Journal of Information & Computational Science 9: 7 (2012) 1921–1929
Available at http://www.joics.com

A Limited Memory Quasi-Newton Method for Large Scale Unconstrained Optimization ⋆

Xiaowei Jiang, Yueting Yang*, Yunlong Lu

School of Mathematics & Institute of Applied Mathematics, Beihua University, Jilin 132013, China

Abstract

We present a new expression for symmetric rank-one formula, and advance a limited memory method based on it. We give a two-loop recursive formula that takes advantage of the symmetry of the update matrices to compute the search direction efficiently. The numerical tests show that the method is efficient for large scale problems.

Keywords: Unconstrained Optimization; Large Scale Problem; Limited Memory Method; SR1 Update; Two-loop Recursion

1 Introduction

We consider the unconstrained optimization problem

$$\min \ \ f(x), \tag{1}$$

where $f: \ R^n \to R$ is a smooth nonlinear function, the number of variables n is large, and analytic expressions for the function f and the gradient g are available.

Different methods have been proposed to solve the above problem, such as: (i) Newton's method and variations of it [1, 2]; (ii) The partitioned Quasi-Newton methods [3]; (iii) The (preconditioning)conjugate gradient methods [4]; (iv) Limited memory Quasi-Newton methods [5], the last of which has attracted much attention for its advantage. The method can be seen as extension of the conjugate gradient method, in which additional storage is used to accelerate convergence. They are suitable for large scale problems because the amount of storage required by the algorithms can be controlled by the user. Alternatively, limited memory methods can be viewed as implementations of Quasi-Newton methods, in which storage is restricted. Their simplicity is one of their main appeals: they do not require knowledge of the sparsity structure of the Hessian, or knowledge of the separability of the objective function, and they are very simple to program.

⋆Project supported by the National Nature Science Foundation of China (No. 11171003) and Key Project of Chinese Ministry of Education (No. 211039).

*Corresponding author.
Email address: jxwbh@126.com, yueting_yang@126.com (Yueting Yang).

1548–7741 / Copyright © 2012 Binary Information Press
July 2012

Among the limited memory Quasi-Newton methods, many are concentrated on the limited memory BFGS methods [5, 6, 7, 8], while a famous example is the limited memory BFGS method (L-BFGS) presented by Nocedal [10]. It is a variation of the standard BFGS method, which is given by

$$x_{k+1} = x_k - \lambda_k H_k g_k \quad k = 0, 1, 2, \cdots \tag{2}$$

where λ_k is a step-length, g_k is the gradient of the objective function f at iterate x_k, and the inverse Hessian approximation H_k is updated at every iteration by the formula

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{3}$$

where

$$\rho_k = 1/y_k^T s_k, V_k = I - \rho_k y_k s_k^T, \tag{4}$$

and

$$s_k = x_{k+1} - x_k, y_k = g_{k+1} - g_k.$$

The limited memory BFGS method is an adaption of the BFGS method to large scale problems. The implementation is almost identical to that of the standard BFGS method—the only difference is in the matrix update. Instead of storing the matrices H_k, one stores a certain number, say m, of pairs $\{s_i, y_i\}$ that define them implicitly. The product $H_k g_k$ is obtained by performing a sequence of inner products involving g_k and the latest m vector pairs $\{s_i, y_i\}$. After computing the new iterate, the oldest pair is deleted from the set $\{s_i, y_i\}$, and replaced by the newest one. The algorithm therefore always keeps the latest m pairs $\{s_i, y_i\}$ to define the iteration matrix. Let $\hat{m} = min\{k, m-1\}$, and from (3) we see that H_k can be written as

$$\begin{aligned} H_{k+1} \;=\; & V_k^T V_{k-1}^T \cdots V_{k-\hat{m}+1}^T H_0 V_{k-\hat{m}+1} \cdots V_{k-1} V_k \\ & + V_k^T \cdots V_{k-\hat{m}+2}^T \rho_{k-\hat{m}+1} s_{k-\hat{m}+1} s_{k-\hat{m}+1}^T V_{k-\hat{m}+2} \cdots V_k \\ & \cdots \\ & + V_k^T \rho_{k-1} s_{k-1} s_{k-1}^T V_k \\ & + \rho_k s_k s_k^T. \end{aligned} \tag{5}$$

The results of numerical tests show that L-BFGS method is very competitive for large scale problems [10].

We also see that limited memory Quasi-Newton methods taken advantage of SR1 update are rather fewer. A major cause is that the SR1 Quasi-Newton method is numerical instable due to the facts that the denominator of the update matrix may be zero or nearly zero, and that updates may not preserve the positive definiteness. While some recent researches show that the sequence of Hessian approximations generated by the SR1 update converges to the actual Hessian at the solution [9], and the SR1 update with improvement techniques is more efficient in comparison with other updates such as the BFGS and DFP updates etc.. In this paper, we present a new expression for SR1 formula, and advance a limited memory SR1(L-SR1) method based on it.

The rest of this paper is organized as follows: we briefly gives the L-SR1 update formula and its properties is given in Section 2. An algorithm based on L-SR1 is presented in Section 3. Finally, in Section 4, numerical results are reported to show the feasibility and effectiveness of the new method.

2 The L-SR1 Update

It is well known that the SR1 update formula is

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}. \tag{6}$$

Denote $a_k = y_k^T H_k y_k$ and $b_k = s_k^T y_k$. Provided $b_k - a_k > 0$, (6) can be written in the product form

$$H_{k+1} = (I - u_k y_k^T) H_k (I - y_k u_k^T) + \rho_k s_k s_k^T, \tag{7}$$

where $u_k = \frac{1}{\sqrt{b_k(b_k - a_k)}} s_k + \frac{\sqrt{b_k - a_k} - \sqrt{b_k}}{a_k \sqrt{b_k - a_k}} H_k y_k$, $\rho_k = b_k^{-1}$. Define $W_k = I - y_k u_k^T$, we have

$$H_{k+1} = W_k^T H_k W_k + \rho_k s_k s_k^T. \tag{8}$$

In general, for $k < m$, we have the usual SR1 update

$$\begin{aligned} H_{k+1} \;=\; & W_k^T W_{k-1}^T \cdots W_1^T W_0^T H_0 W_0 W_1 \cdots W_{k-1} W_k \\ & + W_k^T W_{k-1}^T \cdots W_1^T \rho_0 s_0 s_0^T W_1 \cdots W_{k-1} W_k \\ & \cdots \\ & + W_k^T \rho_{k-1} s_{k-1} s_{k-1}^T W_k \\ & + \rho_k s_k s_k^T. \end{aligned} \tag{9}$$

For $k \geq m$, the special update

$$\begin{aligned} H_{k+1} \;=\; & W_k^T W_{k-1}^T \cdots W_{k-m+1}^T H_0 W_{k-m+1} \cdots W_{k-1} W_k \\ & + W_k^T \cdots W_{k-m+2}^T \rho_{k-m+1} s_{k-m+1} s_{k-m+1}^T W_{k-m+2} \cdots W_k \\ & \cdots \\ & + W_k^T \rho_{k-1} s_{k-1} s_{k-1}^T W_k \\ & + \rho_k s_k s_k^T. \end{aligned} \tag{10}$$

For the sake of convenience, the formulas (9) and (10) can be written as a unified form

$$\begin{aligned} H_{k+1} \;=\; & W_k^T W_{k-1}^T \cdots W_{k-\hat{m}+1}^T H_0 W_{k-\hat{m}+1} \cdots W_{k-1} W_k \\ & + W_k^T \cdots W_{k-\hat{m}+2}^T \rho_{k-\hat{m}+1} s_{k-\hat{m}+1} s_{k-\hat{m}+1}^T W_{k-\hat{m}+2} \cdots W_k \\ & \cdots \\ & + W_k^T \rho_{k-1} s_{k-1} s_{k-1}^T W_k \\ & + \rho_k s_k s_k^T \end{aligned} \tag{11}$$

for all k, where $\hat{m}$ is the same as the one in (5).

Similar to [11], we have the same properties as follows:

Theorem 1 *If H_0 is positive definite, then the matrices defined by* (9) *and* (10) *are positive definite(provided $b_k - a_k > 0$ for all k).*

Proof It is easy to see from (8).

Theorem 2 *Let f be a strictly convex quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$, and suppose that vectors s_k are conjugate, i.e.*

$$s_i^T A s_j = 0, (i \neq j) \tag{12}$$

then the matrices satisfy the Quasi-Newton equation

$$H_k y_j = s_j, \quad j = k-1, \cdots, k-m(for \quad k > m) \tag{13}$$

in the last m directions.

Proof Proceeding by induction (see [11]).

3 L-SR1 Method

In this section, we will advance a L-SR1 method based on (11). Note that where the update is well defined only if the denominator $(s_k - H_k y_k)^T y_k$(i.e. $b_k - a_k$) is nonzero. Also, the SR1 update does not have the property of hereditary positive definiteness in the case of $b_k - a_k < 0$, and the SR1 method is numerical instable when $|b_k - a_k|$ is too small, then we use L-BFGS update (5).

Algorithm 3.1 (L-SR1 method)

Step 1 Choose $x_0 \in R^n, m, 0 < \tau_1 < 1/2, \tau_1 < \tau_2 < 1, \epsilon, \eta$ and a symmetric and positive definite starting matrix H_0. Set $d_0 = -H_0 g_0, k = 0$.

Step 2 Select a step factor α_k such that

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \tau_1 \alpha_k g_k^T d_k,$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \tau_2 g_k^T d_k.$$

(We always try the step length $\alpha_k = 1$ first).

Step 3 Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 4 If $\|g_{k+1}\| < \epsilon$ or $f_k - f_{k+1} \leq \epsilon \max\{1.0, |f_k|\}$, then stop.

Step 5 Compute a search direction

$$d_{k+1} = -H_{k+1} g_{k+1}.$$

If $|b_k - a_k| > \eta$, we compute H_{k+1} according to (11); else we produce H_{k+1} according to (5).

Step 6 Set $k = k + 1$ and goto Step 2.

At Step 5, we must update H_k to get H_{k+1}, and compute the search direction $-H_{k+1} g_{k+1}$. Note that each matrix H_k is not formed explicitly. When updating H_k according to (11), we give an approach for handling limited memory matrices. The approach exploits the symmetry and structure of (11), giving rise to an efficient two-loop recursion for computing products $H \cdot g$ and $H \cdot y$ using the inverse Hessian approximation.

Algorithm 3.2 (The two-loop recursion for product $H \cdot v$)

1) $\bar{m} = \min\{k, m\}$
2) $q = v$
3) for $i = k-1, \cdots, k-\bar{m}$
   $a_i = u_i^T q$
   $q = q - a_i y_i$
   $p_i = q$
4) if $k \leq m$
   $r = H_0 q$
   else
   $r = H_k^{(0)} q$
5) for $i = k-\bar{m}, \cdots, k-1$
   $b_i = y_i^T r$
   $r = r - b_i u_i$
   if $i < k-1$
   $r = r + \rho_i s_i^T p_{i+1} s_i$
   else
   $r = r + \rho_i (s_i^T g_{i+1} s_i)$
6) $Hv = r$

Remark 3.1 Let v be g and y respectively, and we can derive the products $H \cdot g$ and $H \cdot y$.

Remark 3.2 When computing u, the product $H \cdot y$ must be used, so we save u in each iteration, which can be used in next iteration. Anyway, we also need save s, y, ρ in each iteration. We save them for the last k steps for $k \leq m$, but only the last m steps for $k > m$, when we denote $S = [s_{k-m}, \cdots, s_{k-1}], Y = [y_{k-m}, \cdots, y_{k-1}], \rho = [\rho_{k-m}, \cdots, \rho_{k-1}]$ and $U = [u_{k-m}, \cdots, u_{k-1}]$.

Remark 3.3 H_0 is usually the identity or a scalar multiple of the identity.

When updating H_k according to (5), we compute the search direction using the two-loop recursion described in [12].

4 Numerical Results

Some numerical results of the L-SR1 have been reported in this section. The test problems [13, 14] are listed in the Table 1.

Table 1: Test problems

No.	Problem name	No.	Problem name
1	Boundary value function	9	Penalty function I
2	Broyden tridiagonal function	10	Variable dimension function
3	Separable cubic function	11	Trigonometric function
4	Nearly separable function	12	Chebyquad function
5	Schittkowski function 302	13	Gould
6	Freudenstein and Roth function	14	Yang tridiagonal function
7	Extended Rosenbrock function	15	Conn
8	Extended Powell singular function	16	Tridia

Table 2: Numerical results (The lower dimensions)

No.	*dim*	Methods	CPU	NI	Nf	Ng	f^*
1	4	L-SR1	0.02	11	17	12	7.4314×10^{-9}
		L-BFGS	0.02	12	19	13	2.5985×10^{-7}
2	4	L-SR1	0.01	9	16	10	2.3074×10^{-8}
		L-BFGS	0.01	13	18	15	3.0216×10^{-8}
3	4	L-SR1	0.01	10	11	11	6.5173×10^{-10}
		L-BFGS	0.02	10	11	11	4.7041×10^{-8}
4	4	L-SR1	0.02	13	21	14	3.6681×10^{-6}
		L-BFGS	0.02	15	18	16	4.1527×10^{-5}
5	4	L-SR1	0.01	8	13	9	3.1254×10^{-15}
		L-BFGS	0.02	10	12	11	2.1346×10^{-10}
6	4	L-SR1	0.01	18	31	19	48.9843
		L-BFGS	0.02	19	21	20	48.9412
7	4	L-SR1	0.03	34	63	35	2.6499×10^{-9}
		L-BFGS	0.04	38	63	39	3.5016×10^{-9}
8	4	L-SR1	0.02	21	39	22	2.0183×10^{-5}
		L-BFGS	0.02	24	53	25	9.2000×10^{-5}
9	4	L-SR1	0.02	8	15	9	6.0757×10^{-5}
		L-BFGS	0.03	14	21	15	8.2731×10^{-4}
10	4	L-SR1	0.01	12	25	13	6.6868×10^{-19}
		L-BFGS	0.02	15	20	18	2.1536×10^{-15}
11	4	L-SR1	0.03	19	58	20	3.3007×10^{-4}
		L-BFGS	0.03	18	45	20	6.3645×10^{-5}
12	4	L-SR1	0.01	7	12	8	-3
		L-BFGS	0.02	9	14	10	-3
13	10	L-SR1	0.01	6	28	7	-8.2333
		L-BFGS	0.01	3	20	5	-8.8057
14	4	L-SR1	0.01	6	10	7	2.8652×10^{-14}
		L-BFGS	0.01	8	13	10	3.7414×10^{-12}
	10	L-SR1	0.01	7	12	8	4.9981×10^{-13}
		L-BFGS	0.01	10	27	15	4.9301×10^{-10}
15	4	L-SR1	0.01	6	8	7	1.3256×10^{-8}
		L-BFGS	0.01	5	7	6	3.1542×10^{-7}
	10	L-SR1	0.01	9	13	10	1.6810×10^{-8}
		L-BFGS	0.02	7	11	9	1.3964×10^{-7}
16	4	L-SR1	0.01	10	12	11	8.3245×10^{-11}
		L-BFGS	0.01	9	13	12	3.3452×10^{-10}
	10	L-SR1	0.02	15	18	16	1.0120×10^{-10}
		L-BFGS	0.03	14	20	17	2.4532×10^{-10}

All tests are done on a PC with 1.8GHz Pentium IV and 512 MB SDRAM using MATLAB6.5. The parameter values are $\tau_1 = 0.01, \tau_2 = 0.9, \epsilon = 10^{-8}$ and $\eta = 0.01$. The results of our numerical experiments are summarized in Tables 2 and 3, where the algorithms L-SR1 and L-BFGS ([5]) are implemented, respectively. Both of them use the same line search procedure (Wolfe conditions). Tables 2 and 3 report tests with $m = 3$ and $m = 5$ respectively. The basic data reported for each

Table 3: Numerical results (The higher dimensions)

No.	*dim*	Methods	CPU	NI	Nf	Ng	f^*
1	100	L-SR1	0.28	197	216	198	1.6241×10^{-5}
		L-BFGS	0.33	218	238	219	1.7447×10^{-4}
	500	L-SR1	4.78	319	332	320	1.2232×10^{-5}
		L-BFGS	4.84	320	341	321	1.3285×10^{-4}
2	100	L-SR1	0.31	29	35	30	7.1254×10^{-4}
		L-BFGS	0.45	35	38	36	3.0216×10^{-4}
3	100	L-SR1	0.02	18	20	19	7.6687×10^{-10}
		L-BFGS	0.03	21	28	21	8.5462×10^{-8}
	500	L-SR1	0.30	37	41	38	5.2643×10^{-8}
		L-BFGS	0.38	38	48	39	3.2247×10^{-7}
	1000	L-SR1	4.84	85	95	86	3.5424×10^{-8}
		L-BFGS	6.32	90	101	91	8.6044×10^{-7}
4	100	L-SR1	0.47	72	93	73	4.9949×10^{-5}
		L-BFGS	0.82	85	108	86	4.2528×10^{-5}
7	100	L-SR1	0.31	39	69	40	1.3353×10^{-8}
		L-BFGS	0.39	45	48	46	3.5124×10^{-8}
	500	L-SR1	8.28	67	72	68	4.1751×10^{-8}
		L-BFGS	9.18	78	87	79	7.1435×10^{-8}
	1000	L-SR1	35.94	125	144	126	4.4475×10^{-7}
		L-BFGS	37.56	154	171	155	8.6425×10^{-7}
8	100	L-SR1	0.31	21	39	22	2.2320×10^{-5}
		L-BFGS	0.34	24	53	25	9.2000×10^{-5}
	500	L-SR1	6.41	35	54	36	1.1127×10^{-5}
		L-BFGS	6.72	37	60	38	1.8824×10^{-4}
	1000	L-SR1	26.72	84	123	85	2.2254×10^{-5}
		L-BFGS	28.44	92	163	93	4.0376×10^{-4}
14	1000	L-SR1	6.24	34	54	35	4.8945×10^{-8}
		L-BFGS	8.52	42	58	52	7.2381×10^{-6}
	10000	L-SR1	54.91	87	98	88	4.3214×10^{-6}
		L-BFGS	68.12	89	105	112	1.2543×10^{-6}
15	5000	L-SR1	8.46	17	24	18	3.2547×10^{-4}
		L-BFGS	8.42	17	28	25	2.3445×10^{-4}
	10000	L-SR1	28.14	27	38	28	5.2538×10^{-4}
		L-BFGS	25.24	30	35	42	2.2458×10^{-4}

method are the dimension of the objective function (dim), the arithmetic time (CPU) which are reported in seconds, the number of iteration (NI), the number of calls to the function evaluation routine (Nf), the number of calls to the gradient evaluation (Ng) and the final function value (f^*).

The numerical results listed in Table 2 and 3 show the algorithm L-SR1 is competitive. For most of test problems, we can obtain more accurate results with less time and the number of iteration.

5 Conclusions

We present a new expression for SR1 formula, which is very simple in the form, then advance a limited memory method based on it. We give a two-loop recursive formula that takes advantage of the symmetry of the update matrices to compute the product $H_k g_k$ efficiently, and it is convenient for program. As a result, the computation of the search direction in the limited memory SR1 method for unconstrained optimization can be performed very economically.

References

[1] Ph. L. Toint, Towards an efficient sparsity exploiting Newton method for minimization, in: I. S. Duff, ed., Sparse Matrices and Their Uses (Academic Press, London, 1981), 57-88

[2] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, SIAM Journal on Numerical Analysis. 20 (1983), 626-637

[3] Ph. L. Toint, A view of nonlinear optimization in a large number of variables, Report Nr 86/16, Department of Mathematics, Facultes Universitaires de Namur (Namur, Belgium, 1986)

[4] R. Fletcher, Practical Methods of Optimization, John Wiley and Sons: Chichester and New York, 1987

[5] D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Methematical Programming. 45 (1989), 503-528

[6] J. L. Morales, A numerical study of limited memory BFGS methods, Applied Mathematics Letters. 15 (2002), 481-487

[7] M. Al-Baali, Extra update for the BFGS method, Optimization Methods Software. 13 (2000), 159-179

[8] Y. T. Yang, C. X. Xu, A compact limited memory method for large scale unconstrained optimization, European Journal of Operational Research. 180 (2007), 48-56

[9] A. R. Conn, N. I. M. Gould, Ph. Toint, Convergence of Quasi-Newton matrices generated by the symmetric rank one update, Methematical Programming. 50 (1991), 177-195

[10] J. Nocedal, Updating Quasi-Newton matrices with limited storage, Mathematics of Computation. 35(1980), 773-782

[11] W. B. Deng, A limited memory Quasi-Newton method for large scale problem, Neumerical Mathematics: A Journal of Chinese Universities (English series). 5 (1996), 71-79

[12] R. H. Byrd, J. Nocedal, R. B. Schnabel, Representations of Quasi-Newton matrices their use in limited memory methods, Math. Programming. 63 (1994), 129-156

[13] Ph. L. Toint, Test problems for partially separable optimization and results for the routine PSP-MIN, Report Nn8314, Department of Mathematics, Faculte's Universitaries de Namur (Namur, Belgium, 1983a)

[14] J. J. More, B. S. Garbow, K. E. Hillstrom, Testing unconstrained optimization software, ACM Transactions on Mathematical Software. 7 (1981), 17-41