The Local Backward-Forward Algorithm

A. Kehagias
Division of Applied Mathematics
Brown University
Providence, RI 02912

E-Mail Address: st401843@brownvm.bitnet

Abstract

We introduce *Stochastic Recurrent Networks* which are collections of interconnected finite state units. Each unit goes into a new state at every discrete time step following a probability law that is conditional on the state of neighboring units at the previous time step. A network of this type can be trained to *learn* a stochastic process, where "training" means maximizing the probability Likelihood function of the model. A new training (i.e. Likelihood maximization) algorithm is introduced, the *Local Backward-Forward Algorithm.* The new algorithm is based on the fast Backward-Forward Algorithm of Hidden Markov Models training and improves speed of learning (as compared to Back Propagation) substantially. Essentially, the local Backward-Forward Algorithm is a version of Baum's algorithm which estimates local transition probabilities rather than the global transition probability matrix.

0 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, $Prob(X^{t+1} = x^1, ..., X^{t+\tau} = x^\tau; \mathcal{P})$. 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 $x_S \in A^{|S|}$. Finally, $Bin(m, n)$, where m is an integer, means the n-th digit of integer m written in binary notation.

1 Introduction

We develop a network that combines probabilistic and dynamic behavior. In our framework, deterministic and/or static behavior appear as special cases.

We will give a precise definition of SRN in the next section; informally, the subject of our inquiry is collections of interconnected finite state units that change states synchronously, according to a probabilistic mechanism. For any particular unit, the probabilistic change of state depends on the previous state of this unit and its **parents**. We only deal with finite state units; the theory for units with continuous valued states is exactly analogous but mathematically more involved and will be developed elsewhere. Here we define precisely the SRN model, define the learning task for stochastic processes as a Likelihood Maximization problem and, finally, derive the **Local Backward Forward Algorithm**, a very efficient training algorithm that is based on the Baum Backward Forward algorithm [BE67] used in Hidden Markov Modelling and Speech Recognition [Rab88]. We also present a training example using the 8-3-8 encoder problem.

2 The SRN Model

Consider a network of interconnected units. For simplicity of presentation assume each unit is binary, that is, it can be either on or off. When the unit is off its state is 0; when the unit is on its state is 1. Following standard connectionist practice, the units are separated in three layers: input, hidden and output. The state of the network can be fully described by three vectors of 0's and 1's: one vector for each layer, one vector component for each unit. At a given time $t-1$ every unit of the network is in some state. At time t a new *epoch* starts, during which the units update their state as follows.

1. First the input units turn on or off with a probability that is independent of the other network units and completely determined by the external environment.

2. Then each of the hidden units receives as input the states of its "parents" (which can be input or hidden units, including that same unit, but NOT output units). Depending on the configuration of its parents, there is a certain probability that each hidden unit will turn on or off.

3. Finally, each of the output units receives as input the states of its parents (which can be input or hidden units, but NOT output units) and turns on or off with a configuration-dependent probability.

Let us now formulate this mechanism in a mathematically precise manner. Call U_t the M_i-long vector of input units states at time t, X_t the M_h-long vector of hidden units states at time t, Y_t the M_o-long vector of output units states at time t. The components of these vectors all come from the same finite set $A = \{0, 1, ..., K-1\}$. The sequences $U_t, t = 1, ...,$ $X_t, t = 1, ...$ and $Y_t, t = 1, ...$ are the input, hidden and output stochastic processes, respectively.

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 the **probability functions** $p_X(x_1, ..., x_m)$, $p_Y(y_1, ..., y_m)$ for all m, x_1, ..., x_m, y_1, ..., y_m (to be defined presently).

The directed graph $\mathcal{G}$ is itself a pair $\mathcal{G} = (S, \mathcal{N})$, where $S = \{s_1, ..., s_L\}$ 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 $|S_i| = M_i$, $|S_h| = M_h$, $|S_o| = M_o$.

If s reads the state of r before changing state then there is a directed edge from unit r to unit s ($r, s \in S$). 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:

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

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

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

The probabilistic state update mechanism is described by $\mathcal{P}$, which is the set of **local conditional probabilities**. As already described, the state update takes place synchronously and locally for every unit:

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

$$\prod_{s \in S_h} Prob(X^t_s = x^0_s | X^{-1}_{N_h(s)} = x^{-1}_{N_h(s)}, U^t_{N_i(s)} = u^0_{N_i(s)}),$$

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

$$\prod_{s\in S_o} Prob(Y_s^t = y_s^0 | X^t_{N_h(s)} = x^0_{N_h(s)}, U^t_{N_i(s)} = u^0_{N_i(s)}).$$

We define the local conditional probabilities for all $s \in S_h \cup S_i$, $a \in A$, $b \in A^{|N_h(s)|}$, $c \in A^{|N_i(s)|}$

$$p_s(a|b,c) \doteq Prob(X_s^t = a | X^{t-1}_{N_h(s)} = b, U^t_{N_i(s)} = c)$$

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

$$Prob(X^t = x^0 | X^{t-1} = x^1, X^{t-2} = x^2, ..., U^t = u^0, U^{t-1} = u^1, ...) = \prod_{s\in S_h} p(x_s^0 | x^1_{N_h(s)}, u^0_{N_i(s)}).$$

Similarly we can compute

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

$$\prod_{s\in S_o} p(y_s^0 | x^0_{N_h(s)}, u^0_{N_i(s)}).$$

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

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

Now suppose $((S,\mathcal{N}),\mathcal{P})$, p_U, $Prob(X_0)$ are known. We will first compute $p_X(x^1...x^m)$. We have

$$p_X(x^1...x^m) \doteq Prob(X^1...X^m = x^1...x^m) =$$

$$\sum_{x^0\in A^{M_h}, u^1,...,u^m \in A^{M_i}} Prob(X^0...X^m = x^0...x^m, U^1...U^m = u^1...u^m) =$$

$$\sum_{x^0\in A^{M_h}, u^1,...,u^m \in A^{M_i}} \left(\prod_{t=1}^m \prod_{s\in S_h} p_s(x_s^t | x^{t-1}_{N_h(s)}, u^t_{N_i(s)}) \right) p_U(u^1...u^m) Prob(X_0 = x^0).$$

Similarly we can compute $p_Y(y^1...y^m)$:

$$p_Y(y^1...y^m) \doteq Prob(Y^1...Y^m = y^1...y^m) =$$

$$\sum_{x^0,x^1,...,x^m\in A^{M_h}, u^1,...,u^m \in A^{M_i}} Prob(Y^1..Y^m = y^1..y^m, X^0..X^m = x^0..x^m, U^1..U^m = u^1..u^m) =$$

$$\sum_{x^0,x^1,...,x^m, u^1,...,u^m} \left(\prod_{t=1}^m \prod_{s\in S_h\cup S_o} p_s(x_s^t | x^{t-1}_{N_h(s)}, u^t_{N_i(s)}) \right) p_U(u^1...u^m) Prob(X^0 = x^0).$$

This completes the computation of $p_X(y^1...y^m)$ and $p_Y(y^1...y^m)$. This computation can be done for any m, $y^1, ..., y^m$, hence we have shown that $(S,\mathcal{N})$ and p_U and $Prob(X_0)$ are sufficient to determine p_X, p_Y.

3 The Local Backward Forward Algorithm

Recall the Maximum Likelihood Learning problem. We are given an initial condition x^0, input sample $u^1, u^2, ..., u^T$ and output sample $y^1, y^2, ..., y^T$. We are also given a fixed network topology $\mathcal{G} = (S,\mathcal{N})$. Now we want to select a set of local conditionals $\mathcal{P}$ such that the Likelihood function $L(\mathcal{P})$ is maximized; where $L(\mathcal{P})$ is defined to be:

$$L(\mathcal{P}) \doteq Prob(Y^1 = y^1, ..., Y^T = y^T | X^0 = x^0, U^1 = u^1, ..., U^T = u^T; (\mathcal{G},\mathcal{P})).$$

We will now develop the local BF algorithm and show that it solves the ML learning problem. First define some useful quantities:

$$\Psi_{\mathcal{P}}(x^1,...,x^T) = Prob(Y^1 = y^1, ..., Y^T = y^T, X^1 = x^1, ..., X^T = x^T | X^0 = x^0, U^1 = u^1, ..., U^T = u^T; (\mathcal{G},\mathcal{P})),$$

$$\Phi(\mathcal{P},\mathcal{Q}) = \sum_{x^1,...,x^T \in A^{M_h}} \Psi_{\mathcal{P}}(x^1,...,x^T) \log \Psi_{\mathcal{Q}}(x^1,...,x^T).$$

Now we will prove the following theorem:

Theorem 1 *(Baum's Theorem) Suppose $\Phi(\mathcal{P},\mathcal{Q}) \geq \Phi(\mathcal{P},\mathcal{P})$. Then $L(\mathcal{P}) \geq L(\mathcal{Q})$.*

Proof: In [BE67]. •

Now choose $\mathcal{P}_0$ arbitrarily, then maximize $\Phi(\mathcal{P}_0,\mathcal{P})$ with respect to $\mathcal{P}$; call the maximizer $\mathcal{P}_1$. Obviously $\Phi(\mathcal{P}_0,\mathcal{P}_1) \geq \Phi(\mathcal{P}_0,\mathcal{P}_0)$, hence also $L(\mathcal{P}_1) \geq L(\mathcal{P}_0)$. Now maximize $\Phi(\mathcal{P}_1,\mathcal{P})$ with respect to $\mathcal{P}$; call maximizer $\mathcal{P}_2$. Obviously $\Phi(\mathcal{P}_1,\mathcal{P}_2) \geq \Phi(\mathcal{P}_1,\mathcal{P}_1)$, hence also $L(\mathcal{P}_2) \geq L(\mathcal{P}_1) \geq L(\mathcal{P}_0)$. Proceeding in this manner we get a sequence $\mathcal{P}_0, \mathcal{P}_1, ...$ such that

$$L(\mathcal{P}_0) \leq L(\mathcal{P}_1) \leq L(\mathcal{P}_2) \leq ....$$

This is an iterative, greedy algorithm: at every iteration the value of the Likelihood is increased. It has been proven that this procedure guarantees convergence to a local maximum. Convergence to the global maximum is not guaranteed. In all these respects the algorithm is similar to a steepest ascent procedure. However there is one important difference: we can explicitly compute the maximizer at every step (as we will show presently) and hence are not confined to a small step.

We want to minimize the function $\Phi(\mathcal{P},\mathcal{Q})$ with respect to $\mathcal{Q}$, where $\mathcal{P} = \{p_s, s \in S_h \cup S_o\}$, $\mathcal{Q} = \{q_s, s \in S_h \cup S_o\}$ are two sets of local conditionals. So we want to minimize $\Phi(\mathcal{P},\mathcal{Q})$ with respect to $q_s(a|b,c)$, $s \in S$, $a \in A$, $b \in A^{|N_h(s)|}$, $c \in A^{|N_i(s)|}$. These are positive real variables in the range $[0,1]$; however they are not independent. They must satisfy:

$$\sum_{a \in A} q_s(a|b,c) = 1 \qquad \forall s \in S, b \in A^{|N_h(s)|}, c \in A^{|N_o(s)|}. \tag{1}$$

We incorporate these constraints to the Likelihood function by using Lagrange multipliers $\mu_s(b,c)$, $s \in S, b \in A^{|N_h(s)|} c \in A^{|N_i(s)|}$:

$$\Phi^*(\mathcal{P},\mathcal{Q}) = \Phi(\mathcal{P},\mathcal{Q}) + \sum_s \sum_b \mu_s(b,c) \sum_a q_s(a|b,c).$$

Maximization of $\Phi(\mathcal{P},\mathcal{Q})$ under the constraints (1) is equivalent to maximization of $\Phi^*(\mathcal{P},\mathcal{Q})$ without constraints. To maximize Φ^* we apply the usual condition that the partial derivatives with respect to the q's equal zero.

$$\frac{\partial \Phi^*}{\partial q_s(a \mid b,c)} = \mu_s(a \mid b,c) +$$

$$\frac{\sum_t \sum_x P_{\mathcal{P}}(Y^1..Y^T = y^1..y^T, X^1..X^T = x^1..x^T \mid U^1..U^T = u^1..u^T, X^0 = x^0) \mathbf{1}_{a,bc}(x_s^t, x_{N_h(s)}^{t-1}, u_{N_i(s)}^t)}{q_s(a \mid b,c)}. \tag{2}$$

Here the function $\mathbf{1}_{a,bc}(x,z,u)$ equals 1 when $x=a$, $z=b$ and $u=c$; it is zero otherwise. Setting (2) equal to 0 we obtain that

$$q_s(a \mid b,c) = \frac{\sum_{t:u_{N_i(s)}^t = c} Prob(Y^1..Y^T = y^1..y^T, X_s^t = a, X_{N_h(s)}^{t-1} = b \mid U^1..U^T = u^1..u^T, X^0 = x^0)}{\sum_{t:u_{N_i(s)}^t = c} Prob(Y^1..Y^T = y^1..y^T, X_{N_h(s)}^{t-1} = b \mid U^1..U^T = u^1..u^T, X^0 = x^0)}$$

This suggests the following reestimation iteration (for m=1,2,...):

$$p_s^{m+1}(a \mid bc) = \frac{\sum_{t:u_{N_i(s)}^t = c} Prob(Y^1..Y^T = y^1.y^T, X_s^t = a, X^{t-1} N_h(s) = b \mid U^1..U^T = u^1.u^T, X^0 = x^0; \mathcal{P}^m)}{\sum_{t:u_{N_i(s)}^t = c} Prob(Y^1..Y^T = y^1..y^T, X^{t-1} N_h(s) = b \mid U^1..U^T = u^1..u^T, X^0 = x^0; \mathcal{P}^m)} \tag{3}$$

We can compute the terms in the fraction entirely in terms of the p's. To do so, assume $U^1 = u^1, ..., U^T = u^T$, $Y^1 = y^1, ..., Y^T = y^T$, $X^0 = x^0$ are fixed and define some auxiliary quantities. The **transition matrix** is defined for all $z, x \in A^{|S_h|}$, for $t = 1, 2, ..., T$:

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

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

$$P_{zx}^m(t) = \prod_{s \in S_h} p_s^m(x_s \mid z_{N_h(s)}, u_{N_i(s)}^t). \tag{4}$$

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

$$Q^m_{xy}(t) \doteq Prob(Y^t = y \mid X^t = x, U^t = u^t; \mathcal{P}^m).$$

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

$$Q^m_{x,y}(t) = \prod_{s \in S_o} p_s(y_s \mid x_{N_h(s)}, u^t_{N_i(s)}). \tag{5}$$

Next we define the *forward* and *backward probabilities* for all $x \in A^{|S_h|}$, $t = 1, ..., T$

$$\alpha^m_t(x) \doteq Prob(Y^1...Y^t = y^1...y^t, X^t = x | U^1..U^T = u^1..u^T, X^0 = x^0; \mathcal{P}^m),$$

$$\beta^m_t(x) \doteq Prob(Y^{t+1} = y^{t+1}...Y^T = y^T | X^t = x, U^1..U^T = u^1..u^T, X^0 = x^0; \mathcal{P}^m).$$

The α's are called *forward probabilities* and the β's *backward probabilities.* The forward probabilities obey the forward evolution equation for all $x \in A^{|S_h|}$, $t = 0, ..., T-1$:

$$\alpha^m_{t+1}(x) = \sum_{z \in A^{M_h}} \alpha^m_t(z) P^m_{zx}(t) Q^m_{xy^{t+1}} \tag{6}$$

with initial condition $\alpha^m_0(x^0) = 1$, $\alpha^m_1(x) = 0$ for all $x \neq x^0$. The backward probabilities satisfy the backward evolution equation: for all $x \in A^{|S_h|}$, $t = 1, ..., T-1$:

$$\beta^m_t(x) = \sum_z P^m_{xz}(t) Q^m_{zy^{t+1}}(t+1) \beta^m_{t+1}(z) \tag{7}$$

with final condition $\beta^T_m(x) = 1$, $\forall x \in A^{M_h}$. Using the forward and backward probabilities we can write the following relationship:

$$Prob(Y^1..Y^T = y^1..y^T, X^t_s = a, X^{t-1}_{N_h(s)} = b | U^1..U^T = u^1..u^T, X^0 = x^0, \mathcal{P}^m) =$$

$$\sum_{t: u^t_{N_i(s)} = c} \sum_{x^t, x^{t+1}: xt+1_s = a, x^t_{N_h(s)} = b} \alpha^m_t(x^t) P^m_{x^t x^{t+1}}(t+1) Q^m_{x^{t+1} y^{t+1}}(t+1) \beta^m_{t+1}(x^{t+1}).$$

Now we can rewrite the fraction in (3) as

$$p^{m+1}_s(a \mid b, c) = \frac{\sum_{t: u^t_{N_i(s)} = c} \sum_{x^t, x^{t+1}: x^{t+1}_s = a, x^t_{N_h(s)} = b} \alpha^m_t(x^t) P^m_{x^t x^{t+1}}(t+1) Q^m_{x^{t+1} y^{t+1}}(t+1) \beta^m_{t+1}(x^{t+1})}{\sum_{t: u^t_{N_i(s)} = c} \sum_{x^t, x^{t+1}: x^t_{N_h(s)} = b} \alpha^m_t(x^t) P^m_{x^t x^{t+1}}(t+1) Q^m_{x^{t+1} y^{t+1}}(t+1) \beta^m_{t+1}(x^{t+1})} \tag{8}$$

This completes the local BF algorithm. It consists of (4), (5), (6), (7), (8).

The local BF algorithm is essentially a version of the BF algorithm that estimates local conditional probabilities, rather than the global transition matrix which is estimated by the Baum BF algorithm [BE67].

4 An Example: 8-3-8 Encoder

In this section we will use the local BF algorithm to solve the 8-3-8 encoder problem and so illustrate the most important features of the algorithm and also to introduce some practical tricks that improve efficiency.

This is a "static" problem. discussed in [A+85] and elsewhere. We have a set of input/output pairs $(u_1, y_1), ..., (u_8, y_8)$ and we want to build a network that takes u_n in and produces y_n as output for $n = 1, ..., 8$. The input/output pairs are the following 8-long binary vectors: $u_1 = y_1 = 10000000$, $u_2 = y_2 = 01000000$, ..., $u_8 = y_8 = 00000001$. The network has three hidden units. Hence the name "8-3-8 encoder". We use the following network topology: 8 binary input units (call them $i1, i2, ..., i8$), 3 binary hidden units (call them $h1, h2, h3$) fully connected to each other and all the input units and 8 binary output units (call them

$o1, o2, ..., o8$) fully connected to the hidden units. We change somewhat the local BF algorithm: we will **not** estimate the local conditionals of the output units, rather we fix them as follows ($n = 1, ..., 8$):

$$p_{on}(1|x_1 x_2 x_3) = \begin{cases} 1 & \text{iff } x_1 + 2x_2 + 4x_3 = n - 1 \\ 0 & \text{else} \end{cases}$$

So we are looking for a set of local conditionals of the hidden units: $\{p_{hn}(x|u_1 u_2 ... u_8),\ n = 1, 2, 3, ..., x, u_1, ..., u_8 \in \{0, 1\}\}$.

We want to teach a static input/output relationship to a recurrent network. We use the following input/output sample sequence: take an input sample $u_1, ..., u_{200}$ and an output sample $y_1, ..., y_{200}$. We take $u_t = y_t = 000$ for $t = 1, ..., 25$, $u_t = y_t = 001$ for $t = 26, ..., 50$ etc.

There is at least one solution:

$$p_{hn}(x|u_1 ... u_8) = \begin{cases} 1 & \text{iff } Bin(m - 1, n) = 1, \text{ for some m=1,2,...,8} \\ 0 & \text{else.} \end{cases}$$

This is not the only solution (the problem is underdetermined) but is, in an obvious sense, the best. The question is whether the local BF will pick up this solution and how fast We initialize the hidden units for completely random output: $p_{hn}(x|u_1 ... u_8) = .5$, $n = 1, 2, ..., 8$, $\forall x, u_1, ..., u_8 \in \{0, 1\}$. After 5 iterations of the BF algorithm (seven minutes on a Sun/4) we arrive to the following solution:

$$p_{hn}(x|u_1 ... u_8) = \begin{cases} 1 - \epsilon & \text{iff } Bin(m - 1, n) = 1, \text{ for some m=1,2,...,8} \\ \epsilon & \text{else,} \end{cases}$$

where ϵ is less than .001 for all $x, u_1, ..., u_8$. So this is an almost perfect solution. Of all the sets of conditional probabilities that solve the learning task, the local BF algorithm picks up the best solution very quickly.

5 Conclusions

We have developed a new SRN model and a Maximum Likelihood training algorithm to go with it. The model can reproduce both static and dynamic behavior. The training algorithm is essentially a local version of the Baum BF algorithm (that is, the local BF estimates local conditional probabilities, whereas the Baum BF algorithm estimates the global conditional state transition matrix).

Remark: This is a very shortened version of a Brown Un. technical report [Keh91]; in particular the author regrets that he is not able to include here a more complete reference list.

References

[A$^+$85] D.H. Ackley et al. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 1985.

[BE67] L.E. Baum and J.A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov Processes. *Ann. of Math. Stat.*, pages 36–363, 1967.

[Keh91] A. Kehagias. Stochastic Recurrent Network Training and the local Backward Forward Algorithm. Technical report, Division of Applied Mathematics, Brown Un., Providence, Rhode Island, January 1991.

[Rab88] L.R. Rabiner. A tutorial on HMM and selected applications in speech recognition. *IEEE Proc.*, 1988.