Optimization

Find local minimum of $f(x)$

Brent's Method → bracketing

Newton's Method: find $f'(x)=0$

let x_n be a tentative value near the minimum x^*.

Taylor Series

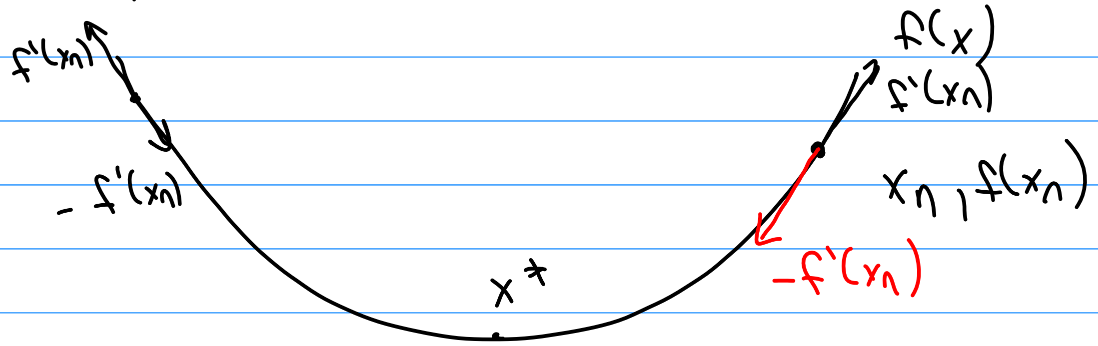
$$f'(x) = f'(x_n) + f''(x_n)(x-x_n) + \cancel{\tfrac{1}{2} f'''(x_n)(x-x_n)^2 + H.O.T.}$$

$$\hat{f}'(x) = f'(x_n) + f''(x_n)(x-x_n) = 0$$

$$\Rightarrow x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

1) might diverge if $f''(x_n) \sim 0$
2) might oscillate

Steepest Gradient Descent

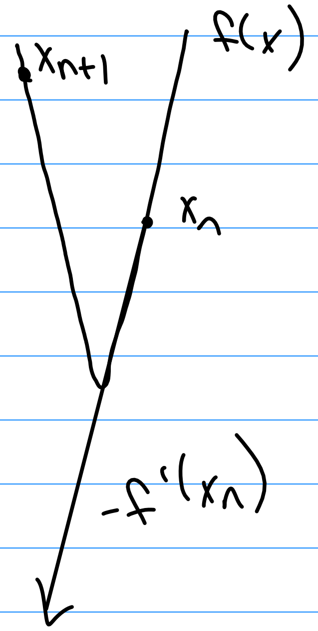


$$x_{n+1} = x_n - \alpha_n f'(x_n)$$

↑ will need damping

You need damping as you do not know how $f'(x)$ behaves

ex.)



Multivariable Minimization

Minimize $f(\underline{x})$ w/ $\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

A scalar

ex.) minimize $f(x_1, x_2) = \sin(x_1)\cos(x_2)$

Minimum occurs where

$$\nabla f(\underline{x}^*) = \begin{bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Taylor Series for $\underline{x}_n$ near minimum, $\underline{x}^*$

$$\nabla f(\underline{x}) = \nabla f(\underline{x}_n) + \underline{H}(\underline{x}_n)(\underline{x} - \underline{x}_n) + \cancel{H.O.T.}$$

$\underline{H}(\underline{x}) =$ Hessian Matrix of $f(\underline{x})$

$$= \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \cdots & \partial^2 f/\partial x_1 \partial x_n \\ \partial^2 f/\partial x_2 \partial x_1 & \partial^2 f/\partial x_2^2 & \cdots & \partial^2 f/\partial x_2 \partial x_n \\ \vdots & & & \\ \partial f/\partial x_n \partial x_1 & \cdots & & \partial^2 f/\partial x_n^2 \end{bmatrix}$$

$$\underline{H}(\underline{x}) = \underline{H}^T(\underline{x})$$

To first-order, $\nabla \hat{f}(\underline{x}) = \nabla f(\underline{x}_n) + \underline{H}(\underline{x}_n)(\underline{x} - \underline{x}_n)$

$$\nabla \hat{f}(\underline{x}_{n+1}) = \underline{0}$$

$$\Rightarrow \underline{x}_{n+1} = \underline{x}_n - \underline{H}_n^{-1} \nabla f(\underline{x}_n)$$

Write as:

1) Solve $\underline{H}_n \underline{\delta}_n = -\nabla f_n$
2) Update: $\underline{x}_{n+1} = \underline{x}_n + \alpha_n \underline{\delta}_n$

Notes: - $\underline{H}_n$ changes every iteration
- Need to solve a linear system each iteration
- $\underline{H}_n$ might be expensive or unknown

Quasi-Newton Methods

Idea: Find a $\underline{B}_n \approx \underline{H}_n$ where
$\underline{B}_n$ is cheap to compute &
solving $\underline{B}_n \underline{\delta}_n = -\nabla f_n$ is also cheap

Need: A sequence $\underline{B}_0, \underline{B}_1, \underline{B}_2$, etc.
1) How to chose $\underline{B}_0$?
2) How does $\underline{B}_n$ depend on $\underline{B}_{n-1}, \underline{B}_{n-2}$, solution, etc.

General Quasi-Newton Algorithm
let x_0 & $\underline{B}_0$ be known
$n=0$
until converged
Solve $\underline{B}_n \underline{\delta}_n = -\nabla f_n$
$\underline{x}_{n+1} = x_n + \alpha_n \underline{\delta}_n$ via line search
$\underline{y}_n = \nabla f(\underline{x}_{n+1}) - \nabla f(\underline{x}_n) = \nabla(f(\underline{x}_{n+1}) - f(\underline{x}_n))$
$\underline{B}_{n+1} = $ function of $\underline{B}_n, \underline{\delta}_n, \underline{y}_n$

Simple choice for $\underline{B}_0$

let $\hat{\underline{\delta}}_0 = -\nabla f(\underline{x}_0)$ $\quad \underline{y}_0 = \nabla(f(\underline{x} + \hat{\underline{\delta}}_0) - f(\underline{x}_0))$

$$\underline{B}_0 = \frac{\underline{y}_0^T \underline{y}_0}{\underline{y}_0^T \underline{\delta}_0} \underline{I} \qquad \underline{B}_0^{-1} = \frac{\underline{y}_0^T \hat{\underline{\delta}}_0}{\underline{y}_0^T \underline{y}_0} \underline{I}$$

Thus, in iteration $\underline{\delta}_0 = -\dfrac{\underline{y}_0^T \hat{\underline{\delta}}_0}{\underline{y}_0^T \underline{y}_0} \nabla f_0$

Many choices for B_n

- Davidon - Fletcher - Powell (DFP) method

$$B_{n+1} = \left(I - \frac{y_n \delta_n^T}{y_n^T \delta_n} \right) B_n \left(I - \frac{\delta_n y_n^T}{y_n^T \delta_n} \right) + \frac{y_n y_n^T}{y_n^T y_n}$$

$$B_{n+1}^{-1} = B_n^{-1} + \frac{\delta_n \delta_n^T}{\delta_n^T y_n} - B_n^{-1} \left(\frac{y_n y_n^T}{y_n^T B_n^{-1} y_n} \right) B_n^{-1}$$

Given B_0^{-1}, B_1^{-1} is known,
Given B_1^{-1}, B_2^{-1} is known, etc.

- Broyden - Fletcher - Goldfarb - Shanno (BFGS)

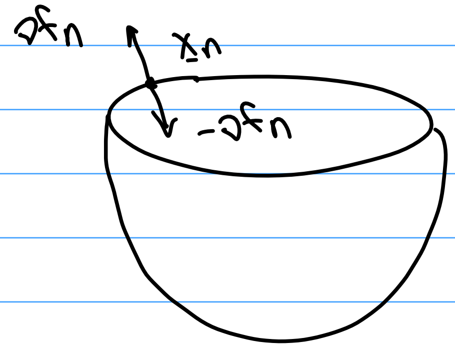
Based on outer products:

$$B_{n+1} = B_n + \frac{y_n y_n^T}{y_n^T \delta_n} - B_n \left(\frac{\delta_n \delta_n^T}{\delta_n^T B_n \delta_n} \right) B_n^T$$

$$B_{n+1}^{-1} = \left(I - \frac{\delta_n y_n^T}{y_n^T y_n} \right) B_n^{-1} \left(I - \frac{y_n \delta_n^T}{y_n^T \delta_n} \right) + \frac{\delta_n y_n^T}{y_n^T \delta_n}$$

Multidimensional Gradient Descent

A 2D convex function $f(x_1, x_2)$



$$\Rightarrow \underline{x}_{n+1} = \underline{x}_n - \alpha_n \nabla f(\underline{x}_n)$$

Line Search / Backtracking

Many methods written as $\underline{x}_{n+1} = \underline{x}_n + \alpha_n \underline{\delta}_n$

Newton: $\underline{\delta}_n = -\underline{J}_n^{-1} \underline{f}_n$

Gradient Descent: $\underline{\delta}_n = -\nabla f(\underline{x}_n)$

Minimization: $\underline{\delta}_n = -\underline{H}_n^{-1} \nabla f(\underline{x}_n)$

In all cases you want

$f(\underline{x}_n + \alpha_n \underline{\delta}_n) < f(\underline{x}_n)$ minimization

or

$\|\underline{f}(\underline{x}_n + \alpha \underline{\delta}_n)\| < \|\underline{f}(\underline{x}_n)\|$ root finding

Focus on $f(\underline{x})$ (or $f(\underline{x}) = \|g(\underline{x})\|$)

It is not sufficient to simply require $f(\underline{x}_n + \alpha_n \underline{\delta}_n) < f(\underline{x}_n)$ ←

ex) Some minimization problem has
$f(\underline{x}_n) = 5$

minimum is never achieved.

Wolfe Conditions

Require that

1) $f(\underline{x}_n + \alpha_n \underline{\delta}_n) \le f(\underline{x}_n) + c_1 \alpha_n (\nabla f_n)^T \underline{\delta}_n$
for some $c_1 \in (0, 1)$

This requires sufficient progress towards the solution.

In practice c_1 is small, say $c_1 \sim 10^{-4}$

2) To avoid small α_n require that

$$(\nabla f(\underline{x}_n + \alpha_n \underline{\delta}_n))^T \underline{\delta}_n \ge c_2 (\nabla f_n)^T \underline{\delta}_n$$

for some $c_2 \in (c_1, 1)$

Thus $0 < c_1 < c_2 < 1$

It can be shown that if $f: \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, $\underline{\delta}_n$ is a descent direction at $\underline{x}_n$, and if $0 < c_1 < c_2 < 1$ there exists a range of α_n that satisfy the Wolfe Conditions

One method for α_n:

$$\alpha_n = \frac{(\overbrace{\underline{x}_n - \underline{x}_{n-1}}^{\underline{\delta}_{n-1}})^T (\overbrace{\nabla(f(\underline{x}_n) - f(\underline{x}_{n-1}))}^{\underline{y}_{n-1}})}{\| \nabla f(\underline{x}_n) - \nabla f(\underline{x}_{n-1}) \|_2^2}$$

Another method

Define $\phi(\alpha) = f(\underline{x}_n + \alpha \underline{\delta}_n)$ (known: $\underline{x}_n$, $\underline{\delta}_n$) $\quad \phi(0) = f(\underline{x}_n)$

$\phi'(\alpha) = \underline{\delta}_n \cdot \nabla f(\underline{x}_n + \alpha \underline{\delta}_n) \quad \phi'(0) = \underline{\delta}_n \cdot \nabla f(\underline{x}_n)$

Wolfe Condition #1: $\phi(\alpha) \le \phi(0) + c_1 \alpha \phi'(0)$

General Idea: choose α_0, say $\alpha_0 = 1$ (full step)

If $\phi(\alpha_0) \le \phi(0) + c_1 \alpha_0 \phi'(0)$ use α_0

If not satisfied actual $\alpha \in (0, \alpha_0)$

Currently Given: $\phi(0)$, $\phi'(0)$, $\phi(\alpha_0)$

Form an interpolant

$$\hat{\phi}(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \right) \alpha^2 + \phi'(0)\alpha + \phi(0)$$

Find minimum of $\hat{\phi}(\alpha)$

$$\alpha_1 = \frac{-\phi'(0)\, \alpha_0^2}{2[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0]}$$

If $\phi(\alpha_1) \le \phi(0) + c_1 \alpha_1 \phi(0)$ use α_1

If not form a cubic polynomial w/ $\phi(0)$, $\phi'(0)$, $\phi(\alpha_0)$, $\phi(\alpha_1)$

$$\tilde{\phi}(\alpha) = a\alpha^3 + b\alpha^2 + c\alpha + d$$

Solve $\tilde{\phi}(\alpha_2) = 0$

$$\alpha_2 = \frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_0^2 \alpha_1^2 (\alpha_1 - \alpha_0)} \begin{bmatrix} \alpha_0^2 & -\alpha_1^2 \\ -\alpha_0^3 & \alpha_1^3 \end{bmatrix} \begin{bmatrix} \phi(\alpha_1) - \phi(0) - \phi'(0)\alpha_1 \\ \phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0 \end{bmatrix}$$

Check if $\phi(\alpha_2) \le \phi(0) + c_1 \alpha_2 \phi'(0)$

If not true repeat $\phi(0)$, $\phi'(0)$, $\phi(\alpha_{k-1})$, $\phi(\alpha_{k-2})$

Sequence of

$\alpha_0 > \alpha_1 > \alpha_2 > \dots > \alpha_{k-1} > \alpha_k > \dots > 0$

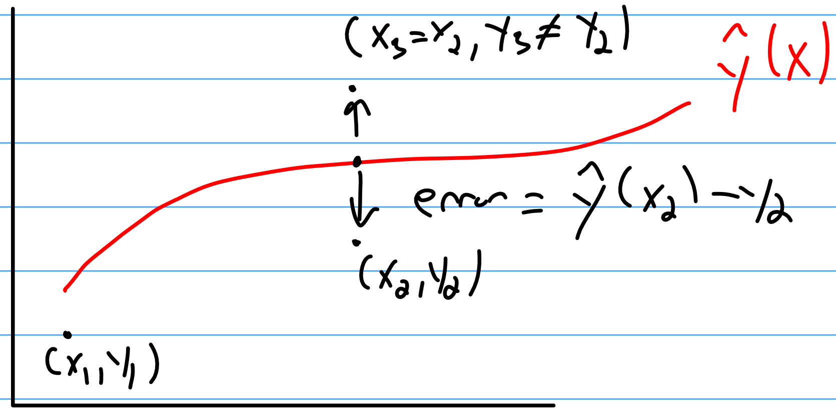
If any $\alpha_k - \alpha_{k-1} < \varepsilon$ or $\alpha_k << \alpha_{k-1}$

or $\alpha_k = 1/2\, \alpha_{k-1}$

See "Numerical Optimization" by Nocedal & Wright

Regression : Curve fitting

Regression → fit an equation to a set of Data but do not require that it pass through the Data,



Goal: minimize the error

Define the error @ point (x_i, y_i) as

$$r_i = y_i - \hat{y}(x_i, \underline{a})$$

$\underline{a}$ = list of coefficients in the model equation

ex.) $\hat{y}(x, \underline{a}) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$
example of linear regression

ex.) $\hat{y}(x, \underline{a}) = a_0 \sin(x) + a_1 \sin(2x) + a_2 \cos(x) + a_3 \cos(2x)$
Linear regression

ex.) $\hat{y}(x, \underline{a}) = \dfrac{a_1 x}{a_2 + x}$ non linear regression ex.) $\hat{y}(x, \underline{a}) = a_0 \sin(a_1 x) + a_2 \cos(a_2 x)$

non-linear regression

For both, need to minimize $\|\underline{r}\|_2$

$$\underline{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} y_1 - \hat{y}(x_1, \underline{a}) \\ y_2 - \hat{y}(x_2, \underline{a}) \\ \vdots \\ y_n - \hat{y}(x_n, \underline{a}) \end{bmatrix}$$ for n-data points

For Linear regression:

$$\hat{y}(x, \underline{a}) = a_1 f_1(x) + a_2 f_2(x) + \cdots + a_p f_p(x)$$

$$\hat{y}(x_1, \underline{a}) = a_1 f_1(x_1) + a_2 f_2(x_1) + \cdots + a_p f_p(x_1) = y_1$$

$$\hat{y}(x_2, \underline{a}) = a_1 f_1(x_2) + \cdots + a_p f_p(x_2) = y_2$$

$$\vdots$$

$$\hat{y}(x_n, \underline{a}) = a_1 f_1(x_n) + \cdots + a_p f_p(x_n) = y_n$$

$$\Downarrow$$

$$\underbrace{\begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_p(x_1) \\ f_2(x_1) & \cdots & & f_p(x_2) \\ \vdots & & & \vdots \\ f_p(x_1) & & & f_p(x_n) \end{bmatrix}}_{\substack{\underline{F} \\ n \times p}} \underbrace{\begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix}}_{\substack{\underline{a} \\ p \times 1}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}}_{\substack{\underline{y} \\ n \times 1}}$$

# of data points

# of functions

In general $n > p \Rightarrow$ minimize $\|\underline{r}\|_2$

The Normal Equations minimize $\|\underline{r}\|_2$

Given $\underline{F}\,\underline{a} = \underline{y}$, the solution to

$$\underline{F}^T \underline{F}\,\hat{\underline{a}} = \underline{F}^T \underline{y}$$ minimizes

$$\|\underline{F}\,\hat{\underline{a}} - \underline{y}\|_2$$

Also called the least squares solution

$$\hat{\underline{a}} = (\underline{F}^T\underline{F})^{-1}\underline{F}^T\underline{y}$$ is unique

Never actually form the Normal Equations, why?

Condition Number

$$\kappa(\underline{A}^T) = \kappa(\underline{A})$$
$$\kappa(\underline{A}\,\underline{B}) = \kappa(\underline{A})\,\kappa(\underline{B})$$
$$\kappa(\underline{A}^T\underline{A}) = \kappa(\underline{A}^T)\,\kappa(\underline{A}) = (\kappa(\underline{A}))^2$$

but, other methods can find the least squares solution (QR, SVD, etc.)

In Matlab: backslash $a = F \backslash y$

Nonlinear: No simple linear system

Define a objective function $S(\underline{a}) = \|\underline{r}\|_2^2$

$S(\underline{a}) = \sum_{i=1}^{n} r_i^2$ Use any minimization method.

look at Gauss - Newton

(The information below has been expanded on past that presented in lecture)

$\underline{a}_{k+1} = \underline{a}_k - \underline{H}_k \underline{g}_k$

$\underline{g}_k = \nabla S(\underline{a}_k) \rightarrow g_j = \frac{\partial s}{\partial a_j}$

$\underline{H}_k =$ Hessian of s @ $\underline{a}_k$

$$\Rightarrow H_{jk} = \frac{\partial^2 s}{\partial a_j \partial a_k} = \frac{d}{da_k}\left(\frac{ds}{da_j}\right) = \frac{dg_j}{\partial a_k}$$

$$s(\underline{a}) = \sum_{i=1}^{n} (y_i - \hat{y}(x_i, \underline{a}_k))^2 = \sum_{i=1}^{n} r_i^2$$

$$g_j = \frac{d}{da_j}\left(\sum_{i=1}^{n} r_i^2\right) = 2\sum_{i=1}^{n} r_i \frac{\partial r_i}{\partial a_j} = 2\sum_{i=1}^{n} r_i J_{ij}$$

↑ Jacobian of $r(\underline{a})$

$$\Rightarrow \underline{g} = 2\underline{r}^T \underline{J}$$

$$H_{jk} = 2\sum_{i=1}^{n}\left(\underbrace{\frac{\partial r_i}{\partial a_j}\frac{\partial r_i}{\partial a_k}}_{1^{st}\text{-order}} + \underbrace{r_i \frac{\partial^2 r_i}{\partial a_j \partial a_k}}_{2^{nd}\text{-order}}\right)$$

neglect 2^{nd}-order part

Thus $H_{jk} \approx 2 \sum_{i=1}^{n} \frac{\partial r_i}{\partial a_j} \frac{\partial r_i}{\partial a_k} = 2 \sum_{i=1}^{n} J_{ij} J_{ik}$

$\Rightarrow \underline{H} = 2 \underline{J}^T \underline{J}$

$\Rightarrow \underline{a}_{k+1} = \underline{a}_k - \underline{H}_k^{-1} g_k \approx \underline{a}_k - \frac{1}{\cancel{2}} (\underline{J}_k^T \underline{J}_k)^{-1} (\cancel{2} \underline{J}_k^T \underline{r}_k)$

$\underline{a}_{k+1} = \underline{a}_k - (\underline{J}_k^T \underline{J}_k)(\underline{J}_k^T \underline{r}_k)$

Define $\underline{\delta}_k = \underline{a}_{k+1} - \underline{a}_k$

$\Rightarrow \underline{\delta}_k = -(\underline{J}_k^T \underline{J}_k)^{-1} (\underline{J}_k^T \underline{r}_k)$

$\underline{J}_k^T \underline{J}_k \underline{\delta}_k = -\underline{J}_k^T \underline{r}_k$ $\leftarrow$ normal equations for $\underline{J} \underline{\delta}_k = -\underline{r}_k$!

Thus, solve $\underline{J}_k \underline{\delta}_k = -\underline{r}_k$ in least-squares sense, then $\underline{a}_{k+1} = \underline{a}_k + \alpha_k \underline{\delta}_k$

Advantage: No Hessian!

Disadvantage: Only converges if

$$\left| r_i \frac{\partial^2 r_i}{\partial a_j \partial a_k} \right| \ll \left| \frac{\partial r_i}{\partial a_j} \frac{\partial r_i}{\partial a_k} \right|$$

Part that is ignored — Part that is kept

Typically valid if $||\underline{r}||$ is initially small or if $\hat{y}(x, \underline{a})$ is mildly non-linear (no large oscillations).