https://doi.org/10.26682/sjuod.2019.22.2.12

Journal of University of Duhok, Vol. 22, No.2 (Pure and Eng. Sciences), Pp 109-114, 2019

ON MODIFICATION OF SYMMETRIC RANK ONE FOR TRAINING NEURAL NETWORK BASED ON GRADIENT VECTOR

HASAN HAZIM JAMEEL*, and **SALAH GHAZE SHAREEF****

*Dept. of Mathematics, College of Basic Education, University of Duhok, Kurdistan region – Iraq.

**Dept. of Mathematics, Faculty of Science, University of Zakho, Zakho, Kurdistan region – Iraq.

(Received: July 8, 2019; Accepted for Publication: November 28, 2019)

ABSTRACT

In this paper, a modification of Symmetric Rank One (SR1) is propounded on the grounds of Modifying gradient-difference vector which meets Quasi condition and positive definite conditions. The new method is compared with the standard test results of the SR1 algorithm. In general, the modified method is more superior and efficient when compared to the standard Quasi-Newton method

KEY WORDS: SR1 method, Quasi-Newton method, Neural Network, optimization.

1. INTRODUCTION

Artificial Neural Networks (ANNs) are greatly applied in various areas of science, in pattern classification, function approximation, optimization, pattern matching and associative memories [2, 6]. Though the success and the promise of artificial neural networks in addressing practical problems, their design procedure still requires trial-and-The error. Attainment of the optimal structure of artificial neural networks for a problem is one of the typical problems in neural network design. Back propagation (BP) learning can recognize the training of feed-forward multilayer neural network. The algorithm primarily revises neural network weights pursuant to the gradient descent methods to decrease errors.

The ANN is comprised of a set of processing factors known as neurons or nodes which are interconnected with each other [1]. Usually, the node transfer function is a nonlinear function suchlike a sigmoid function, a Gaussian function, etc. In this study, sigmoid function is employed.

The optimization aim is to minimize the objective function by optimizing the network weight. E(w) means that a square function is selected as a sum error function[10].

$$E(w) = \frac{1}{2}\sum_{r=1}^{m} \sum_{l=1}^{p}\left(O_l^p - t_r^m\right)^2 \qquad (1.1)$$

where w is weight vector at each iteration; O_l^p and t_r^m represent respectively the actual and predicted value. For effectively training neural networks in scientific and engineering, SR1 algorithm is considered one of the most competitive formulations among the Quasi-Newton methods which is known as one of the most efficient manners to solve nonlinear unconstrained or bound constrained optimization problems. These methods are mostly utilized when the second derivative matrix of the objective function is either unavailable or too pricey to compute, they allow. Therefore, the curvature of the problem to be exploited in the numerical algorithm, despite the fact that only first derivatives (gradients) and function values are required see [4, 5, 11, 12].

The Symmetric rank one produce the sequence of weight $\{w_j\}$, is given by

$$w_{j+1} = w_j + \delta_j d_j \qquad (1.2)$$

Where the search direction is and satisfies the descent condition

$$d_j = -H_j \nabla E(w_j) \quad j > 0 \qquad (1.3)$$

H_j is usually required to be positive definite to assure a descent direction for E . H_j is upgraded at each iteration using gradient vector, and $\delta_j > 0$ is learning rate in machine learning. The H_j is defined using sequences of vectors s_j and y_j, which are given as

$$s_j = w_{j+1} - w_j \text{ and } y_j = \nabla E(w_{j+1}) - \nabla E(w_j) \qquad (1.4)$$

hasan.hazim@uod.ac

The SR1 is the unique rank-one in the class of Quasi Newton update satisfying Quasi-Newton condition:

$H_{j+1}y_j = s_j$ (1.5)

where $H_{j+1} = H_j + \frac{(s_j - H_j y_j)(s_j - H_j y_j)^T}{y_j^T (s_j - H_j y_j)}$, (1.6)

where y_j and s_j are defined in (1.4) and $y_j^T\left(s_j - H_j y_j\right) \neq 0$ for each j, see [4,7].

Proving the positive definite condition of Quasi-Newton method, SR1, usually needs that the step size of δ_j achieves the following Wolfe Conditions [13].

$E(w_j + \delta_j d_j) \leq E\left(w_j\right) + \sigma_1 \delta_j \nabla E\left(w_j\right)^T d_j$ (1.7)

$\left|\nabla E\left(w_j + \delta_j d_j\right)^T d_j\right| \leq \sigma_2 |\nabla E_j^T d_j|$ (1.8)

where $0 < \sigma_1 < \sigma_2 < 1$.

So, the standard Wolfe condition is (1.7) and

$\nabla E\left(w_j + \delta_j d_j\right)^T d_j \geq \sigma_2 \nabla E_j^T d_j$ (1.9)

This study is outlined as follows: in section one, we present an introduction to neural network. Section Two sheds light on SR1 modification whereas the third one deals with the proof of quasi and positive definite condition which verifies the SR1 modification. The forth section illustrates the numerical results which are compared with standard test results.

2. SYMMETRIC RANK ONE MODIFICATION

In this section, modified SR1 is suggested for modifying SR1 by the usage of gradient-difference vector given in (1.6)

$\bar{y_j} = y_j + (1-\theta)\left(G_j s_j - y_j\right)$ where $0 < \theta < 1$ (2.1)

For more details see [8].

In a bid to use the Hessian in H_j Andrei in [9] suggested a nonlinear conjugate gradient algorithm in which the Hessian/vector product $\nabla^2 E\left(w_{j+1}\right)s_j$ is approximated by finite differences:

$\bar{y_j} = y_j + (1-\theta)\left(\frac{y_j}{\sigma} - y_j\right)$ (2.2)

where $\sigma = \frac{2\sqrt{\epsilon_m}(1+\|w_{j+1}\|)}{\|s_j\|}$, and ϵ_m is error machine used for accuracy which is the smallest positive < 1.

Thus

$H_{j+1}^{new} = H_j + \frac{(s_j - H_j \bar{y}_j)(s_j - H_j \bar{y}_j)^T}{\bar{y}_j{}^T (s_j - H_j \bar{y}_j)}$ (2.3)

Algorithm 1.

Step (1) Let w_0, an initial point, be given as well as an identity nxn symmetric positive definite H_0 , ε is a termination scalar and set $j = 0$.

Step (2) compute $d_j = -H_j g_j$ where $g_j = \nabla E\left(w_j\right)$.

Step (3) Calculate δ_j to minimize $E(w_j + \delta_j d_j)$.

Step (4) find new point of weight from (1.2)

Step (5) If $\|g_{j+1}\| < \varepsilon$ then $w^* = w_j$ then stop

Else find s_j from $s_j = w_{j+1} - w_j$ go to step (5)

Step (6) Evaluate the hessian matrix by using (2.2) and (2.3).

Step (7) set $j = j + 1$ go to step 2.

3. QUASI AND POSITIVE DEFINITE CONDITIONS

In this section, we have proved that the modification of SR1 is satisfying Quasi and positive definite conditions.

Theorem3.1 If the new algorithm is applied to the quadratic with Hessian $G = G^T$, then

$H_{j+1}\bar{y}_j = s_j$, $j > 0$. (3.1)

Proof: Multiplying both sides of (2.3) by $\bar{y}_j$ from right, we have:

$H_{j+1}^{new}\bar{y}_j = H_j\bar{y}_j + \frac{(s_j - H_j\bar{y_j})(s_j - H_j\bar{y_j})^T}{\bar{y}_j{}^T(s_j - H_j\bar{y_j})}\bar{y}_j$ (3.2)

It is clear that $(s_j - H_j\bar{y_j})^T\bar{y}_j$ is scalar and $\bar{y}_j{}^T(s_j - H_j\bar{y_j})$ is also scalar

$\therefore (s_j - H_j\bar{y}_j)^T\bar{y_j} = \bar{y}_j{}^T(s_j - H_j\bar{y_j})$ (3.3)

Therefore we have

$H_{j+1}^{new}\bar{y}_j = H_j\bar{y}_j + \left(s_j - H_j\bar{y}_j\right)$ (3.4)

$H_{j+1}^{new}\bar{y_j} = s_j$ ■

Theorem3.2 If H_j is a positive definite, then the matrix H_{j+1} generated by the H_{j+1}^{new} algorithm is also positive definite.

Proof: Multiplying both sides of (2.3) by $\bar{y}_j$ from right and by $\bar{y}_j{}^T$ from left, we have

$$\bar{y}_j^T H_{j+1}^{new} \bar{y}_j = \bar{y}_j^T H_j \bar{y}_j + \frac{\bar{y}_j^T (s_j - H_j\bar{y}_j)(s_j - H_j\bar{y}_j)^T \bar{y}_j}{\bar{y}_j^T (s_j - H_j\bar{y}_j)}$$

So

$$\bar{y}_j^T H_{j+1}^{new} \bar{y}_j = s_j^T \bar{y}_j \qquad (3.5)$$

By substituting (2.2) in (3.5) we get

$\bar{y}_j^T H_{j+1} \bar{y}_j = s_j^T [(y_j + (1-\theta)\left(\frac{y_j}{\sigma} - y_j\right)]$, where $0 < \theta < 1$.

$$= s_j^T y_j \left[1 + (1-\theta)\left(\frac{1}{\sigma} - 1\right)\right]$$

Suppose that $k = \left[1 + (1-\theta)\left(\frac{1}{\sigma} - 1\right)\right]$

Because σ is between 0 & 1 so k will always be greater than zero.

$$s_j^T y_j = s_j^T \left(\nabla E(w_{j+1}) - \nabla E(w_j)\right)$$
$$= s_j^T \nabla E(w_{j+1}) - s_j^T \nabla E(w_j)$$
$$= \alpha_j d_j^T \nabla E(w_{j+1}) + \alpha_j \nabla E(w_j)^T H_j \nabla E(w_j)$$

By using Wolfe condition in (1.9)

$$s_j^T y_j \geq \alpha_j \sigma_2 d_j^T \nabla E(w_j) + \alpha_j \nabla E(w_j)^T H_j \nabla E(w_j)$$
$$= -\alpha_j \sigma_2 \nabla E(w_j)^T H_j \nabla E(w_j) + \alpha_j \nabla E(w_j)^T H_j \nabla E(w_j), \textit{ where } 0 < \sigma_2 < 1$$
$$= (1-\sigma_2)\alpha_j \nabla E(w_j)^T H_j \nabla E(w_j)$$

Since $0 < \sigma_2 < 1$, and H_j is positive definite

Then, $(1-\sigma_2)\alpha_j \nabla E(w_j)^T H_j \nabla E(w_j) > 0$

$s_j^T y_j > 0$.

$$\therefore\ s_j^T y_j \left[1 + (1-\theta)\left(\frac{1}{\sigma.} - 1\right)\right] > 0.$$

So $\bar{y}_j^T H_{j+1} \bar{y}_j \geq 0$ ■

4.1 Numerical Results

This section is devoted to testing the implementation of the modified methods. The modified method is compared to the standard SR1. The results given in Table 1 specifically quote the NOI and NOF. Table 1 shows that the modified SR1 method is superior to standard (SR1) method with respect to NOI and NOF. Furthermore, the modified SR1 algorithms and the standard SR1 algorithms are compared when the input $p = [0.1\ 0.1]$ and target $t = [1\ 1]$. The target error has been set to 0.01 and the maximum epochs to 3000. The numerical results can be seen in Table 3.

Table (1): Comparison between Modified and Standard SR1 Algorithm

Test Function	**N**	**Standard formula SR1**		**MODIFIED SR1**	
		NOI	NOF	NOI	NOF
G-Central	4	36	253	15	90
	100	43	331	29	215
	500	60	496	32	247
	1000	66	554	49	411
	5000	72	616	63	549
Miler	4	34	329	26	105
	100	47	182999	43	938
	500	53	183098	51	170
	1000	53	183098	48	162
	5000	65	189123	57	203
	4	31	90	30	81
	100	32	94	30	81
Rosen	500	33	98	30	81
	1000	37	115	30	80
	5000	37	120	31	84
	4	11	24	11	24
	100	44	89	44	89
GWolfe	500	47	95	47	95
	1000	50	101	49	99
	5000	106	294	105	212
Cubic	4	15	48	12	34
	100	16	66	16	46
	500	16	51	16	46
	1000	16	55	16	46

	5000	16	50	16	46
Gpowell3	4	14	35	13	31
	100	15	37	14	33
	500	15	37	15	35
	1000	15	37	15	35
	5000	16	40	15	35
sum	4	3	11	3	11
	100	14	83	14	80
	500	21	119	21	118
	1000	23	123	23	121
	5000	38	176	38	176
Total	-	1210	742985	1067	4909

Table (2): The Rate of Improvement between Modified Algorithm and Standard SR1 algorithm.

zTools	SR1	Modified SR1
NOI	100%	88.1818
NOF	100%	0.6607

The above table illustrates the rate of improvement in the modified standard SR1 algorithm. The numerical results of the new algorithm are better than the standard algorithm. As noted, the number of iterations and the number of function evaluations of the standard algorithm are about 100%. In other words, the new algorithm has improvement as compared to standard algorithm with 11.8182% in NOI and 99.3393% in NOF when $\theta \in (0,1)$.

Table (3): Comparing the Performance of Modified Algorithm with Standard SR1 Algorithm.

Methods	No. Running	Epochs
SR1	1	1000
	2	1000
	3	833
	4	1000
	5	1000
Modified	1	677
	2	165
	3	268
	4	224
	5	1000

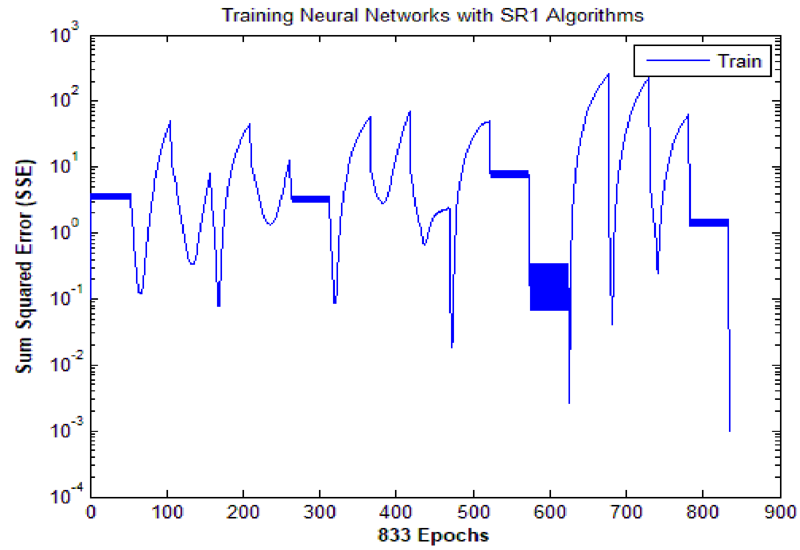

Fig.(1): Performance of standard SR1 algorithm for training neural networks.

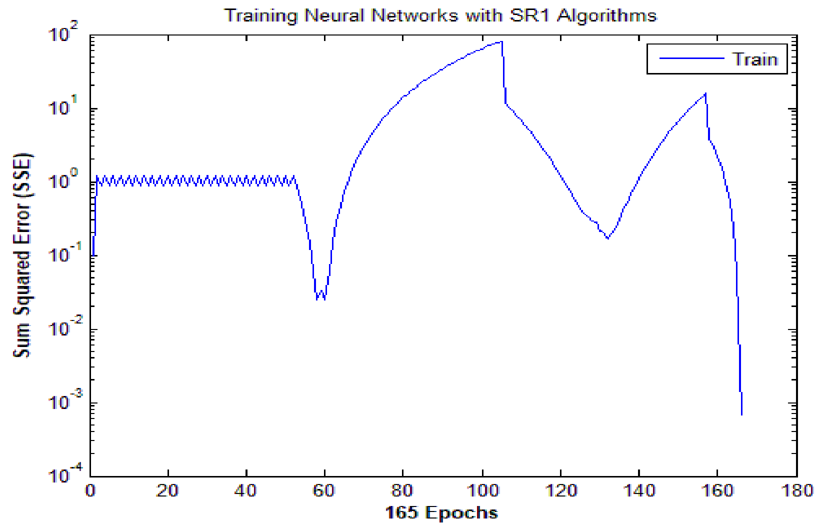

Fig. (2): Performance of modified SR1 algorithm for training neural networks.

4. CONCLUSION

This work, propounded a modification of SR1 by using gradient-difference vector. The quasi-newton condition and positive definite have been proved. In addition, the modification algorithms are used for training neural networks according to outcomes a modified method is more superior and effective than the standard SR1.

REFERENCES

X. Yao, **Evolving artificial neural networks**, Proceedings of the IEEE, 1999, vol. 87, no. 9, pp. 1423-1447,.

J. Dayhoff, An **Introduction to Neural Network** Architectures, New York: Van Nostrand Reinhold, 1990.

J. Nocedal and S. J. Wright. **Numerical Optimization**. Springer, New York, 2006, 2nd edition.

J.E. Dennis and J.J. Mor6, **Quasi-Newton methods, motivation and theory**, SIAM Review 1977, 19, 46-89.

J.E. Dennis and R.B. Schnabel, **Numerical Methods for Unconstrained Optimization and Nonlinear Equations,** Prentice-Hall, Englewood Cliffs, NJ, 1983, 4.

K. Mehrotra, C. K. Mohan, and S. Ranka, **Elements of Artificial Neural Networks**, Cambridge, MA: MIT Press, 1997.

L. Bottou, F. Curtis, and J. Nocedal. **Optimization methods for large-scale machine learning,** SIAM Review, 2018, 60(2):223311.

M. Al-Baali and H. Khalfan, **An Overview of Some Practical QuasiNewton Methods** for Unconstrained Optimization, SQU Journal For Science, 2007, 12 (2) 199-209.

N. Andrei, **Accelerated conjugate gradient algorithm with finite difference Hessian/vector product approximation** for unconstrained optimization, J. Comput. Appl. Math, 2009, 230, no. 2, 570–582.

Ngoc Tam Bui and Hiroshi Hasegawa, **Training Artificial Neural Network Using Modification of Differential Evolution Algorithm**, International Journal of Machine Learning and Computing, February 2015, Vol. 5, No. 1.

P.E. Gill, W. Murray and M.H. **Wright, Practical Optimization** Academic Press, New York, 1981, 8.

R. Fletcher, **Practical Methods of Optimization**: Unconstrained Optimization, Wiley, Chichester, 1980.

Stephen J. Wright and Jorge Nocedal, **numerical optimization**. Springer new york 2006.