Convergence of approximated gradient method for Elman network

Dongpo Xu, Zhengxue Li, Wei Wu

*Department of Applied Mathematics, Dalian University of Technology, Dalian, 116024, P. R. China

Abstract

An approximated gradient method for training Elman networks is considered. For finite sample set, the error function is proved to be monotone in the training process, and the approximated gradient of the error function tends to zero if the weights sequence is bounded. Furthermore, after adding a moderate condition, the weights sequence itself is also proved to be convergent. A numerical example is given to support the theoretical findings.

Key words: Elman network; Approximated gradient method; Convergence

1 Introduction

Elman networks are a class of recurrent networks with one hidden layer, of which each hidden neuron has a feedback connection to every hidden neuron [8]. This feedback path allows Elman networks to learn to recognize and generate temporal patterns, as well as spatial patterns. Gradient learning algorithms have been used to train neural networks [1,2,4,10,11]. In particular, an approximated gradient method is introduced for training Elman networks in the neural network toolbox of MatLab [7]. The approximated gradient method takes the the recurrent weights and biases as constants in the computation of the gradient of the error function, so as to reduce greatly the computational effort. There have been many convergence results for the exact gradient algorithms for recurrent neural networks in the literature, see e.g. [5,6,13]. But

* This work is partly supported by the National Natural Science Foundation of China (10471017).
* Corresponding author.

Email address: wuwei@dlut.edu.cn (Wei Wu).

Manuscript submitted to NNW 21 May 2008
we have not found any theoretical analysis on the convergence of the approximated gradient methods for Elman networks, and this becomes our primary concern in this paper. We consider a finite set of training samples. We show that the error function is monotonically decreasing and its approximated gradient goes to zero in the training process. Some techniques in [11,12] for gradient methods has been exploited here in the proof.

The rest of the paper is organized as follows. The network architecture and the learning algorithm are described in the next two sections respectively. Section 4 presents our main theorem on the monotonicity and the convergence. A numerical example is given in Section 5 to illustrate our theoretical findings. A summary of our work in this paper consists of Section 6. The detailed proof of the theorem is given as an appendix.

![Fig. 1. An Elman network with N-M-1 structure.](image)

2 Structure of Elman networks

As shown in Fig.1, we consider an Elman network with $N$ input neurons, $M$ hidden neurons and 1 output neuron. The input and recurrent weights are collected into weight matrices $V_I \in \mathbb{R}^{M \times N}$ and $V_R \in \mathbb{R}^{M \times M}$, respectively. To simplify the presentation, we write

$$W_1 = (V_I, V_R) \in \mathbb{R}^{M \times (N+M)},$$

and write $w_0 \in \mathbb{R}^{M}$ as the weight vector between the hidden neurons and the output neuron. For a temporal sequence $\{x(t), t = 1, 2, \cdots \} \subset \mathbb{R}^{N}$ supplied to the network, we denote by $y(t) \in \mathbb{R}^{M}$ the corresponding output of the
hidden layer at time $t$, and define $y(0) = 0$. Let $g : \mathbb{R} \to \mathbb{R}$ be a chosen transfer function for the hidden and output neurons, which is typically, but not necessarily, a sigmoid function. For any vector $a = (a_1, a_2, \cdots, a_M)^T \in \mathbb{R}^M$, we define the following vector function

$$G(a) = (g(a_1), g(a_2), \cdots, g(a_M))^T.$$  \hspace{1cm} (2)

For convenience, we concatenate $x(t)$ and $y(t-1)$ to form an $N + M$ dimensional vector as follows

$$u(t) = \begin{pmatrix} x(t) \\ y(t-1) \end{pmatrix}, \hspace{1cm} t = 1, 2, \cdots$$  \hspace{1cm} (3)

The net input vector $s(t) \in \mathbb{R}^M$ to the hidden layer is computed by

$$s(t) = W_1 u(t) = V_I x(t) + V_R y(t-1), \hspace{1cm} t = 1, 2, \cdots$$  \hspace{1cm} (4)

The output of the hidden layer is then computed by

$$y(t) = G(s(t)).$$  \hspace{1cm} (5)

The final output of the network is

$$z(t) = g(w_0 \cdot y(t)), \hspace{1cm} t = 1, 2, \cdots$$  \hspace{1cm} (6)

3 Approximated gradient learning algorithm

Let us introduce some notations for convenience of presentation.

**Definition 1** Given an $m \times n$ matrix $A = (a_{ij})$, vec $A$ is defined as an $mn$ dimensional vector obtained by stacking the columns of the matrix $A$ on top of one another:

$$\text{vec } A = (a_{11}, a_{21}, \cdots, a_{m1}, a_{12}, a_{22}, \cdots, a_{m2}, \cdots, a_{1n}, a_{2n}, \cdots, a_{mn})^T.$$  \hspace{1cm} (7)

**Definition 2** Let $A = (a_{ij})$ be an $m \times n$ matrix, then the element-by-element powers $A \wedge 2$ is also an $m \times n$ matrix defined as $A \wedge 2 = (a_{ij}^2)$.

**Definition 3** Let $A = (a_{ij})$ be an $m \times n$ matrix and $B = (b_{ij})$ a $p \times q$ matrix,
then the Kronecker product $A \otimes B$ is an $mp \times nq$ matrix defined by

$$A \otimes B = \begin{pmatrix}
a_{11}B & a_{12}B & \cdots & a_{1n}B \\
a_{21}B & a_{22}B & \cdots & a_{2n}B \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1}B & a_{m2}B & \cdots & a_{mn}B
\end{pmatrix}. \quad (8)$$

**Definition 4** For $G(a)$ defined in (2), define $G'(a)$ (res. $G''(a)$) as a diagonal matrix with $g'(a_1), \cdots, g'(a_M)$ (res. $g''(a_1), \cdots, g''(a_M)$) as the diagonal elements.

Let $\{x(t), O(t)\}_{t=1}^Q$ be a given set of input-output training samples and write

$$w_1 = \text{vec } W_1, \quad (9)$$

$$w = \text{vec } (w_0, W_1). \quad (10)$$

Define the following square error function

$$E(w) = \frac{1}{2} \sum_{t=1}^{Q} (O(t) - g(w_0 \cdot G(W_1 u(t))))^2. \quad (11)$$

The aim of the network training is to find $w^*$ such that

$$E(w^*) = \min E(w). \quad (12)$$

The approximated gradient of the error function is defined by

$$p(w) = \begin{pmatrix} p_0(w) \\ p_1(w) \end{pmatrix}, \quad (13)$$

where

$$p_0(w) = -\sum_{t=1}^{Q} (O(t) - z(t)) g'(w_0 \cdot y(t)) y(t), \quad (14)$$

$$p_1(w) = -\sum_{t=1}^{Q} (O(t) - z(t)) g'(w_0 \cdot y(t)) u(t) \otimes G'(s(t)) w_0. \quad (15)$$

Actually,

$$p_0(w) = \frac{\partial E}{\partial w_0}, \quad (16)$$

and $p_1(w)$ is a part of $\frac{\partial E}{\partial w_1}$:

$$\frac{\partial E}{\partial w_1} = p_1(w) - \sum_{t=1}^{Q} (O(t) - z(t)) g'(w_0 \cdot y(t)) \left[ \frac{\partial y(t - 1)}{\partial w_1} \right]^T V_R^T G'(s(t)) w_0. \quad (17)$$
We observe that the influence of the recursion has been ignored by using $p_1(w)$ in place of $\frac{\partial E}{\partial w_1}$, and this strategy greatly reduces the computational effort.

We now describe the approximated gradient training algorithm. Given an arbitrary initial weight $w^0$, the weight sequence $\{w^k\}$ are computed iteratively by

$$w^{k+1} = w^k - \eta p(w^k), \quad k = 0, 1, 2, \cdots$$

(18)

where $\eta \in (0, 1)$ is a constant learning rate.

Write

$$\Delta w^k = w^{k+1} - w^k.$$  

(19)

Then,

$$\Delta w_0^k = \eta \sum_{t=1}^{Q}(O(t) - z^k(t))g'(w_0^k \cdot y^k(t))y^k(t),$$

(20)

$$\Delta w_1^k = \eta \sum_{t=1}^{Q}(O(t) - z^k(t))g'(w_0^k \cdot y^k(t))u^k(t) \otimes G'(s^k(t))w_0^k,$$

(21)

where

$$u^k(t) = u(w_1^k, t), \quad s^k(t) = s(w_1^k, t),$$

(22)

$$y^k(t) = y(w_1^k, t), \quad z^k(t) = z(w^k, t).$$

(23)

4 Main result

The following assumptions will be used in our discussion.

**Assumption (A1)** $|g(r)|, |g'(r)|, |g''(r)|$ are bounded for $r \in \mathbb{R}$.

**Assumption (A2)** $\|w_0^k\|, \|V^R_k\|$ ($k = 0, 1, 2, \cdots$) (cf. (1), (9) and (10)) are bounded in the learning process (18).

**Assumption (A3)** There exists a closed bounded region $\Phi \subset \mathbb{R}^{M(N+M+1)}$ such that $\{w^k\} \subset \Phi$ and the set $\Phi_0 = \{w \in \Phi : p(w) = 0\}$ (cf. (13)) contains only finite number of points.

**Remark 5** Assumption (A1) is valid for Sigmoid functions which are the most often used activation functions. An assumption like (A2) is often used in literature (see e.g. [3]) for a nonlinear iteration procedure to guarantee the convergence. Assumption (A3) will be used to get a strong convergence result.

Now we are in a position to present the main theorem. Its proof has been relegated to the Appendix.

**Theorem 6** Suppose that the error function is given by (11), that the weight sequence $\{w^k\}$ is generated by the algorithm (18) for an arbitrary initial value $w^0$, that Assumptions (A1) and (A2) are valid, and that $\eta$ is small enough such that (37) below is valid. Then, we have
(a) \(E(w^{k+1}) \leq E(w^k), \ k = 0, 1, 2, \ldots\);
(b) There is \(E^* \geq 0\) such that \(\lim_{k \to \infty} E(w^k) = E^*;\)
(c) \(\lim_{k \to \infty} \|\Delta w^k\| = 0, \ \lim_{k \to \infty} \|p(w^k)\| = 0.\)

Moreover, if Assumption (A3) is also valid, then we have the strong convergence:
(d) There exists \(w^* \in \Phi_0\) such that \(\lim_{k \to \infty} w^k = w^*.\)

5 Simulation experiment

In this section, the XOR problem with two-cycle delay (see e.g. [10]) is simulated. We use a network with two input units and a bias unit \((N = 3\) in Fig. 1), four hidden units \((M = 4)\) and one output unit. The transfer function for both hidden and out layers is \(g(x) = (e^x - e^{-x})/(e^x + e^{-x}).\) The learning rate \(\eta\) is set to be 0.2, the initial values of \(w^0\) are chosen stochastically in \([-3, 3]\), and the training iteration stops when it reaches 1000 training epoches or when the error \(E < 0.0001.\)

\[\begin{array}{|c|c|c|c|}
\hline
\text{number of epoch} & \text{square error} & \text{norm of approximated gradient} & \text{norm of gradient} \\
\hline
0 & 10^0 & 10^0 & 10^0 \\
200 & 10^{-4} & 10^{-4} & 10^{-4} \\
400 & 10^{-3} & 10^{-3} & 10^{-3} \\
600 & 10^{-2} & 10^{-2} & 10^{-2} \\
800 & 10^{-1} & 10^{-1} & 10^{-1} \\
1000 & 10^0 & 10^0 & 10^0 \\
\hline
\end{array}\]

Fig. 2. XOR with two-cycle delay: Square error and norm of approximated gradient.

The simulation results are shown in Fig. 2 and Table 1. From Fig. 2, we see that the error decreases monotonically and the corresponding approximated gradient tends to zero when the number of iteration increases, as predicted by our convergence theorem. Table 1 gives the the network output \(z(t)\) compared with the target output \(O(t)\) after 1000 epochs of training. In this table, the rows labeled \(t, x_i, O\) and \(z\) represent the sample number, the \(i\)th component of the external input vector, the target output and the network output, respectively. We observe that, for instance, for the same external input \((x_1, x_2) = (0, 1)\)
when \( t = 4 \) and \( t = 7 \), the network responses are totally different due to the recurrence.

Table 1

<table>
<thead>
<tr>
<th>( t )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( O )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( z )</td>
<td>-0.01</td>
<td>0.98</td>
<td>0.00</td>
<td>0.99</td>
<td>0.95</td>
<td>0.99</td>
<td>0.00</td>
<td>0.99</td>
<td>0.95</td>
<td>0.99</td>
<td>0.01</td>
<td>0.99</td>
<td>0.95</td>
<td>0.99</td>
<td>0.00</td>
<td>0.90</td>
</tr>
</tbody>
</table>

6 Conclusion

This paper considers the convergence of an approximated gradient method for training Elmen network with finite training sample set. The method requires less computation effort and uses a constant learning rate. It has been included in a widely used neural network toolbox of Matlab, but its theoretical convergence has not been considered before. The key for the convergence analysis is the monotonicity of the error function during the learning iteration, which is proved under the assumption that the weights are bounded and the learning rate is small enough (cf. (37) below). Some weak and strong convergence results are established. Here the weak convergence means \( \lim_{k \to \infty} \| p(w^k) \| = 0 \), while the strong convergence means that the sequence \( \{ w^k \} \) itself converges. The validity of the theoretical study is demonstrated through numerical experiment.

Appendix

Let us introduce a few symbols to be used in our proof later on

\[
\Delta s^k(t) = s^{k+1}(t) - s^k(t), \quad (24)
\]

\[
\Delta y^k(t) = y^{k+1}(t) - y^k(t). \quad (25)
\]

To begin with, let us present three lemmas. In the following argument, we use \( C \) for a generic positive constant which may be different in different places. The proofs to the following two lemmas are straightforward, and thus are omitted.

**Lemma 7** For the term \( \Delta s^k(t) (t \geq 2) \) given in (24), there holds

\[
\Delta s^k(t) = \left( u^k(t)^T \otimes I_M \right) \Delta w_1^k + \delta^k(t),
\]
where
\[
\delta^k(t) = V_{t+1}^{k+1} G'(\sigma^k(t - 1)) \Delta s^k(t - 1),
\]
and \(I_M\) is the \(M\)-by-\(M\) identity matrix, and \(\sigma^k(t - 1) \in \mathbb{R}^M\) with its component in between the two corresponding components of \(s^k(t - 1)\) and \(s^{k+1}(t - 1)\).

**Lemma 8** Suppose that (A1) and (A2) are satisfied, then
\[
\|\Delta s^k(t)\| \leq C\|\Delta w_1^k\|. \tag{26}
\]
The next lemma is basically the same as Theorem 14.1.5 in [9] (also see [14]). Its proof is thus omitted.

**Lemma 9** Let \(F : \Phi \subset \mathbb{R}^m \to \mathbb{R}^m (m \geq 1)\) be continuous for a bounded closed region \(\Phi\) and \(\Phi_0 = \{w \in \Phi : F(w) = 0\}\) contains finite points. Let the sequence \(\{w^k\} \subset \Phi\) satisfy \(\lim_{k \to \infty} F(w^k) = 0\) and \(\lim_{k \to \infty} \|w^{k+1} - w^k\| = 0\). Then, there exists \(w^* \in \Phi_0\) such that \(\lim_{k \to \infty} w^k = w^*\).

Now, we are ready for the proof of our main result.

**Proof to Theorem 6.** By (22)–(25), the Taylor expansion, Lemma 7 and the learning rule (21), we have
\[
- \sum_{t=1}^{Q} \left( O(t) - z^k(t) \right) g' \left( w_0^k \cdot y^k(t) \right) w_0^k \cdot \Delta y^k(t) = - \frac{1}{\eta} \|\Delta w_1^k\|^2 + \rho_1^k, \tag{27}
\]
where
\[
\rho_1^k = - \sum_{t=1}^{Q} \left( O(t) - z^k(t) \right) g' \left( w_0^k \cdot y^k(t) \right) \left( w_0^k \right)^T G'(\sigma^k(t)) \delta^k(t) \]
\[
- \frac{1}{2} \sum_{t=1}^{Q} \left( O(t) - z^k(t) \right) g' \left( w_0^k \cdot y^k(t) \right) \left( w_0^k \right)^T G''(\sigma^k(t)) \left( w_0^k \right) \delta^k(t) \delta^k(t)^T, \tag{28}
\]
and \(\tau^k(t) \in \mathbb{R}^M\) is a vector, of which each component lies in between the two corresponding components of \(s^{k+1}(t)\) and \(s^k(t)\).

Again applying the Taylor expansion and noting (10), (11), (20), (27) and (28), we have
\[ E(w^{k+1}) - E(w^k) \]
\[ = -\sum_{t=1}^{Q} (O(t) - z^k(t)) \cdot g'(w_0^k \cdot y^k(t)) \left( w_0^{k+1} \cdot y^{k+1}(t) - w_0^k \cdot y^k(t) \right) + \rho_2^k \]
\[ = -\sum_{t=1}^{Q} (O(t) - z^k(t)) \cdot g'(w_0^k \cdot y^k(t)) \Delta w_0^k \cdot y^k(t) \]
\[ - \sum_{t=1}^{Q} (O(t) - z^k(t)) \cdot g'(w_0^k \cdot y^k(t)) w_0^k \cdot \Delta y^k(t) + \rho_2^k + \rho_3^k \]
\[ = -\frac{1}{\eta} \| \Delta w^k \|^2 + \rho_2^k + \rho_3^k, \tag{29} \]

with
\[
\rho_2^k = -\frac{1}{2} \sum_{t=1}^{Q} (O(t) - z^k(t)) \cdot g''(\theta^k(t)) \left( w_0^{k+1} \cdot y^{k+1}(t) - w_0^k \cdot y^k(t) \right)^2, \tag{30} 
\]
\[
\rho_3^k = -\sum_{t=1}^{Q} (O(t) - z^k(t)) \cdot g'(w_0^k \cdot y^k(t)) \Delta w_0^k \cdot \Delta y^k(t) , 
\]

and \( \theta^k(t) \) is a real number between \( w_0^k \cdot y^k(t) \) and \( w_0^{k+1} \cdot y^{k+1}(t) \).

By Assumption \((A1), (2)\) and \((5)\), we get
\[
\| y(a) \| \leq \sqrt{M} \sup_{r \in \mathbb{R}} | g(r) | \leq C, \quad a \in \mathbb{R}^M . \tag{31} 
\]

It results from \((25)\), the mean value theorem, Assumption \((A1)\) and Lemma 8 that
\[
\| \Delta y^k(t) \| \leq C \| \Delta w_1^k \| . \tag{32} 
\]

It follows from \((A1), (A2), \) Lemma 7 and Lemma 8 that
\[
|\rho_2^k| \leq C \left( \| \Delta w^k \|^2 + \| \Delta w^k \| \right) . \tag{33} 
\]

A combination of \((30)-(32), (A1), (A2)\) and the Cauchy-Schwartz inequality gives
\[
|\rho_2^k| \leq C \sum_{t=1}^{Q} \left( \| \Delta w_0^k \| + \| \Delta y^k(t) \| \right)^2 \leq C \| \Delta w^k \|^2 . \tag{34} 
\]

Similarly, we can obtain
\[
|\rho_3^k| \leq C \sum_{t=1}^{Q} \left( \| \Delta w_0^k \|^2 + \| \Delta y^k(t) \|^2 \right) \leq C \| \Delta w^k \|^2 . \tag{35} 
\]

A combination of \((29)\) and \((33)-(35)\) leads to
\[
E(w^{k+1}) - E(w^k) \leq - \left( \frac{1}{\eta} - C \right) \left( \| \Delta w^k \|^2 + \| \Delta w^k \| \right). \tag{36} 
\]
Hence, Conclusion (a) is valid if the learning rate is small enough such that

\[ 0 < \eta < \frac{1}{C}, \]  

where \( C \) is the constant in (36).

Since the nonnegative sequence \( \{E(w^k)\} \) is monotone and bounded below, there must be a limit value \( E^* \geq 0 \) such that \( \lim_{k \to \infty} E(w^k) = E^* \). So Conclusion (b) is proved.

Next, we prove the weak convergence (c). Let

\[ \beta = \frac{1}{\eta} - C. \]  

By (36) we have

\[ E(w^{K+1}) \leq E(w^K) - \beta \left( \|\Delta w^K\|^2 + \|\Delta w^K\| \right) \leq E(w^K) - \beta \|\Delta w^K\| \]  

\[ \leq \cdots \leq E(w^0) - \beta \sum_{k=0}^{K} \|\Delta w^k\|. \]  

Since \( E(w^{K+1}) \geq 0 \) for any \( K \geq 0 \), we let \( K \to \infty \) to get

\[ \sum_{k=0}^{\infty} \|\Delta w^k\| \leq \frac{1}{\beta} E(w^0) < \infty, \]  

and thus

\[ \lim_{k \to \infty} \|\Delta w^k\| = 0. \]  

This together with (18), (19) and (42) leads to

\[ \lim_{k \to \infty} \|p(w^k)\| = 0. \]  

Finally, we prove the strong convergence (d). Let us use Lemma 9 by taking \( F(w) = p(w) \). Then, Conclusion (d) immediately results from a combination of Conclusion (c), the finiteness of \( \Phi_0 \) (cf. Assumption (A3)), (42) and (43).

This completes the proof. \( \square \)

References


