

Computational Optimization



Newton's Method

2/5/08

Newton's Method

- Method for finding a zero of a function.

- Recall FONC $\nabla f(x) = 0$

- For quadratic case:

$$g(x) = \frac{1}{2} x' Q x + b' x$$

$$\nabla g(x) = Qx + b = 0$$

Minimum must satisfy, $Qx^* = -b$

$$\Rightarrow x^* = -Q^{-1}b \quad (\text{unique if } Q \text{ is invertible})$$

General nonlinear functions

- For non-quadratic f (twice cont. diff):
- Approximate by 2nd order TSA
- Solve for FONC for quadratic approx.

$$f(y) \approx f(x) + (y-x)' \nabla f(x) + \frac{1}{2} (y-x)' \nabla^2 f(x) (y-x)$$

Calculate FONC

$$\nabla f(y) = \nabla f(x) + \nabla^2 f(x)(y-x) = 0$$

Solve for y

$$\nabla^2 f(x)(y-x) = -\nabla f(x)$$

$$y = x - \underbrace{\left[\nabla^2 f(x) \right]^{-1} \nabla f(x)}_{\text{Pure Newton Direction}}$$


Pure Newton
Direction



Basic Newton's Algorithm

- Start with x_0
- For $k = 1, \dots, K$
 - If x_k is optimal then stop
 - Solve: $\nabla^2 f(x_k) p = -\nabla f(x_k)$
 - $x_{k+1} = x_k + p$





Theorem 3.5 (NW)


Convergence of Pure Newton's

Let f be twice cont. diff with a Lipschitz Hessian in the neighborhood of a solution x^* that satisfies the SOSC.

- For x_0 sufficiently close to x^* , Newton's method converges to x^* .
 - The rate of convergence of $\{x_k\}$ is quadratic
 - $\|\nabla f(x_k)\|$ converges quadratically to 0.
-

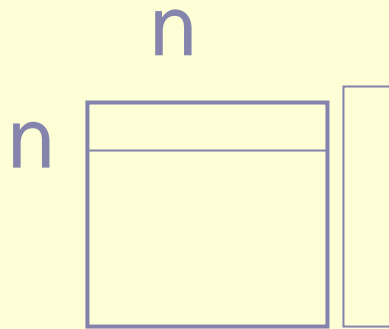


Analysis of Newton's Method

- Newton's Method converges to a zero of a function very fast (quadratic convergence)
 - Expensive both in storage and time.
 - Must compute and store Hessian
 - Must solve Newton Equation
 - May not find a local minimum. Could find any stationary point.
- 

Method 1: $d = -H^{-1}g$

- Require Matrix Vector multiplication for $n \times n$ matrix $H^{-1}g$



$n(n$ multiplies $+(n-1)$ adds)

$2n^2 - n$ FLOPS

Say $O(n^2)$

- Also requires computing H^{-1} $O(n^3)$
- Matlab comand $d = -\text{inv}(H) * g$


Computing Inverse

Gaussian Elimination

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 3 & 7 \end{bmatrix} \begin{array}{l} \text{multiply by } 1/4 \\ \text{multiply row 1 by } -2 \\ \text{add to row} \end{array} \begin{array}{l} n+1 \\ n+1 \\ n+1 \end{array} \quad (n-1 \text{ times})$$
$$= n + 1 + (n - 1)(n + 1) = n^2 + n$$

$$\begin{bmatrix} \frac{1}{4} & 0 & 0 \\ \frac{-1}{2} & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{4} \\ 0 & 4 & \frac{5}{2} \\ 0 & \frac{5}{2} & \frac{27}{4} \end{bmatrix} \text{repeat roughly } n \text{ times}$$

$$O(n^3)$$



Method 2: $Hp=-g$


- Solve by Gaussian Elimination


Take about $2n^3/3$

Faster but still $O(n^3)$

Matlab command

$P=-H\backslash g$





Method 3: Cholesky Factorization $O(n^3)$

- Factorize Matrix -


$H = LDL' = L^*U$ where L is lower diagonal and DL' is upper diagonal matrix

- Why? $Ax=b$ for X by solving
 $LUx=b$

First solve $Ly=b$ by forward elimination

Then $U'x=y$ by backward elimination

Matlab command to compute cholesky factorization is $R=\text{chol}(H)$ but only works if H is p.d. Gives factorization $R'^*R=H$



Forward elimination for $Ly=b$

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & 1_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$l_{11} y_1 = b_1 \Rightarrow y_1 = \frac{b_1}{l_{11}}$$

$$l_{21} y_1 + l_{22} y_2 = b_2 \Rightarrow y_2 = \frac{b_2 - l_{21} y_1}{l_{22}}$$

\vdots

$$l_{n1} y_1 + l_{n2} y_2 + \cdots + l_{nn} y_n = b_n \Rightarrow y_n = \frac{b_n - \sum_{m=1}^{n-1} l_{nm} y_m}{l_{nn}}$$

$O(n^2)$ operations

Back Substitution for $Ux=y$

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$u_{nn} x_n = y_n \quad \Rightarrow \quad x_n = \frac{y_n}{u_{nn}}$$

$$u_{(n-1)(n-1)} x_{(n-1)} + u_{n(n-1)} x_n = y_{(n-1)} \quad \Rightarrow \quad x_{n-1} = \frac{y_{(n-1)} - u_{n(n-1)} x_n}{u_{(n-1)(n-1)}}$$


$$\vdots$$

$$u_{n1} x_n + u_{(n-1)1} y_2 + \cdots + u_{11} x_1 = y_1 \quad \Rightarrow \quad x_1 = \frac{y_1 - \sum_{m=i+1}^n u_{m1} x_m}{u_{11}}$$

$O(n^2)$ operations



Problems with Newton's

- Newton's method may converge to a local max or stationary problem
 - Would like to make sure we decrease function at each iteration.
 - Newton equation may not have solution or may not have unique solution
- 

Guarantee Descent

- Would like to guarantee descent direction is picked $p_k' \nabla f(x_k) < 0$

$$p_k = -H_k \nabla f(x_k)$$

- Any H_k p.d. works since

$$p_k' \nabla f(x_k) = -\nabla f(x_k)' H_k \nabla f(x_k) < 0$$




What if Hessian is not p.d.

Add diagonal matrix Δ^k to $\nabla^2 f(x)$

so that $\nabla^2 f(x) + \Delta^k$ is p.d.

Modified Cholesky Factorization does this automatically.





Theorem: Superlinear Convergence of Newton Like Methods

Assume f be twice cont. differentiable on open set S .

$\nabla^2 f(x)$ p.d. and Lipschitz cont on S i.e.


$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L \|x - y\|$$

for all $x, y \in S$ and some fixed finite L

the sequence $\{x_k\} \subset S$ generated by

$$x_{k+1} = x_k + p_k \quad \lim_{k \rightarrow \infty} x_k = x^* \subset S$$





Theorem 10.1 (continued)

$\{x_k\} \rightarrow x^*$ superlinearly and $\nabla f(x^*) = 0$
if and only if

$$\lim_{k \rightarrow \infty} \frac{\|p_k - (p_n)_k\|}{\|p_k\|} = 0 \quad \text{where } (p_n)_k$$

is the Newton direction at x_k

$$x_{k+1} = x_k + p_k$$



Alternative results

In NW superlinear convergence results iff


$$\lim_{k \rightarrow \infty} \frac{\| (B_k - \nabla^2 f(x^*)) p_k \|}{\| p_k \|} = 0$$

So we can use a quasi-newton algorithm with positive definite matrix approximating Hessian

$$B_k$$




Problems with Newton's

- Newton's method converge to a local max or stationary problem
 - Pure Newton's method may not converge at all. Only has local convergence, i.e. only starts if sufficiently close to the solution.
- 

Stepsize problems too

- Newton's has only local convergence. If too far from solution may not converge at all.
- Try this example starting from 1.1:

$$f(x) = \ln(e^x + e^{-x})$$

$$f'(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

$$f''(x) = 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} > 0$$

Need adaptive stepsize

- Add stepsize in each iteration

$$x_{k+1} = x_k + \alpha_k p_k$$

- Could use exact stepsize algorithm like golden section search to solve

$$\alpha_k = \arg \min_{\alpha} f(x_k + \alpha p_k)$$

- But unnecessarily expensive
- Can do approximate linesearch like Armijo search (next lecture)

Final Newton's Algorithm

- Start with x_0
- For $k = 1, \dots, K$
 - If x_k is optimal then stop
 - Solve:

$$\nabla^2 f(x_k) + E = LDL'$$

$$LDL' p_k = -\nabla f(x_k) \quad \text{using modified cholesky factorization}$$

- Perform linesearch to determine

$$x_{k+1} = x_k + \alpha_k p_k$$



Newton's Method Summary

● Pure Newton

- Very fast convergence (if converges)
 - Each iteration expensive
 - Must add linesearch and modified Cholesky factorization to guarantee convergence globally.
 - Requires Calculation/storage of Hessian
-

