

Lecture XXI: Conjugate Gradient Routines

Charles B. Moss

October 8, 2012

1 Quasi-Newton Methods for Optimization

The Newton-Raphson methodology can be used in U2 in the prototype algorithm. Specifically, the search direction can be determined by

$$p_t = - \left(\nabla_{xx}^2 f(x_t) \right)^{-1} \nabla_x f(x_t) \quad (1)$$

Quasi-Newton algorithms involve an approximation to the Hessian matrix. Replacing the Hessian may be advantageous because the true Hessian is analytically or numerically complex.

1.1 Steepest Descent

For example, we could replace the Hessian matrix in Equation 1 with the negative of the identity matrix (i.e., a constant negative definite matrix) and the applying the process will yield a new x_{t+1} that is closer to the optimum than our beginning estimate. In this case the search direction would be

$$p_t = - (-I(n)) \nabla_x f(x_t) \quad (2)$$

where $I(n)$ is the identity matrix conformable with the gradient vector. This replacement is referred to as the steepest descent method. In our sample problem, this methodology requires 990 iterations and 0.141 seconds. This result highlights some interesting features regarding the Quasi-Newton methods. The steepest descent method requires more overall iterations. In this example, the steepest descent method requires 99 times as many iterations as the Newton-Raphson method. However, the time spent on each iteration is reduced. Again, in the current comparison each the steepest descent

method requires .123 seconds per iteration while Newton-Raphson requires .030 seconds per iteration. **This section needs updating with the new computers.**

Obviously substituting the identity matrix uses no real information from the Hessian matrix. An alternative to this drastic reduction would be to systematically derive a matrix H_k which uses curvature information akin to the Hessian matrix. The projection could then be derived as

$$p_t = -H_k^{-1} \nabla_x f(x_t) \quad (3)$$

1.2 Conjugate Gradient Methods

One class of Quasi-Newton methods are the conjugate gradient methods which "build" up information on the Hessian matrix. From our standard starting point, we take a Taylor series expansion around the point $x_t + s_t$

$$\nabla_x f(x_t + s_t) = \nabla_x f(x_t) + \nabla_{xx}^2 f(x_t) s_t \quad (4)$$

for some $s_t \in N(x_t, \delta)$. Solving this expression for the term involving the Hessian yields

$$\begin{aligned} \nabla_x f(x_t + s_t) - \nabla_x f(x_t) &= \nabla_{xx}^2 f(x_t) s_t \\ s_t' (\nabla_{xx}^2 f(x_t + s_t) - \nabla_{xx}^2 f(x_t)) &= s_t' \nabla_{xx}^2 f(x_t) s_t. \end{aligned} \quad (5)$$

Thus, the change in the gradient contains the same curvature information as the original Hessian.

Defining the change in the gradient as y_t for simplification

$$y_t = \nabla_x f(x_t + s_t) - \nabla_x f(x_t) \Rightarrow y_t = \nabla_{xx}^2 f(x_t) s_t. \quad (6)$$

So we want to approximate the information in the Hessian using the some matrix B_{t+1}

$$y_t = B_{t+1} s_t. \quad (7)$$

One way to generate B_{t+1} is to start with the current B_t and add new information on the current solution (or to conjugate information based on our previous information and new information in the change of the gradient)

$$\begin{aligned} B_{t+1} &= B_t + uv' \\ y_t &= (B_t + uv') s_t \end{aligned} \quad (8)$$

where u and v are $n \times 1$ vectors. Carrying through the vector multiplication

$$u (v' s_t) = y_t - B_t s_t \quad (9)$$

where $v' s_t$ is a scalar. Solving for u given v and information from the last iteration

$$u = \frac{1}{v' s_t} (y_t - B_t s_t) \quad (10)$$

Substituting the result for u in Equation 10 into the expression for B_{t+1} in Equation 8 the update is based only on the value of v . In other words, by designing v we could update the conjugate value of the Hessian.

Given this derivation, a logical choice of v would be $y_t + B_t s_t$. This approximation yields an updated Hessian of

$$B_{t+1} = B_t + \frac{1}{(y_t - B_t s_t)' s_t} (y_t - B_t s_t) (y_t - B_t s_t)' \quad (11)$$

This needs a little more development. It is a rank-one update what is wrong with that? Also an example walk through may be a good idea This update results in a symmetric approximation to the Hessian. In general, any approximation to the Hessian can be made symmetric by

$$B^{(2)} = \frac{1}{2} (B^{(1)} + B^{(1)'}) \quad (12)$$

Substituting such an expression into the general updating procedure results in

$$B_{t+1} = B_t + \frac{1}{v' s_t} \left((y_t - B_t s_t) v' + v (y_t - B_t s_t)' \right) - \frac{(y_t - B_t s_t)' s_t}{(v' s_t)^2} v v'. \quad (13)$$

The advantage of this formulation is that the approximation of the Hessian matrix will be symmetric for any vector v .

Two prominent conjugate gradient methods are the Davidon-Fletcher-Powell (DFP) update and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update. In the DFP update v is set equal to y_t yielding

$$B_{t+1} = B_t - \frac{1}{s_t' B_t s_t} B_t s_t s_t' B_t + \frac{1}{y_t' s_t} y_t y_t' + (s_t' B_t s_t) w_t w_t'$$

$$w_t = \frac{1}{y_t' s_t} y_t - \frac{1}{s_t' B_t s_t} B_t s_t.$$
(14)

The BFGS update is then

$$B_{t+1} = B_t - \frac{1}{s_t' B_t s_t} B_t s_t s_t' B_t + \frac{1}{y_t' s_t} y_t y_t'.$$
(15)

1.3 A Numerical Example

Using the previously specified problem and starting with an identity matrix as the original Hessian matrix, each algorithm was used to maximize the utility function. Each of the procedures will have the same first iteration because they have the same original Hessian matrix and analytical gradient. The initial gradient for all the Quasi-Newton procedures was

$$B_t = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$
(16)

compared with an analytical gradient of

$$B_t^* = \begin{pmatrix} -0.5275 & 0.2885 & 0.0718 \\ 0.2885 & -0.6085 & 0.0954 \\ 0.0718 & 0.0954 & -0.2244 \end{pmatrix}.$$
(17)

This substitution implies a step of

$$s_t = \begin{pmatrix} 0.7337 & 0.9766 & 0.2428 \end{pmatrix}$$
(18)

compared with an optimal step of

$$s_t^* \begin{pmatrix} 4.3559 & 4.3476 & 4.3226 \end{pmatrix}$$
(19)

In discussing the difference in step, I will focus on two attributes. The first attribute is the relative length of the step (the 2-norm). If the conjugate gradient routines can be made identical with Newton-Raphson with the appropriate step length algorithm, we could then focus on designing step

lengths. The second attribute is the direction of the step. As we discussed above one of gains to Newton-Raphson is the refinement of the search direction through the information in the Hessian. The value of this refinement can be seen in the relative gain of the Newton-Raphson over the steepest descent method. Dividing each vector by its 2-norm yields the results depicted in Table 1.

1.3.1 Rank One Approximation

- Iteration 1

$$B_t = \begin{pmatrix} -0.6690 & 0.3842 & 0.1536 \\ 0.3842 & -0.5540 & 0.1783 \\ 0.1536 & 0.1783 & -0.2280 \end{pmatrix} \quad (20)$$

$$B_t^* = \begin{pmatrix} -0.2780 & 0.1276 & 0.0511 \\ 0.1276 & -0.2501 & 0.059 \\ 0.0511 & 0.0592 & -0.2280 \end{pmatrix} \quad (21)$$

- Iteration 2

$$B_{t+1} = \begin{pmatrix} -0.6315 & 0.4256 & 0.1776 \\ 0.4256 & -0.5083 & 0.2048 \\ 0.0106 & 0.0113 & -0.0846 \end{pmatrix} \quad (22)$$

$$B_{t+1}^* = \begin{pmatrix} -0.0628 & 0.0171 & 0.0106 \\ 0.0171 & -0.0612 & 0.0113 \\ 0.0106 & 0.0113 & -0.0846 \end{pmatrix} \quad (23)$$

1.3.2 Powell Symetric Broyden Update

- Iteration 1

$$B_t = \begin{pmatrix} -0.6703 & 0.3860 & 0.1504 \\ 0.3860 & -0.5565 & 0.1827 \\ 0.1504 & 0.1827 & -0.9365 \end{pmatrix} \quad (24)$$

Table 1: Comparison of Conjugate Gradient with Newton Raphson

	x_1	x_2	x_3	Step Length	x_1	x_2	x_3
	<i>Steepest Descent</i>						
Newton-Raphson	4.3559	4.3476	4.3226	7.5207	0.5792	0.5781	0.5748
Conjugate Gradient	0.7337	0.9766	0.2428	1.2454	0.5891	0.7842	0.1950
	<i>Rank One Approximation - Iteration 1</i>						
Newton-Raphson	6.7466	7.6270	4.8254	11.2682	0.5987	0.6769	0.4282
Conjugate Gradient	4.3187	5.0130	2.0042	6.9136	0.6247	0.7251	0.2899
	<i>Rank One Approximation - Iteration 2</i>						
Newton-Raphson	10.8541	11.6909	6.0789	17.0717	0.6358	0.6848	0.3561
Conjugate Gradient	7.7298	8.9067	3.7912	12.3876	0.6240	0.7190	0.3060
	<i>Powell Symmetric Broyden - Iteration 1</i>						
Newton-Raphson	10.8541	11.6909	6.0789	17.0717	0.6358	0.6848	0.3561
Conjugate Gradient	7.7298	8.9067	3.7912	12.3876	0.6240	0.7190	0.3060
	<i>Powell Symmetric Broyden - Iteration 2</i>						
Newton-Raphson	10.8525	11.7000	6.0686	17.0732	0.6356	0.6853	0.3554
Conjugate Gradient	7.7228	8.9130	3.7768	12.3834	0.6236	0.7198	0.3050

	Step				
	x_1	x_2	x_3	Length	x_1 x_2 x_3
<i>DFP - Iteration 1</i>					
Newton-Raphson	6.7466	7.6270	4.8255	11.2682	0.5987 0.6769 0.4282
Conjugate Gradient	4.1395	5.0091	1.7613	6.7327	0.6148 0.7440 0.2616
<i>DFP - Iteration 2</i>					
Newton-Raphson	6.7466	7.6270	4.8255	11.2682	0.5987 0.6769 0.4282
Conjugate Gradient	4.1395	5.0091	1.7613	6.7327	0.6148 0.7440 0.2616
<i>BFGS - Iteration 1</i>					
Newton-Raphson	10.8069	11.9291	5.8015	17.1099	0.6316 0.6972 0.3391
Conjugate Gradient	7.5224	9.0598	3.3966	12.2557	0.6138 0.7392 0.2771
<i>BFGS - Iteration 2</i>					
Newton-Raphson	10.8447	11.7436	6.0193	17.0807	0.6349 0.6875 0.3524
Conjugate Gradient	7.6884	8.9427	3.7078	12.3625	0.6219 0.7234 0.2999

- Iteration 2

$$B_{t+1} = \begin{pmatrix} -0.6328 & 0.4274 & 0.1745 \\ 0.4274 & -0.5109 & 0.2096 \\ 0.1745 & 0.2096 & -0.9223 \end{pmatrix} \quad (25)$$

$$B_{t+1}^* = \begin{pmatrix} -0.0629 & 0.0171 & 0.0106 \\ 0.0171 & -0.0612 & 0.0114 \\ 0.0106 & 0.0114 & -0.0850 \end{pmatrix} \quad (26)$$

1.3.3 Davidon-Fletcher-Powell Update

- Iteration 1

$$B_t = \begin{pmatrix} -0.7187 & 0.4517 & 0.0326 \\ 0.4517 & -0.6455 & 0.3424 \\ 0.0326 & 0.3424 & -1.2232 \end{pmatrix} \quad (27)$$

- Iteration 2

$$B_{t+1} = \begin{pmatrix} -0.6788 & 0.4945 & 0.0589 \\ 0.4945 & -0.6021 & 0.3766 \\ 0.0589 & 0.3766 & -0.0971 \end{pmatrix} \quad (28)$$

$$B_{t+1}^* = \begin{pmatrix} -0.0653 & 0.0177 & 0.0119 \\ 0.0177 & -0.0602 & 0.0124 \\ 0.0119 & 0.0124 & -0.0971 \end{pmatrix} \quad (29)$$

1.3.4 Broyden-Fletcher-Goldfarb-Shanno Update

- Iteration 1

$$B_t = \begin{pmatrix} -0.6771 & 0.3952 & 0.1338 \\ 0.3952 & -0.5690 & 0.2051 \\ 0.1338 & 0.2051 & -0.9768 \end{pmatrix} \quad (30)$$

- Iteration 2

$$B_{t+1} = \begin{pmatrix} -0.6391 & 0.4369 & 0.1585 \\ 0.4369 & -0.5238 & 0.2333 \\ 0.1585 & 0.2333 & -0.9644 \end{pmatrix} \quad (31)$$

$$B_{t+1}^* = \begin{pmatrix} -0.0634 & 0.0172 & 0.0109 \\ 0.0172 & -0.0610 & 0.0115 \\ 0.0109 & 0.0115 & -0.0871 \end{pmatrix} \quad (32)$$