Numerical Optimization

Unit 4: Quasi-Newton and Conjugate Gradient Methods

Che-Rung Lee

Scribe: 周宗毅

May 26, 2011

Three problems of Newton's method

Three problems of Newton's method:

1. Hessian matrix H may not be positive definite.
2. Hessian matrix H is expensive to compute.
3. The system $\vec{p} = -H^{-1}\vec{g}$ is expensive to solve.

We want to discuss methods to solve the second and the third problems.

Secant equation

- Recall that in the one dimensional optimization problem , the secant method approximate $f''(x_k)$ by $\tilde{h}_k = \dfrac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$ and we use $\tilde{h}_k$ in the secant's method.

$$f'(x_k) - f'(x_{k-1}) = \tilde{h}_k(x_k - x_{k-1}) = \tilde{h}_k p_k.$$

- In multivariable optimization, we want to find an "approximate" Hessian matrix B_k such that

$$\nabla f_{k+1} - \nabla f_k = B_k \vec{p}_k. \tag{1}$$

- The above equation is called the "secant equation" in multivariable function.

The SR1 update

- B_k and B_{k-1} should be "similar": the symmetric rank 1 (SR1) update: where $\sigma_k = +1$ or -1 and $\vec{u}$ is a vector.

$$B_k = B_{k-1} + \sigma_k \vec{u}_k \vec{u}_k^T, \tag{2}$$

- What is $\vec{u}$?
  - Let $\vec{y}_k = \nabla f_k - \nabla f_{k-1} = B_k \vec{p}_k = (B_{k-1} + \sigma_k \vec{u}\vec{u}^T)\vec{p}_k = B_{k-1}\vec{p}_k + \sigma_k \vec{u}\vec{u}^T \vec{p}_k$.
  - $\vec{y}_k - B_{k-1}\vec{p}_k = (\sigma \vec{u}^T \vec{p}_k) \cdot \vec{u} \Rightarrow \vec{u}$ is parallel to $\vec{y}_k - B_{k-1}\vec{p}_k$.
  - Let $\vec{u} = \delta(\vec{y}_k - B_{k-1}\vec{p}_k)$. Using (1) and (2), one can derive

$$\sigma = \mathrm{sign}(\vec{y}_k^T \vec{p}_k - \vec{p}_k^T B_{k-1} \vec{p}_k) \tag{3}$$

$$\delta = \pm(\vec{y}_k^T \vec{p}_k - \vec{p}_k^T B_{k-1} \vec{p}_k)^{-1/2} \tag{4}$$

- By substituting (3) and (4) back to (2), one can show that

$$\begin{aligned} B_k &= B_{k-1} + \sigma \cdot \vec{u}\vec{u}^T \\ &= B_{k-1} + \frac{(\vec{y}_k - B_{k-1}\vec{p}_k)(\vec{y}_k - B_{k-1}\vec{p}_k)^T}{(\vec{y}_k - B_{k-1}\vec{p}_k)^T \vec{p}_k} \end{aligned} \tag{5}$$

The Sherman—Morrison—Woodbury formula

- What we really need is not an approximation to H_k, but an approximation to H_k^{-1}.
- If we know B_{k-1}^{-1}, and $B_k = B_{k-1} + \sigma \vec{u}\vec{u}^T$, can we compute B_k^{-1} efficiently?
- The Sherman - Morrison - Woodbury formula.

$$
\begin{aligned}
\hat{A} &= A + \vec{a}\vec{b}^T \\
\hat{A}^{-1} &= A^{-1} - \frac{A^{-1}\vec{a}\vec{b}^T A^{-1}}{1 + \vec{b}^T A^{-1}\vec{a}}
\end{aligned}
$$

- Thus, the formula of SR1 update is (see note 3 for details.)

$$
B_k^{-1} = B_{k-1}^{-1} + \frac{(\vec{p}_k - B_{k-1}^{-1}\vec{y}_k)(\vec{p}_k - B_k^{-1}\vec{y}_k)^T}{\vec{y}_k^T B_{k-1}^{-1}\vec{y}_k - \vec{y}_k^T\vec{p}_k}
$$

Numerical properties of the SR1 update

Convergence for a quadratic function

Suppose $f(\vec{x}) = \vec{b}^T\vec{x} + \frac{1}{2}\vec{x}^T A\vec{x}$ and A is symmetric positive definite. Then for any starting point $\vec{x}_0$ and any starting H_0, SR1 converges to the minimizer in at most n steps, where n is the problem size, provided that $(\vec{p}_k - B_k^{-1}\vec{y}_k)^T\vec{y}_k \neq 0$ for all k.

Problems of the SR1 method

1. $(\vec{y}_k - B_k\vec{p}_k)^T\vec{p}_k$ may be 0 $\Rightarrow$ Just use $B_k = B_{k-1}$.
2. B_k may be indefinite $\Rightarrow$ Use BFGS.

BFGS

The BFGS method (1970) (Broyden, Fletcher, Goldtarb, Shanno)

- BFGS can keep B_k symmetric positive definite with the curvature condition:

$$\vec{y}_k = \vec{B}_k \vec{p}_k \Rightarrow \vec{p}_k^T \vec{B}_k \vec{p}_k = \vec{p}_k^T \vec{y}_k > 0$$

- We need a rank 2 update

$$B_k^{-1} = (I - \rho_k \vec{p}_k \vec{y}_k^T) B_{k-1}^{-1} (I - \rho_k \vec{y}_k \vec{p}_k^T) + \rho_k \vec{p}_k \vec{p}_k^T \text{ where } \rho_k = \frac{1}{\vec{y}_k^T \vec{p}_k}$$

Theorem (Convergence of BFGS)

Suppose $f = \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Consider the iteration $\vec{x}_{k+1} = \vec{x}_k + \vec{p}_k$ where $\vec{p}_k = -B_k^{-1} \nabla f_k$. If $\{\vec{x}_k\}$ converges to $\vec{x}^*$ s.t. $\nabla f(\vec{x}^*) = 0$ and $\nabla^2 f(\vec{x}^*)$ is positive definite, then $\{\vec{x}_k\}$ converges superlinearly if and only if $\lim_{k\to\infty} \dfrac{\|\nabla f_k + \nabla^2 f_k \vec{p}_k\|}{\|\vec{p}_k\|} = 0$

Review of the quadratic model

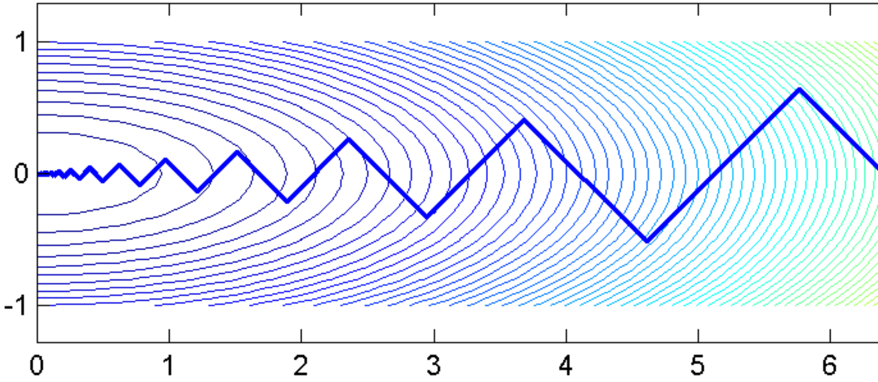
- Consider a quadratic function

$$f(\vec{x}) = \frac{1}{2}\vec{x}^T A\vec{x} - \vec{x}^T\vec{b} + c$$

- To find the optimal solution of $f(\vec{x})$ is equivalent to find $\nabla f(\vec{x}) = A\vec{x} - \vec{b} = 0$, which is to solve the linear system $A\vec{x} = \vec{b}$.
- We call $\vec{r} = \vec{b} - A\vec{x}$ the *residual* for the linear system $A\vec{x} = \vec{b}$. The smaller $\|\vec{r}\|$ is, the better solution $\vec{x}$ is.

$$\begin{aligned} \vec{r} &= \vec{b} - A\vec{x} = A\vec{x}^* - A\vec{x} \\ \|\vec{x}^* - \vec{x}\| &= \|A^{-1}\vec{r}\| \leq \|A^{-1}\|\|\vec{r}\| \\ &= \|A^{-1}\|\|A\|\frac{\|\vec{r}\|}{\|A\|} = \kappa(A)\frac{\|\vec{r}\|}{\|A\|} \end{aligned}$$

- Recall the steepest descent method: $\vec{p}_k = -\nabla f(\vec{x}) = \vec{b} - A\vec{x}$ and $\alpha_k = \dfrac{-\vec{p}_k^T \vec{g}_k}{\vec{p}_k^T A_k \vec{p}_k}$.
- From homework 2, the trace of $\{x_k\}$ shows a zigzag patten.



Conjugate gradient method (CG)

- A symmetric positive definite matrix can define an "inner product":

$$\langle \vec{a}, \vec{b} \rangle_A \equiv \vec{a}^T A \vec{b}.$$

- Vector $\vec{a}, \vec{b}$ are called A-conjugate or A-orthogonal if $\langle \vec{a}, \vec{b} \rangle_A = 0$
- Let $\vec{p}_{k+1} = -\vec{r}_{k+1} + \beta_k \vec{p}_k$. We want $\vec{p}_{k+1}$ and $\vec{p}_k$ to be A-conjugate.

$$\langle \vec{p}_{k+1}, \vec{p}_k \rangle_A = \vec{p}_k^T A(-\vec{r}_{k+1} + \beta_k \vec{p}_k) = -\vec{p}_k^T A \vec{r}_{k+1} + \beta_k \vec{p}_k^T A \vec{p}_k = 0.$$

$$\Rightarrow \beta_k = \frac{\vec{p}_k^T A \vec{r}_{k+1}}{\vec{p}_k^T A \vec{p}_k}$$

- Use the same $\alpha_k = \dfrac{-\vec{p}_k^T \vec{g}_k}{\vec{p}_k^T A_k \vec{p}_k}$ as the steepest descent method.
- To save one matrix-vector multiplication, residuals can be updated as

$$\vec{r}_{k+1} = \vec{b} - A\vec{x}_{k+1} = \vec{b} - A(\vec{x}_k + \alpha_k \vec{p}_k) = \vec{r}_k - \alpha_k A \vec{p}_k$$

The conjugate gradient algorithm

Put everything together...

The conjugate gradient algorithm

1. Given $\vec{x}_0$. Let $\vec{p}_0 = \vec{b} - A\vec{x}_0$ and $\vec{r}_0 = \vec{p}_0$.
2. For $k = 0, 1, 2, \ldots$ until $\|\vec{r}_k\| \leq \epsilon$

$$
\begin{aligned}
\alpha_k &= \frac{\vec{p}_k^T \vec{r}_k}{\vec{p}_k^T A \vec{p}_k} \\
\vec{x}_{k+1} &= \vec{x}_k + \alpha_k \vec{p}_k \\
\vec{r}_{k+1} &= \vec{r}_k - \alpha_k A \vec{p}_k \\
\beta_k &= \frac{\vec{r}_{k+1}^T A p_k}{\vec{p}_k^T A \vec{p}_k} \\
\vec{p}_{k+1} &= -\vec{r}_{k+1} + \beta_k \vec{p}_k
\end{aligned}
$$

Example

$f(\vec{x}) = \frac{1}{2}\vec{x}^T \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix} \vec{x}$ and $\vec{x}_0 = \begin{pmatrix} 9 \\ 1 \end{pmatrix}$, in which

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}, \vec{b} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Initially,

$$\vec{p}_0 = \vec{b} - A\vec{x}_0 = \begin{pmatrix} -9 \\ -9 \end{pmatrix} = \vec{r}_0.$$

The first iteration,

$$A\vec{p}_0 = \begin{pmatrix} -9 \\ -81 \end{pmatrix}$$

$$\alpha_0 = \frac{2 \times 81}{81 + 9 \times 81} = \frac{1}{5}$$

$$\vec{x}_1 = \begin{pmatrix} 9 \\ 1 \end{pmatrix} + \frac{1}{5}\begin{pmatrix} -9 \\ -9 \end{pmatrix} = \begin{pmatrix} 7.2 \\ -0.8 \end{pmatrix}$$

Example–continue

$$\vec{r}_1 = \begin{pmatrix} -9 \\ -9 \end{pmatrix} - \frac{1}{5} \begin{pmatrix} -9 \\ -81 \end{pmatrix} = \begin{pmatrix} -7.2 \\ 7.2 \end{pmatrix}$$

$$\beta_1 = \frac{7.2^2 \times 2}{9^2 \times 2} = \left(\frac{7.2}{9}\right)^2 = 0.64$$

$$\vec{p}_1 = \begin{pmatrix} -7.2 \\ 7.2 \end{pmatrix} + 0.8 \times 0.8 \begin{pmatrix} -9 \\ -9 \end{pmatrix} = \begin{pmatrix} -1.8 \times 7.2 \\ 0.2 \times 7.2 \end{pmatrix}$$

The second iteration

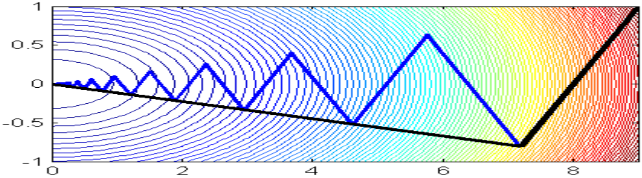
$$\alpha_1 = \frac{7.2^2 \times 2}{12.96^2 + 12.96 \times 1.44} = \frac{1}{1.8}$$

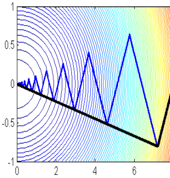
$$\vec{x}_2 = \begin{pmatrix} 7.2 \\ -0.8 \end{pmatrix} + \frac{1}{1.8} \begin{pmatrix} -12.96 \\ 1.44 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\vec{r}_2 = \begin{pmatrix} -7.2 \\ 7.2 \end{pmatrix} - \frac{1}{1.8} \begin{pmatrix} -1.8 \times 7.2 \\ 0.2 \times 7.2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Properties of the CG

Trace of the example (compared with Steepest-descent direction and Newton's direction.)





Theorem (Convergence)

For any $\vec{x} \in \mathbb{R}^n$, if A has m distinct eigenvalues, the CG will terminate at the solution at most m iterations. Moreover, if A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$,

$$\|\vec{x}_{k+1} - \vec{x}^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|\vec{x}_0 - \vec{x}^*\|_A^2$$

Preconditioned CG (PCG)

- The convergence of the CG can be very small if If $\kappa(A)^{-1} = \frac{\lambda_{min}}{\lambda_{max}}$ is small.
- If we can find a matrix M such that the ratio of the smallest eigenvalue and the largest eigenvalue of $MA \approx I$, then the convergence can be faster.
- The original problem $A\vec{x} = \vec{b}$ becomes $MA\vec{x} = M\vec{b}$.

$$\vec{x} = (MA)^{-1}M\vec{b} = A^{-1}M^{-1}M\vec{b} = A^{-1}\vec{b}$$

Truncated Newton method

1. Hessian matrix A may fail to be positive definite.
2. The linear system $A\vec{x} = \vec{b}$ need not be solved "exactly". (Recall the modified Newton's method.)
3. Therefore, we can stop the iterations as soon as we found the indefiniteness of A or when $\|\vec{r}\| < \epsilon$.

Hessian free CG

- When the problem is large, generating and storing matrix A are expensive. (Matrix A may not be sparse in many cases.)
- We don't really need the Hessian matrix A explicitly. What we need is $A\vec{v}$.
- Matrix A is a special matrix $\nabla^2 f_k$. Recall the definition of the directional derivative (See note 2),

$$A\vec{v} = \nabla^2 f_k \vec{v} \approx \frac{\nabla f(\vec{x}_k + h\vec{v}) - \nabla f(\vec{x}_k)}{h}$$

  for some small enough h.
- Other methods that can solve large-scale problems include Limited memory BFGS, etc.

Nonlinear CG

- When the problem is not quadratic, similar methods can be used for the nonlinear optimization.
- Two differences:
  1. Step length α_k is computed by the line search algorithm.
  2. The formula of computing β_k.
     - Ex: The Fletcher-Reeves method, $\beta_k = \dfrac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$.
     - Ex: The Polak-Ribière method, $\beta_k = \dfrac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$.