Newton and Quasi-Newton Methods

GIAN Short Course on Optimization: Applications, Algorithms, and Computation

Sven Leyffer

Argonne National Laboratory

September 12-24, 2016

Outline

Quadratic Models and Newton's Method

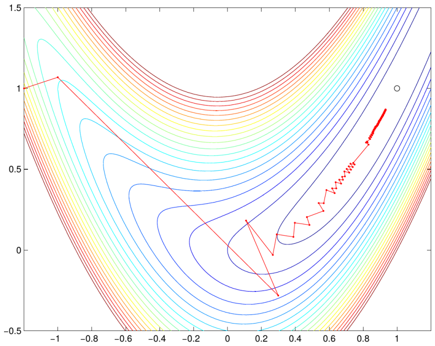
Consider unconstrained optimization problem:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ twice continuously differentiable.

Motivation for Newton:

- Steepest descend is easy, ... but can be slow
- Quadratics approximate nonlinear $f(x)$ better
- Faster local convergence
- More "robust" methods



Quadratic Models and Newton's Method

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x)$$

Main Idea Behind Newton

- Quadratic function approximates a nonlinear $f(x)$ well.
- First-order conditions of quadratics are easy to solve.

Consider minimizing a quadratic function (wlog cons t= 0)

$$\underset{x}{\text{minimize}} \; q(x) = \frac{1}{2} x^T H x + b^T x$$

First-order conditions, $\nabla q(x) = 0$, are

$$Hx = -b$$

... a linear system of equations

Quadratic Models and Newton's Method

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ f(x)$$

Newton's method uses truncated Taylor series:

$$f(x^{(k)} + d) = f^{(k)} + g^{(k)^T} d + \frac{1}{2} d^T H^{(k)} d + o(\|d\|^2)$$

where $a = o(\|d\|^2)$ means that $\frac{a}{\|d\|^2} \to 0$ as $\|d\|^2 \to 0$.

Notation Convention

Functions evaluated at $x^{(k)}$ are identified by superscripts:

- $f^{(k)} := f(x^{(k)})$
- $g^{(k)} := g(x^{(k)}) := \nabla f(x^{(k)})$
- $H^{(k)} := H(x^{(k)}) := \nabla^2 f(x^{(k)})$

Quadratic Models and Newton's Method

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x)$$

Newton's method defines quadratic approx. at $x^{(k)}$

$$q^{(k)}(d) := f^{(k)} + g^{(k)^T} d + \frac{1}{2} d^T H^{(k)} d,$$

and steps to minimum of $q^{(k)}(d)$.

If $H^{(k)}$ positive definite, solve linear system:

$$\min_d \; q^{(k)}(d) \quad \Leftrightarrow \quad \nabla q^{(k)}(d) = 0 \quad \Leftrightarrow \quad \nabla H^{(k)} d = -g^{(k)}.$$

... then sets $x^{(k+1)} := x^{(k)} + d$

Simple Version of Newton's Method

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x)$$

Simple Newton Line-Search Method

Given $x^{(0)}$, set $k = 0$.

repeat

Solve $H^{(k)} s^{(k)} := -g(x^{(k)})$ for Newton direction

Find step length $\alpha_k := \text{Armijo}(f(x), x^{(k)}, s^{(k)})$

Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.

until $x^{(k)}$ *is (local) optimum*;

See Matlab demo

Theory of Newton's Method

Newton direction is a descend direction if $H^{(k)}$ is positive definite:

Lemma

If $H^{(k)}$ is positive definite, then $s^{(k)}$ from solve of $H^{(k)} s^{(k)} := -g(x^{(k)})$ is a descend direction.

Proof.

Drop superscripts (k) for simplicity

H is positive definite $\Rightarrow$ H^{-1} inverse exists and is pos. definite

$\Rightarrow g^T s = g^T H^{-1}(-g) < 0$

$\Rightarrow s$ is a descend direction. □

Theory of Newton's Method

Newton's method converges quadratically

... steepest descend only linearly

Theorem

$f(x)$ twice continuously differentiable and that $H(x)$ is Lipschitz:

$$\|H(x) - H(y)\| \leq L\|x - y\|$$

near local minimum x^*.
If $x^{(k)}$ sufficiently close x^*, and if H^* positive definite, then Newton's method converges quadratically and $\alpha_k = 1$.

Theory of Newton's Method

Newton's method converges quadratically

... steepest descend only linearly

Theorem

$f(x)$ twice continuously differentiable and that $H(x)$ is Lipschitz:

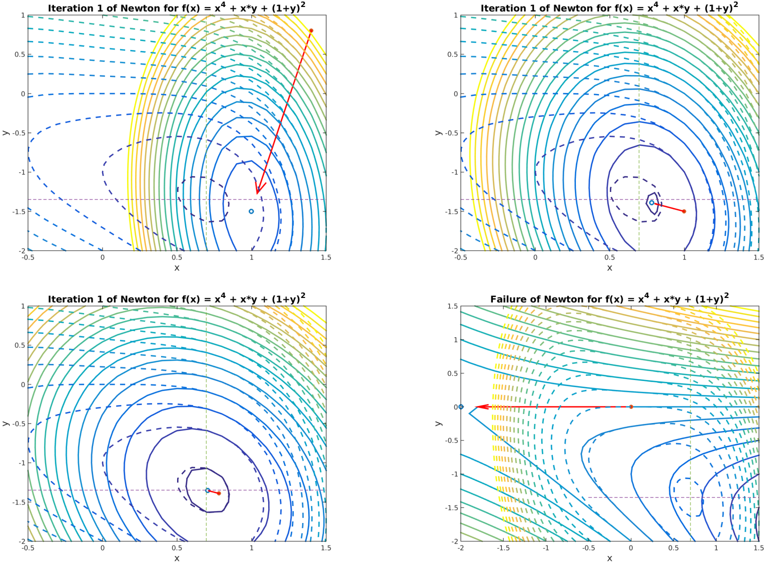
$$\|H(x) - H(y)\| \leq L\|x - y\|$$

near local minimum x^*.
If $x^{(k)}$ sufficiently close x^*, and if H^* positive definite, then Newton's method converges quadratically and $\alpha_k = 1$.

This is a remarkable result:

- Near a local solution, we do not need a line search.
- Convergence is quadratic ... double the significant digits.

Illustrative Example of Newton's Method



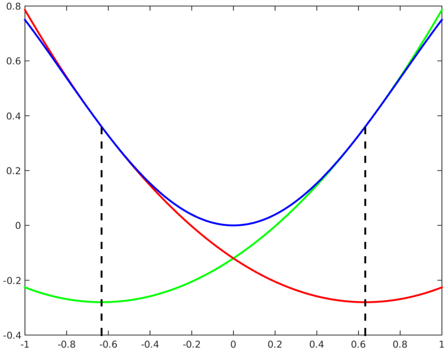
Convergence & failure of Newton: $f(x) = x_1^4 + x_1 x_2 + (1 + x_2)^2$

Discussion of Newton's Method I

Full Newton step may fail to reduce $f(x)$, E.g.

$$\underset{x}{\text{minimize}} \;\; f(x) = x^2 - \frac{1}{4}x^4.$$

$x^{(0)} = \sqrt{2/5}$ creates alternating iterates $-\sqrt{2/5}$ and $\sqrt{2/5}$.



Remedy: Use a line search.

Discussion of Newton's Method II

- Newton's method solves linear system at every iteration.
  Can be computationally expensive, if n is large.
  Remedy: Apply iterative solvers, e.g. conjugate-gradients.

- Newton's method needs first and second derivatives.
  Finite differences are computationally expensive.
  Use automatic differentiation (AD) for gradient
  ... Hessian is harder, get efficient Hessian products: $H^{(k)}v$
  Remedy: Code efficient gradients, or use AD tools.

Discussion of Newton's Method III

Problem, if Hessian, $H^{(k)}$ not positive definite

- Newton direction may not be defined
  If $H^{(k)}$ singular, then $H^{(k)}s = -g^{(k)}$ not well defined:
  - Either $H^{(k)}s = -g^{(k)}$ has no solution,
  - or $H^{(k)}s = -g^{(k)}$ has infinitely many solutions!
- Even if Newton direction exists, it may not reduce $f(x)$
  $\Rightarrow$ Newton's method fails even with line search

Discussion of Newton's Method IV

Problem, if Hessian, $H^{(k)}$, has indefinite curvature:

Consider

$$\underset{x}{\text{minimize}} \;\; f(x) = x_1^4 + x_1 x_2 + (1 + x_2)^2$$

Starting Newton at $x^{(0)} = 0$, get

$$x^{(0)} = 0, \quad g^{(0)} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad H^{(0)} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}, \quad \Rightarrow s^{(0)} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

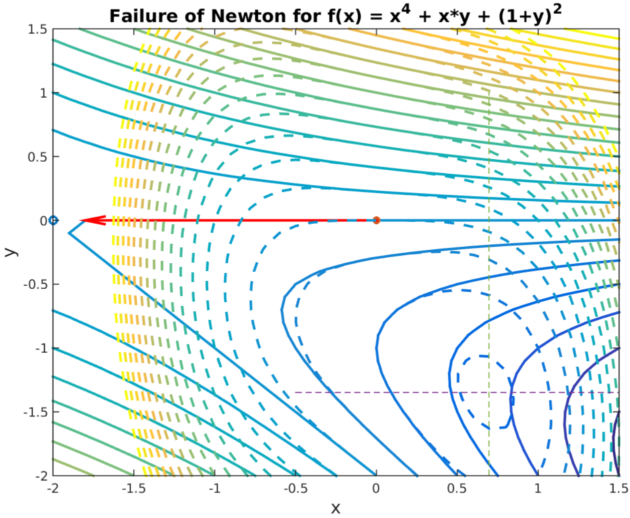
Line-search from $x^{(0)}$ in direction $s^{(0)}$:

$$x^{(0)} + \alpha s^{(0)} = \begin{pmatrix} -2\alpha \\ 0 \end{pmatrix} \quad \Rightarrow \; f(x^{(0)} + \alpha s^{(0)}) = (-2\alpha)^4 + 1 = 16\alpha^4 + 1 > 1$$

for all $\alpha > 0$, hence cannot decrease $f(x) \Rightarrow \alpha_0 = 0$

$\Rightarrow$ Newton's method stalls

Failure of Newton's Method



Steepest descend works fine

⇒ *Remedy: Modify Hessian to make it positive definite.*

Modifying the Hessian to Ensure Descend I

Newton's method can fail, if $H^{(k)}$, is not positive definite.

To modify the Hessian, estimate smallest eigenvalue, $\lambda_{\min}(H^{(k)})$,

Define modification matrix, M_k:

$$M_k := \max\left(0, \epsilon - \lambda_{\min}(H^{(k)})\right) I,$$

where $\epsilon > 0$ small, and $I \in \mathbb{R}^{n \times n}$ identity matrix

Use modified Hessian, $H^{(k)} + M_k$, which is positive definite

Matlab get smallest eigenvalue: `Lmin = min(eig(H))`

Modifying the Hessian to Ensure Descend II

Alternative modification

- Compute Cholesky factors of $H^{(k)}$:

$$H^{(k)} + M_k = L_k L_k^T$$

  where L_k lower triangular with positive diagonal
- $L_k L_k^T$ is positive definite
- Choose $M_k = 0$ if $H^{(k)}$ is positive definite
- Choose M_k not unreasonably large
- Related to $L_k D_k L_k^T$ factors

... perform modification as we solve the Newton system,

$$H^{(k)} s^{(k)} := -g(x^{(k)})$$

Modified Newton Line-Search Method

Given $x^{(0)}$, set $k = 0$. **repeat**

  Form M_k from eigenvalue est. or mod. Cholesky factors.

  Get modified Newton direction: $\left(H^{(k)} + M_k\right) s^{(k)} := -g(x^{(k)})$.

  Get step length $\alpha_k :=$ Armijo$(f(x), x^{(k)}, s^{(k)})$.

  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.

until $x^{(k)}$ *is (local) optimum*;

Modification $H^{(k)} - \lambda_{\min}(H^{(k)})I$ bias towards steepest descend:
Let $\mu = \lambda_{\min}(H^{(k)})^{-1}$, then solve

$$\lambda_{\min}(H^{(k)}) \left(\mu H^{(k)} + I\right) s^{(k)} := -g(x^{(k)}),$$

As $\mu \to 0$, recover steepest-descend direction, $s^{(k)} \simeq -g(x^{(k)})$

Outline

Quasi-Newton Methods

Quasi-Newton Methods avoid pitfalls of Newton's method:

1. Failure Newton's, if $H^{(k)}$ not positive definite;
2. Need for second derivatives;
3. Need to solve linear system at every iteration.

Study quasi-Newton and more modern limited-memory quasi-Newton methods

- Overcome computational pitfalls of Newton
- Retain fast local convergence (almost)

Quasi-Newton methods work with approx. $B^{(k)} \simeq {H^{(k)}}^{-1}$
$\Rightarrow$ Newton solve becomes matrix-vector product: $s^{(k)} = -B^{(k)} g^{(k)}$

Quasi-Newton Methods

Choose initial approximation, $B^{(0)} = \nu I$ Define

$$\begin{aligned} \gamma^{(k)} &:= g^{(k+1)} - g^{(k)} \text{ gradient difference} \\ \delta^{(k)} &:= x^{(k+1)} - x^{(k)} \text{ iterate difference,} \end{aligned}$$

then, for quadratic $q(x) := q_0 + g^T x + \frac{1}{2} x^T H x$, get

$$\gamma^{(k)} = H\delta^{(k)} \;\Leftrightarrow\; \delta^{(k)} = H^{-1}\gamma^{(k)}$$

Because $B^{(k)} \simeq {H^{(k)}}^{-1}$, ideally want $B^{(k)}\gamma^{(k)} = \delta^{(k)}$

Not possible, because need $B^{(k)}$ to compute $x^{(k+1)}$, hence use

Quasi-Newton Condition

$$B^{(k+1)}\gamma^{(k)} = \delta^{(k)}$$

Rank-One Quasi-Newton Update

Goal: Find rank-one update such that $B^{(k+1)}\gamma^{(k)} = \delta^{(k)}$

Express symmetric rank-one matrix as outer product:

$$uu^T = [u_1 u; \ldots ; u_n u], \quad \text{and set } B^{(k+1)} = B^{(k)} + auu^T.$$

Choose $a \in R$ and $u \in \mathbb{R}^n$ such that update, $B^{(k+1)}$, satisfies

$$\delta^{(k)} = B^{(k+1)}\gamma^{(k)} = B^{(k)}\gamma^{(k)} + auu^T\gamma^{(k)}$$

... quasi-Newton condition

Rewrite Quasi-newton condition as

$$\Leftrightarrow \quad \delta^{(k)} - B^{(k)}\gamma^{(k)} = auu^T\gamma^{(k)}$$

"Solving" last equation of u, then quasi-Newton condition implies

$$u = \left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right) / \left(au^T\gamma^{(k)}\right)$$

assuming $au^T\gamma^{(k)} \neq 0$

Rank-One Quasi-Newton Update

From previous page: Quasi-Newton condition implies

$$u = \left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right) / \left(au^T\gamma^{(k)}\right)$$

assuming $au^T\gamma^{(k)} \neq 0$

We are looking for update auu^T

- Assume $au^T\gamma^{(k)} \neq 0$ (can be monitored)
- Choose $u = \delta^{(k)} - B^{(k)}\gamma^{(k)}$

Given this choice of u, we must set a as

$$a = \frac{1}{u^T\gamma^{(k)}} = \frac{1}{\left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right)^T \gamma^{(k)}}.$$

Double check that we satisfy the quasi-Newton condition:

$$B^{(k+1)}\gamma^{(k)} = B^{(k)}\gamma^{(k)} + auu^T\gamma^{(k)}$$

Rank-One Quasi-Newton Update

Substituting values for a and u we get ...

$$B^{(k+1)}\gamma^{(k)} = B^{(k)}\gamma^{(k)} + \frac{\left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right)\left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right)^T \gamma^{(k)}}{\left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right)^T \gamma^{(k)}}$$

$$= B^{(k)}\gamma^{(k)} + \delta^{(k)} - B^{(k)}\gamma^{(k)} = \delta^{(k)}$$

Rank-One Quasi-Newton Update

Assuming that $(\delta - B\gamma)^T \gamma \neq$ we use:

$$B^{(k+1)} = B + \frac{(\delta - B\gamma)(\delta - B\gamma)^T}{(\delta - B\gamma)^T \gamma}.$$

Properties of Rank-One Quasi-Newton Update

Rank-One Quasi-Newton Update

$$B^{(k+1)} = B + \frac{(\delta - B\gamma)(\delta - B\gamma)^T}{(\delta - B\gamma)^T \gamma}.$$

Theorem (Quadratic Termination of Rank-One)

If rank-one update is well defined, and $\delta^{(1)}, \ldots, \delta^{(n)}$ linearly independent, then rank-one method terminates in at most $n+1$ steps with $B^{(n+1)} = H^{-1}$ for quadratic with pos. definite Hessian.

Remark (Disadvantages of Rank-One Formula)

1. *Does not maintain positive definiteness of $B^{(k)}$*
   $\Rightarrow$ *steps may not be descend directions*
2. *Rank-one breaks down, if denominator is zero or small.*

BFGS Quasi-Newton Update

BFGS rank-two update ... method of choice

BFGS Quasi-Newton Update

$$B^{(k+1)} = B - \left(\frac{\delta\gamma^T B + B\gamma\delta^T}{\delta^T\gamma} \right) + \left(1 + \frac{\gamma^T B\gamma}{\delta^T\gamma} \right) \frac{\delta\delta^T}{\delta^T\gamma}.$$

... works well with low-accuracy line-search

Theorem (BFGS Update is Positive Definite)

If $\delta^T\gamma > 0$, then BFGS update remains positive definite.

Picture of BFGS Quasi-Newton Update

We can visualize the BFGS update ...

Picture of BFGS Quasi-Newton Update

We can visualize the BFGS update ...

Convergence of BFGS Updates

Question (Convergence of BFGS with Wolfe Line Search)

Does BFGS converge for nonconvex $f(x)$ with Wolfe line-search?

Wolfe Line-Search Conditions

Wolfe line search finds α:

$$f(x^{(k)} + \alpha_k s^{(k)}) - f^{(k)} \leq \delta \alpha_k {g^{(k)}}^T s^{(k)}$$

$$g(x^{(k)} + \alpha_k {s^{(k)}}^T s^{(k)}) \geq \sigma {g^{(k)}}^T s^{(k)}.$$

Convergence of BFGS Updates

Question (Convergence of BFGS with Wolfe Line Search)

Does BFGS converge for nonconvex $f(x)$ with Wolfe line-search?

Wolfe Line-Search Conditions

Wolfe line search finds α:

$$f(x^{(k)} + \alpha_k s^{(k)}) - f^{(k)} \leq \delta \alpha_k g^{(k)^T} s^{(k)}$$

$$g(x^{(k)} + \alpha_k s^{(k)^T} s^{(k)}) \geq \sigma g^{(k)^T} s^{(k)}.$$

Unfortunately, the answer is no!

Dai [2013] Example of Failure of BFGS

Constructs "perfect 4D example" for BFGS method:

- Steps $s^{(k)}$, gradients, $g^{(k)}$, satisfy

$$s^{(k)} = \begin{bmatrix} R_1 & 0 \\ 0 & \tau R_2 \end{bmatrix} s^{(k-1)} \quad \text{and} \quad g^{(k)} = \begin{bmatrix} \tau R_1 & 0 \\ 0 & R_2 \end{bmatrix} g^{(k-1)},$$

where τ parameter, and R_1, R_2 rotation matrices

$$R_1 = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \quad \text{and} \quad R_2 = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}$$

Can show that

- $\alpha_k = 1$ satisfies Wolfe or Armijo line search
- $f(x)$ is polynomial of degree 38 (strongly convex along $s^{(k)}$.
- Iterates converge to circle around vertices of octagon
  ... not stationary points.

Limited-Memory Quasi-Newton Methods

Disadvantage of quasi-Newton: Storage & computat^n: $\mathcal{O}(n^2)$

- Quasi-Newton matrices are dense ($\exists$ sparse updates).
- Storage & computation of $\mathcal{O}(n^2)$ prohibitive for large n
  ... solve inverse problems from geology with 10^{12} unknowns

Limited memory method are clever way to re-write quasi-Newton

- Store $m \ll n$ most recent difference pairs $m \simeq 10$
- Cost per iteration only $\mathcal{O}(nm)$ not $\mathcal{O}(n^2)$

Limited-Memory Quasi-Newton Methods

Recall BFGS update:

$$B^{(k+1)} = B - \left(\frac{\delta\gamma^T B + B\gamma\delta^T}{\delta^T\gamma} \right) + \left(1 + \frac{\gamma^T B\gamma}{\delta^T\gamma} \right) \frac{\delta\delta^T}{\delta^T\gamma}.$$

$$= B - \left(\frac{\delta\gamma^T B + B\gamma\delta^T}{\delta^T\gamma} \right) + \left(\frac{\gamma^T B\gamma}{\delta^T\gamma} \right) \frac{\delta\delta^T}{\delta^T\gamma} + \frac{\delta\delta^T}{\delta^T\gamma}$$

Rewrite BFGS update as (substitute and prove for yourself!)

$$B^{(k+1)}_{\text{BFGS}} = V_k^T B V_k + \rho_k \delta\delta^T,$$

where

$$\rho_k = \left(\delta^T\gamma\right)^{-1}, \quad \text{and} \quad V_k = I - \rho_k \gamma\delta^T.$$

Recur update back to initial matrix, $B^{(0)} \succ 0$

Limited-Memory Quasi-Newton Methods

Idea: Apply $m \ll n$ quasi-Newton updates at iteration k, corresponding to difference pairs, (δ_i, γ_i) for $i = k-m, \ldots, k-1$:

$$
\begin{aligned}
B^{(k)} = & \left[V_{k-1}^T \cdots V_{k-m}^T\right] B^{(0)} \left[V_{k-1} \cdots V_{k-m}\right] \\
& + \rho_{k-m} \left[V_{k-1}^T \cdots V_{k-m+1}^T\right] B^{(0)} \left[V_{k-1} \cdots V_{k-m+1}\right] \\
& + \ldots \\
& + \rho_{k-1} \delta^{(k-1)} \delta^{(k-1)^T}
\end{aligned}
$$

... can be implemented recursively!

Limited-Memory Quasi-Newton Methods

Recursive procedure to compute BFGS direction, s:

Limited Memory BFGS Search Direction Computation

Given initial $B^{(0)}$, memory m, set gradient, $q = \nabla f(x^{(k)})$.

for $i = k-1, \ldots, k-m$ **do**

  Set $\alpha_i = \rho_i {\delta^{(i)}}^T \gamma^{(i)}$

  Update gradient: $q = q - \alpha_i \gamma^{(i)}$

end

Apply initial quasi-Newton matrix: $r = H^{(0)} q$

for $i = k-1, \ldots, k-m$ **do**

  Set $\beta = \rho_i {\gamma^{(i)}}^T r$

  Update direction: $r = r + \delta^{(i)}(\alpha_i - \beta)$

end

Return search direction: $s^{(k)} := r \left(= H^{(k)} g^{(k)}\right)$

Cost of recursion is $\mathcal{O}(4nm)$ if $H^{(0)}$ is diagonal

General Quasi-Newton Methods

Given any of updates discussed, quasi-Newton algorithm is

General Quasi-Newton (qN) Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat

Get quasi-Newton direction, $s^{(k)} = -B^{(k)} g^{(k)}$

Step length $\alpha_k :=$ Armijo$(f(x), x^{(k)}, s^{(k)})$

Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$.

Form $\gamma^{(k)}, \delta^{(k)}$, update qN matrix, $B^{(k+1)}$, set $k = k + 1$.

until $x^{(k)}$ *is (local) optimum*;

Summary: Newton and Quasi-Newton Methods

Methods for unconstrained optimization:

$$\underset{x}{\text{minimize}} \;\; f(x)$$

- Quadratic model provides better approx. of $f(x)$
- Newton's method minimizes quadratic for step d:

$$\underset{d}{\text{minimize}} \;\; q^{(k)}(d) := f^{(k)} + {g^{(k)}}^T d + \frac{1}{2} d^T H^{(k)} d,$$

  - Modify if $H^{(k)}$ not pos. def. (no descend): $H^{(k)} + M_k \succeq 0$
  - Converges quadratically (near solution)
- Quasi-Newton methods avoid need for Hessian $H^{(k)}$
  - Update quasi-Newton approx. $B^{(k)} \approx {H^{(k)}}^{-1}$
  - Limited memory version for large-scale optimization