

Computational Optimization

Quasi Newton Methods 2/22
NW Chapter 8

Theorem 3.4

- Suppose f is twice cont diff and the sequence of steepest descent converge to x^* satisfying SOSOC.

Let $r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right) = \left(\frac{\rho - 1}{\rho + 1} \right)$ where $\lambda_1 \leq \dots \leq \lambda_n$ eigenvalues of $\nabla^2 f(x^*)$

The for all k sufficient large

$$f(x_{k+1}) - f(x^*) \leq r^2 [f(x_k) - f(x^*)]$$



Choosing better directions

- Steepest Descent – simple and cheap per iterations but can converge very slowly if conditioning is bad.
 - Modified Newton's – expensive per iterations but converges quickly.
 - Goal – first order methods with Newton like behavior
-

Scaled Steepest Descent

- Pick approximation of Hessian D_k

$$x_{k+1} = x_k - \alpha_k D_k \nabla f(x_k)$$

$$\text{Let } S_k = (D_k)^{1/2}$$

$$x = Sy$$

Do change
of variables

Now problem is

$$\min h(y) = f(Sy)$$

Scaled Steepest Descent...

$$\begin{aligned}y_{k+1} &= y_k - \alpha_k \nabla h(y_k) \\ &= y_k - \alpha_k S \nabla f(Sy_k)\end{aligned}$$

Multiple by S

$$S y_{k+1} = S y_k - \alpha_k S S \nabla f(Sy_k)$$

$$S y_{k+1} = S y_k - \alpha_k D \nabla f(Sy_k)$$

$$x_{k+1} = x_k - \alpha_k D \nabla f(x_k)$$

Thus convergence rate of steepest descent
applies in this space

$$g(y) = f(Sy) = y' S Q S y$$



Scaled Steepest Descent...

Convergence rate governed by eigs of SQS

λ_1 = smallest eigenvalue of SQS

λ_n = largest eigenvalue of SQS

Choose S close to $(Q^{-1})^{1/2}$ to make λ_n / λ_1 close to 1

(note $(Q^{-1})^{1/2} Q (Q^{-1})^{1/2} = I$)

Cheap Newton Approximation

- Use just diagonal of Hessian

$$S = D_k = \begin{bmatrix} \left(\frac{\partial^2 f}{\partial x_1 \partial x_1} \right)^{-1} & 0 & 0 \\ 0 & \left(\frac{\partial^2 f}{\partial x_2 \partial x_2} \right)^{-1} & 0 \\ 0 & 0 & \left(\frac{\partial^2 f}{\partial x_3 \partial x_3} \right)^{-1} \end{bmatrix}$$

- Linear storage and computation, inverse is trivial.
- Limited effectiveness.



Quasi-Newton Methods

- Newton's Method

$$\nabla^2 f(x_k) p = -\nabla f(x_k)$$

- Instead substitute B_k

$$B_k p = -\nabla f(x_k)$$


to get Newton-like directions



Quasi-Newton Methods

- Better yet – estimate Newton inverse directly

$$B_k p = -\nabla f(x_k)$$

$$H_k = B_k^{-1}$$


1-dimensional case

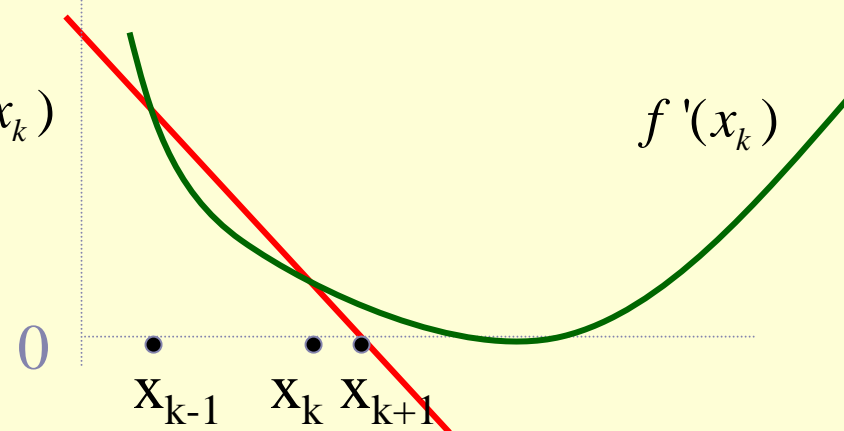
- In 1-d case might estimate

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

change in derivative
change in x

- If you do this you get secant method

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})} f'(x_k)$$





1-d convergence

- Secant method has superlinear convergence with rate

$$r = \frac{1}{2}(1 + \sqrt{5}) \quad (\text{the "golden ratio" again!})$$

- But Secant Method only applies to 1-d
- 

Secant Condition

- 1-d condition

$$f''(x_k)(x_k - x_{k-1}) = f'(x_k) - f'(x_{k-1})$$

- Generalizes to

$$\nabla^2 f(x_k)(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1})$$

- So we want

$$B_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1})$$

Another way to think about it

Approximating quadratic model

$$m_k(p) = f(x_k) + \nabla f(x_k)' p + \frac{1}{2} p' B_k p$$

• Gradient = grad of current iterate

$$\nabla m_k(0) = \nabla f(x_k)$$

• Want gradient = gradient of old iterate

$$\nabla m_k(x_k - x_{k-1}) = \nabla m_k(-\alpha_k p_{k-1}) = \nabla f(x_k) - B_k \alpha_k p_{k-1} = \nabla f(x_{k-1})$$

• So

$$B_k \alpha_k p_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$$

Quadratic Case

- For $\min \frac{1}{2}x'Qx - b'x$

$$\begin{aligned}\nabla f(x_k) - \nabla f(x_{k-1}) &= (Qx_k - b) - (Qx_{k-1} - b) \\ &= Q(x_k - x_{k-1})\end{aligned}$$

- So B_k should act like Q along direction

$$s_k = x_k - x_{k-1}$$


- Let $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$

So Quasi Newton Condition becomes

$$B_{k+1}s_k = y_k$$



Choice of B

- At each step we get information about Q along direction $x_k - x_{k-1}$
 - Use it to update our estimate of Q
 - Many possible ways to do this and still satisfy quasi-Newton condition
- 

BFGS Update

- Update by adding two matrices

$$B_{k+1} = B_k + \alpha a_k a_k' + \beta b_k b_k' \quad \text{Note outer product}$$

- Need

$$B_{k+1} s_k = B_k s_k + \alpha a_k a_k' s_k + \beta b_k b_k' s_k = y_k \quad \text{from QNC}$$

$$\text{So we make } \alpha a_k a_k' s_k = y_k$$

$$\text{and } \beta b_k b_k' s_k = -B_k s_k$$

BFGS Update

• To make $\beta b_k b_k' s = -B_k s_k$

Define $b_k = (B_k s_k)$

$$\begin{aligned}\text{So } \beta b_k b_k' s_k &= \beta (B_k s_k) (B_k s_k)' s_k \\ &= \beta (s_k' B_k s_k) (B_k s_k)\end{aligned}$$

• So pick $\beta = -\frac{1}{s_k' B_k s_k}$

BFGS Update

● To make $\alpha a_k a_k' s_k = y_k$

Define $a_k = y_k$

$$\begin{aligned}\text{So } \alpha a_k a_k' s_k &= \alpha y_k y_k' s_k \\ &= \alpha \left(y_k' s_k \right) y_k\end{aligned}$$

$$\text{So define } \alpha = \frac{1}{\left(y_k' s_k \right)}$$

BFGS Update

- Final Update is

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)'}{s_k' B_k s_k} + \frac{y_k y_k'}{y_k' s_k}$$

- This is called a BFGS family update for Broyden Fletcher Goldfarb and Shanno



Key Ideas

- This update is called a rank 2 update since it adds two rank one matrices.
- We want B_k to be p.d. and symmetric.
- Want to solve $B_k p_k = -\nabla f(x_k)$ efficiently.

Two possible ways





Descent directions

Need B to be positive definite.

Necessary condition=Curvature Condition

$$B_{k+1}s_k = y_k \Rightarrow s_k' B_{k+1}s_k = s_k' y_k > 0$$

Enforce for general conditions using
Wolfe or Strong Wolfe Conditions



Wolfe Conditions

For $0 < c_1 < c_2 < 1$

$$f(x_{k+1}) \leq f(x_k) + c_1 \alpha \nabla f(x_k)' p_k$$

$$\nabla f(x_{k+1})' p_k \geq c_2 \nabla f(x_k)' p_k$$

Implies

$$\nabla f(x_{k+1})' s_k \geq c_2 \nabla f(x_k)' s_k$$

$$\begin{aligned} y_k' s_k &= (\nabla f(x_{k+1}) - \nabla f(x_k))' s_k \geq (c_2 - 1) \nabla f(x_k)' s_k \\ &= (c_2 - 1) \nabla f(x_k)' (\alpha_k p_k) > 0 \end{aligned}$$



Guaranteeing B p.d. and sym.

- Lemma 11.5 in Nash and Sofer
if B_k is p.d. and symmetric then B_{k+1} is p.d. if and only if $y_k' s_k > 0$
- So enforce this condition in linesearch procedure using wolfe conditions

$$[\nabla f(x_k) - \nabla f(x_{k-1})]' [x_k - x_{k-1}] > 0$$



Quasi-Newton Algorithm with BFGS update

- Start with x_0, B_0 e.g. $B_0=I$
- For $k = 1, \dots, K$
 - If x_k is optimal then stop
 - Solve:
$$B_k p_k = -\nabla f(x_k) \quad \text{using modified cholesky fact.}$$
 - Perform linesearch satisfying Wolfe conditions
$$x_{k+1} = x_k + \alpha_k p_k$$
 - Update $s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
 - $$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)'}{s_k' B_k s_k} + \frac{y_k y_k'}{y_k' s_k}$$



Add Wolfe Condition to Linesearch

- Wolfe condition is approximation to optimality condition for the exact linesearch.

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


$$\text{Optim. Cond. } g'(\alpha) = 0 = p_k' \nabla f(x_k + \alpha p_k)$$

$$\text{want } p_k' \nabla f(x_k + \alpha p_k) \leq \eta p_k' \nabla f(x_k) \text{ for } 1 > \eta > 0$$

- Used with Armijo search condition
- 




Theorem 8.5 – global convergence

- Assumes start with symmetric pd B_0
 F is twice continuously differentiable
 X_0 forms a convex level set, and
eigenvalues of hessian on that level are
bounded and strictly positive
 - Then BFGS converges to minimizer of f .
- 



Theorem 8.6

- Assumes BFGS converges to x^* and Hessian is Lipschitz in neighborhood of x^*
 - Then quasi-Newton BFGS algorithm has superlinear convergence.
- 

Easy update of Cholesky Fact.

- Don't need to refactorize whole matrix each time. Just much simpler matrix.

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)'}{s_k' B_k s_k} + \frac{y_k y_k'}{y_k' s_k}$$

$O(n^2)$

$$= LL' - \frac{(LL' s_k)(LL' s_k)'}{s_k' LL' s_k} + \frac{y_k y_k'}{y_k' s_k}$$


$$= L \left(I - \frac{ss'}{s's} + \frac{\hat{y}\hat{y}'}{y'y} \right) L' \text{ where } s = L' s'_k, L\hat{y} = y_k$$

$$= L\tilde{L}\tilde{L}'L' \text{ where } \tilde{L} \text{ are factors of inner matrix}$$

$$= \hat{L}\hat{L}'$$



Practical considerations see pages 200-201

- Linesearches that don't satisfy Wolfe conditions may not satisfy curvature condition. Then no descent direction so need some kind of recovery strategy. (Book suggest damped Newton)
 - Can eliminate solving Newton equation
- 

Calculating H

Want $H_k = B_k^{-1}$

$$H_k = (I - \rho_k s_k y_k') H_k (I - \rho_k s_k y_k') + \rho_k s_k s_k'$$

where $\rho_k = \frac{1}{y_k' s_k}$

Book shows derivation of this directly



Finding H


Want $H_{k+1} y_k = s_k$

Such that H_{k+1} is as close as possible to H_k

$$\min \|H - H_k\|$$

subject to $H = H' \quad Hy_k = s_k$

Can go back and forth between H and B using Sherman-Morrison-Woodbury Formula (see page 605)





Quasi Newton Methods

Pros and Cons

- Globally converges to a local min
 - Superlinear convergence w/o computing Hessian
 - Works great in practice. Widely used.
 - More complicated than steepest descent
 - Best implementations require sophisticated linear algebra, linesearch, dealing with curvature conditions. Have to watch out for numerical error.
-