Journal of Applied Mathematics and Physics, 2018, 6, 880-887
http://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

A Study on the Convergence of Gradient Method with Momentum for Sigma-Pi-Sigma Neural Networks

Xun Zhang, Naimin Zhang*

School of Mathematics and Information Science, Wenzhou University, Wenzhou, China
Email: *nmzhang@wzu.edu.cn

How to cite this paper: Zhang, X. and Zhang, N.M. (2018) A Study on the Convergence of Gradient Method with Momentum for Sigma-Pi-Sigma Neural Networks. *Journal of Applied Mathematics and Physics*, **6**, 880-887.
https://doi.org/10.4236/jamp.2018.64075

Received: December 20, 2017
Accepted: April 23, 2018
Published: April 26, 2018

Abstract

In this paper, a gradient method with momentum for sigma-pi-sigma neural networks (SPSNN) is considered in order to accelerate the convergence of the learning procedure for the network weights. The momentum coefficient is chosen in an adaptive manner, and the corresponding weak convergence and strong convergence results are proved.

Keywords

Sigma-Pi-Sigma Neural Network, Momentum Term, Gradient Method, Convergence

1. Introduction

Pi-sigma network (PSN) is a kind of high order feedforward neural network which is characterized by the fast convergence rate of the single-layer network, and the unique high order network nonlinear mapping capability [1]. In order to further improve the application capacity of the network, Li introduces more complex network structures based on PSN called sigma-pi-sigma neural network (SPSNN) [2]. SPSNN can be learned to implement static mapping in the similar manner to that of multilayer neural networks and the radial basis function networks.

The gradient method is often used for training neural networks, and the main disadvantages of this method are the slow convergence and the local minimum problem. To speed up and stabilize the training iteration procedure for the gradient method, a momentum term is often added to the increment formula for the weights, in which the present weight updating increment is a combination of the present gradient of the error function and the previous weight updating increment [3]. Many researchers have developed the theory about momentum and extended its applications. For the back-propagation algorithm, Phansalkar and Sastry give a stability analysis with adding the momentum term [4]. Torii and Bhaya discuss the convergence of the gradient method with momentum under the restriction that the error function is quadratic [5] [6]. Shao *et al.* study the adaptive momentum for both batch gradient method and online gradient method, and compare the efficiency of momentum with penalty [7] [8] [9] [10] [11]. The key for the convergence analysis for momentum algorithms is the monotonicity of the error function during the learning procedure, which is generally proved under the uniformly boundedness assumption of the activation function and its derivatives. In [8] [10] [12] [13], for the gradient method with momentum, some convergence results are given for both two-layer and multi-layer feedforward neural networks. In this paper, we will consider the gradient method with momentum for sigma-pi-sigma neural networks and discuss its convergence.

The rest of the paper is organized as follows. In Section 2 we introduce the neural network model of SPSNN and the gradient method with momentum. In Section 3 we give the convergence analysis of the gradient method with momentum for training SPSNN. Numerical experiments are given in Section 4. Finally, in Section 5, we end the paper with some conclusions.

2. The Neural Network Model of SPSNN and Gradient Method with Momentum

In this section we introduce the sigma-pi-sigma neural network that is composed of multilayer neural network. The output of SPSNN has the form $\sum_{n=1}^{K}\prod_{i=1}^{n}\sum_{j=1}^{N_v} f_{nij}\left(x_j\right)$, where x_j is an input, N_v is the number of inputs, $f_{nij}\left(\ \right)$ is a function to be generated through the network training, and K is the number of pi-sigma network(PSN) that is the basic building block for SPSNN. The expression of the function $f_{nij}\left(x_j\right)$ is $\sum_{k=1}^{N_q+N_e-1} w_{nijk}B_{ijk}\left(x_j\right)$, where the function $B_{ijk}\left(\ \right)$ is either 0 or 1, and w_{nijk} is weight values stored in memory. N_q and N_e are information numbers stored in x_j. For a K-th order SPSNN, the total weight value will be

$$\frac{1}{2}\times K\times\left(K+1\right)\times N_v\times\left(N_q+N_e-1\right).$$

For a set of training examples $\left\{\left(S_t,O_t\right)\in R^{N_v}\times R\right\}$, where O_t is the ideal output, $t=1,2,\cdots,T$, we have the following actual output:

$$y_t=\sum_{n=1}^{K}\prod_{i=1}^{n}\sum_{j=1}^{N_v}\sum_{k=1}^{N_q+N_e-1} w_{nijk}B_{ijk}\left(x_j^{(S_t)}\right),$$

where $x_j^{(S_t)}$ denotes the *j*th element of a given input vector S_t.

In order to train the SPSNN, we choose a quadratic error function $E\left(W\right)$:

$$E\left(W\right)=\frac{1}{2}\sum_{t=1}^{T}\left(O_t-y_t\right)^2\equiv\sum_{t=1}^{T}g_t\left(W\right),\ g_t\left(W\right)=\frac{1}{2}\left(O_t-y_t\right)^2$$

where $W=\left(w_{1111},w_{1112},\cdots,w_{K,K,N_v,N_q+N_e-1}\right)^{\mathrm{T}}$. For convenience we denote $w_\alpha=w_{K,K,N_v,N_q+N_e-1}$.

The gradient method with momentum is used to train weights. The gradients of $E(W)$ and $g_t(W)$ are denoted by

$$\nabla E(W)=\left(\frac{\partial E(W)}{\partial w_{1111}},\frac{\partial E(W)}{\partial w_{1112}},\cdots,\frac{\partial E(W)}{\partial w_{nijk}},\cdots,\frac{\partial E(W)}{\partial w_\alpha}\right)^{\mathrm{T}},$$

$$\nabla g_t(W)=\left(\frac{\partial g_t(W)}{\partial w_{1111}},\frac{\partial g_t(W)}{\partial w_{1112}},\cdots,\frac{\partial g_t(W)}{\partial w_{nijk}},\cdots,\frac{\partial g_t(W)}{\partial w_\alpha}\right)^{\mathrm{T}},$$

and the Hessian matrices of $g_t\left(W^m\right)$ and $E\left(W^m\right)$ at W^m are denoted by

$$\nabla^2 g_t\left(W^m\right)=\begin{pmatrix}\frac{\partial^2 g_t\left(W^m\right)}{\partial w_{1111}^m\partial w_{1111}^m} & \cdots & \frac{\partial^2 g_t\left(W^m\right)}{\partial w_{1111}^m\partial w_\alpha^m}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2 g_t\left(W^m\right)}{\partial w_\alpha^m\partial w_{1111}^m} & \cdots & \frac{\partial^2 g_t\left(W^m\right)}{\partial w_\alpha^m\partial w_\alpha^m}\end{pmatrix},$$

$$\nabla^2 E\left(W^m\right)=\begin{pmatrix}\frac{\partial^2 E\left(W^m\right)}{\partial w_{1111}^m\partial w_{1111}^m} & \cdots & \frac{\partial^2 E\left(W^m\right)}{\partial w_{1111}^m\partial w_\alpha^m}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2 E\left(W^m\right)}{\partial w_\alpha^m\partial w_{1111}^m} & \cdots & \frac{\partial^2 E\left(W^m\right)}{\partial w_\alpha^m\partial w_\alpha^m}\end{pmatrix}.$$

Given any arbitrarily initial weight vectors W^0, W^1, the gradient method with momentum updates the weight vector W by

$$W^{m+1}=W^m-\eta\nabla E\left(W^m\right)+\tau^m\left(W^m-W^{m-1}\right),\ m=1,2,\cdots,\tag{1}$$

where $\eta>0$ is the learning rate, W^m-W^{m-1} is called the momentum term, τ^m is the momentum coefficient.

Similar to [12] [14], in this paper, we choose τ^m as follows:

$$\tau^m=\begin{cases}\frac{\mu\left\|\nabla E\left(W^m\right)\right\|}{\left\|\Delta W^m\right\|}, & \text{if } \Delta W\neq 0\\ 0, & \text{else}\end{cases}$$

where μ is a positive number and $\Delta W^m=W^m-W^{m-1}$, and $\|\cdot\|$ is 2-norm in this paper.

Notice the component form of (1) is

$$w_{nijk}^{m+1}=w_{nijk}^m-\eta\frac{\partial E\left(W^m\right)}{\partial w_{nijk}^m}+\tau^m\left(w_{nijk}^m-w_{nijk}^{m-1}\right).$$

In fact,

$$y_t=PSN_1+PSN_2+\cdots+PSN_K,$$

where $PSN_n=\prod_{i=1}^n U_{ni}$. Recalling $f_{nij}\left(x_j\right)=\sum_{k=1}^{N_q+N_e-1}w_{nijk}B_{ijk}\left(x_j\right)$, then

$$\begin{aligned}\frac{\partial E(W)}{\partial w_{nijk}} &= \sum_{t=1}^{T}(y_t - O_t)\frac{\partial y_t}{\partial w_{nijk}} \\ &= \sum_{t=1}^{T}(y_t - O_t)\frac{\partial PSN_n}{\partial U_{ni}}\frac{\partial U_{ni}}{\partial f_{nij}}\frac{\partial f_{nij}}{\partial w_{nijk}} \\ &= \sum_{t=1}^{T}(y_t - O_t)\left\{\prod_{p\neq i} U_{np}\right\} B_{ijk}(x_j)\end{aligned}$$

3. Convergence Results

Similar to [12] [14], we need the following assumptions.

(A1): The elements of the Hessian matrix $\nabla^2 E(W^m)$ be uniformly bounded for any W^m.

(A2): The number of the elements of $\Omega = \{W \mid \nabla E(W) = 0\}$ be finite.

From (A1), it is easy to see that there exists a constant $M > 0$ such that

$$\left\|\nabla^2 E(W^m)\right\| \le M, m = 0,1,2,\cdots.$$

Lemma 3.1 ([15]) Let $f: R^n \to R$ be continuously differentiable, the number of the elements of the set $\Omega = \{x \mid \nabla f(x) = 0\}$ be finite, and the sequence $\{x^k\}$ satisfy

$$\lim_{k\to\infty}\left\|x^k - x^{k-1}\right\| = 0,$$

$$\lim_{k\to\infty}\left\|\nabla f(x^k)\right\| = 0.$$

Then there exists a $x^* \in R^n$ such than

$$\lim_{k\to\infty} x^k = x^*, \nabla f(x^*) = 0.$$

Theorem 3.2 If Assumption (A1) is satisfied. Then there exists $E^* \ge 0$ such that for $\eta \in \left(0, \frac{2}{M}\right)$ and $\mu \in \left(0, \frac{-1 - M\eta + \sqrt{1 + 4M\eta}}{M}\right)$, it holds the following weak convergence result for the iteration (1):

$$E(W^{m+1}) \le E(W^m),$$

$$\lim_{m\to\infty} E(W^m) = E^*,$$

$$\lim_{m\to\infty}\left\|\nabla E(W^m)\right\| = 0.$$

Furthermore, if Assumption (A2) is also valid, then it holds the strong convergence result, that is there exists W^* such that

$$\lim_{m\to\infty} W^m = W^*, \nabla E(W^*) = 0.$$

Proof.

Using Taylor's formula, we expand $g_t(W^{m+1})$ at W^m:

$$\begin{aligned}g_t(W^{m+1}) &= g_t(W^m) + \left(\nabla g_t(W^m)\right)^{\mathrm{T}}(W^{m+1} - W^m) \\ &\quad + \frac{1}{2}(W^{m+1} - W^m)^{\mathrm{T}}\nabla^2 g_t(\xi^m)(W^{m+1} - W^m)\end{aligned} \tag{2}$$

where ξ^m lies in between W^m and W^{m+1}.

From (2) we have

$$\sum_{t=1}^{T} g_t\left(W^{m+1}\right) = \sum_{t=1}^{T} g_t\left(W^m\right) + \sum_{t=1}^{T}\left(\nabla g_t\left(W^m\right)\right)^{\mathrm{T}}\left(W^{m+1} - W^m\right) + \sum_{t=1}^{T}\frac{1}{2}\left(W^{m+1} - W^m\right)^{\mathrm{T}}\nabla^2 g_t\left(\xi^m\right)\left(W^{m+1} - W^m\right)$$

The above equation is equivalent to

$$E\left(W^{m+1}\right) = E\left(W^m\right) + \delta_1 + \delta_2 \tag{3}$$

where

$$\delta_1 = \sum_{t=1}^{T}\left(\nabla g_t\left(W^m\right)\right)^{\mathrm{T}}\left(W^{m+1} - W^m\right),$$

$$\delta_2 = \sum_{t=1}^{T}\frac{1}{2}\left(W^{m+1} - W^m\right)^{\mathrm{T}}\nabla^2 g_t\left(\xi^m\right)\left(W^{m+1} - W^m\right).$$

It is easy to see that

$$\begin{aligned}
\delta_1 &= \left(\nabla E\left(W^m\right)\right)^{\mathrm{T}}\Delta W^{m+1} \\
&= \left(\nabla E\left(W^m\right)\right)^{\mathrm{T}}\left(-\eta\nabla E\left(W^m\right) + \tau^m\Delta W^m\right) \\
&= -\eta\left(\nabla E\left(W^m\right)\right)^{\mathrm{T}}\nabla E\left(W^m\right) + \tau^m\left(\nabla E\left(W^m\right)\right)^{\mathrm{T}}\Delta W^m \\
&\le -\eta\left\|\nabla E\left(W^m\right)\right\|^2 + \mu\left\|\left(\nabla E\left(W^m\right)\right)^{\mathrm{T}}\right\|\frac{\left\|\nabla E\left(W^m\right)\right\|}{\left\|\Delta W^m\right\|}\left\|\Delta W^m\right\| \\
&= \left(-\eta + \mu\right)\left\|\nabla E\left(W^m\right)\right\|^2
\end{aligned}$$

$$\begin{aligned}
\delta_2 &= \frac{1}{2}\left(\Delta W^{m+1}\right)^{\mathrm{T}}\sum_{t=1}^{T}\nabla^2 g_t\left(\xi^m\right)\Delta W^{m+1} \\
&= \frac{1}{2}\left(\Delta W^{m+1}\right)^{\mathrm{T}}\nabla^2 E\left(\xi^m\right)\Delta W^{m+1} \\
&\le \frac{1}{2}\left|\left(\Delta W^{m+1}\right)^{\mathrm{T}}\nabla^2 E\left(\xi^m\right)\Delta W^{m+1}\right| \\
&\le \frac{1}{2}\left\|\left(\Delta W^{m+1}\right)^{\mathrm{T}}\right\|\left\|\nabla^2 E\left(\xi^m\right)\right\|\left\|\Delta W^{m+1}\right\| \\
&\le \frac{1}{2}M\left\|\Delta W^{m+1}\right\|^2 \\
&= \frac{1}{2}M\left\|-\eta\nabla E\left(W^m\right) + \tau^m\Delta W^m\right\|^2 \\
&\le \frac{1}{2}M\left(\left\|-\eta\nabla E\left(W^m\right)\right\| + \left\|\tau^m\Delta W^m\right\|\right)^2 \\
&= \frac{1}{2}M\left(\left\|-\eta\nabla E\left(W^m\right)\right\|^2 + \left\|\tau^m\Delta W^m\right\|^2 + 2\left\|-\eta\nabla E\left(W^m\right)\right\|\left\|\tau^m\Delta W^m\right\|\right) \\
&\le \frac{1}{2}M\left(\eta^2\left\|\nabla E\left(W^m\right)\right\|^2 + \mu^2\left\|\nabla E\left(W^m\right)\right\|^2 + 2\eta\mu\left\|\nabla E\left(W^m\right)\right\|^2\right) \\
&= \frac{1}{2}M\left(\eta + \mu\right)^2\left\|\nabla E\left(W^m\right)\right\|^2
\end{aligned}$$

Together with (3), we have

$$
\begin{aligned}
E\left(W^{m+1}\right) &= E\left(W^{m}\right)+\delta_1+\delta_2 \\
&\le E\left(W^{m}\right)-\left(\eta-\mu-\frac{1}{2}M\left(\eta+\mu\right)^2\right)\left\|\nabla E\left(W^{m}\right)\right\|^2
\end{aligned}
$$

Set $\beta=\eta-\mu-\frac{1}{2}M\left(\eta+\mu\right)^2$. Then

$$E\left(W^{m+1}\right)\le E\left(W^{m}\right)-\beta\left\|\nabla E\left(W^{m}\right)\right\|^2 . \tag{4}$$

It is easy to see that $\beta>0$ when

$$\begin{cases}\eta\in\left(0,\dfrac{2}{M}\right)\\ \mu\in\left(0,\dfrac{-1-M\eta+\sqrt{1+4M\eta}}{M}\right)\end{cases}. \tag{5}$$

If η and μ satisfy (5), then the sequence $\left\{E\left(W^{m}\right)\right\}$ is monotonically decreasing. Since $E\left(W^{m}\right)$ is nonnegative, it must converge to some $E^*\ge 0$, that is

$$\lim_{m\to\infty}E\left(W^{m}\right)=E^* .$$

By (4) it is easy to see for any positive integer N, it holds

$$\beta\sum_{m=0}^{N-1}\left\|\nabla E\left(W^{m}\right)\right\|^2\le E\left(W^{0}\right)-E\left(W^{N}\right).$$

Let $N\to\infty$, then we have $\sum_{m=0}^{\infty}\left\|\nabla E\left(W^{m}\right)\right\|^2\le\infty$, so $\lim_{m\to\infty}\left\|\nabla E\left(W^{m}\right)\right\|=0$, which finishes the proof for the weak convergence.

By (1), we have

$$\left\|W^{m+1}-W^{m}\right\|\le\eta\left\|\nabla E\left(W^{m}\right)\right\|+\tau^m\left\|\Delta W^{m}\right\|\le\left(\eta+\mu\right)\left\|\nabla E\left(W^{m}\right)\right\|,$$

which indicates

$$\lim_{m\to\infty}\left\|W^{m+1}-W^{m}\right\|=0 .$$

From Lemma 3.1, it holds

$$\lim_{m\to\infty}W^{m}=W^*,\nabla E\left(W^*\right)=0,$$

which finishes the proof for the strong convergence.

4. Numerical Results

In this section, we propose an example to illustrate the convergence behavior of the iteration (1) by comparing the iteration steps (IT), elapsed CPU time in seconds (CPU) and relative residual error (RES). The experiment is terminated when the current iteration satisfies $\text{RES}\le 10^{-8}$ or the number of the max iteration steps k = 1000 are exceeded. The computations are implemented in MATLAB on a PC computer with Intel (R) Core (R) CPU 1000 M @ 1.80 GHz, and 2.00 GB memory.

Example 4.1 ([16]) Four-dimensional parity problem (Table 1)

Table 1. The data samples.

input	*output*	*input*	*output*	*input*	*output*	*input*	**output**
1 1 1 1	0	−1 1 −1 −1	1	−1 −1 −1 1	1	1 −1 −1 1	0
−1 1 1 1	1	1 1 −1 −1	0	1 −1 1 −1	0	−1 −1 1 −1	1
1 1 1 −1	0	1 −1 1 1	1	−1 −1 −1 −1	0	1 1 −1 1	1
−1 1 1−1	0	−1 −1 1 1	0	1 −1 −1 −1	1	−1 1 −1 1	0

Table 2. Optimal parameters, CPU times, iteration numbers, and residuals.

Algorithm	*OPT*	*CPU*(*s*)	*IT*	**RES**
no momentum	$\eta = 10^{-5}$	0.3984	223	1.3178×10^{-9}
momentum	$\eta = 10^{-5}, \mu = 5 \times 10^{-5}$	0.2106	28	9.3907×10^{-9}

In this simulation experiment, the initial weights W^0 is a zero vector of 24 dimensional and W^1 is a 24 dimensional vector whose elements are all 1. The learning rate $\eta = 0.00001$ and momentum factor $\mu = 0.00005$. The number of training samples is $T = 16$. In the above Table 2, we compare the convergence behavior of the gradient method with momentum and the gradient method with no momentum. It can be seen that the network training is improved significantly after added the momentum item.

5. Conclusion

In this paper, we study the gradient method with momentum for training sigma-pi-sigma neural networks. We take the momentum coefficient in an adaptive manner, and the corresponding weak convergence and strong convergence results are proved. The Assumptions A1 and A2 in this paper seem to be a little severe, so how to weaken the one or two assumptions will be our future work.

Support Information

This author is supported by National Natural Science Foundation of China under grant No. 61572018 and Zhejiang Provincial Natural Science Foundation of China under Grant No. LY15A010016.

References

[1] Shin, Y. and Ghosh, J. (1991) The Pi-Sigma Network: An Efficient Higher-Order Neural Network for Pattern Classification and Function Approximation. *International Joint Conference on Neural Networks*, **1**, 13-18. https://doi.org/10.1109/IJCNN.1991.155142

[2] Li, C.K. (2003) A Sigma-Pi-Sigma Neural Network (SPSNN). *Neural Processing Letters*, **17**, 1-19. https://doi.org/10.1023/A:1022967523886

[3] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) Learning Representations by Back-Propagating Errors. *Nature*, **323**, 533-536. https://doi.org/10.1038/323533a0

[4] Phansalkar, V.V. and Sastry, P.S. (1994) Analysis of the Back-Propagation Algorithm with Momentum. *IEEE Transactions on Neural Networks*, **5**, 505-506. https://doi.org/10.1109/72.286925

[5] Torii, M. and Hagan, M.T. (2002) Stability of Steepest Descent with Momentum for Quadratic Functions. *IEEE Transactions on Neural Networks*, **13**, 752-756. https://doi.org/10.1109/TNN.2002.1000143

[6] Bhaya, A. and Kaszkurewicz, E. (2004) Steepest Descent with Momentum for Quadratic Functions Is a Version of the Conjugate Gradient Method. *Neural Networks*, **17**, 65-71. https://doi.org/10.1016/S0893-6080(03)00170-9

[7] Qian, N. (1999) On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Networks*, **12**, 145-151. https://doi.org/10.1016/S0893-6080(98)00116-6

[8] Shao, H. and Zheng, G. (2011) Convergence Analysis of a Back-Propagation Algorithm with Adaptive Momentum. *Neurocomputing*, **74**, 749-752. https://doi.org/10.1016/j.neucom.2010.10.008

[9] Shao, H. and Zheng, G. (2011) Boundedness and Convergence of Online Gradient Method with Penalty and Momentum. *Neurocomputing*, **74**, 765-770. https://doi.org/10.1016/j.neucom.2010.10.005

[10] Shao, H., Xu, D., Zheng, G. and Liu, L. (2012) Convergence of an Online Gradient Method with Inner-Product and Adaptive Momentum. *Neurocomputing*, **747**, 243-252. https://doi.org/10.1016/j.neucom.2011.09.003

[11] Xu, D., Shao, H. and Zhang, H. (2012) A New Adaptive Momentum Algorithm for Split-Complex Recurrent Neural Networks. *Neurocomputing*, **93**, 133-136. https://doi.org/10.1016/j.neucom.2012.03.013

[12] Zhang, N., Wu, W. and Zheng, G. (2006) Convergence of Gradient Method with Momentum for Two-Layer Feedforward Neural Networks. *IEEE Transactions Neural Networks*, **17**, 522-525. https://doi.org/10.1109/TNN.2005.863460

[13] Wu, W., Zhang, N., Li, Z., Li, L. and Liu, Y. (2008) Convergence of Gradient Method with Momentum for Back-Propagation Neural Networks. *Journal of Computational Mathematics*, **4**, 613-623.

[14] Gori, M. and Maggini, M. (1996) Optimal Convergence of On-Line Backpropagation. *IEEE Transactions on Neural Networks*, **7**, 251-254. https://doi.org/10.1109/72.478415

[15] Yuan, Y. and Sun, W. (1997) Optimization Theory and Methods. Science Press, Beijing.

[16] Yan, X. and Chao, Z. (2008) Convergence of Asynchronous Batch Gradient Method with Momentum for Pi-Sigma Networks. *Mathematica Applicata*, **21**, 207-212.