[12] Q. T. Zhang, K. M. Wong, P. C. Yip, and J. P. Reilly, "Statistical analysis of the performance of information criteria in the detection of the number of signals in array processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 1557–1567, 1989.

A Unified Approach on Fast Training of Feedforward and Recurrent Networks Using EM Algorithm

Sheng Ma and Chuanyi Ji

***Abstract*— In this work, we provide a theoretical framework that unifies the notions of hidden representations and moving targets through the expectation and maximization (EM) algorithm. Based on such a framework, two fast training algorithms can be derived consistently for both feedforward networks and recurrent networks.**

I. Introduction

Although multilayer feedforward and recurrent neural networks have found many important applications in signal processing, training such networks is usually slow due to the nonlinearity and, possibly, the compact structure of these networks. This hinders wide applications of these networks in such important signal processing areas as speech and communication, where the networks need to be adapted rapidly to a nonstationary environment [4].

One of the interesting approaches used to reduce the training time was based on hidden representations or moving targets [3], [6], [9]. Targets were obtained adaptively for either hidden or recurrent units and then used to decompose training an entire (multilayer or recurrent) network into training several single-layer feedforward networks. The approach resulted in fast training algorithms for feedforward networks with hard threshold units [3] but was not effective in training networks with sigmoidal units [6], [9]. In addition, the algorithms were heuristic, which was less desirable.

On the other hand, the expectation-maximization (EM) algorithm [2] in statistics was developed based on so-called hidden (random) variables. Although similar to moving targets, hidden (random) variables were defined through a sound theoretical framework, and the EM algorithms were developed based on a theoretical foundation for maximum likehood estimation (MLE). When applied to training tree-structured networks with the localized modules [5], the EM algorithm reduced the training time significantly.

Motivated by the idea of moving targets and the EM algorithms, we have developed in our previous work two fast training algorithms for feedforward [7] and recurrent networks [8]. These algorithms have demonstrated that EM algorithms can also be applied effectively to layered and recurrent networks to speed up training. However, since the two algorithms have been developed in separate contexts, their inter-relationships have not been carefully investigated.

Manuscript received July 23, 1997; revised January 5, 1998. This work was supported by the National Science Foundation under Grant ECS-9 312 504 and CAREER Grant IRI-9 502 518. The associate editor coordinating the review of this paper and approving it for publication was Prof. Yu-Hen Hu.

The authors are with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA.

Publisher Item Identifier S 1053-587X(98)05244-1.

In this correspondence, we will focus on providing a unified framework, based on the EM algorithm, from which both algorithms can be derived in a similar fashion. We will show that the two algorithms are based on a similar probabilistic formulation on hidden targets. Such a formulation will result in an EM algorithm for feedforward networks that reduces the training a nonlinear network into training individual neurons. By the Markov properties of recurrent networks and the mean-field approximation [11], a recurrent network can be unfolded at each EM step into an equivalent feedforward network. Then, the EM algorithm developed for feedforward networks can be directly applied to improve the training time.

II. Unified Notations

Consider the discrete time recurrent networks with a fully connected hidden layer as shown in Fig. 1. The network has one linear output, d inputs, and L fully connected recurrent (hidden) sigmoid units. Let $\vec{w}_j^{(1)}$ and $\vec{w}_j^{(3)}$ denote the weight vectors connecting the external inputs and the other recurrent units to the jth hidden unit, respectively, for $1 \leq j \leq L$. Let $W^{(1)}$ and $W^{(3)}$ be the weight matrices containing all $\vec{w}_j^{(1)}$ and $\vec{w}_j^{(3)}$, respectively. Let $\vec{w}^{(2)}$ be the weight vector connecting the recurrent hidden units to the output. $\{\vec{x}(n), t(n+1)\}_{n=1}^{N}$ is a set of N training samples, where $\vec{x}(n)$ is a $(d+1) \times 1$ input vector at the nth time unit,[1] and $t(n+1)$ is the corresponding desired output. Let $h_j(n+1)$ be the output of the recurrent hidden unit. Then, $h_j(n+1) = g(\vec{w}_j^{(1)T}\vec{x}(n) + \vec{w}_j^{(3)T}\vec{h}(n))$, where $g(\cdot)$ is a sigmoid function. The output of the network $y(n+1)$ corresponding to an input $\vec{x}(n)$ can be expressed as $y(n+1) = \vec{w}^{(2)T}\vec{h}(n+1)$. The recurrent network is reduced to a feedforward network when the recurrent connections do not exist, i.e., $W^{(3)} = 0$. Therefore, we can use the above notations for both feedforward and recurrent networks.[2]

III. The EM Algorithm

The method we will use to develop our algorithm is based on the EM algorithm for MLE [2]. Let D_I be a set of data observed directly, and let Θ be a set of parameters characterizing the corresponding distribution of the random variables. The maximum likelihood problem is to find an optimal set of parameters Θ^{opt} through maximizing the log likelihood of the data. Since it can be very difficult to maximize the original log likelihood function directly, the EM algorithm has been introduced to simplify this process. A set of hidden (random) variables are introduced. The data for the hidden variables are called missing data. Together with the missing data, the so-called incomplete data D_I form a complete data set D_c. The EM algorithm converts the original maximum likelihood procedure into a two-step process: the E-step (expectation) and the M-step (maximization). In the E-step, the following conditional expectation can be computed (with respect to hidden variables) of the complete data log likelihood given incomplete data D_I and the previous parameters $\Theta^{(p)}$ at the pth step

$$Q(\Theta|\Theta^{(p)}) = E\{\ln\ P(D_c|\Theta)|D_I, \Theta^{(p)}\} \tag{1}$$

where Θ are the parameters to be obtained in the (next) M-step. The notation $Q(\Theta|\Theta^{(p)})$ indicates that the conditional expectation

[1] The bias corresponds to an input $x_0(n) = 1$.

[2] No delays exist for feedforward neurons, i.e., the time $n+1$ reduces to n.

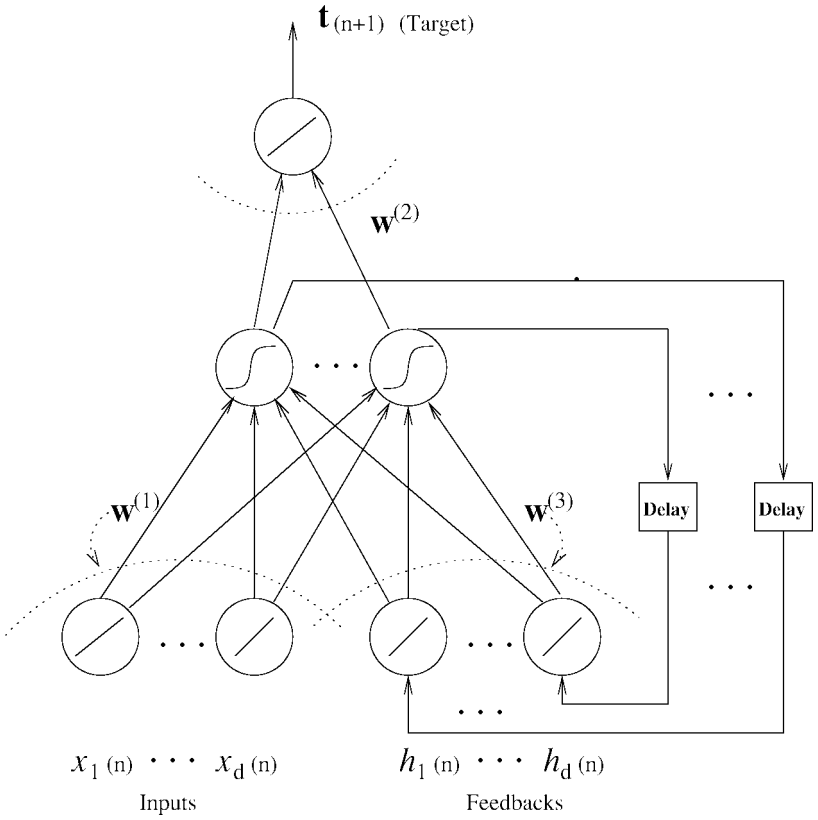

Fig. 1. Recurrent network.

is a function of Θ and that the expectation is evaluated using $\Theta^{(p)}$ as parameters in the probability density function. $P(\cdot)$ represents a probability density function. E stands for the expectation operation. In the M-step, new parameters $\Theta^{(p+1)}$ are found by maximizing the expected log likelihood $Q(\Theta|\Theta^{(p)})$. The algorithm alternates between the E- and M-steps and terminates when a certain convergence criterion is satisfied. Local convergence of the EM algorithm is guaranteed [2]. There is an extended survey on the EM algorithm in [1].

IV. Probabilistic Models

A. Choice of Hidden Variables

For feedforward and recurrent networks, we choose the hidden variables $\vec{z}$ to be either the desired hidden targets at the hidden layer for the feedforward networks or the desired hidden states for recurrent neurons of the recurrent networks. Then, $\vec{z}(n)$ and $\vec{z}(n+1)$ denote the corresponding missing data for the feedforward and recurrent networks, respectively, corresponding to the input $\vec{x}(n)$ (for $1 \leq n \leq N$). The complete data set therefore contains the labeled training samples and the desired hidden target, which is $\{\vec{x}(n), t(n), \vec{z}(n)\}_{n=1}^{N}$ for feedforward networks, and $\{\vec{x}(n), t(n+1), \vec{z}(n+1)\}_{n=1}^{N}$ for recurrent networks. The incomplete data set is simply the training set $\{\vec{x}(n), t(n)\}_{n=1}^{N}$. For simplicity, we use the notation $\{\vec{x}\}, \{y\}$ and $\{\vec{z}\}$ to represent $\{\vec{x}(n)\}_{n=1}^{N}, \{t(n)\}_{n=1}^{N}$ or $\{t(n+1)\}_{n=1}^{N}, \{\vec{z}(n)\}_{n=1}^{N}$ or $\{\vec{z}(n+1)\}_{n=1}^{N}$, respectively, and we use Θ to represent all the weights in a network. We assume that training samples (sequences for recurrent networks) are drawn independently. So are the desired hidden targets.

Using the notations, we have that the conditional expectation of the complete-data log likelihood defined in (1) can then be expressed as $Q(\Theta|\Theta^{(p)}) = \ln P(\{\vec{x}\}|\Theta) + \int P(\{\vec{z}\}|\{t\}, \{\vec{x}\}, \Theta^{(p)}) \ln P(\{\vec{z}\}, \{t\}|\{\vec{x}\}, \Theta)\, d\{\vec{z}\}$.

B. Choice of Probabilistic Models

To evaluate the expected log-likelihood $Q(\Theta|\Theta^{(p)})$, we need to obtain $P(\{\vec{z}\}|\{t\}, \{\vec{x}\}, \Theta)$ and $P(\{\vec{z}\}, \{t\}|\{\vec{x}\}, \Theta)$. In this work, Gaussian distributions are chosen to facilitate the analysis.

1) A Probabilistic Model for Two Layer Feedforward Networks: For two-layer feedforward networks, we choose

$$P(\{\vec{z}\}|\{\vec{x}\}, \Theta) = B_1 \exp(-\lambda_1 E_1) \tag{2}$$

$$P(\{t\}|\{\vec{z}\}, \Theta) = B_2 \exp(-2\lambda_2 E_2) \tag{3}$$

where $E_1 = \Sigma_{n=1}^{N} \|\vec{x}(n) - \vec{h}(n)\|^2$, which measures the error between the desired and the actual outputs of hidden units. $E_2 = \Sigma_{n=1}^{N} (t(n) - \vec{z}(n)^T \cdot \vec{w}^{(2)})^2$, which characterizes the error between the desired and the actual outputs of the network when the desired hidden values are used as inputs to the second layer. B_1 and B_2 are normalization constants. λ_1 and λ_2 are weighting factors. Then, we can obtain [7]

$$P(\{t\}, \{\vec{z}\}|\{\vec{x}\}, \Theta) = A_{tz}^{N} \exp(-\lambda_1 E_1 - \lambda_2 E_2) \tag{4}$$

$$P(\{\vec{z}\}|\{t\}, \{\vec{x}\}, \Theta^{(p)}) = A_z^{N/2} \prod_{n=1}^{N} \exp\left(\tfrac{1}{2}(\vec{z}(n) - \hat{\vec{z}}(n))^T \Sigma^{-1} (\vec{z}(n) - \hat{\vec{z}}(n))\right) \tag{5}$$

where $A_{tz} = \sqrt{(\lambda_2/\pi)(\lambda_1/\pi)^L}$, and $A_z = (1/\sqrt{(2\pi)^L \det(\Sigma)}$. Σ is a $(L \times L)$ covariance matrix whose inverse is $\Sigma^{-1} = 2(\lambda_1 I +$

$\lambda_2 \vec{w}^{(2)^{(p)}} (\vec{w}^{(2)^{(p)}})^T$). The superscript p indicates the corresponding value at the pth step. I is an $(L \times L)$ identity matrix. $\hat{\vec{z}}(n)$ is the expectation of hidden targets $\vec{z}(n)$ given the nth training pairs and the previous parameter set $\Theta^{(p)}$. The jth $(j \le j \le L)$ element of $\hat{\vec{z}}(n)$ can be computed as [7]

$$\hat{z}_j(n) = h_j^{(p)}(n) + \frac{\lambda_2 w_j^{(2)^{(p)}}}{\lambda_1 + \lambda_2 \|\vec{w}^{(2)p}\|^2} e(n) \tag{6}$$

where $h_j^{(p)}(n)$ represents the jth element of $\vec{h}(n)$. $e(n)$ is the training error of the nth training sample satisfying $e(n) = t(n) - y^{(p)}(n)$.

2) A Probabilistic Model for Recurrent Networks: For recurrent networks, we note that the output of the recurrent network and the hidden states at the current time step depend only on the hidden states at the previous time step. This results in the Markov property that leads to the conditional probability density functions [8]

$$\begin{aligned} &P(\{\vec{z}\}|\{t\},\{\vec{x}\},\Theta) \\ &\quad = \prod_{n=1}^{N} P(\vec{z}(n+1)|\vec{z}(n), t(n+1), \vec{x}(n), \Theta) \end{aligned} \tag{7}$$

$$\begin{aligned} &P(\{t\},\{\vec{z}\}|\{\vec{x}\},\Theta) \\ &\quad = \prod_{n=1}^{N} P(t(n+1), \vec{z}(n+1)|\vec{z}(n), \vec{x}(n), \Theta). \end{aligned} \tag{8}$$

Similar Gaussion models to those used for feedforward networks are chosen as the conditional distributions

$$P(\vec{z}(n+1)|\vec{z}(n),\vec{x}(n),\Theta) = B_1 \exp(-\lambda_1 E_1(n+1)) \tag{9}$$

$$P(t(n+1)|\vec{z}(n+1),\Theta) = B_2 \exp(-\lambda_2 E_2(n+1)) \tag{10}$$

where B_1, B_2, λ_1, and λ_2 are the same as for feedforward networks. $E_1(n+1) = \|\vec{z}(n+1) - \vec{h}'(n+1)\|^2$, where the jth element of $\vec{h}'(n+1)$ for $1 \le j \le L$ is $h'_j(n+1) = g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \vec{z}(n)^T \cdot \vec{w}_j^{(3)})$. Therefore, $E_1(n+1)$ measures the error between the desired hidden states $\vec{z}(n+1)$ and $\vec{h}'(n+1)$. Likewise, $E_2(n+1) = (t(n+1) - \vec{z}(n+1)^T \cdot \vec{w}^{(2)})^2$, which characterizes the error between the desired and the actual output of the network when the desired hidden states are used as inputs to the output layer. These Gaussian assumptions lead to

$$\begin{aligned} &P(t(n+1), \vec{z}(n+1)|\vec{z}(n),\vec{x}(n),\Theta) \\ &\quad = A_{tz} \exp(-\lambda_1 E_1(n+1) - \lambda_2 E_2(n+1)) \end{aligned} \tag{11}$$

$$\begin{aligned} &P(\vec{z}(n+1)|t(n+1),\vec{z}(n),\vec{x}(n),\Theta) \\ &\quad = A_z \exp(-\tfrac{1}{2}(\vec{z}(n+1) \\ &\qquad - \hat{\vec{z}}(n+1))^T \sigma^{-1} (\vec{z}(n+1) - \hat{\vec{z}}(n+1))) \end{aligned} \tag{12}$$

where A_{tz}, A_z, Σ, and I are the same as for feedforward networks. $\hat{\vec{z}}(n+1)$ is the conditional expectation of $\vec{z}(n+1)$ conditioned on $t(n+1), \vec{z}(n)$ and $\vec{x}(n)$. The jth element of $\hat{\vec{z}}(n+1)$ can be expressed as

$$\hat{z}_j(n+1) = h'_j(n+1) + \frac{\lambda_2 w_j^{(2)}}{\lambda_1 + \lambda_2 \|\vec{w}^{(2)}\|^2} e(n+1) \tag{13}$$

with $e(n+1) = t(n+1) - \vec{h}'(n+1)^T \cdot \vec{w}^{(2)}$. Then, we can obtain the two distributions $P(\{\vec{z}\}|\{t\},\{\vec{x}\},\Theta)$ and $P(\{t\},\{\vec{z}\}|\{\vec{x}\},\Theta)$ similar to those for feedforward networks.

V. Applying the EM Algorithm to Feedforward and Recurrent Networks

A. The E-Step for Feedforward Networks

For feedforward networks, the expected log likelihood can be evaluated by using the established probability models [7] as

$$\begin{aligned} Q(\Theta|\Theta^{(p)}) &= E\{\ln\ P(\{t\},\{\vec{z}\},\{\vec{x}\}|\Theta)|\{t\},\{\vec{x}\},\Theta^{(p)}\} \\ &= E_c - E_p - E_h - E_o \end{aligned} \tag{14}$$

where E_c is a constant, and $E_p = \lambda_2 N (\vec{w}^{(2)})^T \sigma \vec{w}^{(2)}$. E_h and E_o can be expressed as

$$E_h = \lambda_1 \sum_n \sum_j (\hat{z}_j(n) - g(w_j^T \vec{x}(n)))^2 \tag{15}$$

$$E_o = \lambda_2 \sum_n ((\vec{w}^{(2)})^T \hat{\vec{z}}(n) - t(n))^2. \tag{16}$$

Therefore, the evaluation of the expectation of the log likelihood in the E-step is reduced to finding the expected hidden targets through (6).

B. The E-Step and the Mean-Field Approximation for Recurrent Networks

To evaluate $Q(\Theta|\Theta^p)$ for recurrent networks, we also need to compute the expectation of the hidden targets $\vec{z}(n)$. Since $\vec{z}(n)$ now depends on $\vec{z}(n-1)$, due to the nonlinearity of sigmoidal functions, computing the expectation directly is impossible. Therefore, the self-consistent mean-field approximation [11] is used to approximate the expectation.[3] This approximation replaces the condition on $\vec{z}(n)$ by its expected value

$$\begin{aligned} &P(\vec{z}(n+1)|t(n+1),\vec{z}(n),\vec{x}(n),\Theta^p) \\ &\quad \approx P(\vec{z}(n+1)|t(n+1),\tilde{\vec{z}}(n),\vec{x}(n),\Theta^p) \end{aligned} \tag{17}$$

where the jth element $\tilde{z}_j(n)$ of $\tilde{\vec{z}}(n)$ can be computed recursively through

$$\tilde{h}_j(n+1) = g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \tilde{\vec{z}}(n)^T \cdot \vec{w}_j^{(3)}) \tag{18}$$

$$\tilde{z}_j(n+1) \approx \tilde{h}_j(n+1) + \frac{\lambda_2 w_j^{(2)p}}{\lambda_1 + \lambda_2 \|\vec{w}^{(2)p}\|^2} \tilde{e}(n+1) \tag{19}$$

for every $1 \le j \le L$ and $1 \le n \le N$. $\tilde{e}(n+1) = t(n+1) - \tilde{\vec{h}}(n+1)^T \cdot \vec{w}^{(2)^p} \cdot \tilde{\vec{h}}(n+1)$ is a $L \times 1$ vector whose jth element is defined by (18). It is easy to observe that (19) is strikingly similar to (6). If $\vec{w}_j^{(3)}$'s equal to zero so that the recurrent networks reduce to the feedforward networks, $\tilde{\vec{z}}(n)$ will equal to $\hat{\vec{z}}(n)$ for the feedforward networks.

The mean-field approximation essentially unfolds a recurrent network into a two-layer feedforward network shown in Fig. 2. The weights at the first layer of the unfolded network consist of $W^{(1)}$ and $W^{(3)}$. The total inputs to the two-layer network consist of the original input at time n and the estimated mean value of the previous hidden targets. Then, the rest of the results will follow those for two layer feedforward networks except that

$$E_h = \lambda_1 \sum_n \sum_j (\tilde{z}_j(n) - g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \tilde{\vec{z}}(n)^T \cdot \vec{w}_j^{(3)}))^2 \tag{20}$$

$$E_o = \lambda_2 \sum_n (\vec{w}^{(2)^T} \cdot \tilde{\vec{z}}(n) - t(n))^2. \tag{21}$$

[3] Please see [8] for more references.

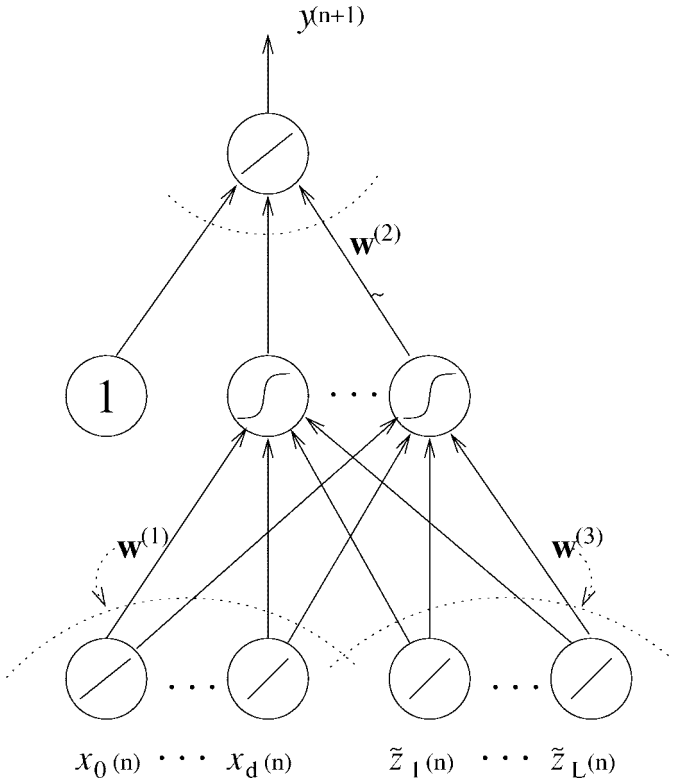

Fig. 2. Unfolded two-layer feedforward network.

C. The M-Step

New weights are obtained through maximizing the expected log likelihood $Q(\Theta|\Theta^{(p)})$ in the M-step, which involves solving two separated quadratic minimization problems

$$(\vec{w}_j^{(1)(p+1)}, \vec{w}_j^{(3)(p+1)}) = \arg \min_{\vec{w}_j^{(1)}, \vec{w}_j^{(3)}} \sum_n (\tilde{z}_j(n+1) - g(\vec{x}(n)^T \cdot \vec{w}_j^{(1)} + \tilde{\vec{z}}(n)^T \cdot \vec{w}_j^{(3)}))^2 \quad (22)$$

where $\vec{w}_j^{(1)(p+1)}$ and $\vec{w}_j^{(3)(p+1)}$ stand for the new weight vectors connecting the jth unit to the inputs and the other recurrent hidden units for $1 \le j \le L$. Solving this maximization problem is equivalent to training individual (nonrecurrent) hidden units in the unfolded network in parallel. In addition, we have

$$\vec{w}^{(2)(p+1)} = \arg \min_{\vec{w}^{(2)}} \sum_n (\vec{w}^{(2)^T}(n+1) - t(n+1))^2 + E_p \quad (23)$$

where $\vec{w}^{(2)(p+1)}$ are the new weights connecting hidden units with the output unit, and $E_p = (\lambda_2 N \|\vec{w}^{(2)}\|^2/2(\lambda_1 + \lambda_2 \|\vec{w}^{(2)^p}\|^2)) + (\lambda_2 N(\|\vec{w}^{(2)}\|^2 \|\vec{w}^{(2)^p}\|^2 - (\vec{w}^{(2)})^T(\vec{w}^{(2)^{(p)}})(\vec{w}^{(2)^{(p)}})^T \vec{w}^{(2)})/2\lambda_1(\lambda_1 + \lambda_2 \|\vec{w}^{(2)^p}\|^2))$. This step is equivalent to training the output neuron in the unfolded two layer feedforward network.

When $w_j^{(3)}$'s are set to zero, we have the M-step for feedforward networks. Therefore, the major computational cost of the M-step for both the feedforward and recurrent networks is in training individual neurons. This can be done rapidly through linear weighted regressions [7].

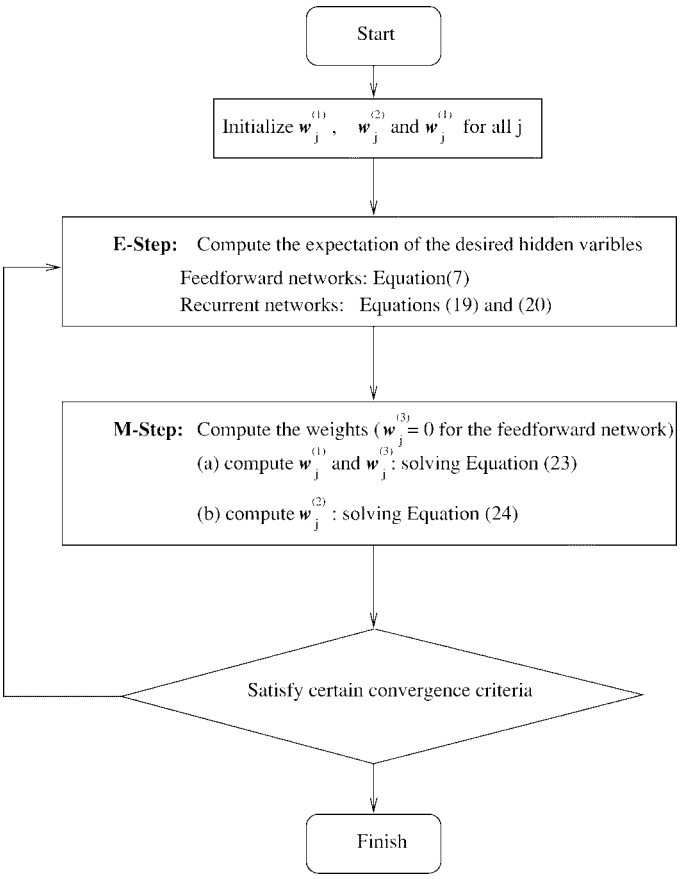

Fig. 3. Flowchart of the algorithm.

D. Summary of the Algorithm

In this work, we have established a unified framework to derive fast training algorithms for both feedforward and recurrent networks. The key steps can be summarized as follows. Probabilistic models are chosen for hidden states for both two-layer feedforward networks and the recurrent networks. Based on these models, the idea of moving targets have been formulated through the EM algorithm. For the feedforward networks, the expectation of hidden targets are computed at each E-step. In the M-step, they then serve as the targets at the hidden layer to train the first layer weights. They are also used as the inputs to the output unit to train the second-layer weights. Therefore, training two-layer feedforward networks is decomposed to training individual neurons. The mean-field approximation effectively unfolds a recurrent network into an equivalent feedforward network at each EM step. Such a feedforward network uses the expected hidden states at the previous step as a part of the inputs, and the expected hidden states at the current step as the targets at the hidden layer. Then, the maximization method developed for the feedforward network can be applied to achieve fast training for each equivalent feedforward network. The major steps of the unified EM algorithms are summarized in a flowchart given in Fig. 3. The training time has been reduced 5–15 times by both algorithms on various bench-mark problems [7], [8].

Further investigations are needed on how widely our simple models for desired hidden states is applicable for real problems.

ACKNOWLEDGMENT

The authors would like to thank helpful comments from anonymous reviewers.

REFERENCES

[1] S. Amari, "Information geometry of EM algorithms for neural networks," *Neural Networks*, vol. 8, no. 9, pp. 1379–1408, 1995.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via EM algorithm," *J. R. Statist. Soc.*, vol. B39, pp. 1–33, 1977.

[3] T. Grossman, "Learning, capacity and implementation of neural network models," Ph.D. dissertation, Weizmann Inst. Sci., Rehovot, Israel, 1992.

[4] S. Haykin and L. Li, " Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Processing*, vol. 43, pp. 526535, Feb. 1995.

[5] M. Jordan and R. A. Jacobs, "Hierarchical mixture of experts and the EM algorithm," *Neural Comput.*, vol. 6, pp. 181–214, 1994.

[6] A. Krough, G. I. Thorberaso, and J. A. Hertz, "A cost function for internal representations," in D. Touretzky, Ed., *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufmann, 1990, pp. 733–745.

[7] S. Ma, C. Ji, and J. Farmer, "An efficient EM-based training algorithm for feedforward neural networks," *Neural Networks*, vol. 10, no. 2, pp. 243–256, 1997.

[8] S. Ma and C. Ji, "Fast training of recurrent networks based on the EM algorithm," submitted for publication.

[9] R. Rohwer, "The moving target training algorithm," *Adv. Neural Inform. Process. Syst.*, vol. 2, pp. 558–565, 1990.

[10] D. Saad and E. Marom, "Learning by choice of internal representations: An energy minimization approach," *Complex Syst.*, vol. 4, pp. 107–118, 1990.

[11] J. Zhang, "The mean-field theory in EM procedures for Markov random fields," *IEEE Trans. Signal Processing*, vol. 40, pp. 2570–2583, Oct. 1992.