IE510 Applied Nonlinear Programming

Lecture 9: Quasi-Newton and BB Methods

Ruoyu Sun

Mar 29 2018. Week 10 Thur.

Review of Previous Lectures

- Methods we have learned so far:
  - Gradient descent methods
  - GD with momentum (heavy ball and Nesterov's method)
  - Decomposition-type methods: coordinate descent, SGD
- Today: methods based on approximating Newton method
- After this lecture, you should be able to
  - Properly apply BFGS method
  - Properly apply BB method
  - Know the application range of BFGS and BB methods

Outline

Pros and Cons of Newton Method

- Drawback of GD/HB/SGD/CD: convergence speed limited by some kind of condition number 100×100, $\kappa = 10^6$, 10^9. $\kappa \to \infty$.
- Recall Newton method:

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

- **Advantage** of Newton method
  - Convergence speed not necessarily limited by condition number
  - Locally quadratic convergence (see next slide)   # of iterations: 20, 50,
- **Drawback** of Newton method: expensive iteration, since inverting a matrix takes time $O(n^3)$

Quadratic and superlinear convergence

Linear convergence: $\{e_k\}$ $e_k = f(x_k) - f^*$, $e_k = \|\nabla f(x_k)\|$, etc.

$$\lim_{k\to\infty} e_k/e_{k-1} = c < 1.$$

$c \in (0,1)$.

to achieve ε, need $O(\log(\frac{1}{\varepsilon}))$ iterations

or $e_k = 1/10^k$.

e.g. $e_k = 1/2^k$, or $0.1, 0.01, 0.001, \ldots$

Quadratic convergence:

$$e_k/e_{k-1}^2 \le M, \text{ for large enough } k,$$

to achieve ε, need $O(\log\log\frac{1}{\varepsilon})$ iterations

e.g. $e_k = 1/10^{2^k}$ or 0.1, $0.01, 0.0001, 0.00000001, \ldots$

$e_k = \frac{1}{10^{2k}} = \frac{1}{100^k}$: linear rate

Superlinear convergence:

$$\lim_{k\to\infty} e_k/e_{k-1} = 0.$$

e.g. $e_k = 1/3^{2^k}$, or $e_k = 1/3^{k^2}$, or $1, 1/3^1, 1/3^4$, $1/3^9, 1/3^{16}, 1/3^{25}, \ldots$

Sublinear: $e_k = \frac{1}{k}$, $e_k = \frac{1}{k^2}$

General Framework of Quasi-Newton Method

- Instead of exact Hessian, assume we have some way to approximate $\nabla^2 f(x_k)^{-1}$, i.e.

$$H_k \approx \nabla^2 f(x_k)^{-1}.$$

- **Quasi-Newton method** framework:

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

where H_k is a certain matrix that approximates $\nabla^2 f(x_k)^{-1}$.

- Stepsize choice, often Wolfe rule ($d_k =$ $H_k \nabla f(x_k)$):

Armijo rule $f(x_k + \alpha_k d_k) \le f(x_k) + c_1 \alpha_k d_k^T \nabla f(x_k),$ (sufficient decrease)

$\nabla f(x_k + \alpha_k d_k)^T d_k \ge c_2 \nabla f(x_k)^T d_k.$ (desired curvature)

e.g. $c_1 = 10^{-4}, c_2 = 0.9.$

How to approximate Hessian?

- Many possible ideas

$\nabla^2 f(x)^{-1}$: Just use $\nabla^2 f(x_0)^{-1}$ as approximation

- 1-dim case:

$$f'(x) \approx$$

$$f''(x) \approx$$

- n-dim case: Hessian satisfies

$$\nabla^2 f(x)$$

How to approximate Hessian?

- Many possible ideas

- 1-dim case:

$$f'(x) \approx \frac{f(x+\delta) - f(x)}{\delta}$$

$$f''(x) \approx \frac{f'(x+\delta) - f'(x)}{\delta} \qquad (1)$$

- n-dim case: Hessian satisfies

$$\underbrace{\nabla^2 f(x)}_{n \times n} \cdot \underbrace{\delta}_{n \times 1} \approx \underbrace{\nabla f(x+\delta) - \nabla f(x)}_{n \times 1}$$

Secant Equation

nxn matrix

- Use B_k to approximate $\nabla^2 f(x_k)$.
- We hope B^k satisfies the secant equation

s: difference of iterates.

$$B_{k+1} s_k = y_k, \qquad (*)$$

difference of gradients

where

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

- Is there a unique B_{k+1} sastisfying the secant equation?

(*) is an equation on B_{k+1}.

\# of variables: n^2

\# of equations: n

DFP Method

- Originally proposed by **D**avidon 1959 (physist at Argonne), later analyzed/popularized by **Fl**etcher and **P**owell.

$$\begin{cases} \min_{B_{k+1}} \|B_{k+1} - B_k\| \\ \text{s.t. } B_{k+1} s_k = y_k \\ B_{k+1} \text{ is symmetric} \end{cases}$$

- Additional idea: want B^{k+1} to be close to B^k
- In a certain sense, the following B^{k+1} is the "best" ("closest" to B^k under certain metric; and satisfies the secant equation)

$$\text{DFP:} \quad B_{k+1} = (I - \rho_k y_k s_k^T) B_k (I - \rho_k s_k y_k^T) + \rho_k y_k y_k^T,$$

symmetric.

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

$(I - \rho_k s_k y_k^T) s_k = s_k - s_k \cdot \underbrace{\rho_k y_k^T s_k}_{1} = 0$;

$\underbrace{\rho_k y_k y_k^T s_k}_{1} = y_k$.

where $\rho_k = 1/(y_k^T s_k)$.

we want $\nabla^2 f^{-1} \nabla f$:

- Use $H_k = B_k^{-1}$ to approximate $\nabla^2 f(x_k)^{-1}$, then

by matrix inversion lemma,

$$\text{DFP:} \quad H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}.$$

BFGS Method

- The most popular version of quasi-Newton algorithm is BFGS method.
    - Named after Broyden, Fletcher, Goldfarb and Shanno (independent discovery)
- Instead of forcing $B_{k+1}s_k = y_k$ then inverting B_{k+1}, directly require

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

- Again, the "best" (in what sense?) H_{k+1} is the following

$$\text{BFGS:} \quad H_{k+1} = (I - \rho_k s_k y_k^T)H_k(I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

- Relation between BFGS and DFP?

Switch y_k and s_k. (dual)

Time and Storage

GD: each iteration $x_{k+1} = x_k - \alpha \nabla f(x_k)$. $O(n^2)$ for $n \times n$ least square.

$O(n)$ memory

BFGS: $x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$, with H_k computed as

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

where $\rho_k = 1/s_k^T y_k$.

How much time does per-iteration take? $O(\cancel{n^3})$.

$O(n^2)$: $H_k(I - y_k s_k^T) = H_k - \underbrace{(H_k y_k)}_{\text{matrix} \times \text{vector}} s_k^T$

Storage?

$O(n^2)$.

Global Convergence of BFGS

For strongly convex functions, BFGS (with proper stepsize) is shown to converge (under some other conditions).

Assumption 1: the level set $\mathcal{C} = \{x \mid f(x) \leq f(x_0)\}$ is convex, and f is strongly convex in $\mathcal{C}$.

If f is strongly convex in $\mathbb{R}^n$, then f satisfies Assumption 1; not vice versa.

Theorem 9.1: Suppose

- f is twice-continuously differentiable
- x_0 satisfies Assumption 1
- B_0 is any symmetric positive definite matrix

Then BFGS method with stepsize satisfying Wolfe condition converges to the global minimizer x^*.

Question: Why not "every limit point is stationary"?
Answer: Strongly convex, unique global minimizer x^*.

Local Superlinear Convergence of BFGS

Exact convergence rate of BFGS is hard to obtain.

Result: For stronlgy convex functions, BFGS with Wolfe stepsize rule achieves local superlinear convergence (under some minor conditions).

# of iterations: 5264 34 21

steepest descent	BFGS	Newton
1.827e-04	1.70e-03	3.48e-02
1.826e-04	1.17e-03	1.44e-02
1.824e-04	1.34e-04	1.82e-04
1.823e-04	1.01e-06	1.17e-08

Simulation

$f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$.

All 3 methods use Wolfe stepsize rule.

Figure: Compare Last Few Iterations of GD, BFGS, Newton

Non-Convergence of BFGS

BFGS needs NOT converge for nonconvex functions.

- See "a perfect example" in [Dai'2013] complicated example
- 4-dim nonconvex functions, each subproblem is strongly convex

  CD converges for this problem
- BFGS with several popular line search rules all fail to converge to stationary points

Other Variants of Newton Methods

Quasi-Newton methods: very popular

- Variants of BFGS: L-BFGS (Limited-memory BFGS), stochastic BFGS
- Parallel to BFGS: SR1 (Symemtric Rank 1), Broyden family

Other variants of Newton methods: subsampled Newton method (recent years)

Outline

Revisit Secant equation

- Algorithm
$$x_{k+1} = x_k - H_k \nabla f(x_k)$$
where H_k approximates $\nabla^2 f(x_k)$.
- Simplest choice? Pick $H_k = \alpha_k I$.
- Still approximately satisfies secant equation
$$H_k^{-1} \quad =$$
where $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.
- **BB stepsize 1**:
$$\alpha_k^{\text{RBB}} = \underset{H_k = \alpha I}{\operatorname{argmin}} \|H_k^{-1} s_{k-1} - y_{k-1}\|^2,$$
yielding
$$\alpha_k^{\text{LBB}} = \frac{s_{k-1}^T s_{k-1}}{y_{k-1}^T s_{k-1}}.$$

Revisit Secant equation

- Algorithm

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

where H_k approximates $\nabla^2 f(x_k)$.

- Simplest choice? Pick $H_k = \alpha_k I$.
- Still approximately satisfies secant equation

$$H_k^{-1} s_{k-1} = y_{k-1}$$

where $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

- **BB stepsize 1:**

$$\alpha_k^{\text{RBB}} = \operatorname*{argmin}_{H_k = \alpha I} \|H_k^{-1} s_{k-1} - y_{k-1}\|^2,$$

yielding

$$\alpha_k^{\text{LBB}} = \frac{s_{k-1}^T s_{k-1}}{y_{k-1}^T s_{k-1}}.$$

Revisit Secant equation

- Algorithm

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

where H_k approximates $\nabla^2 f(x_k)$.

- Simplest choice? Pick $H_k = \alpha_k I$.

- Still approximately satisfies secant equation

$$H_k^{-1} s_{k-1} = y_{k-1}$$

where $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

- **BB stepsize 1**:

$$\alpha_k^{\text{RBB}} = \operatorname*{argmin}_{H_k = \alpha I} \|H_k^{-1} s_{k-1} - y_{k-1}\|^2,$$

yielding

$$\alpha_k^{\text{LBB}} = \frac{s_{k-1}^T s_{k-1}}{y_{k-1}^T s_{k-1}}.$$

Revisit Secant equation

- Algorithm

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

where H_k approximates $\nabla^2 f(x_k)$.

- Simplest choice? Pick $H_k = \alpha_k I$.
- Still approximately satisfies secant equation

$$H_k^{-1} s_{k-1} = y_{k-1}$$

where $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

- **BB stepsize 1**:

$$\alpha_k^{\overset{L}{\cancel{\text{R}}}\text{BB}} = \operatorname*{argmin}_{H_k = \alpha I} \|H_k^{-1} s_{k-1} - y_{k-1}\|^2,$$

yielding

$$\alpha_k^{\text{LBB}} = \frac{s_{k-1}^T s_{k-1}}{y_{k-1}^T s_{k-1}}.$$

BB Stepsize Choice 2

- Another secant equation:

$$s_{k-1} = H_k y_{k-1}. \tag{1}$$

Here $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

- **BB stepsize 2**:

$$\alpha_k^{\text{SBB}} = \operatorname*{argmin}_{H_k = \alpha I} \|s_{k-1} - H_k y_{k-1}\|^2,$$

yielding

$$\alpha_k^{\text{SBB}} = \frac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}}.$$

- Meaning of “L” and “S”:

$$\alpha_{k+1}^{\text{LBB}} = \frac{s_k^T s_k}{y_k^T s_k} \geq \frac{y_k^T s_k}{y_k^T y_k} = \alpha_{k+1}^{\text{SBB}}.$$

BB Stepsize Choice 2

- Another secant equation:

$$s_{k-1} = H_k y_{k-1}. \tag{1}$$

Here $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

- **BB stepsize 2**:

$$\alpha_k^{\mathsf{SBB}} = \operatorname*{argmin}_{H_k = \alpha I} \|s_{k-1} - H_k y_{k-1}\|^2,$$

yielding

$$\alpha_k^{\mathsf{SBB}} = \frac{y_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}}.$$

- Meaning of "L" and "S":

$$\alpha_{k+1}^{\mathsf{LBB}} = \frac{s_k^T s_k}{y_k^T s_k} \geq \frac{y_k^T s_k}{y_k^T y_k} = \alpha_{k+1}^{\mathsf{SBB}}.$$

Convergence Theory of BB Method

For strongly convex quadratic problems: linear convergence is known.

For non-quadratic problems, may not converge.

Safeguard:

$$\alpha_k^{\text{BB}} = \max(\alpha_{\min}, \overset{\text{min}}{\cancel{\max}}(\alpha_{\max}, |\alpha_k^{\text{BB}}|,)).$$

$= \text{Projection of } |\alpha_k^{BB}| \text{ onto } [\alpha_{\min}, \alpha_{\max}]$

e.g $[0.01, 100]$.

Not too small, not too large.

Comment on BB Method

- It is often the best choice of stepsize for GD
- Pros: Little tuning; adaptive ; In practice, can be faster than Nesterov's gradient method.
- Cons: less theory
- Example: SparSA for compressive sensing uses BB stepsize

Comments on Unconstrained Algorithms

- Classical nonlinear programming put emphasis on (nonlinear) conjugate gradient methods, quasi-Newton methods
- Line search is often a default choice
- GD with constant stepsize, BB method, CD and SGD are less emphasized
- In this class, we shift the focus mainly due to large-scale optimization needs

Outline

Summary

In this lecture, we learned the following:

- BFGS method and BB method
- Convergence theory of BFGS and BB method
- Pros and Cons of BFGS and BB methods

Summary

In this lecture, we learned the following:

- BFGS method and BB method
- Convergence theory of BFGS and BB method
- Pros and Cons of BFGS and BB methods

References and Further Reading

Wright, Nocedal, Numerical Optimization, 1999.

- Chapter 6 quasi-Newton method

Bertsekas, Nonlinear Programming (textbook), subsection on “Quasi-Newton method” (Sec 1.7 in 1999 version).

Fletcher 2005, On the Barzilai-Borwein Method