Newton & Quasi-Newton Method

Feb. 16. 2022

Quiz:

① Newton's method under appropriate conditions & assumptions can converge quadratically.

A. True

B. False.

② In BFGS method, in every iteration we need to keep the "hessian" approximate positive definite.

A. True.

B. False.

Newton's Method

- Linear cge: $\lim_{k\to\infty} \frac{|x_{k+1}-x^*|}{|x_k-x^*|} = u \in (0,1)$.
- Super-linear cge: $\lim_{k\to\infty} \frac{|x_{k+1}-x^*|}{|x_k-x^*|} = 0$
- Quadratic convergence: $\lim_{k\to\infty} \frac{|x_{k+1}-x^*|}{|x_k-x^*|^2} < M$.

Optimization problem:

$\min_x : f(x)$ (f twice cts diff).

* replace $f(x)$ by quad approx:

$m_k(x) = f(x_k) + \nabla f(x_k)^T (x-x_k) + \frac{1}{2}(x-x_k)^T \nabla f(x_k)(x-x_k)$

or let $s = x - x_k$

$= f(x_k) + \nabla f_k^T s + \frac{1}{2} s^T B_k s$

if non-convex, what can happen ?

If $m_k(x)$ is convex, then to min

$\Rightarrow B_k s + \nabla f_k = 0$

If B_k is invertable,

$\Rightarrow s_k = -\underbrace{B_k^{-1}}_{\text{hessian}}(\nabla f_k)$ (*1)

* local rate of convergence:

Suppose f is twice diff, and $\nabla^2 f(x)$ is L-lip-cts in a neighbourhood of a slon x^*, where $\nabla^2 f(x^*) \succ 0$, $\nabla f(x^*) = 0$ (x^*: strict local min)

Newton's method (*1) $\overset{\text{cge}}{\Rightarrow} x^*$ quadratically.

pf: $x_{k+1} - x^* = x_k + s_k - x^*$

$= x_k - x^* - B_k^{-1}(\nabla f_k)$

$= B_k^{-1}\left(B_k(x_k - x^*) - (\nabla f_k - \nabla f_*)\right)$

By taylor's thm:

$\nabla f_k - \nabla f_* = \int_0^1 \nabla^2 f(x_k + t(x^* - x_k))(x_k - x^*)\, dt$

$\therefore \|B_k(x_k - x^*) - (\nabla f_k - \nabla f_*)\|$

$= \|\int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))\right)(x_k - x^*)\, dt\|$

$\le \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))\| \|x_k - x^*\|\, dt$

$\le \int_0^1 Lt \|x^* - x_k\|^2\, dt$

$= \frac{L}{2}\|x^* - x_k\|^2$

* We can now choose r so small that $\|x_k - x^*\| \le r$,

$\|\nabla^2 f_k^{-1}\| \le 2\|\nabla^2 f(x^*)^{-1}\|$ ($\because \nabla^2 f(x^*) \succ 0$).

$$\therefore \|x_{k+1} - x^*\| \le \frac{L}{2} \|B_k^{-1}\| \|x^* - x_k\|^2$$
$$\le \underbrace{\frac{L}{2} \|\nabla^2 f(x^*)^{-1}\|}_{\text{constant}} \cdot \|x^* - x_k\|^2.$$

#.

Quasi-Newton method:

Main drawback of Newton's method:
if dim is large, Hessian is expensive to obtain!

Quasi-Newton is a great alternative!
- NO hessian needed, just "approx" it.
- can still have superlinear convergence.

* Most popular Quasi-Newton: BFGS.
- update B_k or $H_k (\succ 0)$ at every iteration.
- $x_{k+1} = x_k + \alpha_k (-H_k \nabla f_k)$
  (line search) $\cdot$ p_k.

How to update B_k and get B_{k+1}?

$$m_{k+1}(p) = f_{k+1} + \nabla f_{k+1}^T p + \frac{1}{2} p^T B_{k+1} p.$$

reasonable to require: ∇m_{k+1} and ∇f match at x_{k+1} and x_k.

x_{k+1} – $\nabla m_{k+1}(0) = \nabla f_{k+1}$ matches $\nabla f(x_{k+1})$.

x_k – $\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_k \Rightarrow B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k$

Let: $s_k = x_{k+1} - x_k = \alpha_k p_k$, $y_k = \nabla f_{k+1} - \nabla f_k$.

$\therefore B_{k+1} s_k = y_k$ (secant equation).

Q1: This can only happen if $s_k^T y_k > 0$. why?

But if f is strongly convex, it's always true.

Q2: why for strongly convex functions $s_k^T y_k > 0$?

For (strongly) convex functions:

$$\begin{cases} f(y) \geq f(x) + \nabla f(x)\cdot(y-x) + \frac{u}{2}\|y-x\|^2 \\ f(x) \geq f(y) + \nabla f(y)\cdot(x-y) + \frac{u}{2}\|y-x\|^2 \end{cases}$$

$$\Rightarrow 0 \geq (\nabla f(y) - \nabla f(x))\cdot(x-y) + u\|y-x\|^2$$

$$\Rightarrow \langle \nabla f(x) - \nabla f(y), x-y \rangle \geq u\|y-x\|^2$$

(For strongly convex fun, inequality is strict)

- For non-convex f, we need to enforce it.
(by line search: Wolfe conditions $\Rightarrow$ $f(x_k+\alpha_k p_k) \leq f(x_k) + c_1\alpha_k \nabla f_k^T p_k$, $\nabla f(x_k+\alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$, $(0<c_1<c_2<1)$)

- Given $s_k^T y_k > 0$, we can update B_k!
$n(n+1)/2$ degree of freedom, $B_{k+1}s_k = y_k$ takes n.
We can have many choices of B_{k+1}!

so we want to make minimal adjustment to B_k, (satisfy $B_{k+1}s_k = y_k$), at same time keep $B_{k+1} \succ 0$.
turns out satisfy needs:

$$B_{k+1} = B_k + \underbrace{\frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}}_{\text{rank 2 update}}$$

(*) By Sherman-Morrison formula: update $H_{k+1} = B_{k+1}^{-1}$.

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

(BFGS)

Given starting point x_0, convergence tolerance $\epsilon > 0$,
inverse Hessian approximation H_0;
$k \leftarrow 0$;
while $\|\nabla f_k\| > \epsilon$;
Compute search direction

$$p_k = -H_k \nabla f_k;$$

Set $x_{k+1} = x_k + \alpha_k p_k$ where α_k is computed from a line search procedure to satisfy the Wolfe conditions (3.6); (← back track LS)
Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
Compute H_{k+1} by means of (6.17);
$k \leftarrow k + 1$;
end (while)

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

Each iteration has cost $O(n^2)$, not $O(n^3)$!

θ_3: Do you think the requirement $B_k, H_k \succ 0$ make sense?

Superlinear convergence:

Rosenbrock function: $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

$\|x_k - x^*\|$

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

itr: 5264 34 21
(till $\|\nabla f\| \le 10^{-5}$)

Thm: Let $B_0 \succ 0$, x_0 any starting pt with:

(i) *The objective function f is twice continuously differentiable.*

(ii) *The level set $\mathcal{L} = \{x \in \mathbb{R}^n \mid f(x) \le f(x_0)\}$ is convex, and there exist positive constants m and M such that*

$$m\|z\|^2 \le z^T G(x) z \le M\|z\|^2 \tag{6.39}$$

for all $z \in \mathbb{R}^n$ and $x \in \mathcal{L}$.

then BFGS converges to a minimizer x^*, if the

hessian is L-Lip near x^* then:
BFGS converges super-linearly.
(or $\sum_{k=1}^{\infty} \|x_k - x^*\| < \infty$)

Limited memory BFGS

What if our dim is huge? storing the "hessian" matrix will be too expensive.

* So now we want to get rid of storing H_k.
* H_k is fully determined by H_0, s_i's, y_i's. (just need to do some algebra).

This allows us to not remember H_k's, and also re-initialization of H_k's.

* Re-initialization:

Instead of going back to $H_0 = I$, just use H_0 and last "m" s_i's and y_i's.
(m is some fixed small int).

* L-BFGS:

Simply do this for every iteration, and only work with the last "m" s_i, and y_i's.

Performs very well in practice, but no theoretical guarantees!