1 General

line search conditions (1st and 2nd order), how to choose step length and direction:

sufficient decrease, proportional to α_k step length and directional derivative $\nabla f_k^T p_k$:

$$f_{k+1}^T \leq f_k + c_1 \alpha_k \nabla f_k^T p_k \tag{1}$$
$$\nabla f_k^T p_k \leq 0 \ //by\ construction \tag{2}$$

curvature:

$$\nabla f_{k+1}^T p_k \geq c_2 \nabla f_k^T p_k \tag{3}$$
$$\frac{\partial \phi(\alpha_k)}{\partial \alpha_k} \geq \frac{\partial \phi(0)}{\partial \alpha_k} \tag{4}$$
$$c_1, \alpha_k \in (0,1) \tag{5}$$
$$0 < c_1 < c_2 < 1 \tag{6}$$

$$\begin{aligned} \text{where :} f_k &= f(x_k) \\ f_{k+1} &= f(x_k + \alpha_k p_k) \\ \phi(\alpha_k) &= f(x_k + \alpha_k p_k) \\ \frac{\partial \phi(\alpha_k)}{\partial \alpha_k} &= \nabla f_{k+1}^T p_k \\ \frac{\partial \phi(0)}{\partial \alpha_k} &= \nabla f_k^T p_k //initial\ slope \end{aligned}$$

strong Wolfe curvature condition, to restrict large positive derivative:

$$|\nabla f_{k+1}^T p_k| \leq c_2 |\nabla f_k^T p_k| \tag{7}$$

Form of search direction:

$$p_k = -B_k^{-1} \nabla f_k \tag{8}$$

B_k symmetric, non-singular, postive definite $\implies p_k$ is a descent direction:

$$\nabla f_k^T (-B_k^{-1} \nabla f_k) < 0 \tag{9}$$

Goldstein condition (may miss minimizer of f):

$$f_k + (1-c)\alpha_k \nabla f_k^T p_k \leq f_{k+1} \leq f_k + c\alpha \nabla f_k^T p_k \tag{10}$$

$$c \in (0, \frac{1}{2}) \tag{11}$$

Algorithm 1: Line Search

$f, x, d, c_1, \alpha, \beta$**:** function, x, direction, gradient threshold, initial step length, contraction

α **:** found step length

while $f(x + \alpha d) > f(x) + c_1 \alpha \nabla f(x)^T d$ **do**

  $\alpha \leftarrow \alpha * \beta$

return α

2 Quasi Newton

2.1 Concept

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

Derivation, using 2nd order model, with B_k SPSD:

$$m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$$

taking gradient wrt. p and solve, assuming minimum exists:

$$\begin{aligned} 0 &= \nabla f_k + B_k p \\ p &= -B_k^{-1} \nabla f_k \end{aligned}$$

update equation:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k p_k \\ x_{k+1} &= x_k - \alpha_k B_k^{-1} \nabla f_k \end{aligned}$$

pick α to satisfy Wolfe conditions

updated model at next iterate x_{k+1}:

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

impose reasonable conditions:

1. gradient of m_{k+1} matches gradient of objective function f at iterate x_k
2. gradient of m_{k+1} matches gradient of objective function f at iterate x_{k+1}

$$\begin{aligned} \nabla m_{k+1}(0) &= \nabla f_{k+1} \\ &\implies m_{k+1} \text{ condition 1 satisfied} \end{aligned}$$

condition 2 (gradient of m_{k+1} match gradient of objective f at iterate x_k):

$$\begin{aligned} \nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_{k+1} p_k &= \nabla f_k \\ \nabla f_{k+1} - \nabla f_k &= \alpha_k B_{k+1} p_k \\ y_k &= \nabla f_{k+1} - \nabla f_k \\ s_k &= x_{k+1} - x_k = \alpha_k p_k \\ y_k &= B_{k+1} s_k \text{ [secant equation]} \end{aligned}$$

B_{k+1} SPD $\implies s_k^T B_{k+1} s_k = s_k^T y_k > 0$
f strongly convex $\implies s_k^T y_k > 0$ satisfied for any 2 points x_k and x_{k+1}.
f nonconvex $\implies$ need to enforce $s_k^T y_k$ by Wolfe conditions, and use line search for step length α.

Using line search ensures curvature condition:

Proof.

$s_k = x_{k+1} - x_k = \alpha_k p_k$
$y_K = \nabla f_{k+1} - \nabla f_k$
Wolfe curvature condition:
$\nabla f_{k+1}^T p_k \geq c_2 \nabla f_k^T p_k, c_2 \in (0,1)$
$y_k^T s_k = (\nabla f_{k+1} - \nabla f_k)^T (\alpha_k p_k)$
$y_k^T s_k = \alpha_k (\nabla f_{k+1} - \nabla f_k)^T p^k$
$y_k^T s_k = \alpha_k (c_2 \nabla f_k - \nabla f_k)^T p^k$
$y_k^T s_k = \alpha_k (c_2 - 1) \nabla f_k^T p^k$
$\alpha > 0 \wedge c_2 - 1 < 0 \wedge \nabla f_k^T p_k < 0 (p_k$ is a descent dir)
$\implies y_k^T s_k > 0$

Curvature condition holds when Wolfe line search is performed. ☐

Choosing approximate Hessian, B:

Force a unique solution among infinite many due to extra degrees of freedom in the matrix, few (n) constraining conditions imposed by secand equation, few (n) constraining conditions of PD. One approach is solving a optimization problem to make row rank update to previous iterate:

$$\begin{aligned} &\min_B ||B - B_k|| \\ &s.t.\ B = B^T \\ &\quad Bs_k = y_k \text{ [seccant equation]} \\ &\quad B \succ 0 \end{aligned}$$

alternatively, constrain B's inverse, H:

$$\begin{aligned} &\min_H ||H - H_k|| \\ &s.t.\ H = H^T \\ &\quad Hy_k = s_k \qquad H_{k+1} y_k = s_k \text{ [secant equation]} \end{aligned}$$

Different norms can be used. One choice: weighted Frobenius norm:

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

2.2 DFT update algorithm

let $\bar{G_k}$ be the average Hessian:

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

using Taylor's theorem:
$y_k = \bar{G_k} s_k = \bar{G_k} \alpha_k p_k$
let $W = \bar{G_k}^{-1}$
solve optimization problem:

$$\begin{aligned} &\min_B ||B - B_k||_W \\ s.t.\ &W = \bar{G_k}^{-1} \\ &Bs_k = y_k \\ &B = B^T \end{aligned}$$

Solution:

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

Computation simplification: inverse Hessian used in update of search direction: $p + k = -B_k^{-1} \nabla f_k$
Use Sherman-Morrison-Woodbury formula to obtain inverse.
let $H_k = B_k^{-1}$, then DFP update becomes:

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

This is a rank 2 modification to previous iterate for efficiency.

2.3 BFGS update algorithm

Idea: impose condition on inverse of Hessian instead of Hessian approximation.

let H be approximate inverse of Hessian, then impose:

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

solve for H in the optimization problem:

$$\begin{aligned} &\min_H ||H - H_k||_W \\ s.t.\ &H = H^T \\ &Hy_k = s_k \end{aligned}$$

where $y_k = W s_k$

let W be the average Hessian:
$\bar{G}_k = \int_0^1 \nabla^2 f(x_k + \tau \alpha_k p_k) d\tau$

solve to obtain:

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

initial H_0 can be chosen approximately:

- finite diferences
- αI

Algorithm 2: BFGS Algorithm

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

$k \leftarrow 0$
$p_k \leftarrow -B^{-1} \nabla f(x_k) = -H \nabla f(x_k)$
while $||\nabla f_k|| > \epsilon$ **do**
  $\alpha_k \leftarrow$ LineSearch(..)
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$
  $s_k \leftarrow x_{k+1} - x_k$
  $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$
  $\rho_k \leftarrow \frac{1}{y_k^T s_k}$
  $H_{k+1} \leftarrow (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$
  $k \leftarrow k + 1$
return x

Cost: $O(n^2)$ + cost of eval $f(.)$ + cost of eval $\nabla f(.)$. Order of convergence: superlinear, worse than Newton but more computationally efficient than Newton.

Using Sherman-Morrison-Woodbury formula to obtain Hessian update equation,

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

, but is it more efficient to use the inverse version.

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

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

$$\begin{aligned}\Delta_k &:= \text{trust region radiius}\\ g_k &= \nabla f(x_k)\\ B_k &\succeq 0\end{aligned}$$

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 3: Trust Region Algorithm

```
k ← 0
while ||∇f_k|| > ε do
    p_k ←
      argmin_p f_k + g_k^T p + (1/2) 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))  //ratio test
    //change trust region for next iterate
    if ρ_k < γ(: 1/4) then
        Δ_{k+1} ← α(: 1/4) Δ_k  //shrink
    else if ρ_k > β(: 3/4) and ||p_k|| == Δ_k then
        Δ_{k+1} ← min(2Δ_k, Δ̂)  //expand
    else
        Δ_{k+1} ← Δ_k  //keep
    //accept/reject for current iterate
    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 as a global solution for trust region iff:

$$\begin{aligned}(B + \lambda I)p^* &= -g\\ &\text{complementary slackness:}\\ \lambda(\Delta - ||p^*||) &= 0\\ (B + \lambda I) &\succeq 0\\ \lambda &\geq 0\end{aligned}$$

$\lambda \geq 0 \implies p$ parallel to negative gradient of model

Solving 2nd order Taylor series using approx methods:

- dogleg
- 2-D subspace minimization
- conjugate gradient based, effective when B large, sparse

These reduce at least as much as Cauchy Point.

Cauchy Point (sufficient reduction in model value): 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)
Resulting algo, interpolate between p^U and p^B:

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

$$\begin{aligned} B \succ 0 \implies & \|\tilde{p}(\tau)\| \text{increases wrt. } \tau \wedge \\ & m(\tilde{p}(\tau)) \text{decreases wrt. } \tau \end{aligned}$$

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:
$\|p^U + (\tau - 1)(p^B - p^U)\|^2 = \|\Delta^2\|$
$p_k^S = \underset{p}{\operatorname{argmin}}\, f_k + g_k^T p, \|p\| \leq \Delta_k$
$\tau_k = \underset{\tau \geq 0}{\operatorname{argmin}}\, m_k(\tau p_k^S), \|\tau p_k^S\| \leq \Delta_k$
$p_k^S = \frac{-\Delta_k g_k|}{\|g_k\|}$
$p_k^C = \tau_k p_k^S$
$p_k^C = -\tau_k \frac{\Delta_k g_k}{\|g_k\|}$

$$\tau_k = \begin{cases} 1 & , g_k^T B_k g_k \leq 0 \\ min(\frac{\|g_k|^3}{\Delta_k g_k^T B_k g_k}, 1) & , o/w \end{cases}$$

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. $(B + \lambda I) \succeq 0$ 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 4: 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 $\lambda \geq \lambda_1$

3.3 Trust Region Newton CG Method

Idea: use trust region algo for the outer iteration, use iterative CG based algorithm to solve for inner optimization problem.

Outer iteraion: Algo 3.

Inner optimization problem:

$$\begin{aligned} \min_{p \in \mathbb{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p \\ s.t. \|p\| \leq \Delta \end{aligned}$$

Algorithm 5: Trust Region Newton-CG Subproblem (CG Steinhaug)

$\epsilon_k = \eta_k \|\nabla f_k\|$
$z_0 = 0$
$r_0 = \nabla f_k$
$d_0 = -r_0 = \nabla f_k$
if $\|r_0\| < \epsilon_k$ **then**
  return $p_k = z_0 = 0$
for $j = 0, 1, ..$ **do**
  //dir of nonpositive curvature test
  if $d_j^T B_k d_j \leq 0$ **then**
    return $\operatorname{argmin}_{p_k} m_k(p_k = z_j + \tau d_j)$ s.t. $\|p_k\| = \Delta_k$
  $\alpha_j = \frac{r_j^T r_j}{d_j^T B_k d_j}$
  $z_{j+1} = z_j + \alpha_j d_j$
  //trust region check
  if $\|z_{j+1}\| \geq \Delta_k$ **then**
    return $p_k : p_k = z_j + \tau d_j$ *s.t.* $\|p_k\| = \Delta_k, \tau \geq 0$
  $r_{j+1} = r_j + \alpha_j B_k d_j$
  if $\|r_{j+1} < \epsilon_k\|$ **then**
    return $p_k = z_{j+1}$
  $B_{j+1} = \frac{r_{j+1}^T r_{j+1}}{r_j^T r_j}$
  $d_{j+1} = -r_{j+1} + B_{j+1} d_j$

ϵ_j can be chosen similar as in Line Search Newton-CG method, where $\{\eta_k\}$ is the forcing sequence.

4 Conjugate Gradient

4.1 linear method

Assuming unconstrained problem with strict convex quadratic objective function:

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

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

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

$$\begin{aligned}
&Ax_{k+1} - b = 0\\
&A(x_k + \alpha_k p_k) - b = 0\\
&\alpha_k Ap_k = b - Ax_k\\
&r_k = Ax - b\\
&\alpha_k Ap_k = -r_k\\
&\alpha_k p_k^T Ap_k = -p_k^T r_k\\
&\alpha_k = -\frac{p_k^T r_k}{p_k^T Ap_k}
\end{aligned}$$

4.2 Conjugate Direction

Enforce by construction for search directions linearly independent wrt. A.:

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

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:
  - eigenvectors
  - Gram-Schmidt
  - conjugate gradient algo.

4.2.1 Termination Steps

Theorem 4.1 (Conjugate Direction Termination)**.** *Conjugate direction algorithm converges to solution $x \in \mathbb{R}^n$ of linear system in n steps. Theorem [T5.1] in Num. Opt. book.*

Proof.

given:

$x_{k+1} = x_k + \alpha_k p_k$

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

$(\forall i \neq j)p_i^T Ap_j = 0$ [A conjugate]

then:

$x^* - x_0 = \sum_{i=0}^{n-1} \sigma_i p_i$

$p_k^T A(x^* - x_0) = p_k^T A(\sum_{i=0}^{n-1} \sigma_i p_i)$

$\sigma_i = \frac{p_k^T A(x^* - x_0)}{p_k^T Ap_k}$ (from conjugacy)

$x_k = x_0 + \sum_{i=0}^{k-1} \alpha_i p_i$

$p_k^T Ax_k = p_k^T A(x_0 + \alpha_0 p_0 + ..)$

$p_k^T Ax_k = p_k^T Ax_0$ (from conjugacy)

$p_k^T A(x_k - x_0) = 0$

$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k)$

$p_k^T A(x^* - x_0) = p_k^T (b - Ax_k) = -p_k^T r_k$

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

α_k is 1D minimizer of kth coordinate by construction

Conjugate direction algo. terminates in n steps. $\square$

If A is not diagonal, can transform coordinates:

$$\begin{aligned}
S\hat{x} &= x\\
S &= [p_0 \ p_1 \ .. \ p_{n-1}], \ (\forall i \neq j) \ p_i Ap_j = 0\\
\phi(x) &= \frac{1}{2}x^T Ax - b^T x\\
\phi(\hat{x}) &= \frac{1}{2}\hat{x}^T S^T AS\hat{x} - (S^T b)^T \hat{x}
\end{aligned}$$

$S^T AS \ diagonal \implies$ can optimize one coordinate at a time

4.2.2 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 $\frac{1}{2}x^T Ax - b^T x$
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 $\{x_0 + span\{p_0, ...p_{k-1}\}\} \iff r(\tilde{x})^T p_i = 0$
$h(\sigma) = \phi(\tilde{x})$
$$\phi(x) = \frac{1}{2}x^T Ax - b^T x$$
h is also strictly convex quadratic,
with unique σ^* satisfying:
$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = 0, i = [0, k-1]$$
$$\frac{\partial h(\sigma^*)}{\partial \sigma_i} = \nabla\phi(\tilde{x})^T p_i = 0, i = [0, k-1]$$
$\nabla\phi(x) = Ax - b = r$
$r(\tilde{x})^T p_i = 0, i = [0, k-1]$

□

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

Proof.

base case of k=1, i=0 : $x_1 = x_0 + \alpha_0 p_0$ minimizes ϕ along p_0
$\implies r_1^T p_0 = (r_0 + \alpha_0 Ap_0)^T p_0 = 0$
case: $r_{k-1}^T p_i = 0, i = [0, k-2]$:
$r_k = r_{k-1} + \alpha_{k-1} Ap_{k-1}$
$(\forall i \in [0, k-2])p_i^T r_k = p_i^T r_{k-1} + \alpha_{k-1} p_i^T Ap_{k-1}$
$(\forall i = [0, k-2])p_i^T Ap_{k-1} = 0$ (A-conjugacy by construction)
$(\forall i = [0, k-2])p_i^T r_{k-1} = 0$ (by induction hypothesis)
$p_i^T r_k = 0, i = [0, k-1]$

□

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 6: 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
    $\alpha_k \leftarrow \frac{-r_k^T p_k}{p_k^T A p_k} = \frac{r_k^T r_k}{p_k^T A p_k}$
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $r_{k+1} \leftarrow A x_{k+1} - b$
    $\beta_{k+1} \leftarrow \frac{p_k^T A r_{k+1}}{p_k^T A p_k} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
    $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$

p_k 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:

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

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

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

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:

$$\beta_{k+1}^{PR} \leftarrow \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$$
$$\beta_{k+1}^{+} \leftarrow \max(\beta_{k+1}^{PR}, 0)$$

CG Gradient Algo. Property [T.5.3]:

$$(\forall i = [0, k-1])\ r_k^T r_i = 0$$
$$span\{r_0, .., r_k\} = span\{r_0, Ar_0, .., A^k r_0\}$$
$$span\{p_0, .., p_k\} = span\{r_0, Ar_0, .., A^k r_0\}$$
$$(\forall i = [0, k-1])\ p_k^T A p_i = 0$$
$$\implies \{x_k\} \text{ converges to } x^* \text{ in at most n steps}$$

Proof by induction to show generated search direcitons are A-conjugate. Then apply Theorem 4.1 to conclude algo terminates within n steps.

5 Proximal Algorithm

Idea:

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

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

Resolvent of subdifferential operator:

$$z = prox_{\lambda f}(x) \implies z \in (I + \lambda \partial f)^{-1}(x)$$
$$(I + \lambda \partial f)^{-1} := \text{resolvent of operator } \partial f$$

5.1 Proximal Gradient Method

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

$$x^* = prox_{\lambda^k g}(x^k - \lambda^k \nabla f(x^k))$$
$$= \operatorname*{argmin}_x g(x) + \frac{1}{2\lambda^k}\|x - (x^k - \lambda^k \nabla f(x^k))\|_2^2$$

tradeoff between g and and gradient step

$g = I_C(x) \implies$ projected gradient step

$g = 0 \implies$ gradient descent

$f = 0 \implies$ proximal minimization

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:

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

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 7: Proximal Gradient Algorithm

```
f̂(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) ≤ f̂(z, y^k) then
        break
    λ = βλ
return λ^k := λ, x^{k+1} := z
```

5.3 Types of Proximal Operators

- quadratic functions

$$f = \frac{1}{2}\|.\|_x^2 \implies prox_{\lambda f}(v) = (\frac{1}{1+\lambda})v$$
$$f = \frac{1}{2}x^T Ax + b^T x + c, A \in S_+^n \implies$$
$$prox_{\lambda f}(v) = (I+\lambda)^{-1}(v - \lambda b)$$

- 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 Smoothness

β smoothness:

$$\frac{\beta}{2}\|x\|_2^2 - f(x) \text{ is convex (fit quadratic on top)}$$
$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{\beta}{2}\|y-x\|_2^2$$
$$\implies \text{f is smooth}$$

α strong convexity:

$$f(x) - \frac{\alpha}{2}\|x\|_2^2 \text{ is convex (fit quadratic below)}$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\alpha}{2}\|y-x\|_2^2$$
$$\text{f may not be smooth}$$

Smoothness near optimal point: there always exists a step (length $< \frac{2}{\beta}$) that is a descent direction.

Property of f wrt. optimality value:
f is β-smooth $\implies$

$$\frac{1}{2\beta}\|\nabla f(x)\|_2^2 \leq f(x) - f(x^*) \leq \frac{\beta}{2}\|x-x^*\|_2^2$$

f is α-strongly convex $\implies$

$$\frac{\alpha}{2}\|x-x^*\|_2^2 \leq f(x) - f(x^*) \leq \frac{1}{2\alpha}\|\nabla f(x)\|_2^2$$

Co-coercivity for β-smooth f:

$$(\nabla f(x) - \nabla f(y))^T (x-y) \geq \frac{1}{\beta}\|\nabla f(x) - \nabla f(y)\|_2^2$$

Coercivity for α-strongly convex f:

$$(\nabla f(x) - \nabla f(y))^T (x-y) \geq \alpha\|x-y\|_2^2$$

6.1 Oracle based lower bounds

Lipschitz convex function: error=$\mathcal{O}(\frac{1}{\sqrt{T}})$
Smooth convex function: $\epsilon = \mathcal{O}(\frac{1}{T^2})$
Smooth + strongly convex function: $\epsilon = \mathcal{O}(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})$

6.2 Gradient descent rate of convergence

β-smooth f: $\mathcal{O}(\frac{1}{T})$
β-smooth + α-strongly convex f: $\mathcal{O}((\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^T)$

6.3 Gradient descent with momentum

f β-smooth, α-s.c.:

$$x_1 = y_1 = x_{init}$$
$$y_{t+1} = x_t - \frac{1}{\beta}\nabla f(x_t)$$
$$x_{t+1} = \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) y_{t+1} - \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) y_t$$

f β-smooth:

$$x_{t+1} = x_t - \eta\nabla f(x_t) \text{ [normal GD]}$$
momentum, nesterov accel.:
$$d_{t+1} = \gamma_{t+1}(x_{t+1} - x_t)$$
$$x_{t+1} = (x_t + d_t) - \eta\nabla f(x_t + d_t)$$
$$\epsilon = \mathcal{O}\left(\frac{1}{T^2}\right)$$

7 Common Algorithms

7.1 Projected Gradient

$$y_{t+1} = x_t - \eta g_t, g_t \in \partial f(x_t)$$
$$x_{t+1} = proj_X(y_{t+1})$$
$$x^* = proj_X(x^*) \text{ [fixed point at optimality]}$$

7.2 Proximal Gradient

$$prox_{\eta h}(y) = \underset{x}{\operatorname{argmin}}\, h(x) + \frac{1}{2\eta}\|x-y\|_2^2$$
$$x_{t+1} = prox_{\eta h}(x_t - \eta\nabla f(x_t))$$

Special case of projected gradient:
let $h(x) = I_X(x) \implies prox_{\eta h} = proj_X$
where $I_X(x) = \begin{cases} 0 & , x \in X \\ +\infty & , o/w \end{cases}$

7.2.1 Common proximal operators

$h(x) = \|x\|_1$:

$$(prox_{\eta h}(x))_i = \begin{cases} x_i - \eta & , x_i \geq \eta \\ 0 & , |x_i| \leq \eta \\ x_i + \eta & , x_i \leq -\eta \end{cases}$$
$$= \max(|x_i| - \eta, 0)\ sign(x_i)$$

$h(x) = \frac{1}{2}x^TQx + q^Tx + q_0, Q \succeq 0$:

$$prox_{\eta h} = (I + \eta Q)^{-1}(x - \eta q)$$

$h(x) = \sum_i h_i(x_i)$:

$$(prox_{\eta h}(x))_i = prox_{\eta h_i}(x_i) \text{ [parallelism]}$$

Composite function:
$\min_x f(x) = g(x) + h(x)$, g smooth, h not smooth but has prox. operator.
Convergence due to non-smooth function: $\mathcal{O}(\frac{1}{\sqrt{T}})$

7.3 L1 regularization with subgradient descent

$$\begin{aligned}
&\min_x \|Ax - y\|_2^2 + \|x\|_1 \\
&g = \|Ax - y\|_2^2 \\
&h = \lambda\|x\|_1 \\
&\partial_x h(x) = \begin{cases} -1 & , x_i < 0 \\ [-1, 1] & , x_i = 0 \\ 1 & , x_i > 0 \end{cases} \\
&x_{t+1} = x_t - \eta(\nabla g(x_t) + \lambda \partial h(x)) \\
&x_{t+1} = x_t - \eta(2A^T(Ax - y) + \lambda z) \\
&z = \begin{cases} -1 & , x_i < 0 \\ [-1, 1] & , x_i = 0 \\ 1 & , x_i > 0 \end{cases}
\end{aligned}$$

Convergence: $\epsilon = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$

7.4 ISTA (with proximal gardient)

$$\begin{aligned}
&\min_x g(x) + h(x), g \ \beta\text{-smooth} \\
&g(x) = \|Ax - y\|_2^2 \\
&h(x)\lambda\|x\|_1 \\
&x_{t+1} = prox_{\eta h}(x_t - \eta \nabla g(x_t)) \\
&g \ \beta\text{-smooth, select } \eta = \frac{1}{\beta} \\
&x_{t+1} = prox_{\frac{1}{\beta}\lambda\|\dot{\|}_1}(x_t - \frac{1}{\beta}\nabla g(x_t)) \\
&x_{t+1} = \operatorname*{argmin}_x \lambda\|x\|_1 + \frac{\beta}{2}\|x - (x_t - \frac{1}{\beta}\nabla g(x_t))\|_2^2 \\
&\nabla g(x_t) = 2A^T(Ax - y)
\end{aligned}$$

Convergence: $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$

7.5 FISTA

$$\begin{aligned}
&\min_x g(x) + h(x), g \ \beta\text{-smooth} \\
&\mu_0 = 0 \\
&\mu_t = \frac{1 + \sqrt{1 + 4\mu_t^2 - 1}}{2} \\
&\gamma_t = \frac{1 - \mu_t}{\mu_{t+1}} \\
&x_1 = z_1 = \text{arbitrary} \\
&x_{t+1} = (1 - \gamma_t)x_{t+1} + \gamma_t z_t \\
&z_{t+1} = prox_{\frac{\lambda}{\beta}\|\dot{\|}_1}(x_t - \frac{1}{\beta}\nabla g(x_t))
\end{aligned}$$

Convergence: $\epsilon = \mathcal{O}\left(\frac{1}{T^2}\right)$

8 Subgradient Method

TODO

9 Large Scale Optimization