# Chapter 6: Newton method for large-scale problems.

Newton method: $x^{k+1} = x^k - \nabla^2 f(x^k)^{-1} \cdot \nabla f(x^k)$

Newton direction: solving linear system

$$\nabla^2 f(x^k) \cdot p^k = \nabla f(x^k)$$

"Inexact" Newton method:

Use another iterative solver for solving

$$0 = p^{k,0},\ p^{k,1},\ p^{k,2}, \cdots, p^{k,t}, \cdots$$

Until $p^{k,t}$ satisfy the stopping condition.

A typical stopping condition: $\to r^{k,0}$

$$\| r^{k,t} \| \le \sigma_k \cdot \| \nabla f(x^k) \|$$

$$r^{k,t} = \nabla^2 f(x^k) \cdot p^{k,t} - \nabla f(x^k)$$

$\{\sigma_k\}$: forcing sequence.

Inexact Newton method:

For $k = 0, 1, \cdots$

- Solve $\nabla^2 f(x^k) \cdot p^k = \nabla f(x^k)$ to get $p^k$ (use CG)
  until $\| \nabla^2 f(x^k) p^k - \nabla f(x^k) \| \le \sigma_k \| \nabla f(x^k) \|$
- $x^{k+1} = x^k - \eta^k \cdot p^k$ where $\eta^k$ by line search.

end.

When using CG, only need $\underbrace{\nabla^2 f(x^k)}_{n \times n} \cdot \underbrace{p^k}_{n \times 1}$ at each iteration.

Recall: convergece rate.

- Gradiend descent: linear convergence.

$$\lim_{k\to\infty} \frac{f(x^{k+1}) - f(x^*)}{f(x^k) - f(x^*)} \le C. \quad 1 > C > 0$$

- Newton method: (exact. $\nabla^2 f(x^k)^{-1} \nabla f(x^k)$)

$$\lim_{k\to\infty} \frac{f(x^{k+1}) - f(x^*)}{(f(x^k) - f(x^*))^2} \le C \ , \ C > 0$$

$$\lim_{k\to\infty} \frac{\| \nabla f(x^{k+1}) \|}{\| \nabla f(x^k) \|^2} \le C$$

Thm: If $f$ is twice continuous differentiable. Then

① $\{\sigma_k\} \to 0$, then inexact Newton has "superlinear" convergence rate.

e.g. $1/k$

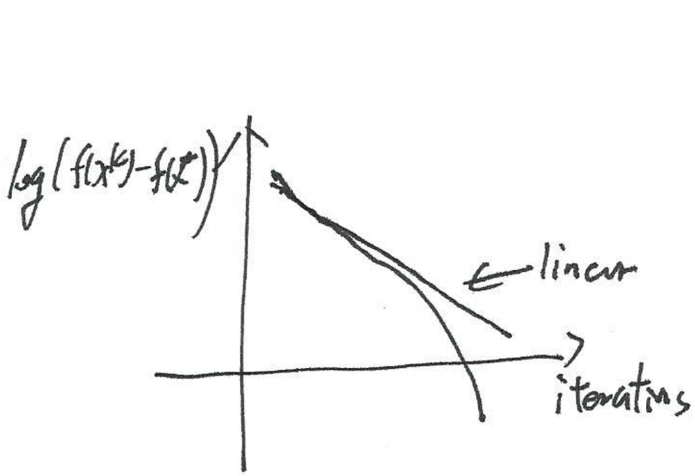
② $\sigma_k = O(\|\nabla f(x^k)\|)$, then inexact Newton has "quadratic" convergence rate.

$= C_1 \cdot \|\nabla f(x^k)\|$

Superlinear:
$$\lim_{k\to\infty} \frac{f(x^{k+1}) - f(x^*)}{f(x^k) - f(x^*)} = 0$$

Quadrac convergence:

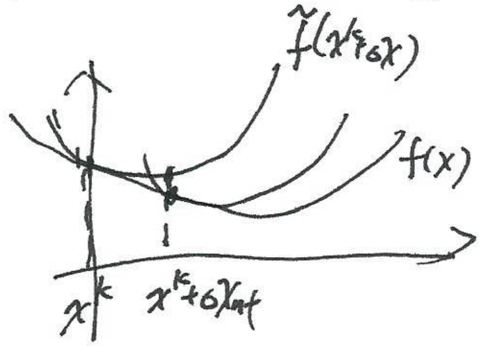
$$\lim_{k\to\infty} \frac{f(x^{k+1}) - f(x^*)}{(f(x^k) - f(x^*))^2} \le C.$$



Quasi-Newton method.

- Newton method.
  - Each iteration. form a quadratic approximation

$$\tilde{f}_k(x^k+\Delta x) = f(x^k) + \nabla f(x^k)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x^k) \Delta x$$




$$\Delta x_{nt} := \underset{\Delta x}{\arg\min}\ \tilde{f}_k(x^k+\Delta x)$$

$$\nabla \tilde{f}(x^k+\Delta x_{nt}) = 0$$

$$\nabla f(x^k) + \nabla^2 f(x^k)\Delta x_{nt} = 0$$

$$\Delta x_{nt} = \underline{-\nabla^2 f(x^k)^{-1} \cdot \nabla f(x^k)}.$$

Quasi-Newton : Replace. $\nabla^2 f(x^k)$ by some $B_k$

st $B_k \succ 0 I$

$$\Delta x_{nt}' = -\underline{B_k}^{-1} \cdot \nabla f(x^k)$$

$$x^{k+1} = x^k + \eta^k \cdot \Delta x_{nt}'$$

How to choose $B_k$?

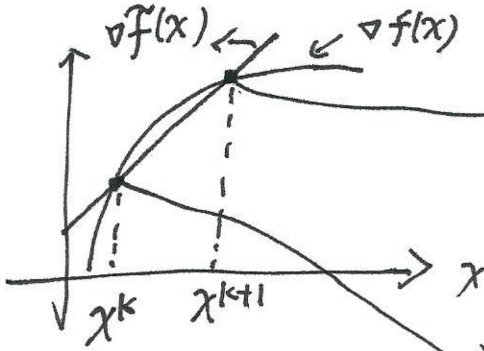
$\boxed{\text{BFGS}}$ : only use the information $\{\nabla f(x^0), \nabla f(x^1), \nabla f(x^2) \cdots ; \nabla f(x^k)\}$

$$\tilde{f}(x^{k+1}+\Delta x) = f(x^{k+1}) + \nabla f(x^{k+1})^T \Delta x + \frac{1}{2}\Delta x^T \underline{B_{k+1}} \Delta x$$

Want $\nabla \tilde{f}(x^k) = \underline{\nabla f(x^k)}$

the gradient for the approximate function at $x^k$

real gradient at $x^k$




$$\underbrace{\nabla \tilde{f}(x^{k+1}) = \nabla f(x^{k+1})}$$

$$\nabla \tilde{f}(x^{k+1} + \Delta x) = \nabla f(x^{k+1}) + B_{k+1}(\Delta x)$$

$$\boxed{\nabla \tilde{f}(x^k) = \nabla f(x^k)}$$

$$\Delta x = x^k - x^{k+1}$$

$$\nabla f(x^{k+1}) + B_{k+1}(x^k - x^{k+1}) = \nabla f(x^k)$$

$$B_{k+1}(x^{k+1} - x^k) = \nabla f(x^{k+1}) - \nabla f(x^k)$$

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

$$\underbrace{B_{k+1} \cdot s_k = y_k} \quad \boxed{\text{Secant equation}}$$

① There exists a pd solution $B_{k+1}$ iff $s_k^T y_k > 0$

($\Rightarrow$) $s_k^T B_{k+1} s_k = s_k^T y_k > 0$

($\Leftarrow$) $B_{k+1}$ : $n^2/2$ freedom.

Secant equation : $n$ equalities.

pd : constraint.

② If $f(x)$ is strongly convex, then $s_k^T y_k > 0$
(strictly)

BFGS:

$$x^{k+1} = x^k - B_{k+1}^{-1} \cdot \nabla f(x^k)$$

$H_{k+1} = B_{k+1}^{-1}$. Secant equation.

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

$$\boxed{H_{k+1} y_k = s_k}$$ Choosing $H_{k+1}$ is equivalent to choosing $B_{k+1}$

BFGS: select $H_{k+1}$ by

$$\underset{H}{\operatorname{argmin}} \|H - H_k\|_W \quad \text{st} \quad s_k = H \cdot y_k \ . \ \text{and} \ H > 0I$$

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

where $\rho_k = 1/y_k^T s_k$ $\quad (s_k = x^{k+1} - x^k, \ y_k = \nabla f(x^{k+1}) - \nabla f(x^k))$

Usually $H_0 = \lambda I$ for some $\lambda > 0$

$$H_0 \to H_1 \to H_2 \to \cdots\cdots$$

Algorithm: (BFGS)

- Initial $x^0, H^0$
- For $k = 0, 1, \cdots$

  Compute $p_k = -H_k \cdot \nabla f(x^k)$

  Set $x^{k+1} = x^k + \eta^k \cdot p_k$ ($\eta^k$ by line search)

  Compute $s_k = x^{k+1} - x^k$, $y_k = \nabla f(x^{k+1}) - \nabla f(x^k)$

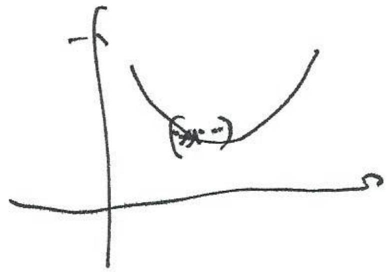
  Compute $H_{k+1} = \underbrace{(I - \rho_k s_k y_k^T)}_{V_k} H_k \underbrace{(I - \rho_k y_k s_k^T)}_{V_k^T} + \rho_k s_k s_k^T$

end.

Thm: If $f$ is twice continuous differentiable, and

$$mI \preceq \nabla^2 f(x) \preceq MI \text{ for some } m, M > 0, \forall x.$$

Then $\{x^k\} \to x^*$ superlinearly.



Computation & storage

Computation: $\nabla f(x^k)$

$$H_{k+1} = V_k^T H_k V_k^T + \rho s_k s_k^T$$

$$H_{k+1} \cdot \nabla f(x^k) \leftarrow O(n^2)$$

Storage: store $s_k, y_k$ $\forall k$ in memory.

$$\underset{\substack{\uparrow \\ \#\text{iterations}}}{O(k \cdot n)} \text{ memory}$$

# Limited memory BFGS (L-BFGS)

Revisit the BFGS update for $H_{k+1}$

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

$$
\begin{aligned}
H_0 &= H_0 \\
H_1 &= V_0^T H_0 V_0 + \rho_0 s_0 s_0^T \\
H_2 &= V_1^T V_0^T H_0 V_0 V_1 + \rho_0 V_1^T s_0 s_0^T V_1 + \rho_1 s_1 s_1^T \\
H_3 &= V_2^T V_1^T V_0^T H_0 V_0 V_1 V_2 + \rho_0 V_2^T V_1^T s_0 s_0^T V_1 V_2 + \rho_1 V_2^T s_1 s_1^T V_2 \\
&\quad + \rho_2 s_2 s_2^T \\
&\ \ \vdots
\end{aligned}
$$

$$
\begin{aligned}
H_k &= V_{k-1}^T V_{k-2}^T \cdots V_0^T H_0 V_0 V_1 \cdots V_{k-1} \\
&\quad + \rho_0 V_{k-1}^T V_{k-2}^T \cdots V_1^T s_0 s_0^T V_1 \cdots V_{k-1} \\
&\quad + \rho_1 V_{k-1}^T \cdots V_2^T s_1 s_1^T V_2 V_3 \cdots V_{k-1} \\
&\quad \ \vdots \\
&\quad + \rho_{k-2} V_{k-1}^T s_{k-2} s_{k-2}^T V_{k-1} \\
&\quad + \rho_{k-1} s_{k-1} s_{k-1}^T
\end{aligned}
$$

$n \times n$

$q$, $q$, $k$

$q,\ V_{k-1} q,\ V_{k-2} V_{k-1} q,\ \ldots\ V_1 V_2 \cdots V_{k-1} q$

---

How to compute $H_k \cdot q$?

$r = q$
For $i = k-1, \cdots, 0$
 $\alpha_i = s_i^T r$
 $r = V_i r$
end

$r = H_0 r$,
For $i = 0, 1, 2, \cdots k-1$
 $r = V_i^T r$
 $r = r + \rho_i s_i \alpha_i$
end

In this algo. each step: $V_i r$ or $V_i^T r$

$$\cancel{V_i = \rho_i s_i y_i}$$

$$V_i = I - \rho_i s_i y_i^T$$

$$V_i \cdot r = r - \rho_i s_i (y_i^T r) \Rightarrow O(n)$$

The total time complexity: $O(nk) \ll O(n^2)$

$$V_i^T r = (I - \rho_i y_i s_i^T) r$$
$$= r - \rho_i y_i (s_i^T r)$$

L-BFGS: Only store $m$ vectors in memory

BFGS: $H^0 \to H^1 \to H^2 \to \cdots H^k$

$\quad\; s^0 y^0 \quad s^1 y^1$

L-BFGS: $\cancel{H}$ $\quad H_{k-m} \to H_{k-m+1} \to \cdots \to H_k$

$\quad\quad\quad\quad \gamma I \quad\quad 2 \quad\quad 2 \quad\quad 2$

$O(2 \cdot m \cdot n)$ memory space

$$H_k = V_{k-1}^T V_{k-2}^T \cdots V_{k-m}^T H_0^k V_{k-m} V_{k-m+1} \cdots V_{k-1}$$
$$+ \rho_{k-m} V_{k-1}^T \cdots V_{k-m+1}^T s_{k-m} s_{k-m}^T V_{k-m+1} \cdots V_{k-1}$$
$$+ \rho_{k-m+1} \cdots\cdots$$
$$\vdots$$
$$+ \rho_{k-1} s_{k-1} s_{k-1}^T$$

Usually. Set $H_0^k = rI$ for some $r > 0$.

Time complexity for $H_k \cdot p$ : $O(mn)$

Space complexity : $O(mn)$

Convergence $\Rightarrow$ linear convergence.