1 General

line search conditions:

$$
\begin{aligned}
f_{k+1}^T &\leq f_k + c_1 \alpha_k \nabla f_k^T p_k, c_1, \alpha_k \in (0,1) && (1)\\
\nabla f_{k+1}^T p_k &\geq c_2 \nabla f_k^T p_k, 0 < c_1 < c_2 < 1 && (2)\\
\text{where :} & && (3)\\
f_k &= f(x_k) && (4)\\
f_{k+1} &= f(x_k + \alpha_k p_k) && (5)
\end{aligned}
$$

2 Quasi Newton

2.1 BFGS

properties: $O(n^2)$, self correcting, slightly more iterations than Newton Method, linear convergence order and superlinear rate of convergence

secant equation:

$$
\begin{aligned}
B_{k+1}(x_{k+1} - x_k) &= \nabla f_{k+1} - \nabla f_k\\
B_{k+1} s_k &= y_k\\
s_k &= \alpha_k p_k\\
y_k &= \nabla f_{k+1} - \nabla f_k\\
B_{k+1} &:= \text{approx. Hessian}
\end{aligned}
$$

$$
\begin{aligned}
B_{k+1} &\succ 0\\
s_k^T B_{k+1} s_k &= s^T y_k > 0
\end{aligned}
$$

Proof.

$$
\begin{aligned}
y_k^T s_k &= (\nabla f_{k+1} - \nabla f_k)^T s_k\\
\nabla f_{k+1}^T s_k &\geq c_2 \nabla f_k^T s_k\\
(\nabla f_{k+1} - \nabla f_k)^T s_k &\geq c_2 \nabla f_k^T s_k - \nabla f_k^T s_k\\
y_k^T s_k &\geq (c_2 - 1) \nabla f_k^T s_k\\
c_2 < 1, s_k \text{ is a descent dir } &\implies s_k^T y_k > 0
\end{aligned}
$$

Curvature condition holds. □

constrain B by solving:

$$
\begin{aligned}
&\min_B ||B - B_k||\\
&s.t.\ B = B^T, B s_k = y_k
\end{aligned}
$$

similarly, constrain B's inverse, H where it satisfy secant equation:

$$
\begin{aligned}
&H_{k+1} y_k = s_k\\
&\min_H ||H - H_k||\\
&s.t.\ H = H^T, H y_k = s_k
\end{aligned}
$$

using weighted Frobenius norm:

$$
\begin{aligned}
||A||_W &:= ||W^{1/2} A W^{1/2}||_F\\
||X||_F &:= (\sum_i \sum_j (X_{ij})^2)^{1/2}
\end{aligned}
$$

solved weight matrix W satisfy $W s_k = y_k$ solution given by:

$$
\begin{aligned}
H_{k+1} &= (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T)\\
&+ \rho_k s_k^T s_k\\
\rho_k &= \frac{1}{y_k^T s_k}
\end{aligned}
$$

W is the average Hessian $\bar{G}$:

$$
\bar{G} = \int_0^1 \nabla^2 f(x_k + \tau \alpha_k p_k) d\tau
$$

initial H_0 can be chosen approximately (eg: finite differences, I)

Algorithm 1: BFGS Algorithm

$H_0, x_0, \epsilon > 0$**:** inverse Hessian approx., initial point, convergence tolerance
x **:** solution

```
k ← 0
while ||∇f_k||> ε do
    α_k ← LineSearch(..)
    x_{k+1} ← x_k + α_k p_k
    s_k ← x_{x+1} − x_k
    y_k ← ∇f_{k+1} − ∇f_k
    ρ_k ← 1/(y_k^T s_k)
    H_{k+1} ← (I − ρ_k s_k y_k^T) H_k (I − ρ_k y_k s_k^T)
    + ρ_k s_k^T s_k
    k ← k + 1
```

using Sherman-Morrison-Woodbury formula to obtain Hessian update equation:

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

proper line search is required so that BFGS algo captures curvature information

inaccurate line search can be used to reduce computation cost

3 Trust Region Methods

idea:

- models local behaviour of the objective function (eg: 2nd order Taylor series)
- set local region to explore, then simultaneously find direction and step size to take
- region size adaptively set using results from previous iterations
- step may fail due to inadequately set region, which need to be adjusted
- superlinear convergence when approximate model Hessian is equal to true Hessian

using 2nd order Taylor series model with symmetric matrix approximating Hessian

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p, \ st. \ ||p|| \leq \Delta_k$$

$$\Delta_k := \text{trust region radiius}$$
$$g_k = \nabla f(x_k)$$

full step is ($p_k = -B_k^{-1} g_k$) taken when $B \succ 0$ and $||B_k^{-1} g_k|| \leq \Delta_k$

evaluate goodness of model with actual function by:

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$$

$$action \leftarrow \begin{cases} \text{expand trust region} & , \rho_k \approx 1 \ (agreement) \\ \text{shrink trust region} & , \rho_k < 0 + \text{thresh} \\ \text{keep trust region} & , o/w \end{cases}$$

Algorithm 2: Trust Region Algorithm

```
k ← 0
while ||∇f_k|| > ε do
    p_k ← argmin_p f_k + g_k^T p + ½ p^T B_k p,  st. ||p|| ≤ Δ_k
    ρ_k = (f(x_k) − f(x_k + p_k)) / (m_k(0) − m_k(p_k))
    if ρ_k < γ(: 1/4) then
        Δ_{k+1} ← α(: 1/4) Δ_k
    else if ρ_k > β(: 3/4) and ||p_k|| == Δ_k then
        Δ_{k+1} ← min(2Δ_k, Δ̂)
    else
        Δ_{k+1} ← Δ_k
    if ρ_k > η(:∈ [0, 1/4)) then
        x_{k+1} ← x_k + p_k
    else
        x_{k+1} ← x_k
```

minimizer of the 2nd order Taylor series satisfy the following:

$$(B + \lambda I) p^* = -g$$
$$\lambda(\Delta - ||p^*||) = 0$$
$$(B + \lambda I) \succeq 0$$

solving 2nd order Taylor series using approx methods:

- dogleg
- 2-D subspace minimization
- conjugate gradient based

Cauchy Point:
Use 1st order approx. of model and gradient descent to get get next iterate, bounded within trust region.

3.1 Dogleg method

if $B \succ 0$:
$p^B = -B^{-1} g$
$p^* = p^B$ if $\Delta \geq \|p^B\|$
$p^U = \frac{-g^T g}{g^T B g} g$ (intermediate point along direction of steepest descent)
interpolate between p^U and p^B:

$$\tilde{p}(\tau) = \begin{cases} \tau p^U & , \tau \in [0, 1] \\ p^U + (\tau - 1)(p^B - p^U) & , \tau \in [1, 2] \end{cases}$$

$B \succ 0 \implies \|\tilde{p}(\tau)\|$ increases wrt. τ, $m(\tilde{p}(\tau))$ decreases wrt. τ

if $\|p^B\| \leq \Delta$: p chosen at p^B
else p chosen at intersection of $\tilde{p}(\tau)$ and trust region boudnary by solving:
$\|p^U + (\tau - 1)(p^B - p^U)\|^2 = \|\Delta^2\|$

$$p_k^S = \operatorname*{argmin}_p f_k + g_k^T p, \|p\| \leq \Delta_k$$
$$\tau_k = \operatorname*{argmin}_{\tau \geq 0} m_k(\tau p_k^S), \|\tau p_k^S\| \leq \Delta_k$$
$$p_k^S = \frac{-\Delta_k g_k|}{\|g_k\|}$$
$$p_k^C = \tau_k p_k^S$$
$$p_k^C = -\tau_k \frac{g_k}{\|g_k\|}$$
$$\tau_k = \begin{cases} 1 & , g_k^T B_k g_k \leq 0 \\ min(\frac{\|g_k|\hat{3}}{\Delta_k g_k^T B_k g_k}, 1) & , o/w \end{cases}$$

3.2 Iterative Solution

Idea: solve subproblem $\min_{\|p\|\leq\Delta} m(p)$ by applying Newton's method to find λ that matches trust region radius. This is slightly more accurate per step compared to Dogleg. Use $(B+\lambda I)p^* = -g$ to solve $\min_{\|p\|\leq\Delta} m(p)$ for λ.

If $lambda = 0$ and $(B+\lambda I)p^* = -g, \|p^*\| \leq \Delta$ and $(B+\lambda I) \succeq 0$: return λ
Else: find λ s.t. $(B+I) \succeq 0$ and $\|p(\lambda)\| = \Delta, p(\lambda) = -(B+\lambda I)^{-1}g$. Solve and return λ.

Solve $\|p(\lambda)\| - \Delta = 0, \lambda > \lambda_1$ via Newton's method (root finding). Approx. this to nearly a linear problem for easy solving:

Algorithm 3: Subproblem Algo

for $l = 0, 1, ..$ **do**
- solve $B + \lambda^l I = R^T R$
- $R^T R p_l = -g$
- $R^T q_l = p_l$
- $\lambda^{l+1} \leftarrow \lambda^l + (\frac{\|p_l\|}{\|q_l\|})^2 (\frac{\|p_l\| - \Delta)}{\Delta})$ check $\lambda \geq \lambda_1$

4 Conjugate Gradient

4.1 linear method

Assuming unconstrained problem with strict convex quadratic objective function:

$$\frac{1}{2}x^TAx - b^Tx, A \succ 0, A^T = A$$

$\nabla(\frac{1}{2}x^TAx - b^Tx) = Ax - b$, thus $\min_x x^TAx - b^Tx$ transformed to $Ax - b = 0$.

$x_{k+1} = x_k + \alpha_k p_k$, solve for α:

$\frac{\partial}{\partial \alpha}(\frac{1}{2}(x_k + \alpha_k p_k)^T A(x_k + \alpha_k p_k) - b^T(x_k + \alpha_k p_k)) = 0$

$r_k = Ax - b$

$\alpha_k = \frac{-p_k^T r_k}{p_k^T A p_k}$

4.2 Conjugate Direction

Search directions linearly independent wrt. A.

$$(\forall i \neq j) p_i^T A p_j = 0$$

Properties:

- Residual elimnated one direction at a time, resulting in max of n iterations.
- Optimal if Hessian is diagonal, if not can try preconditioning.
- Current residual is orthogonal to all previous search directions.
- Any set of conjugate directions can be used (eg: eigenvectors, Gram-Schmidt)

Expanding subspace minimizer:
Using conjugate directions to generate sequence $\{x\}$, then:
$r_k^T p_i = 0, \forall i < k$, x_k is minimizer of $\frac{1}{2}x^TAx - b^Tx$ over $\{x | x = x_0 + span\{p_0, ...p_{k-1}\}$

Proof.

$\tilde{x} = x_0 + \sum_i \sigma_i p_i$

$\tilde{x}$ minimizes over $\{x_0 + span\{p_0, ...p_{k-1}\}\} \iff r(\tilde{x})^T p_i = 0$

$h(\sigma) = \phi(\tilde{x})$

$\phi(x) = \frac{1}{2}x^TAx - b^Tx$

h is also strictly convex quadratic,
with unique σ^* satisfying:

$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, i = [0, k-1]$

$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = \nabla\phi(\tilde{x})^T p_i = 0, i = [0, k-1]$

$\nabla\phi(x) = Ax - b = r$

$r(\tilde{x})^T p_i = 0, i = [0, k-1]$

$\square$

$p_i^T r_k = 0, i = [0, k-1]$ via induction:

Proof.

base case : $x_1 = x_0 + \alpha_0 p_0$ minimizes ϕ along p_0

$\implies r_1^T p_0 = 0$

case: $r_{k-1}^T p_i = 0, i = [0, k-2]$:

$r_k = r_{k-1} + \alpha_{k-1} A p_{k-1}$

$p_{k-1}^T r_k = p_{k-1}^T r_{k-1} + \alpha_{k-1} p_{k-1}^T A p_{k-1} = 0$(by construction)

A-conjugate $\implies p_{k-1}^T A p_{k-1}$

case: $\forall i = [0, k-2] : p_i^T r_k = 0$

$p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T A p_{k-1}$

$p_i^T r_{k-1} = 0(by induction hypothesis$

$\alpha_{k-1} p_i^T A p_{k-1} = 0(conjugacy)$

$p_i^T r_k = 0, i = [0, k-1]$

$\square$

4.3 Conjugate Gradient Method

Idea:

- uses only previous search direction to compute current search direction
- p_k set to linear combination of $-r_k$ and p_{k-1}
- impose $p_k^T A p_{k-1} = 0$

$$\begin{aligned}
p_k &= -r_k + \beta_k p_{k-1} \\
p_{k-1}^T A p_k &= -p_{k-1}^T A r_k + \beta p_{k-1}^T A p_{k-1} \\
0 &= -p_{k-1}^T A r_k + \beta p_{k-1}^T A p_{k-1} \\
\beta &= \frac{p_{k-1}^T A r_k}{p_{k-1}^T A p_{k-1}} \\
p_0 &= -(Ax_0 - b) = -r_0
\end{aligned}$$

Algorithm 4: Basic Conjugate Gradient Algorithm

$r_0 \leftarrow Ax_0 - b$
$p_0 = -r_0$
for $k = [0, ..n-1]$ **do**
  if $r_k == 0$ **then**
    return x_k
  else
    $\alpha_k \leftarrow \frac{-r_k^T p_k}{p_x^T A p_k}$
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $r_{k+1} \leftarrow A x_{k+1} - b$
    $\beta_{k+1} \leftarrow \frac{p_k^T A r_{k+1}}{p_k^T A p_k}$
    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$

p and r_k is within krylov subspace:
$K(r_0; k) = span\{r_0, Ar_0, ..A^k r_0\}$
if $r_k \neq 0$:
$r_k^T r_i = 0, i = [0, k-1]$
$span\{r_0, .., r_k\} = span\{r_0, Ar_0, .., A^k r_0\}$
$span\{p_0, .., p+k\} = span\{r_0, Ar_0, .., A^k r_0\}$
$p_k^T A p_i = 0, i = [0, k-1]$
then, $\{x_k\} \to x^*$ in at most n steps.

Simplification:

$$\begin{aligned}
& p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k \\
& \alpha_k \leftarrow \frac{-r_k^T p_k}{p_k^T A p_k} \\
& \alpha_k \leftarrow \frac{-r_k^T (-r_k + \beta_k p_{k-1})}{p_k^T A p_k} \\
& (\forall i = [0, k-1]) r_k^T p_i = 0 \implies \beta_k r_k^T p_{k-1} = 0 \\
& \alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k} \text{ (simplified)}
\end{aligned}$$

$$\begin{aligned}
& r_{k+1} = r_k + \alpha_k A p_k \\
& A p_k = \frac{r_{k+1} - r_k}{\alpha_k} \\
& \beta = \frac{p_k^T A r_{k+1}}{p_k^T A p_k} \\
& p_k^T A p_k = p_k^T \frac{r_{k+1} - r_k}{\alpha_k} = \frac{-p_k^T r_k}{\alpha} \text{(conjugacy)} \\
& p_k^T A p_k = -\frac{(-r_k + \beta_k p_{k-1})^T r_k}{\alpha} = \frac{r_k^T r_k}{\alpha} \text{(conjugacy)} \\
& p_k^T A r_{k+1} = r_{k+1}^T A p_k \\
& p_k^T A r_{k+1} = r_{k+1}^T \frac{r_{k+1} - r_k}{\alpha_k} \\
& r_k \in span\{p_k, p_{k-1}\} \ and \ r_{k+1}^T p_i = 0, i = [0, k] \implies \\
& p_k^T A r_{k+1} = \frac{r_{k+1}^T r_{k+1}}{\alpha_k} \\
& \beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \text{ (simplified)}
\end{aligned}$$

4.4 Nonlinear Method

Minimize general convex function or nonlinear function. Variants: FR, PR.

4.4.1 FR (Fletcher Reaves)

Modify linear CG by:

- replace residual by gradient of objective, $r_k \to \nabla f_k$
- replace α_k computation by a linear search to find approx. minimum along search direction

Equivalent to linear CG if objective is strongly convex quadratic.

Linear search for α_k with strong Wolfe condition to ensure p_k's are descent directions wrt. objective function.

4.4.2 PR

Replace β_{k+1} computation in FR with:

$$\beta_{k+1} \leftarrow \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$$

5 Proximal Algorithm

Idea:

- reliance on easy to evaluate proximal operators
- separability allows parallel evaluation
- generalization of projection based algorithms

$$prox_{\lambda f}(v) = \underset{x}{\operatorname{argmin}}\, f(x) + \frac{1}{2\lambda} \|x - v\|_2^2$$

5.1 Proximal Gradient Method

Solve $\min_x g(x) + f(x)$, where f, g are closed, convex functions and f differentiable

$$\begin{aligned}
&x^* = prox_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k)) \\
&= \underset{x}{\operatorname{argmin}}\, g(x) + \frac{1}{2\lambda^k} \|x - (x^k - \lambda^k \nabla f(x^k))\|_2^2 \\
&\text{tradeoff between g and and gradient step} \\
&g = I_C(x) \implies \text{projected gradient step} \\
&g = 0 \implies \text{gradient descent} \\
&f = 0 \implies \text{proximal minimization}
\end{aligned}$$

Relation to Pixed Point:
x^* is a fixed point solution of $\min_x g(x) + f(x)$ iff $0 \in \nabla f(x^*) + \partial g(x^*)$ iff $x^* = (I + \lambda \partial g)^{-1}(I - \lambda \nabla f)(x^*)$

Forward Euler, Backward Euler stepping is same as the proximal gradient iteration, $prox_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$

5.2 Accelerated Proximal Gradient Method

Introduce extrapolation:

$$\begin{aligned}
&y^{k+1} = x^k + w^k (x^k - x^{k-1}) \\
&x^{k+1} = prox_{\lambda^k g}(y^{k+1} - \lambda^k \nabla f(y^{k+1})) \\
&w^k \in [0, 1)
\end{aligned}$$

Example: $w^k = \frac{k}{k+3}, w^0 = 0, \lambda^k \in (0, 1/L], L :=$ Lipschitz constant of ∇f, *or* λ^k found via line search. Line search for λ^k (Beck and Teboulle):

Algorithm 5: Proximal Gradient Algorithm

```
ˆf(x,y) := f(y) + ∇f(y)^T (x − y) + (1/2λ)||x − y||_2^2
while True do
    z = prox_{λg}(y^k − λ∇f(y^k))
    if f(x) ≤ ˆf(z, y^k) then
        break
    λ = βλ
return λ^k := λ, x^{k+1} := z
```

5.3 Types of Proximal Operators

- quadratic functions

$$f = \frac{1}{2}\|.\|_x^2 \implies prox_{\lambda f}(v) = (\frac{1}{1+\lambda})v$$

$$f = \frac{1}{2}x^TAx + b^Tx + c, A \in S_+^n \implies$$

$$prox_{\lambda f}(v) = (I+\lambda)^{-1}(v - \lambda b)$$

- unconstrained problem: use gradient methods such as Newton, Quasi-Newton
- constrained: use projected subgradient for non-smooth, projected gradient or interior method for smooth
- separable function: if scalar, may be solved analytically, eg: L1 norm separable to:

$$f(x) = |x| \implies prox_{\lambda f}(v) = \begin{cases} v-\lambda, & v \geq \lambda \\ 0, & |v| \leq \lambda \\ v+\lambda, & v \leq -\lambda \end{cases}$$

$$f(x) = -log(x) \implies prox_{\lambda f}(v) = \frac{v + \sqrt{v^2+4\lambda}}{2}$$

- general scalar function
  - localization: using a subgradient oracle and bisection algorithm
  - twice continuously differentiable: guarded Newton method
- polyhedra constraint, quadratic objective: solve as QP problem
  - duality to reduce number of variables to solve if possible
  - gram matrix caching
  - affine constraint($Ax = b$): use pseudo-inverse, A^+:

    $\Pi_C(v) = v - A^+(Av - b)$

    $$A \in R^{m\times n}, m < n \implies A^+ = A^T(AA^T)^{-1}$$

    $$A \in R^{m\times n}, m > n \implies A^+ = (A^TA)^{-1}A^T$$

  - hyperplane constaint($a^Tx = b$):

    $$\Pi_C(v) = v + (\frac{b - a^Tb}{\|a\|_2^2})a$$

  - halfspace

    $$\Pi_C(v) = v - \frac{max(a^Tv - b, 0)}{\|a\|_2^2}a$$

  - box($l \leq x \leq u$)

    $$\Pi_C(v)_k = min(max(v_k, l_k), u_k)$$

  - probability simplex($1^Tx = 1, x \geq 0$)
    bisection algo on ν:

    $$\Pi_C(v) = (v - \nu 1)_+$$

    $$\text{intial } [l_k, u_k] = [\max_i v_i - 1, \max_i vi]$$

    analytically solve when bounded in between 2 adjacent v˙i's
- cones (κ: proper cone)
  problem of the form:

$$min_x\|x - v\|_2^2$$
$$s.t. : x \in \kappa$$

$$x \in \kappa$$
$$v = x - \lambda$$
$$\lambda \in \kappa^*$$
$$\lambda^T x = 0$$

  - cone $C = \mathbb{R}_+^n$

    $$\Pi_C(v) = v_+$$

  - 2nd order cone $C = \{(x,t) \in \mathbb{R}^{n+1} : \|x\|_2 \leq t\}$

    $$\Pi_C(v,s) = \begin{cases} 0, & \|v\|_2 \leq -s \\ (v,s), & \|v\|_2 \leq s \\ \frac{1}{2}(1 + \frac{s}{\|v\|_2})(v, \|v\|_2), & \|v\|_2 \geq |s| \end{cases}$$

  - PSD cone S_+^n

    $$\Pi_C(V) = \sum_i (\lambda_i)_+ u_i u_i^T$$

    $$V = \sum_i \lambda_i u_i u_i^T \ (eigendecomp)$$

  - exponential cone
    Todo
- pointwise supremum
  - max function
  - support function
- norms
  - L2
  - L1
  - L-inf
  - elastic net
  - sum of norms
  - matrix norm
- sublevel set
- epigraph
- matrix functions Todo

6 Subgradient Method

todo