April 5, 2021

Newton: \( \min_{x \in \mathbb{R}^n} f(x) \)

for \( h = 1, 2, 3, \ldots \)

\[
\begin{align*}
    x_{h+1} &= x_h - H_k^{-1} \nabla f_k \\
    H_k &\text{ SPD } \implies \text{ descent direction}
\end{align*}
\]

How expensive?

1. compute \( H_k, \nabla f_k \)
2. \( H_k P_k = 0f_k \quad O(n^3) \)

Problems:

1. difficult to compute \( H_k \)
2. \( O(n^3) \) expensive
3. \( O(n^2) \) storage

Quasi-Newton: cheap Hessian approx

\[
\begin{align*}
    H(x_h) &\approx B_k \\
    x_{h+1} &= x_h - B_k^{-1} \nabla f_k
\end{align*}
\]
\[ m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p \]

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

\[ x_{k+1} = x_k - B_k^{-1} \nabla f_k \]

Idea: model gradient matches \( \nabla f \) at \( x_{k+1}, x_k \)

\[ \nabla m_{k+1}(p) = \nabla f_{k+1} + B_{k+1} p \]

\[ x_{k+1}(p=0) \Rightarrow \nabla m_{k+1}(0) = \nabla f_{k+1} \]

\[ x_k(p=-p_k) \Rightarrow \nabla m_{k}(p=-p_k) = \nabla f_{k+1} - B_{k+1} p_k = \nabla f_k \]

\[ B_{k+1} p_k = \nabla f_{k+1} - \nabla f_k \]

\[ B_{k+1} (x_{k+1} - x_k) = \nabla f_{k+1} - \nabla f_k \]

\( n^2 \) DOF to satisfy \( n \) linear equations
\( x_0, \nabla f_0, B_0 = \beta I \)

for \( k = 0, 1, 2, \ldots \)

\[
P_k = -B_k^{-1} \nabla f_k
\]

\[
x_{k+1} = x_k + p_k
\]

\[
y_k = \nabla f_{k+1} - \nabla f_k
\]

Compute \( B_{k+1} \) from \( B_k, p_k, y_k \) check for convergence

\( B_{k+1} = \arg \min_B \| \nabla f_k - B_k y_k \| \\
\text{s.t. } B = B^T \)

\[
b_k = f''(x_k)
\]

\[
x_{k+1} = x_k - f'(x_k) / b_k
\]

\[
B_k (x_k - x_{k-1}) = f'(x_k) - f'(x_{k-1})
\]

\[
B_k = f'(x_k) - f'(x_{k-1})
\]

\[
x_{k+1} = x_k - f'(x_k) / b_k
\]

\[
x_{k+1} = x_k - f'(x_k) (x_k - x_{k-1}) / f''(x_k) - f'(x_{k-1})
\]

\[
x_{k+1} = x_k - f'(x_k) (x_k - x_{k-1}) / f'(x_k) - f'(x_{k-1})
\]

\( x_{k+1} \) is the next iteration, \( g = f' \)
BFGS

Idea: maintaining $B^{-1} = G_k$
For a particular choice of norm

$$G_{k+1} = (I - \frac{P_k y_k}{\gamma_k}) G_k (I - \frac{y_k P_k}{\gamma_k})$$
$$+ P_k P_k^T / \gamma_k$$

$$\gamma_k = P_k y_k$$

$$= J G_k^T + F \ (F = F^T)$$

$$O(h^n)$$

$$G_0 = \beta I \quad G_0 \nabla f_0 \quad O(n)$$

$$G_{k+1} \nabla f_{k+1} = \beta_{k+1}$$

$$\nabla f_{k+1} = J^T \nabla f_{k+1} \quad O(h^n)$$

$$z_{k+1} = \nabla f_{k+1} - y_k (P_k^T \nabla f_{k+1}) / \gamma_k$$

$$w_{k+1} = G_k z_{k+1} \quad O(kn)$$

$$J w_{k+1} \quad O(n)$$

$$F \nabla f_{k+1} = \beta_k (P_k^T \nabla f_{k+1}) / \gamma_k$$

$$O(n)$$

$$\alpha (k+1) n$$

Benefits:

1. Only need $\nabla f_k$
2. $P_{k+1} = G_k \nabla f_k$ cheap if $k$ not too large
3. Converges like $\| x_{k+1} \| = \| x_k \|^a \quad \alpha > 0
BFGS

\[ \nabla f_0, \ G_0 = \beta I, \ x_0 \]

for \( h = 0, 1, 2, \ldots \), implicitly

\[ P_h = -G_h \nabla f_h \]

\[ x_{h+1} = x_h + \alpha_h P_h \]

\[ y_h = \nabla f_{h+1} - \nabla f_h \]

store \( y_h, x_h, \alpha_h, P_h \) for \( G_{h+1} \)

Limited memory BFGS (L-BFGS)

only use last \( l \) (\(< 20\)) steps

of \( y_h, x_h, \alpha_h, P_h \) when computing \( G_{h+1} \nabla f_h \)

Why step size is useful?

\[ f(x_{h+1}) = f(x_h) + \nabla f_h^T (x_h P_h) + O(\|x_h P_h\|^2) \]

\[ = f(x_h) - x_h \nabla f_h^T G_h^\top G_h y_h > 0 \]

if \( G_h \) SPD, \( \exists \ x_h > 0 \) s.t.

\[ f(x_{h+1}) < f(x_h) \]

Claim: \( G_{h+1} \) is SPD

if \( G_h \) SPD and \( x_h \) chosen "appropriately"
\[ g_{h+1} = J_h g_h J_h^T + F_h \]
\[ = J_h (J_{h-1} g_{h-1} J_{h-1}^T + F_{h-1}) + F_h \]
\[ = J_h (J_{h-1} (J_{h-2} g_{h-2} J_{h-2}^T + F_{h-2}) + F_{h-1}) + F_h \]

\[ g_{h+1} v = (I - xy^T) v \]
\[ = v - x(y^T v) \]