 352 Fig. 353 Fig. 354 Fig. 355 Fig. 356 Fig. 357 Fig. 358 Fig. 359 Fig. 360 Fig. 361 Fig. 362 Fig. 363 Fig. 364 Fig. 365 Fig. 366 Fig. 367 Fig. 368 Fig. 369 Fig. 370 Fig. 371 Fig. 372 Fig. 373 Fig. 374 Fig. 375 Fig. 376 Fig. 377 Fig. 378 Fig. 379 Fig. 380 Fig. 381 Fig. 382 Fig. 383 Fig. 384 Fig. 385 Fig. 386 Fig. 387 Fig. 388 Fig. 389 Fig. 390 Fig. 391 Fig. 392 Fig. 393 Fig. 394 Fig. 395 Fig. 396 Fig. 397 Fig. 398 Fig. 399 Fig. 400 Fig. 401 Fig. 402 Fig. 403 Fig. 404 Fig. 405 Fig. 406 Fig. 407 Fig. 408 Fig. 409 Fig. 410 Fig. 411 Fig. 412 Fig. 413 Fig. 414 Fig. 415 Fig. 416 Fig. 417 Fig. 418 Fig. 419 Fig. 420 Fig. 421 Fig. 422 Fig. 423 Fig. 424 Fig. 425 Fig. 426 Fig. 427 Fig. 428 Fig. 429 Fig. 430 Fig. 431 Fig. 432 Fig. 433 Fig. 434 Fig. 435 Fig. 436 Fig. 437 Fig. 438 Fig. 439 Fig. 440 Fig. 441 Fig. 442 Fig. 443 Fig. 444 Fig. 445 Fig. 446 Fig. 447 Fig. 448 Fig. 449 Fig. 450 Fig. 451 Fig. 452 Fig. 453 Fig. 454 Fig. 455 Fig. 456 Fig. 457 Fig. 458 Fig. 459 Fig. 460 Fig. 461 Fig. 462 Fig. 463 Fig. 464 Fig. 465 Fig. 466 Fig. 467 Fig. 468 Fig. 469 Fig. 470 Fig. 471 Fig. 472 Fig. 473 Fig. 474 Fig. 475 Fig. 476 Fig. 477 Fig. 478 Fig. 479 Fig. 480 Fig. 481 Fig. 482 Fig. 483 Fig. 484 Fig. 485 Fig. 486 Fig. 487 Fig. 488 Fig. 489 Fig. 490 Fig. 491 Fig. 492 Fig. 493 Fig. 494 Fig. 495 Fig. 496 Fig. 497 Fig. 498 Fig. 499 Fig. 500 Fig. 501 Fig. 502 Fig. 503 Fig. 504 Fig. 505 Fig. 506 Fig. 507 Fig. 508 Fig. 509 Fig. 510 Fig. 511 Fig. 512 Fig. 513 Fig. 514 Fig. 515 Fig. 516 Fig. 517 Fig. 518 Fig. 519 Fig. 520 Fig. 521 Fig. 522 Fig. 523 Fig. 524 Fig. 525 Fig. 526 Fig. 527 Fig. 528 Fig. 529 Fig. 530 Fig. 531 Fig. 532 Fig. 533 Fig. 534 Fig. 535 Fig. 536 Fig. 537 Fig. 538 Fig. 539 Fig. 540 Fig. 541 Fig. 542 Fig. 543 Fig. 544 Fig. 545 Fig. 546 Fig. 547 Fig. 548 Fig. 549 Fig. 550 Fig. 551 Fig. 552 Fig. 553 Fig. 554 Fig. 555 Fig. 556 Fig. 557 Fig. 558 Fig. 559 Fig. 560 Fig. 561 Fig. 562 Fig. 563 Fig. 564 Fig. 565 Fig. 566 Fig. 567 Fig. 568 Fig. 569 Fig. 570 Fig. 571 Fig. 572 Fig. 573 Fig. 574 Fig. 575 Fig. 576 Fig. 577 Fig. 578 Fig. 579 Fig. 580 Fig. 581 Fig. 582 Fig. 583 Fig. 584 Fig. 585 Fig. 586 Fig. 587 Fig. 588 Fig. 589 Fig. 590 Fig. 591 Fig. 592 Fig. 593 Fig. 594 Fig. 595 Fig. 596 Fig. 597 Fig. 598 Fig. 599 Fig. 600 Fig. 601 Fig. 602 Fig. 603 Fig. 604 Fig. 605 Fig. 606 Fig. 607 Fig. 608 Fig. 609 Fig. 610 Fig. 611 Fig. 612 Fig. 613 Fig. 614 Fig. 615 Fig. 616 Fig. 617 Fig. 618 Fig. 619 Fig. 620 Fig. 621 Fig. 622 Fig. 623 Fig. 624 Fig. 625 Fig. 626 Fig. 627 Fig. 628 Fig. 629 Fig. 630 Fig. 631 Fig. 632 Fig. 633 Fig. 634 Fig. 635 Fig. 636 Fig. 637 Fig. 638 Fig. 639 Fig. 640 Fig. 641 Fig. 642 Fig. 643 Fig. 644 Fig. 645 Fig. 646 Fig. 647 Fig. 648 Fig. 649 Fig. 650 Fig. 651 Fig. 652 Fig. 653 Fig. 654 Fig. 655 Fig. 656 Fig. 657 Fig. 658 Fig. 659 Fig. 660 Fig. 661 Fig. 662 Fig. 663 Fig. 664 Fig. 665 Fig. 666 Fig. 667 Fig. 668 Fig. 669 Fig. 670 Fig. 671 Fig. 672 Fig. 673 Fig. 674 Fig. 675 Fig. 676 Fig. 677 Fig. 678 Fig. 679 Fig. 680 Fig. 681 Fig. 682 Fig. 683 Fig. 684 Fig. 685 Fig. 686 Fig. 687 Fig. 688 Fig. 689 Fig. 690 Fig. 691 Fig. 692 Fig. 693 Fig. 694 Fig. 695 Fig. 696 Fig. 697 Fig. 698 Fig. 699 Fig. 700 Fig. 701 Fig. 702 Fig. 703 Fig. 704 Fig. 705 Fig. 706 Fig. 707 Fig. 708 Fig. 709 Fig. 710 Fig. 711 Fig. 712 Fig. 713 Fig. 714 Fig. 715 Fig. 716 Fig. 717 Fig. 718 Fig. 719 Fig. 720 Fig. 721 Fig. 722 Fig. 723 Fig. 724 Fig. 725 Fig. 726 Fig. 727 Fig. 728 Fig. 729 Fig. 730 Fig. 731 Fig. 732 Fig. 733 Fig. 734 Fig. 735 Fig. 736 Fig. 737 Fig. 738 Fig. 739 Fig. 740 Fig. 741 Fig. 742 Fig. 743 Fig. 744 Fig. 745 Fig. 746 Fig. 747 Fig. 748 Fig. 749 Fig. 750 Fig. 751 Fig. 752 Fig. 753 Fig. 754 Fig. 755 Fig. 756 Fig. 757 Fig. 758 Fig. 759 Fig. 760 Fig. 761 Fig. 762 Fig. 763 Fig. 764 Fig. 765 Fig. 766 Fig. 767 Fig. 768 Fig. 769 Fig. 770 Fig. 771 Fig. 772 Fig. 773 Fig. 774 Fig. 775 Fig. 776 Fig. 777 Fig. 778 Fig. 779 Fig. 780 Fig. 781 Fig. 782 Fig. 783 Fig. 784 Fig. 785 Fig. 786 Fig. 787 Fig. 788 Fig. 789 Fig. 790 Fig. 791 Fig. 792 Fig. 793 Fig. 794 Fig. 795 Fig. 796 Fig. 797 Fig. 798 Fig. 799 Fig. 800 Fig. 801 Fig. 802 Fig. 803 Fig. 804 Fig. 805 Fig. 806 Fig. 807 Fig. 808 Fig. 809 Fig. 810 Fig. 811 Fig. 812 Fig. 813 Fig. 814 Fig. 815 Fig. 816 Fig. 817 Fig. 818 Fig. 819 Fig. 820 Fig. 821 Fig. 822 Fig. 823 Fig. 824 Fig. 825 Fig. 826 Fig. 827 Fig. 828 Fig. 829 Fig. 830 Fig. 831 Fig. 832 Fig. 833 Fig. 834 Fig. 835 Fig. 836 Fig. 837 Fig. 838 Fig. 839 Fig. 840 Fig. 841 Fig. 842 Fig. 843 Fig. 844 Fig. 845 Fig. 846 Fig. 847 Fig. 848 Fig. 849 Fig. 850 Fig. 851 Fig. 852 Fig. 853 Fig. 854 Fig. 855 Fig. 856 Fig. 857 Fig. 858 Fig. 859 Fig. 860 Fig. 861 Fig. 862 Fig. 863 Fig. 864 Fig. 865 Fig. 866 Fig. 867 Fig. 868 Fig. 869 Fig. 870 Fig. 871 Fig. 872 Fig. 873 Fig. 874 Fig. 875 Fig. 876 Fig. 877 Fig. 878 Fig. 879 Fig. 880 Fig. 881 Fig. 882 Fig. 883 Fig. 884 Fig. 885 Fig. 886 Fig. 887 Fig. 888 Fig. 889 Fig. 890 Fig. 891 Fig. 892 Fig. 893 Fig. 894 Fig. 895 Fig. 896 Fig. 897 Fig. 898 Fig. 899 Fig. 900 Fig. 901 Fig. 902 Fig. 903 Fig. 904 Fig. 905 Fig. 906 Fig. 907 Fig. 908 Fig. 909 Fig. 910 Fig. 911 Fig. 912 Fig. 913 Fig. 914 Fig. 915 Fig. 916 Fig. 917 Fig. 918 Fig. 919 Fig. 920 Fig. 921 Fig. 922 Fig. 923 Fig. 924 Fig. 925 Fig. 926 Fig. 927 Fig. 928 Fig. 929 Fig. 930 Fig. 931 Fig. 932 Fig. 933 Fig. 934 Fig. 935 Fig. 936 Fig. 937 Fig. 938 Fig. 939 Fig. 940 Fig. 941 Fig. 942 Fig. 943 Fig. 944 Fig. 945 Fig. 946 Fig. 947 Fig. 948 Fig. 949 Fig. 950 Fig. 951 Fig. 952 Fig. 953 Fig. 954 Fig. 955 Fig. 956 Fig. 957 Fig. 958 Fig. 959 Fig. 960 Fig. 961 Fig. 962 Fig. 963 Fig. 964 Fig. 965 Fig. 966 Fig. 967 Fig. 968 Fig. 969 Fig. 970 Fig. 971 Fig. 972 Fig. 973 Fig. 974 Fig. 975 Fig. 976 Fig. 977 Fig. 978 Fig. 979 Fig. 980 Fig. 981 Fig. 982 Fig. 983 Fig. 984 Fig. 985 Fig. 986 Fig. 987 Fig. 988 Fig. 989 Fig. 990 Fig. 991 Fig. 992 Fig. 993 Fig. 994 Fig. 995 Fig. 996 Fig. 997 Fig. 998 Fig. 999 Fig. 1000

