Quasi-Newton Methods

Oswin Krause, NO, 2022

KØBENHAVNS UNIVERSITET

This week: Trust-Region Newton (Idea)

1. Set $m(p) = f(x) + p^T \nabla f(x) + \frac{1}{2} p^T \nabla^2 f(x) p$ (second order Taylor)
2. $p = \min_{p'} m(p')$ such, that $\|p\| \leq \Delta$
3. Adjust Δ based on how well $m(p)$ approximates $f(x+p)$
4. If $f(x+p)$ sufficiently better than $f(x)$
   4.1 Set $x \to x + p$
5. Go to 1.

Idea: Quasi Newton

- Computing $\nabla^2 f(x)$ is expensive
- Value limited if $\nabla^2 f(x)$ not PD.
- If PD, computing the inverse is expensive

Idea: Quasi Newton

- Computing $\nabla^2 f(x)$ is expensive
- Value limited if $\nabla^2 f(x)$ not PD.
- If PD, computing the inverse is expensive
- Quasi Newton Method
    - approximates $B_k \approx \nabla^2 f(x_k)$, $k = 1, \dots$ using gradient information only
    - and $B_k \to \nabla^2 f(x^*)$ as $x_k \to x^*$
- Quasi Newton-Methods have super-linear convergence

Quasi-Newton Trust-Region (Idea)

1. Set $m(p) = f(x) + p^T \nabla f(x) + \frac{1}{2} p^T B p$ (Approximated second order Taylor)
2. $p = \min_{p'} m(p')$ such, that $\|p'\| \leq \Delta$
3. Adjust Δ based on how well $m(p)$ approximates $f(x+p)$
4. Adjust B based on $\nabla f(x)$ and $\nabla f(x+p)$
5. If $f(x+p)$ sufficiently better than $f(x)$
   5.1 Set $x \to x + p$
6. Go to 1.

Secant Equation

1. Idea: model m should predict the change of gradient of the last step

$$\underbrace{\nabla f(x_{k+1}) - \nabla f(x_k)}_{y_k} = \nabla m(\underbrace{x_{k+1} - x_k}_{p_k}) - \nabla m(0)$$

Secant Equation

1. Idea: model m should predict the change of gradient of the last step

$$\underbrace{\nabla f(x_{k+1}) - \nabla f(x_k)}_{y_k} = \nabla m(\underbrace{x_{k+1} - x_k}_{p_k}) - \nabla m(0)$$

2. Inserting gradient $\nabla m(p) = Bp + \nabla f(x_k)$:

$$y_k = (Bp_k + \nabla f(x_k)) - \nabla f(x_k) = Bp_k$$

Secant Equation

1. Idea: model m should predict the change of gradient of the last step

$$\underbrace{\nabla f(x_{k+1}) - \nabla f(x_k)}_{y_k} = \nabla m(\underbrace{x_{k+1} - x_k}_{p_k}) - \nabla m(0)$$

2. Inserting gradient $\nabla m(p) = Bp + \nabla f(x_k)$:

$$y_k = (Bp_k + \nabla f(x_k)) - \nabla f(x_k) = Bp_k$$

3. Secant Equation:

$$Bp_k = y_k$$

Note: the book writes $s_k = x_{k+1} - x_k$, we will use that from now on.

Inverse Secant Equation

1. Let B invertible and $H = B^{-1}$
2. Multiply H from the left:

$$HBs_k = Hy_k$$

Inverse Secant Equation

1. Let B invertible and $H = B^{-1}$
2. Multiply H from the left:

$$HBs_k = Hy_k$$

3. Since $HB = I$

$$s_k = Hy_k$$

Inverse Secant Equation

1. Let B invertible and $H = B^{-1}$
2. Multiply H from the left:
$$HBs_k = Hy_k$$
3. Since $HB = I$
$$s_k = Hy_k$$
4. Quasi-Newton methods can be derived based on the Secant or Inverse Secant Equation.

Updating of B

- We looked at sequence $x_1, x_2, \ldots$

Updating of B

- We looked at sequence $x_1, x_2, \ldots$
- Now we will also add sequence $B_1, B_2, \ldots$

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

- Can we use the Secant Equation to update B_k?

First Attempt: Symmetric-Rank-One update (SR1)

- Assume form
$$B_{k+1} = B_k + \sigma_k q_k q_k^T$$
- $\sigma_k \in \mathbb{R}$, $q_k \in \mathbb{R}^n$

First Attempt: Symmetric-Rank-One update (SR1)

- Assume form

$$B_{k+1} = B_k + \sigma_k q_k q_k^T$$

- $\sigma_k \in \mathbb{R}$, $q_k \in \mathbb{R}^n$
- Secant equation:

$$B_{k+1} \underbrace{s_k}_{x_{k+1}-x_k} = \underbrace{y_k}_{\nabla f(x_{k+1})-\nabla f(x_k)}$$

- Can we find σ_k, q_k?

First Attempt: Symmetric-Rank-One update (SR1)

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

First Attempt: Symmetric-Rank-One update (SR1)

$$\begin{aligned} B_{k+1}s_k &= y_k \\ \Leftrightarrow \left(B_k + \sigma_k q_k q_k^T\right) s_k &= y_k \end{aligned}$$

First Attempt: Symmetric-Rank-One update (SR1)

$$
\begin{aligned}
B_{k+1}s_k &= y_k \\
\Leftrightarrow \left(B_k + \sigma_k q_k q_k^T\right) s_k &= y_k \\
\Leftrightarrow B_k s_k + \sigma_k q_k \left(q_k^T s_k\right) &= y_k
\end{aligned}
$$

First Attempt: Symmetric-Rank-One update (SR1)

$$\begin{aligned} B_{k+1}s_k &= y_k \\ \Leftrightarrow \left(B_k + \sigma_k q_k q_k^T\right) s_k &= y_k \\ \Leftrightarrow B_k s_k + \sigma_k q_k \left(q_k^T s_k\right) &= y_k \\ \Leftrightarrow \left(\sigma_k q_k^T s_k\right) q_k &= y_k - B_k s_k \end{aligned}$$

First Attempt: Symmetric-Rank-One update (SR1)

$$
\begin{aligned}
B_{k+1}s_k &= y_k \\
\Leftrightarrow \left(B_k + \sigma_k q_k q_k^T\right) s_k &= y_k \\
\Leftrightarrow B_k s_k + \sigma_k q_k \left(q_k^T s_k\right) &= y_k \\
\Leftrightarrow \underbrace{\left(\sigma_k q_k^T s_k\right)}_{\text{scalar}} \underbrace{q_k}_{\text{vector}} &= \underbrace{y_k - B_k s_k}_{\text{vector}}
\end{aligned}
$$

For the equality to hold q_k must be a multiple of $y_k - B_k s_k$.
Since we can choose σ_k arbitrarily, set $q_k = y_k - B_k s_k$.
Equation holds for $\sigma_k = \frac{1}{q_k^T s_k}$

First Attempt: Symmetric-Rank-One update (SR1)

We have

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

Observations:

- $y_k - B_k s_k$: Difference of predicted gradient change to observed change
- Issue: If model is perfect, we divide by zero!

First Attempt: Symmetric-Rank-One update (SR1)

We have

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

Observations:

- $y_k - B_k s_k$: Difference of predicted gradient change to observed change
- Issue: If model is perfect, we divide by zero!
- Eigenvalues of B_{k+1} can be negative:

$$\begin{aligned} B_{k+1} s_k &= y_k \\ \Rightarrow s_k^T B_{k+1} s_k &= \underbrace{s_k^T y_k}_{\text{Can be } < 0} \end{aligned}$$

Trust-Region SR1

1. Set $m(p) = f(x) + p^T \nabla f(x) + \frac{1}{2} p^T B p$
2. $p = \min_{p'} m(p')$ such, that $\|p'\| \leq \Delta$
3. Adjust Δ based on how well $m(p)$ approximates $f(x+p)$
4. Update B with $y = \nabla f(x+p) - \nabla f(x)$, if $|(y - Bp)^T p| > \delta$

$$B \to B + \frac{(y - Bp)(y - Bp)^T}{(y - Bp)^T p}$$

5. If $f(x+p)$ sufficiently better than $f(x)$
   5.1 Set $x \to x + p$
6. Go to 1.

Trust-Region SR1

Pro:

- Pro:
  - Simple Adaptation of Trust-Region Method
  - Can show: Given large enough initial Δ, converges after maximal n steps on n-dim Ellipsoid, independent of initial B
  - Arbitrary f: B_k converges to true hessian close to optimum
- Con:
  - Solving the sub-problem is still expensive

Quasi-Newton Line-Search Algorithms

- Idea:
  - B_k PD $\Rightarrow$ Can just use Newton-Step with B_k!
- We need
  - update equation that ensures B_k PD
  - fast inversion formula

Quasi-Newton Line-Search Algorithms

1. Set $m(p) = f(x) + p^T \nabla f(x) + \frac{1}{2} p^T B p$
2. Pick $p = -B^{-1} g = -Hg$
3. Find α such, that $f(x + \alpha p)$ fulfills Wolfe conditions
4. Update $H = B^{-1}$ using $\nabla f(x), \nabla f(x + \alpha p)$ such, that H is PD.
5. Set $x \to x + \alpha p$ and go to 1.

Why Wolfe Conditions?

From the Secant Condition it must hold:

$$s_k B_{k+1} s_k = y_k^T \underbrace{s_k}_{x_{k+1}-x_k=\alpha p}$$

If $y_k^T s_k \leq 0$, B_{k+1} can not be PD

Why Wolfe Conditions?

From the Secant Condition it must hold:

$$s_k B_{k+1} s_k = y_k^T \underbrace{s_k}_{x_{k+1} - x_k = \alpha p}$$

If $y_k^T s_k \leq 0$, B_{k+1} can not be PD

Show that (8min)

$$\underbrace{(\nabla f(x + \alpha p) - \nabla f(x))^T p}_{y_k^T p} > 0$$

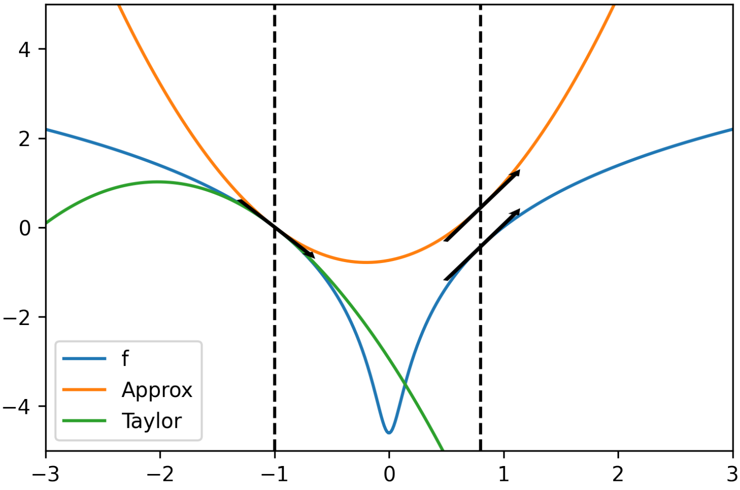
when α is selected by Wolfe Conditions $(0 < c_1 < c_2 < 1)$:

$$\nabla f(x + \alpha p) \leq f(x) + c_1 \alpha \nabla f(x)^T p$$
$$\nabla f(x + \alpha p)^T p \geq c_2 \nabla f(x)^T p$$

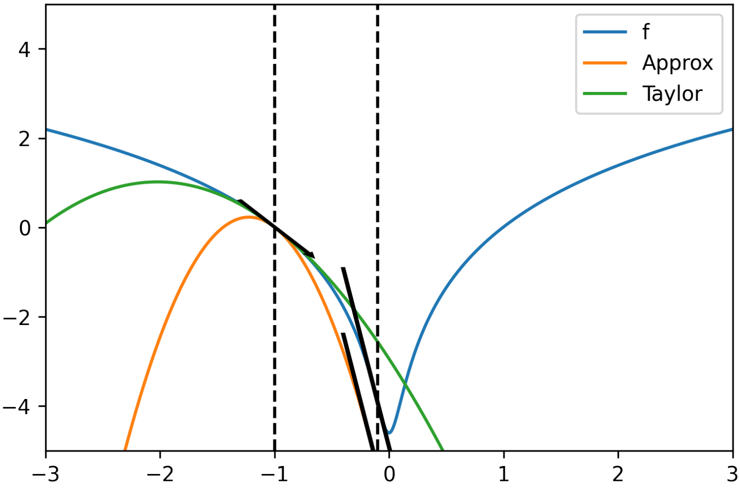
Solution:

$$\begin{aligned}
&\nabla f(x+\alpha p)^T p \geq c_2 \nabla f(x)^T p \\
\Leftrightarrow\ &\nabla f(x+\alpha p)^T p - \nabla f(x)^T p \geq c_2 \nabla f(x)^T p - \nabla f(x)^T p \\
\Leftrightarrow\ &(\nabla f(x+\alpha p) - \nabla f(x))^T p \geq \underbrace{(c_2 - 1)}_{<0} \underbrace{\nabla f(x)^T p}_{<0} \\
\Leftrightarrow\ &(\nabla f(x+\alpha p) - \nabla f(x))^T p > 0
\end{aligned}$$

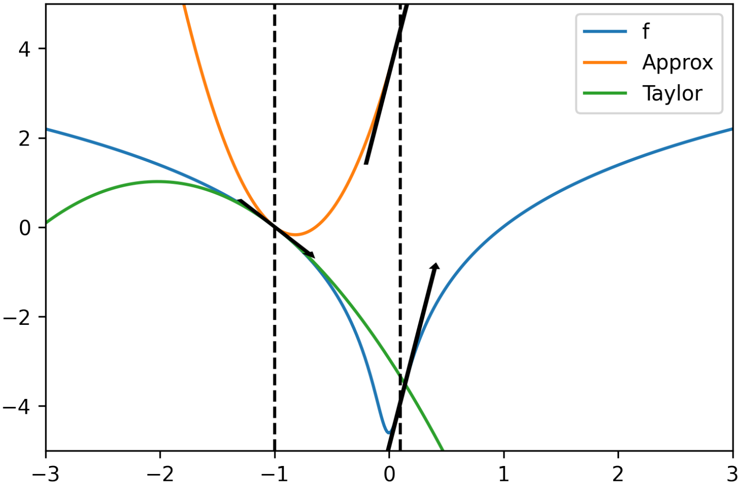
Why Wolfe Conditions? Visualized



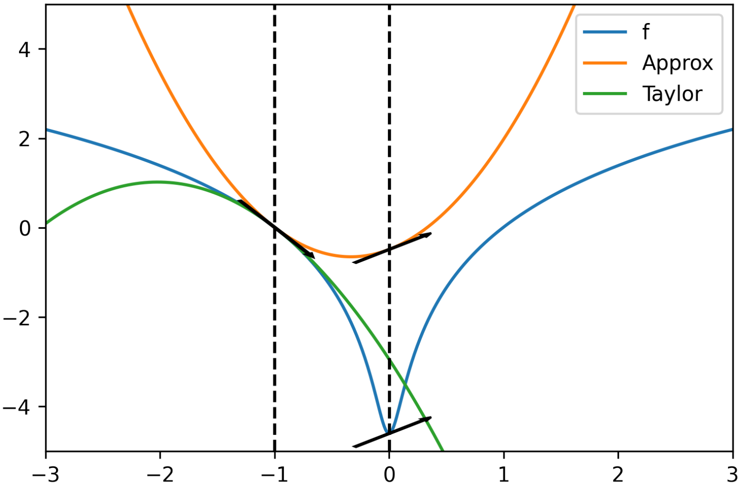
A point not fulfilling curvature condition



Wolfe is not perfect!



Strong Wolfe is better!



Is SR1 With Wolfe Enough?

No!

- Can still come up with cases where SR1 is not PD
- Theoretical Assignment this week!
- Our SR1 update has no degrees of freedom left, update is unique.

$\Rightarrow$ We need a more powerful update

Symmetric-Rank-Two updates

- Assume form with two rank-1 updates:

$$B_{k+1} = \underbrace{B_k + \gamma_k r_k r_k^T}_{B_k'} + \sigma_k q_k q_k^T$$

- For any choice of γ_k, r_k, we can still use the SR1 update to obtain σ_k, q_k:

$$B_{k+1} = B_k' + \frac{(y_k - B_k' s_k)(y_k - B_k' s_k)^T}{(y_k - B_k' s_k)^T s_k}$$

- Which γ_k, r_k should we pick?

Symmetric-Rank-Two updates

- Idea: Ensure $(y_k - B'_k s_k)^T s_k = \underbrace{y_k^T s_k > 0}_{\text{Wolfe Curvature}}$

Symmetric-Rank-Two updates

- Idea: Ensure $(y_k - B'_k s_k)^T s_k = \underbrace{y_k^T s_k > 0}_{\text{Wolfe Curvature}}$

$\Rightarrow s_k^T B'_k s_k = 0$

Symmetric-Rank-Two updates

- Idea: Ensure $(y_k - B'_k s_k)^T s_k = \underbrace{y_k^T s_k > 0}_{\text{Wolfe Curvature}}$

$\Rightarrow s_k^T B'_k s_k = 0$

$$\begin{aligned} 0 =& s_k^T B'_k s_k \\ =& s_k^T (B_k + \gamma_k r_k r_k^T) s_k \\ =& s_k^T B_k s_k + \gamma_k (s_k^T r_k)^2 \end{aligned}$$

Symmetric-Rank-Two updates

- Idea: Ensure $(y_k - B'_k s_k)^T s_k = \underbrace{y_k^T s_k > 0}_{\text{Wolfe Curvature}}$

$\Rightarrow s_k^T B'_k s_k = 0$

$$\begin{aligned} 0 = & s_k^T B'_k s_k \\ = & s_k^T (B_k + \gamma_k r_k r_k^T) s_k \\ = & s_k^T B_k s_k + \gamma_k (s_k^T r_k)^2 \end{aligned}$$

$\Rightarrow \gamma_k = -\frac{s_k^T B_k s_k}{(s_k^T r_k)^2}$

- Need to pick appropriate r_k

Broyden–Fletcher–Goldfarb–Shannon (BFGS) Algorithm

- BFGS picks $r_k = B_k s_k$, because:

$$
\begin{aligned}
B'_k s_k &= B_k s_k + \gamma_k r_k (r_k^T s_k) \\
&= B_k s_k - \frac{s_k^T B_k s_k}{(s_k^T B_k s_k)^2} B_k s_k (s_k^T B_k s_k) \\
&= B_k s_k - B_k s_k = 0
\end{aligned}
$$

Broyden–Fletcher–Goldfarb–Shannon (BFGS) Algorithm

- BFGS picks $r_k = B_k s_k$, because:

$$B'_k s_k = 0$$

Broyden–Fletcher–Goldfarb–Shannon (BFGS) Algorithm

- BFGS picks $r_k = B_k s_k$, because:

$$B'_k s_k = 0$$

- Thus, the SR1 update becomes:

$$\begin{aligned} B_{k+1} &= B'_k + \frac{(y_k - B'_k s_k)(y_k - B'_k s_k)^T}{(y_k - B'_k s_k)^T s_k} \\ &= B'_k + \frac{y_k y_k^T}{y_k^T s_k} \\ &= B_k - \frac{B_k s_k s_k B_k^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \end{aligned}$$

Why choose $r_k = B_k s_k$?

- Intuitive from secant equation:

$$B_{k+1} s_k = \underbrace{B_k' s_k}_{0} + y_k \frac{y_k^T s_k}{y_k^T s_k} = y_k$$

- First term: Forget all that is known about direction s_k
- Second term: Replace by new information of y_k in direction s_k

BFGS Properties

- B_k always PD when $y_k^T s_k > 0$
- Surprising: Converges after maximal n steps on n-dim Ellipsoid (exact line-search)
- There exists an update of the inverse:

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

Assuming $H_k = B_k^{-1}$, $H_{k+1} = B_{k+1}^{-1}$