Quasi-Newton Matrices Comparison

Numerical Linear Algebra Final Project

Jinxin Xia
Mathematics and Statistics Department
Wake Forest University

04/24/2019

Content

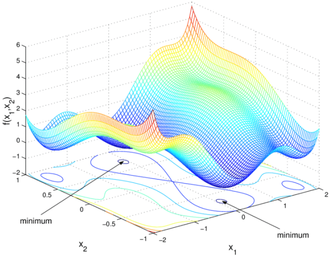
Content

Optimization

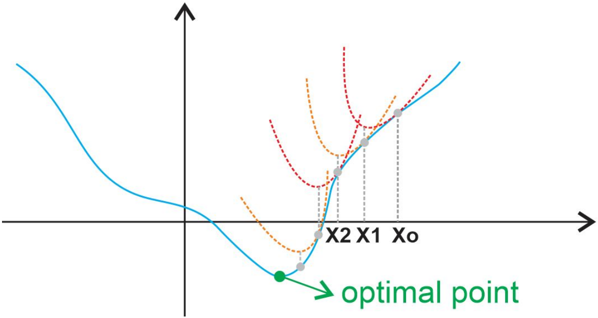
Definition of optimization "the action of making the best or most effective use of a situation or resource". In mathematics world, we use the following to show an optimization problem.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \; i = 1, \ldots, m. \end{aligned}$$

where $f(x)$ is the objective function and $f_i(x)$ are the constrain functions.



Newton Method in Optimization



Newton Method in Optimization

The minimize problem

$$\underset{x}{\text{minimize}}\ f(x)$$

The $f(x)$ is twice continuously differentiable. By using Taylor series expansion of $f(x)$, we can obtain a quadratic approximation at the point $x^{(k)}$

$$f(x) \approx f(x^{(k)}) + (x - x^{(k)})^T g^{(k)} + \frac{1}{2}(x - x^{(k)})^T H(x^{(k)})(x - x^{(k)})$$

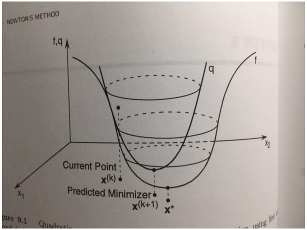
Newton Method in Optimization

To get the minimal we use the First Order Necessary Condition(FONC), which is let the gradient of $f(x)$ equal to zero.

$$0 = g^{(k)} + H(x^{(k)})(x - x^{(k)}))$$

Therefore, we can find the next point will be

$$x^{(k+1)} = x^{(k)} - H(x^{(k)})^{-1}g^{(k)}$$



Newton Method in Optimization

Two main steps in Newton Method:

- Compute Hessian $H(x)$
- Solve the system $H(x)(x^{(k+1)} - x^{(k)}) = g(x)$

Newton Method in Optimization

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2.2ex] \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2.2ex] \vdots & \vdots & \ddots & \vdots \\[2.2ex] \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Content

Quasi-Newton Method in Optimization

Instead of computing the Hessian matrix or solving the linear system, Quasi-Newton method use approximation of Hessian matrix. In this project, there are two ways of updating the inverse of the Hessian matrix.

Method 1

$$B_{k+1} = B_0 + \hat{\Psi}_k \hat{M}_K \hat{\Psi}_k^T$$

Method 2

$$B_k = B_0 + G_k T_k G_k^T$$

Method 1

$$B_{k+1} = B_0 + \hat{\Psi}_k \hat{M}_K \hat{\Psi}_k^T$$

where

$$\hat{M}_k = (\Xi_k^T \Pi_k \quad \begin{pmatrix} -S_k^T B_0 S_k + \Gamma_k & -L_k + \Gamma_k \\ -L_k^T + \Gamma_k & D_k \Gamma_k \end{pmatrix} \quad \Pi_k^T \Xi_k)$$

$$\Pi_k = \begin{pmatrix} I_k & 0 & 0 & 0 \\ 0 & 0 & I_k & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\Psi_k = \begin{pmatrix} B_0 S_k & Y_k \end{pmatrix}$, $\Pi_k \in \mathcal{R}^{2(K+1) \times 2(K+1)}$, $\Psi_k \in \mathcal{R}^{n \times 2(K+1)}$
$S_k = \begin{pmatrix} s_0 & s_1 & s_2 & \dots & s_k \end{pmatrix} \in \mathcal{R}^{n \times (K+1)}$
$Y_k = \begin{pmatrix} y_0 & y_1 & y_2 & \dots & y_k \end{pmatrix} \in \mathcal{R}^{n \times (K+1)}$
$s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$, where g is the gradient vector.

Method 1

$\Gamma_k = \underset{0\leq j\leq k}{diag}(\gamma_j)$, where

$$\gamma_j = \begin{cases} \phi_j\left(-\frac{1-\phi_j}{s_j^T B_j s_j}\frac{\phi_j}{s_j^T y_j}\right)^{-1} & if\, \phi_j \neq \phi_j^{SR11} \\ 0 & otherwise \end{cases}$$

and $S_k^T Y_k = L_K + D_k + R_k$, $\hat{\Psi}_k = \Psi_k \Pi_k^T \Xi_k$. The Ξ_k is as follow

$\Xi_k = \begin{pmatrix} \Pi_{k-1}^T \Xi_{k-1} & 0 \\ 0 & E_k \end{pmatrix}$, *where*

$$E_k = \begin{cases} \begin{pmatrix} -1 & 1 \end{pmatrix}^T & if\, \phi_k \neq \phi_k^{SR11} \\ I_2 & otherwise \end{cases}$$

Method 2

$$B_k = B_0 + \sum_{i=0}^{k-1} [\frac{1}{g_i^T p_i} g_i g_i^T + \frac{1}{\alpha_i (g_{i+1} - gi)^T} (g_{i+1} - gi)(g_{i+1} - gi)^T + \phi_i \omega_i \omega_i^T]$$

where

$$\omega_i = (-g_i^T p_i)^{\frac{1}{2}} (\frac{1}{(g_{i+1} - gi)^T p_i} (g_{i+1} - gi) - \frac{1}{g_i^T p_i} g_i)$$

which can also be written as

Method 2

$$B_k = B_0 + G_k T_k G_k^T$$

Method 2

$G_k = [g_0 \quad g_1 \quad g_2 \quad \dots \quad g_{k-1} \quad g_k]$ and $T_k \in \mathcal{R}^{(K+1)\times(K+1)}$ is a symmetic tridiagonal matrix on the form $T_k = T_k^C + T_k^{\phi}$, and $B_k p_k = -g_k$.

where

$$T_k^C(1,1) = \frac{1}{g_0^T p_0} + \frac{1}{\alpha_0 (g_1 - g_0)^T p_0}$$

$$T_k^C(i,i) = \frac{1}{g_i^T p_i} + \frac{1}{\alpha_{i-1}(g_i - g_{i-1})^T p_{i-1}} + \frac{1}{\alpha_i (g_{i+1} - g_i)^T p_i} \quad i = 1, \quad \dots, \quad k-1$$

$$T_k^C(i+1,i) = T_k^C(i,i+1) = -\frac{1}{\alpha_{i-1}(g_i - g_{i-1})^T p_{i-1}} \quad i = 1, \quad \dots, \quad k$$

$$T_k^C(k+1,k+1) = -\frac{1}{\alpha_{k-1}(g_k - g_{k-1})^T p_{k-1}}$$

Method 2

$$T_k^{\phi}(1,1) = -\phi_0 g_0^T p_0 \left(\frac{1}{(g_1-g_0)^T p_0} + \frac{1}{g_0^T p_0}\right)^2$$

$$T_k^{\phi}(i,i) = -\phi_{i-1} g_{i-1}^T p_{i-1} \left(\frac{1}{(g_i-g_{i-1})^T p_{i-1}}\right)^2 - \phi_i g_i^T p_i \left(\frac{1}{(g_{i+1}-g_i)^T p_i} + \frac{1}{g_i^T p_i}\right)^2$$

$$i = 1, \quad \ldots, \quad k-1$$

$$T_k^{\phi}(i+1,i) = T_k^{\phi}(i,i+1) =$$

$$\phi_{i-1} g_{i-1}^T p_{i-1} \left(\frac{1}{(g_i-g_{i-1})^T p_{i-1}}\right)^2 + \phi_{i-1} \frac{1}{(g_i-g_{i-1})^T p_{i-1}}$$

$$i = 1, \quad \ldots, \quad k$$

$$T_k^{\phi}(k+1,k+1) = -\phi_{k-1} g_{k-1}^T p_{k-1} \left(\frac{1}{(g_k-g_{k-1})^T p_{k-1}}\right)^2$$

Content

Method 1

$$B_{k+1} = B_0 + \hat{\Psi}_k \hat{M}_K \hat{\Psi}_k^T$$

$$\hat{M} = (\Xi_k^T \Pi_k \quad \begin{pmatrix} -S_k^T B_0 S_k + \Gamma_k & -L_k + \Gamma_k \\ -L_k^T + \Gamma_k & D_k \Gamma_k \end{pmatrix} \quad \Pi_k^T \Xi_k)$$

Since Ξ_k and Π_k are permutation matrix, we can ignore them in flops counting.

$$\hat{M} \approx \quad \begin{pmatrix} -S_k^T B_0 S_k + \Gamma_k & -L_k + \Gamma_k \\ -L_k^T + \Gamma_k & D_k \Gamma_k \end{pmatrix}$$

Comparison of two methods

$\hat{\Psi}_k$ is a dense matrix with dimension $\mathbb{R}^{n\times 2(k+1)}$, therefore we can use the following expression to count the updating flops

$$B_{k+1} = B_0 + \hat{\Psi}_k \quad \begin{pmatrix} -S_k^T B_0 S_k + \Gamma_k & -L_k + \Gamma_k \\ -L_k^T + \Gamma_k & D_k \Gamma_k \end{pmatrix} \quad \hat{\Psi}_k^T$$

Notice that the new $\hat{M}$ with dimension $\mathbb{R}^{2(k+1)\times 2(k+1)}$ is also a dense matrix thus the flops of the updating will be

$$\mathcal{O}(4n(k+1)^2) + \mathcal{O}(2n^2(k+1)) + \mathcal{O}(nk^2)$$

$\mathcal{O}(nk^2)$ is from the computation of $S_k^T Y_k$.

Comparison of two methods

Method 2

$$B_k = B_0 + G_k T_k G_k^T x$$

In method 2, G_k is a dense matrix with dimension $\mathbb{R}^{n\times(k+1)}$, and T_k is a tridiagonal matrix with dimension $\mathbb{R}^{(k+1)\times(k+1)}$. To make the analysis easy, we will treat it as a dense matrix first. Then the flops for this expression will be

$$\mathcal{O}(n(k+1)^2) + \mathcal{O}(n^2(k+1))$$

Comparison of two methods

The other thing that we need to check in method 2 is the flops of computing the tridiagonal matrix T_k.

$$T_k^C(1,1) = \frac{1}{g_0^T p_0} + \frac{1}{\alpha_0(g_1-g_0)^T p_0}$$

$$T_k^C(i,i) = \frac{1}{g_i^T p_i} + \frac{1}{\alpha_{i-1}(g_i-g_{i-1})^T p_{i-1}} + \frac{1}{\alpha_i(g_{i+1}-g_i)^T p_i} \quad i = 1, \quad ..., \quad k-1$$

$$T_k^C(i+1,i) = T_k^C(i,i+1) = -\frac{1}{\alpha_{i-1}(g_i-g_{i-1})^T p_{i-1}} \quad i = 1, \quad ..., \quad k$$

$$T_k^C(k+1,k+1) = -\frac{1}{\alpha_{k-1}(g_k-g_{k-1})^T p_{k-1}}$$

Comparison of two methods

All the computation of the elements of matrix T_k is vector times vector. Without too much work, we can get the flops of computing matrix T_k, which is

$$26k\mathcal{O}(n)$$

Therefore, the total flops of method 2 updating is

$$\mathcal{O}(n(k+1)^2) + \mathcal{O}(n^2(k+1)) + 26k\mathcal{O}(n)$$

and the total flops of method 1 updating is

$$\mathcal{O}(4n(k+1)^2) + \mathcal{O}(2n^2(k+1)) + \mathcal{O}(nk^2)$$

Content

Solving a linear system

Method 1

$$B_{k+1} = B_0 + \hat{\Psi}_k \hat{M}_K \hat{\Psi}_k^T$$

$$B_{k+1}x = b$$

Method 2

$$B_k = B_0 + G_k T_k G_k^T$$

$$B_k x = b$$

We will use the direct method to solve the two linear system, which is to use B_k^{-1} multiply b to get the solution for x.

Solving a linear system

Applying the Sherman-Morrison-Woodbury formula we can get the inverse of B_k we can get the following equations

Method 1

$$B_{k+1}^{-1} = B_0^{-1} + B_0^{-1}\hat{\Psi}_k(-\hat{M}_K^{-1} - \hat{\Psi}_k^T B_0^{-1}\hat{\Psi}_k)\hat{\Psi}_k^T B_0^{-1}$$

Method 2

$$B_{k+1}^{-1} = B_0^{-1} + B_0^{-1}G_k(-T_K^{-1} - G_k^T B_0^{-1}G_k)G_k^T B_0^{-1}$$

Solving a linear system

With help from previous work, we can get the flops of the two methods in sovling a linear system.

Method 1

$$B_{k+1}^{-1}b = B_0^{-1}b + B_0^{-1}\hat{\Psi}_k(-\hat{M}_K^{-1} - \hat{\Psi}_k^T B_0^{-1}\hat{\Psi}_k)\hat{\Psi}_k^T B_0^{-1}b$$

- $\mathcal{O}(8n(k+1)^2) + \mathcal{O}(4n(k+1))$

Method 2

$$B_{k+1}^{-1} = B_0^{-1}b + B_0^{-1}G_k(-T_K^{-1} - G_k^T B_0^{-1}G_k)G_k^T B_0^{-1}b$$

- $\mathcal{O}(2n(k+1)^2) + \mathcal{O}(2n(k+1))$

References

- DeGuchy, O, Erway, JB, Marcia, RF. Compact representation of the full Broyden class of quasi-Newton updates. Numer Linear Algebra Appl. 2018; 25:e2186. https://doi.org/10.1002/nla.2186
- Ek, David; Forsgren, Anders. On limited-memory quasi-Newton methods for minimizing a quadratic function. arXiv:1809.10590

Quasi-Newton Matrices Comparison

Numerical Linear Algebra Final Project

Jinxin Xia
Mathematics and Statistics Department
Wake Forest University

04/24/2019