The Third AFSS(1998), 118-121

REVISING THE TRADITIONAL BACKPROPAGATION WITH THE METHOD OF VARIABLE METRIC(QUASI - NEWTON) AND APPROXIMATING A STEP SIZE

Sang Woong CHOE [a], Jin Choon LEE [b]

a. Dept. of Management Information System, Pohang Junior College
Hunghae-Up, Pohang, Kyungpook, 791-940, Korea
Tel: +82-562-45-1198

b. Dept. of Industrial Engineering, Kyungil University
Hayang, Kyungsan, Kyungpook, 712-701, Korea
Tel: +82-53-850-7266, Fax: +82-53-850-7275, E-mail: jinclee@bear.kyungil.ac.kr

Abstract

In this paper, we propose another paradigm(QNBP) to be capable of overcoming limitations of the traditional backpropagation(SDBP). QNBP is based on the method of Quasi-Newton(variable metric) with the normalized direction vectors and computes step size through the linear search. Simulation results showed that QNBP was definitely superior to both the stochastic SDBP and the deterministic SDBP in terms of accuracy and rate of convergence and might surmount the problem of local minima. And there was no difference between DFP+SR1 and BFGS+SR1 combined algorithms in QNBP.

Keywords : Quasi-Newton(variable metric) method, combined algorithm, DFP+SR1, BFGS+SR1, normalized direction vector

1. Introduction

The traditional backpropagation method based on the steepest descent method(here-after referred to as the SDBP) is not good learning algorithm in terms of accuracy and rate of convergence. Especially, the most serious problem in SDBP training is the existence of local minima, where the error at the network outputs may still be unacceptably high. It is undeniable that most researchers have been inclined to blindly accept this method for many subsequent years.

In this paper, we propose another paradigm to be capable of overcoming limitations of SDBP. The proposed backpropagation (here-after referred to as the QNBP) is based on the combined Quasi-Newton method(or variable metric method) with the approximate to step sizes. The term "combined" implys that two rank-two updates in Quasi-Newton methods can be combined with the symmetric rank-one update.

To use the Newton-Raphson method we must know the Hessian matrix of the function and we must invert it. In many cases including backpropagation model, the Hessian matrices are not available, are very expensive to compute. The variable metric method, originally developed by Davidon[1], avoids these difficulties.

At current point $x_k \in R^n$, arbitrary function $f(x)$ can be locally approximated by the quadratic form of Eq. (1) using Taylor series

$$f(x_k) \simeq b + a^T x_k + \frac{1}{2} x_k^T H(x_k) x_k \,,\; H(x_k) > 0 \qquad (1)$$

where $H(x_k)$ is the Hessian matrix of $f(x_k)$.

$f(x_k)$ has the minimum point $x_{k+1} = -H(x_k)^{-1}a$. From the gradient vector g_k, x_{k+1} can be computed in one step as

$$\begin{aligned} x_{k+1} &= -H(x_k)^{-1}a = -H(x_k)^{-1}(g_k - H(x_k)x_k) \\ &= -H(x_k)^{-1}g_k + x_k \end{aligned} \qquad (2)$$

if $H(x_k)^{-1}$ is available.

The optimal direction of descent is along $-H(x_k)^{-1}g_k$ and not along $-g_k$. And $H(x_k)^{-1}g_k$ is the gradient of $f(x_k)$ when the metric is given by $\sqrt{x_k^T H(x_k) x_k}$. In most cases, $H(x_k)$ is not known. Therefore, this method estimates $H(x_k)^{-1}$ successively by constructing a sequence of matrices such that they are the inverse of $H(x_k)$. Hence, the name of the variable metric method or Quasi-Newton method.

The "Quasi" in Quasi-Newton method is because we don't use the actual Hessian matrix, but instead use current approximation of it. This is often better than using the true Hessian. In short, the basic idea of the Quasi-Newton methods is to build up, iteratively, a good estimate to the inverse Hessian matrix.

Quasi-Newton methods come in three main flavors; two rank-two algorithms such as Davidon - Fletcher - Powell(DFP) algorithm[2], Broyden - Fletcher - Goldfarb - Shanno(BFGS) algorithm[3,4,5,6] and symmetric rank-one algorithm(SR1)[7]. This three versions differ mainly in choice of an estimate to the inverse Hessian matrix. The main drawback of SR1 update is that an estimate to inverse Hessian at next point may not be positive definite even when an estimate to inverse Hessian at current point is positive definite.

However, the main advantage of SR1 update is that it requires less amount of storage space and less amount of arithmetic operations in

comparison to two rank-two updates such as DFP and BFGS updates. Hence, if the fatal flaw of SR1 update could be overcome, this update is the most adequate to the backpropagation model.

In this paper, two combined updates, DFP+SR1 and BFGS+SR1, are applied to the backpropagation model. So, we intend to call this another paradigm QNBP.

2. QNBP : Algorithm and step size

First, to describe QNBP algorithm, we introduce the following notation.

[1] r -- * layer : input layer(0), hidden layer($1\sim(L-1)$), and outputlayer(L).

[2] p, k -- * the p_th training data, the k_th iteration. : $1 \le p \le P,\ k \ge 0$

[3] N_r -- * number of the $r-$layer nodes except bias term.
n_r -- * the n_r_th $r-$layer node. : $0 \le n_r \le N_r\ (r \ne L)$ and bias term, if $n_r = N_r$.

[4] x_{pn_0} -- * value of the p_th training data to the n_0_th input layer node.
y_{pn_L} -- * value of the p_th training data to the n_L_th output layer node.

[5] $w^r_{n,n_{r-1}},\ d^r_{n,n_{r-1}}$ -- * weight and direction on the connection from the n_{r-1}_th $(r-1)-$layer node to the n_r_th $r-$layer node. : $n_r \ne N_r$

[6] $net^r_{pn_r},\ f^r_{n_r}(net^r_{pn_r})$ -- * net value and output value of the n_r_th $r-$layer node for the p_th training data.

[7] $f^0_{n_0}(net^0_{pn_0}) = net^0_{pn_0} = x_{pn_0}$ -- * $n_0 \ne N_0$

$f^r_{n_r}(net^r_{pn_r}) = f^r_{n_r}(0) = 1$ -- * $n_r = N_r$

$$f^r_{n_r}(net^r_{pn_r}) = f^r_{n_r}\Big(\sum_{n_{r-1}} w^r_{n,n_{r-1}} f^{r-1}_{n_{r-1}}(net^{r-1}_{pn_{r-1}})\Big) = f^r_{n_r}\Big(\sum_{n_{r-1}\ne N_{r-1}} (w^r_{n,n_{r-1}} f^{r-1}_{n_{r-1}}(net^{r-1}_{pn_{r-1}})) + w^r_{n,N_{r-1}}\Big)$$

-- * $r \ne 0,\ n_r \ne N_r$

[8] $$g^r_{n,n_{r-1}} = \frac{\partial E(\cdot)}{\partial w^r_{n,n_{r-1}}} = -\sum_p \delta^r_{pn_r} f^{r-1}_{n_{r-1}}(net^{r-1}_{pn_{r-1}})$$

$$\delta^r_{pn_r} = \frac{df^r_{n_r}(net^r_{pn_r})}{dnet^r_{pn_r}} \sum_{n_{r+1}} \delta^{r+1}_{pn_{r+1}} w^{r+1}_{n_{r+1}n_r},\ r \ne 0, L \text{ and}$$

$$\delta^L_{pn_L} = \frac{df^L_{n_L}(net^L_{pn_L})}{dnet^L_{pn_L}} (y_{pn_L} - f^L_{n_L}(net^L_{pn_L}))$$

[9] $W^r = [w^r_{n,n_{r-1}}]_{N_r \times (N_{r-1}+1)}$, $D^r = [d^r_{n,n_{r-1}}]_{N_r \times (N_{r-1}+1)}$,
$G^r = [g^r_{n,n_{r-1}}]_{N_r \times (N_{r-1}+1)}$.
-- * $r-$layer weight, direction and gradient matrix : $r \ne 0,\ n_r \ne N_r$

[10] $Vec(A=[a_{ij}]_{I\times J}) = [b_h]_{IJ\times 1}$ -- * $h = jI + i$, where j is a maximum integer less than or equal to $\frac{h}{I}$ and $0 \le h \le (IJ-1),\ 0 \le i \le (I-1),\ 0 \le j \le (J-1)$.

[11] $$E(\theta) = E(Vec(W^1), Vec(W^2), \ldots, Vec(W^L)) = \frac{1}{2}\sum_p\sum_{n_L}[y_{pn_L} - f^L_{n_L}(\sum_{n_{L-1}} w^L_{n_Ln_{L-1}} f^{L-1}_{n_{L-1}}(\sum_{n_{L-2}} w^{L-1}_{n_{L-1}n_{L-2}} f^{L-2}_{n_{L-2}}(\ldots\ldots w^3_{n_3n_2} f^2_{n_2}(\sum_{n_1} w^2_{n_2n_1} f^1_{n_1}(\sum_{n_0} w^1_{n_1n_0} f^0_{n_0}(net^0_{pn_0})))\ldots\ldots)))]^2$$

-- * objective(error) function of the network. : $w^r_{N,n_{r-1}} = 0\ (r \ne 0, L)$

[12] $$\nabla E(\theta) = \rho = \Big[\frac{\partial E(\theta)}{\partial Vec(W^r)}\Big]_{L\times 1},\ \frac{\partial E(\theta)}{\partial Vec(W^r)} = Vec(G^r)$$

-- * gradient vector of $E(\theta)$.

[13] $$H(\theta) = [H_{ab}]_{L\times L},\ H_{ab} = \frac{\partial^2 E(\theta)}{\partial Vec(W^b)\,\partial Vec(W^a)} = \Big[\frac{\partial^2 E(\theta)}{\partial w^b_{n_bn_{b-1}} \partial w^a_{n_an_{a-1}}}\Big]_{p\times q}$$

-- * positive definite Hessian matrix. : $p = N_a\times(N_{a-1}+1),\ q = N_b\times(N_{b-1}+1)$

[14] $\|x\| = (x^Tx)^{1/2}$

[15] $$\theta^r = [Vec(W^1), \ldots, Vec(W^r), Vec(W^{r+1}), \ldots, Vec(W^L)]^T$$
$$\lambda^r = [Vec(D^1), \ldots, Vec(D^r), Vec(D^{r+1}), \ldots, Vec(D^L)]^T$$
$$\rho^r = [Vec(G^1), \ldots, Vec(G^r), Vec(G^{r+1}), \ldots, Vec(G^L)]^T$$
$$\sum_r \theta^r = \theta,\ \sum_r \lambda^r = \lambda,\ \sum_r \rho^r = \rho,\ r \ne 0$$

[16] $$\langle\theta^r\rangle^T = [Vec^T(W^1), \ldots, Vec^T(W^r), Vec^T(W^{r+1}), \ldots, Vec^T(W^L)]$$
$$\langle\lambda^r\rangle^T = [Vec^T(D^1), \ldots, Vec^T(D^r), Vec^T(D^{r+1}), \ldots, Vec^T(D^L)]$$
$$\langle\rho^r\rangle^T = [Vec^T(G^1), \ldots, Vec^T(G^r), Vec^T(G^{r+1}), \ldots, Vec^T(G^L)]$$
$$\sum_r \langle\theta^r\rangle^T = \theta^T,\ \sum_r \langle\lambda^r\rangle^T = \lambda^T,\ \sum_r \langle\rho^r\rangle^T = \rho^T,\ r \ne 0$$

2.1 Algorithm

Let A_k be an estimate of $H(x_k)^{-1}$ at current point θ_k. In QNBP, the next point θ_{k+1} is generated from θ_k by the linear search in the direction of λ_k and is expressed as

$$\theta_{k+1} = \theta_k + \tau_k\lambda_k,\ k \ge 0$$
$$\text{where } \lambda_k = \frac{1}{\|A_k\rho_k\|}(-A_k\rho_k) \tag{3}$$

τ_k is a step size and we select it such that

$$\text{Min}_{\tau_k} E(\theta_{k+1}) \cong \text{Min}_{\tau_k}\Big\{E(\theta_k) + \langle\rho_k\rangle^T\lambda_k\tau_k + \frac{1}{2}\langle\lambda_k\rangle^T H(\theta_k)\lambda_k(\tau_k)^2\Big\} \tag{4}$$

From Eq. (4),

$$\tau_k = \frac{-\langle\rho_k\rangle^T\lambda_k}{\langle\lambda_k\rangle^T H(\theta_K)\lambda_k} > 0 \tag{5}$$

Eq. (5) can be written as

$$\tau_k \cong \frac{2(E(\theta_{k+1}) - E(\theta_k))}{\langle \rho_{k+1} \rangle^T \lambda_k + \langle \rho_k \rangle^T \lambda_k} \quad (6)$$

using Taylor series and $\frac{1}{\tau_{k-1}}(\rho_k - \rho_{k-1}) \cong H(\theta_{k-1})\lambda_{k-1}$.

One important feature of QNBP is the choice of matrices A_k to be positive definite and satisfying the following Quasi-Newton equation

$$A_{k+1} p_k = q_k$$

where $p_k = \rho_{k+1} - \rho_k$ and $q_k = \theta_{k+1} - \theta_k$. (7)

QNBP with DFP, BFGS and SR1 updates generates the sequence of $A_k (k \geq 1)$ respectively as

$$A_{k+1} = A_k + \frac{q_k q_k^T}{q_k^T p_k} - \frac{A_k p_k p_k^T A_k}{p_k^T A_k p_k} \quad (8)$$

$$A_{k+1} = A_k + \frac{1}{q_k^T p_k}\left(\left(1 + \frac{p_k^T A_k p_k}{q_k^T p_k}\right) q_k q_k^T - q_k p_k^T A_k - A_k q_k p_k^T\right) \quad (9)$$

$$A_{k+1} = A_k + \frac{(q_k - A_k p_k)(q_k - A_k p_k)^T}{(q_k - A_k p_k)^T p_k} \quad (10)$$

First combined algorithm, DFP+SR1, computes A_{k+1} as follows

$$\begin{aligned} A_{k+1} &= \text{Eq.(10)} \quad \text{if } (q_k - A_k p_k)^T p_k > 0 \\ &= \text{Eq.(8)} \quad \text{if } (q_k - A_k p_k)^T p_k < 0 \text{ and } q_k^T p_k > 0 \\ &= sI \quad \text{otherwise} \end{aligned} \quad (11)$$

where I is identity matrix and s is a given positive constant.

Second combined algorithm, BFGS+SR1, computes A_{k+1} as follows

$$\begin{aligned} A_{k+1} &= \text{Eq.(10)} \quad \text{if } (q_k - A_k p_k)^T p_k > 0 \\ &= \text{Eq.(9)} \quad \text{if } (q_k - A_k p_k)^T p_k < 0 \text{ and } q_k^T p_k > 0 \\ &= sI \quad \text{otherwise} \end{aligned} \quad (12)$$

And $A_0 = I$

Note that Eq. (11) and (12) check whether the sequence of $A_k (k \geq 1)$ are positive definite. Thus we are led to consider an algorithm, that is, QNBP which have two versions, DFP+SR1 and BFGS+SR1.

2.2 Step size

Eq. (6) may not be a directly available expression since we must estimate ρ_{k+1} and $E(\theta_{k+1})$. For this reason, we will grant two desirable properties to τ_k with the purpose of a successful training.

We can analytically derive the unique τ_k which satisfies two properties and then use this as the good approximate to a true τ_k . These two desirable properties are as follows.

[Property 1] θ_{k+1} lies in a given direction λ_k from θ_k .

[Property 2] $E(\theta_k)$ decrease log-linearly.

In short, Eq. (6) satisfies the above-mentioned properties, it can be written as

$$\tau_k \cong \frac{2(E_1 - E_0)\left(\frac{E_1}{E_0}\right)^k}{\langle \rho_k \rangle^T \lambda_k} < U_b \quad (13)$$

where U_b is upper bound.

3. Simulation example

(See Table 2)

3.1 First case : XOR problem
(See Tables 3, 4)

3.2 Second case : Evaluation of Rosenbrock function value
(See Tables 3, 5)

[Table 2] Input parameters

Simulation	SDBP				QNBP	
	Step Size	Momentum -Rate	Initial Temperature	Type	U_b	s
1	1.0	0.5	0	d*	1.0	1.0
2	1.0	0.5	0.001	s**	0.5	1.0
3	0.5	0.25	0	d	0.3	1.5
4	0.5	0.25	0.001	s	0.2	2.0
5	0.3	0.15	0	d	0.15	2.0
6	0.3	0.15	0.001	s	0.1	2.0
7	0.2	0.1	0	d		
8	0.2	0.1	0.001	s		

* : deterministic ** : stochastic

[Table 3] Simulation environment in two cases

	Input Nodes	First Hidden Nodes	Output Nodes	Output Function	Error Level	Max. Iteration	Initial Weights	Training Data
1_st Case	2	2	1	sigmoid	1E-12	3000	n*	4
2_nd Case	14	10	7	sigmoid	1E-12	5000	n	20

* : normalized

[Table 4] Simulation results of 1_st case, XOR problem

Simulation	SDBP			QNBP (DFP+SR1 AND BFGS+SR1)		
	Run Time	Iterations	Error	Run Time	Iterations	Error
1	0.77	3000	0.00050596986	0.33	500	0.000000000000
2	0.77	3000	0.00051034391	0.27	400	0.000000000000
3	0.77	3000	0.00224905702	0.33	500	0.000000000000
4	0.77	3000	0.00223123949	0.33	500	0.000000000000
5	0.77	3000	0.00853072653	0.33	500	0.000000000000
6	0.77	3000	0.00825091817	0.49	700	0.000000000000
7	0.77	3000	0.09576089606			
8	0.77	3000	0.08420768138			

[Table 5] Simulation results of 2_nd case, Rosenbrock function evaluation

Simulation	SDBP			QNBP (DFP+SR1 AND BFGS+SR1)		
	Run Time	Iterations	Error	Run Time	Iterations	Error
1	61.48	5000	0.00485982260	133.0	1000	0.000000000001
2	61.48	5000	0.00144454306	292.5	2200	0.000000000001
3	61.48	5000	0.00450409402	412.0	3100	0.000000000001
4	61.48	5000	0.00462669594	465.4	3500	0.000000000001
5	61.48	5000	0.00764988971	531.21	4000	0.000000000001
6	61.48	5000	0.00752127151	611.76	4600	0.000000000001
7	61.48	5000	0.01256565752			
8	61.48	5000	0.01190128557			

4. Concluing remarks

From simulation results, compared with SDBP, QNBP has three major advantages. These are as follows.

① QNBP is by far general, i.e., SDBP is a special case of QNBP ($A_k = I \ \forall k$)

② QNBP is definitely superior to SDBP in terms of accuracy and rate of convergence.

③ QNBP may surmount the problem of local minima.

And

① In large scale problem, QNBP is time-consuming.

② There was no difference between DFP+SR1 and BFGS+SR1 combined algorithms in QNBP.

References

[1] W.C. Davidon, "Variable Metric Method for Minimization", Argonne National Laboratory Report ANL-5990, Argonne, Illinois, 1959.

[2] R. Fletcher and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization", Computer Journal, 6, 163-168, 1963.

[3] C.G. Broyden, "The Convergence of a Class of Double-Rank Minimization Algorithms", Journal Inst. Maths. Applics., 6, 76-90, 1970.

[4] R. Fletcher, "A New Approach to Variable Metric Algorithm", Computer Jouranal, 13, 392-399, 1970.

[5] D. Goldfarb, "A Family of Variable Metric Methods Derived by Variational Means", Mathematics of Computations 24, 23-26, 1970.

[6] D.F. Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization", Mathematics of Computations 24, 647-657, 1970.

[7] W.C. Davidon, "Variance Algorithm for Minimization", Computer Journal, 10, 406-410, 1968.