Optimization For ML

Machine learning 2021

Mansoor Rezghi

Regression $\Rightarrow$ $\min \|\Phi w - y\|_2^2 + \lambda \|w\|_1^2$

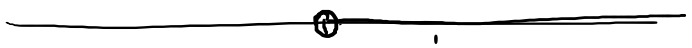
$\min f(x)$ Introduction to Matrix Computation

$x \in \mathbb{R}^n$ $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ V vector space $\mathbb{R}^1$

$\alpha \in \mathbb{R}$ $\alpha v \in V$

$x, y \in V$ $\forall \alpha, \beta \in \mathbb{R}$

$\alpha x + \beta y \in V$



X $V = \{$ min [illegible] $\}$

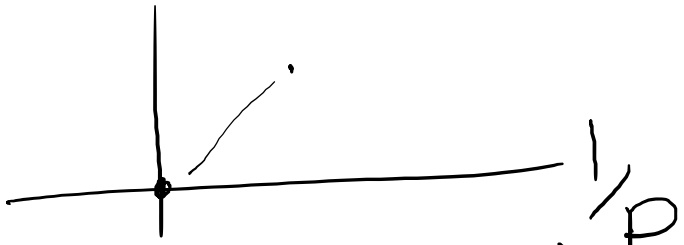
$\|x\|$

$\|x\| \geq 0$ $\quad \|x\| = 0 \iff x = 0$

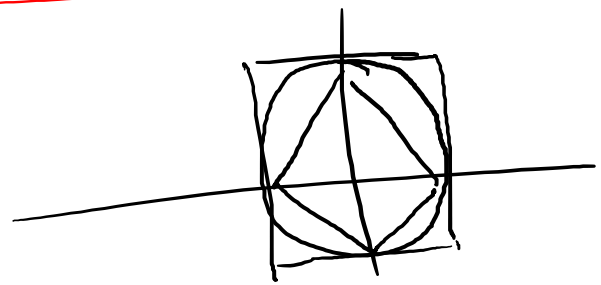
$\|\alpha x\| = |\alpha| \|x\|$

$\mathbb{R} \to$

$\|x+y\| \leq \|x\| + \|y\|$



$\|x\|_p = 1$



$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \to l_p$$

$p = 2 \Rightarrow \|x\|_2 = \sqrt{\sum x_i^2}$ اقلیدسی

$p = 1 \Rightarrow \|x\|_1 = \sum |x_i|$

$p = \infty \Rightarrow \|x\|_\infty = \lim_{p \to \infty} \sqrt[p]{\left(\sum |x_i|^p \right)} = \max_{1 \leq i \leq n} |x_i|$

$\|x-y\|$

$f(x) = \|x\|$

$\|\alpha x + (1-\alpha) y\| \leq \|\alpha x\| + \|(1-\alpha) y\|$

$\leq |\alpha| \|x\| + |1-\alpha| \|y\|$

$\alpha \|x\| + (1-\alpha) \|y\|$

$0 < \alpha < 1$

$f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y)$

$A \in \mathbb{R}^{n \times m}$ = n × m

Data

operator

$A : \mathbb{R}^m \to \mathbb{R}^m$ $\quad y = Ax = f(x)$

$A f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$

$\|A\|_F^2 = \sum_{i,j}^{n,m} a_{ij}^2 = \sum_{j=1}^{m} \|a_j\|_2^2 = \sum_{j=1}^{m} \|s_i\|_2^2$

$A = [a_1 \mid a_2 \mid a_3 \mid \cdots \mid a_m]$

$= \begin{bmatrix} s_1^T \\ \vdots \\ s_n^T \end{bmatrix}$

$\|A\|_1 = \sum_{i,j}^{n,m} |a_{ij}|$ → ML

$\|A\|$ A is a lin op. $A : \mathbb{R}^m \longrightarrow \mathbb{R}^n$

$$\|A\|_q = \max_{x \neq 0} \frac{\|Ax\|_q}{\|x\|_q}$$ تعریف

$$= \min_{x \neq 0} \left\| A\left(\frac{x}{\|x\|_q}\right) \right\|_q = \min_{\|x\|_q = 1} \|Ax\|_q$$

$\{x_1, \dots, x_n\} \Rightarrow$ lin independency $\forall i$

Prove independent vectors $\sum \alpha_i x_i = 0 \Longrightarrow \alpha_i = 0$

$\sum \alpha_i x_i = 0 \quad \alpha_j \neq 0 \Rightarrow \alpha x_j = -\sum_{i \neq j} \alpha_i x_i \Rightarrow x_j = \sum \left(-\frac{\alpha_i}{\alpha_j}\right) x_i$

V : $Rank(V) = \{$ عدد متجهات مستقل خطي $\}$

$\mathbb{R}^2$ $\quad \begin{pmatrix} x \\ y \end{pmatrix} = x\begin{pmatrix} 1 \\ 0 \end{pmatrix} + y\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$\alpha\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta\begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0$

$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = 0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $\quad \alpha = 0$ $\quad \beta = 0$

$\dim(\mathbb{R}^2) = 2$ $\quad \begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$\begin{pmatrix} a \\ b \end{pmatrix} = a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$\{x_1 \dots x_n\} \xrightarrow{\text{Base of } V}$ $\quad \dim(V) = n$

$x_1 \dots x_n$ independent

$x \in V$ $\quad x = \sum \alpha_i x_i$

$V = span\{x_1, \dots, x_n\}$

$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $\mathbb{R}^2$

$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$\begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$A = [a_1 \cdots a_n] \in \mathbb{R}^{m \times n}$ $\quad a_i \in \mathbb{R}^m$

$$= \begin{bmatrix} s_1 \\ s_1 \\ \vdots \\ s_m \end{bmatrix}$$

رتبه سطری ماتریس A: تعداد بیشترین سطرهای مستقل خطی A

رتبه ستونی: تعداد بیشترین ستونهای مستقل خطی A

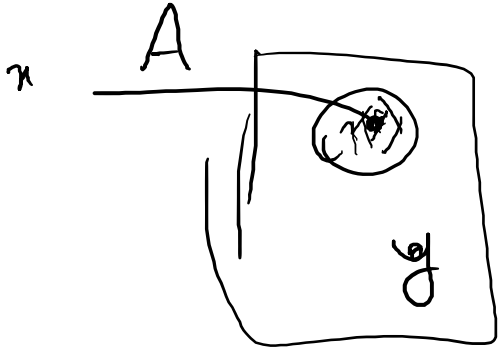
رتبه سطری = رتبه ستونی

- $Rank(A) = \{ \#$ of indepnt column or Row $\}$

$$Rank(A) \leq \min\{m, n\}$$

$A \subset R^{m \times n}$ $\quad A : R^n \longrightarrow R^m$

$$R(A) = \{ y \in R^m \mid \exists\, \alpha \text{ s.t } A\alpha = y \}$$



$n < m$

$R(A) \subset R^m$ $\qquad W \subset V$

$y = A\alpha = \sum \alpha_i a_i$

$R(A) = \text{span}\{a_1 \dots a_n\}$

$R^2 \longrightarrow R^3$

$\dim(R(A)) = \underline{\text{Rank}}(A) \leq \min\{m, n\}$

$A: R^{m\times n} : R^n \to R^m$

$N(A) = \{ x \in R^n \mid Ax = 0 \} \Rightarrow$ $Ax = 0 \to \boxed{\sum x_i a_i = 0}$

columns of A indepent $\Rightarrow$ $N(A) = \{0\}$

$R^m = R(A) \oplus N(A^T)$

$R^n = R(A^T) \oplus N(A)$

$R(A) \perp N(A^T)$

$y \in R(A)$ $\quad z \in N(A^T)$

$y^T z = 0$

$y^T z = (Ax)^T z = x^T A^T z = 0$

Defini: $V = V_1 \oplus V_2 \Rightarrow V_1 \cap V_2 = \{0\} \Rightarrow x \in V \to \exists!\ x_1 \in V_1 \quad x = x_1 + x_2$
$x_2 \in V_2$

$V = V_1 + V_2$ $\quad \forall x \in V \quad \exists\ x_1 \in V_1 \quad x = x_1 + x_2$
$x_1 \in V_1$

$Ax = y$ $\quad y \in R(A)$ $\quad A \in R^{n \times n}$ $\quad Rank(A) = n$:

$\arg\min \|Ax - b\|_2^2$

$\arg\min \|Ax - b_R\|_2^2$

$Ax_1 = y$, $Ax_2 = y$ $\Rightarrow$ $A(x_1 - x_2) = 0$ $\rightarrow$ $x_1 = x_2$

nonsingular

inverteh.

$Ax = b_R$

$A^T(Ax)$

$det(A) \neq 0$

$b_R \in R(A)$

$b_N \in N(A^T)$

$b \in R(A)$ $\quad b = b_R + b_N$

$\min_x \|Ax - b\|_2^2$

$A \in R^{m \times n}$ $\quad b \in R^m$

$\min_x \|Ax - b_R - b_N\|_2^2 = \|z - b_N\|_2^2 = (z^T - b_N^T)(z - b_N)$

$= z^T z - 2 z^T b_N + b_N^T b_N$

$\|Ax - b_R\|_2^2$ $\quad$ $z = Ax$

$\|x\|_2^2 = \sum x_i^2 = x^T x$

$\min \|Ax-b\|_2^2 \iff Ax = b_R \iff Ax - b_R = 0$

$A^T(Ax - b_R) = 0$

$R(A) \perp N(A^T)$

$A^T b_N$

$A^TAx = A^Tb_R + 0$

$= A^T(b_R + b_N)$

$= A^Tb$

$A^TAx = A^Tb \Rightarrow x = (A^TA)^{-1}A^Tb$

$\min \|Ax - b\|_1^2 \iff x = (A^TA)^{-1}A^Tb$

$(A^TA)^{\dagger}A^Tb$

Eigenvalue: $\forall$ $Ax = \lambda x$ $A : \mathbb{R}^{n \times n}$

For a λ if $\exists\, x \neq 0$ s.t $Ax = \lambda x$

eigenvector ← x ; λ → eigen value.

$$\underbrace{(A - \lambda I)}\,\underbrace{x} = 0 \Rightarrow x \in N(A - \lambda I) \Rightarrow \underbrace{\det(A - \lambda I)} = 0$$

حد n تا ریشه مختلف دارد.

λ

$\lambda^{2}(\lambda - 11)^{3}(\lambda - 2)$

$B(x) = 0$

x

$Trac(A) = \sum a_{ii}$

$$\min Trac(X^T A X) = \sum_{i=n-k+1}^{n} \lambda_i$$

s.t $X^T X = I$

A SPD

$A \in \mathbb{R}^{n \times n}$ $X \in \mathbb{R}^{n \times k}$

$A: \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$

k تا از بردارهای ویژه متناظر k تا کوچکترین میکند.

$$\max \operatorname{Tr}(X^T A X) = \sum_{i=1}^{k} \lambda_i \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

s.t.

$$X^T X = I \qquad X = [x_1 \ldots x_k]$$

$\lambda_i \leftarrow$ Corresponding eigenvector $\leftarrow x_i \quad i = 1 \ldots k$

$a \in \mathbb{R}$ a.s. $ax = 0 \Rightarrow x^2 > 0$

$\forall x \neq 0$

$A \in \mathbb{R}^{n \times n}$ semi positive definite if $\forall x \neq 0 \quad x^T A x \geq 0$

Psd $\rightsquigarrow$ $x^T A x > 0$

A spd $\Longleftrightarrow$ $\lambda_i > 0$

A spd $\rightarrow$ $Ax = \lambda x$

$x^T A x = \lambda$

!

$x^T A x \geq 0$

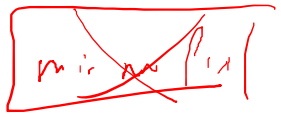
Text:

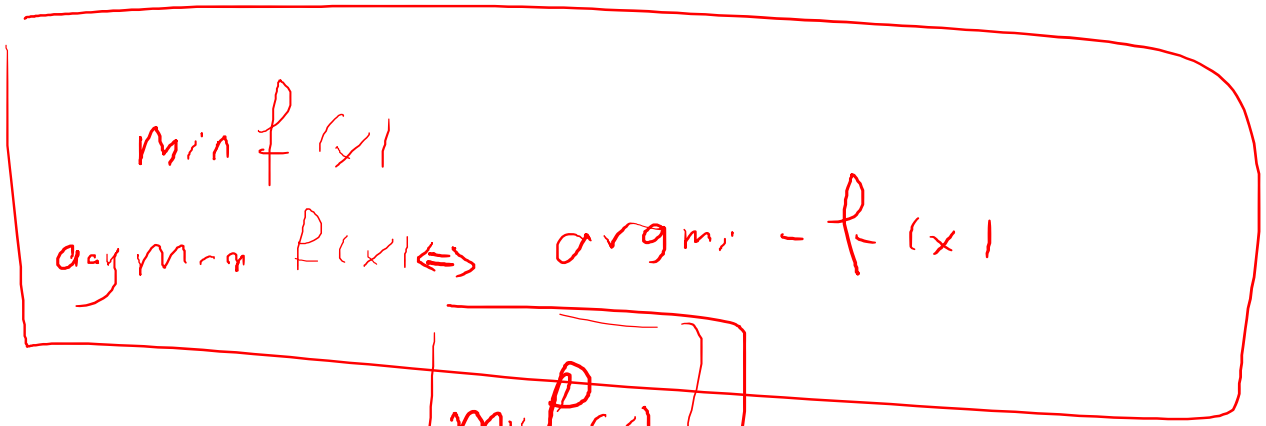
Numerical Optimization Nocedal, springer 2006, Chap 2,3

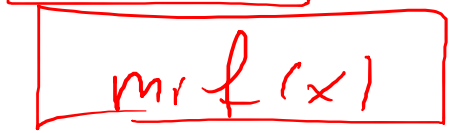
Convex optimization, Boyd, Cambridge University Press

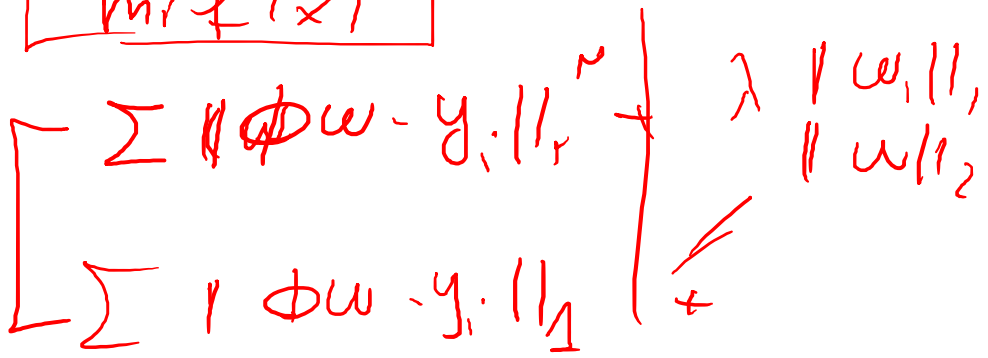
Deep Learning, chap4, chap 8

KM(section 13.3.2)





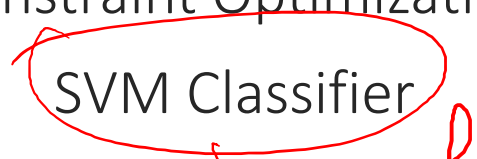


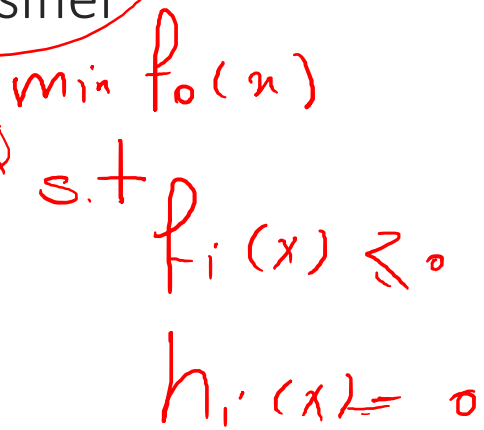


- Unconstraint Optimization

  logistic Regression, Perceptron, …

- Constraint Optimization





Objective function and constraints

- Linear

  Linear Programming (LP)

- Non- linear

  Special case: Quadratic

$$\min \|\Phi w - y\|_2^2$$

$$= (\Phi w - y)^T (\Phi w - y)$$

$$= w^T \underbrace{\Phi^T \Phi}_{A} w + 2 \underbrace{y^T \Phi}_{b^T} w + \underbrace{y^T y}_{c}$$

$$= w^T A w + b^T w + c$$

- Continues domain
- Discrete domain

  Integer Programming

- Differentiable
- Non-differentiable

- Convex
- Non-Convex

$$f(w) = w^T A w + b^T w + c$$

$$= \begin{bmatrix} w_1 & \cdots & w_d \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix} = \sum_{i=1}^{d} w_i^2 \lambda_i$$

Un-constraint Constraint and optimization

- Unconstraint Optimization

$$\text{minimize} \quad f_0(x)$$

- Constraint optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

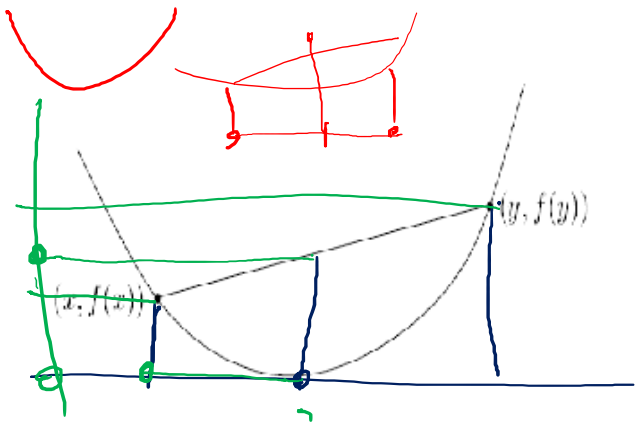
Convex Optimization

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

- f_i, i=0,…,m convex function
- h_i, i=1,…,p Affine function

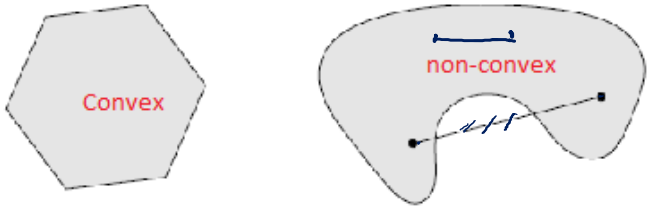
A function $f : \mathbf{R}^n \to \mathbf{R}$ is *convex* if $\mathbf{dom}\, f$ is a convex set and if for all $x, y \in \mathbf{dom}\, f$, and θ with $0 \leq \theta \leq 1$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$



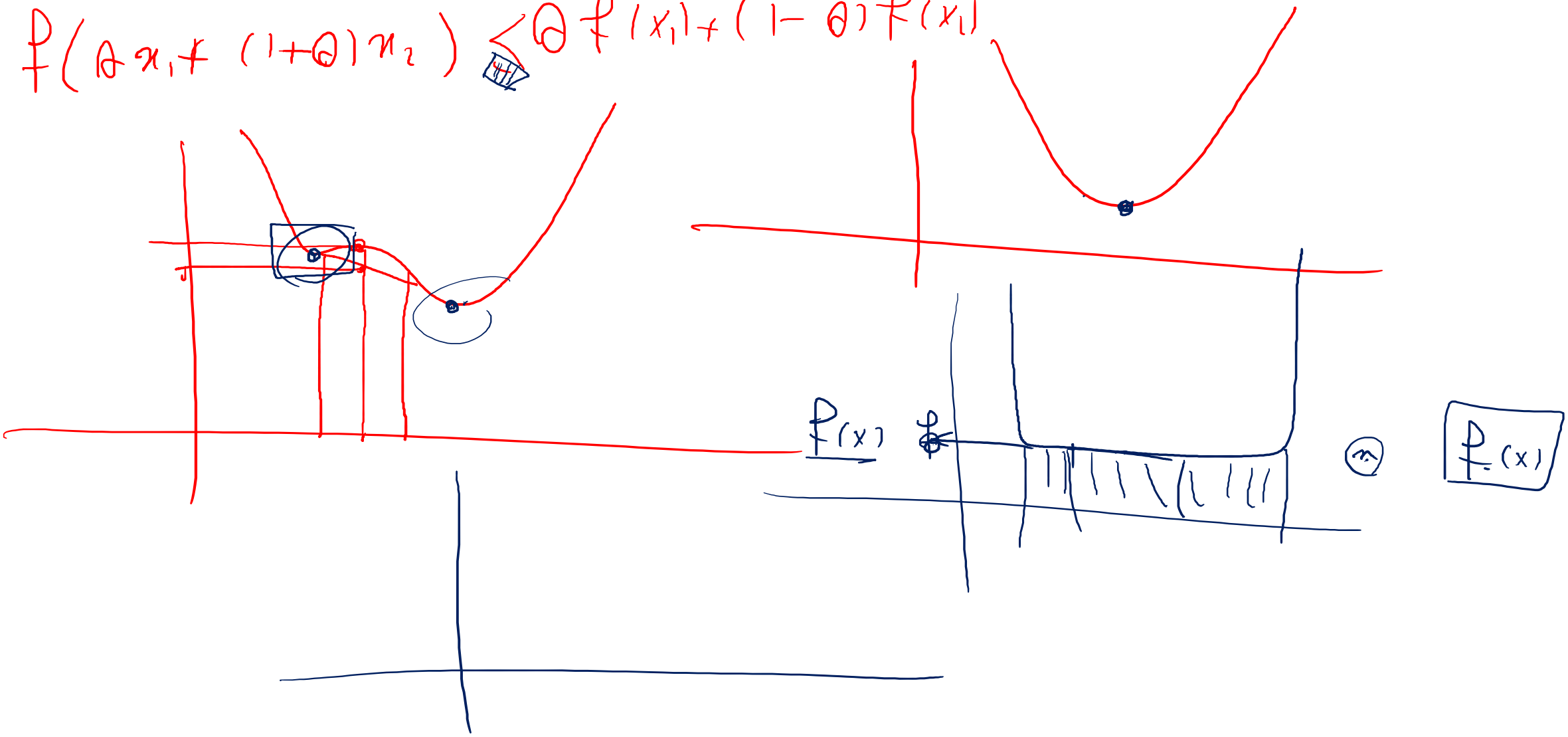
A set C is *convex* if the line segment between any two points in C lies in C, *i.e.*, if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$

$$\theta x_1 + (1-\theta) x_2 \in C.$$



f: dom f conv. set convex.

$$f(\theta x_1 + (1-\theta) x_2) \leq \theta f(x_1) + (1-\theta) f(x_2)$$



$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y)$

$f(x) = \|x\| \rightarrow$ norm. Con

Ex: $\|x\|_2$

$f(x)$

$\ell(x) = f_{x_0}(x) + (x - x_0) f'(x_0)$

Convex Optimization chap3, S. Boyd

$f: \mathbb{R} \rightarrow \mathbb{R}$

$f'(x)$

First-order conditions

Suppose f is differentiable (*i.e.*, its gradient ∇f exists at each point in **dom** f, which is open). Then f is convex if and only if **dom** f is convex and

$$f(y) \ge f(x) + \nabla f(x)^T (y - x)$$

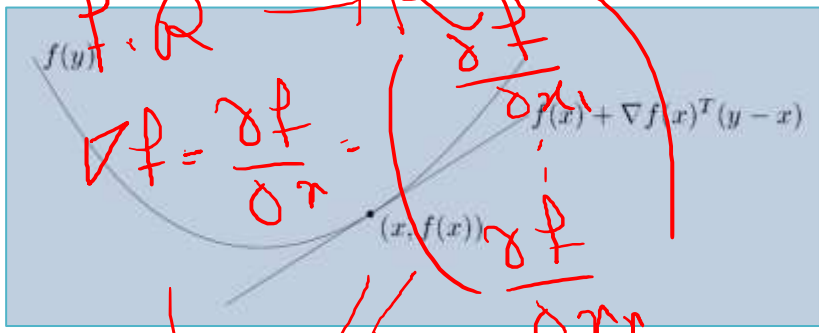
holds for all $x,\ y \in$ **dom** f

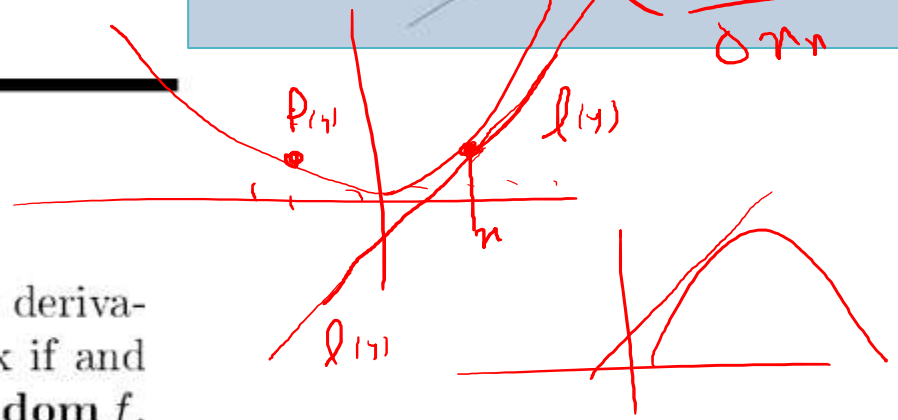
$\ell(y)$

$\ell_x(y) = f(x) + (y-x)^T \nabla f(x)$

$f: \mathbb{R}^n \rightarrow \mathbb{R}$

$\nabla f = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$





Second-order conditions

We now assume that f is twice differentiable, that is, its *Hessian* or second derivative $\nabla^2 f$ exists at each point in **dom** f, which is open. Then f is convex if and only if **dom** f is convex and its Hessian is positive semidefinite: for all $x \in$ **dom** f,

$$\nabla^2 f(x) \succeq 0.$$

$x \in \mathbb{R}$ $\quad f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

$f : \mathbb{R}^n \longrightarrow \mathbb{R}$ $\quad x \in \mathbb{R}^n$ $\quad \nabla f = \frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$

$$\nabla^2 f = \begin{pmatrix} \frac{\partial f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \vdots & & \\ \frac{\partial f}{\partial x_n \partial x_1} & \cdots & \frac{\partial f}{\partial x_n \partial x_n} \end{pmatrix}$$

$f(x) = \|x\|_2^2 = \sum_{i=1}^{n} x_i^2$ $\quad \frac{\partial f}{\partial x} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} = \begin{pmatrix} 2x_1 \\ 2x_2 \\ \vdots \\ 2x_n \end{pmatrix} = 2 \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

$$\nabla^2 f = \frac{\partial f}{\partial x^2} = 2 \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix} = 2I$$

$\forall z \neq 0 \quad z^T \nabla^2 f \, z = 2 z^T z = 2\|z\|_2^2 > 0$

Linear regression

$$\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{y}\|_2^2$$

$$\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{y}\|_1$$

Nonnegative weighted sums

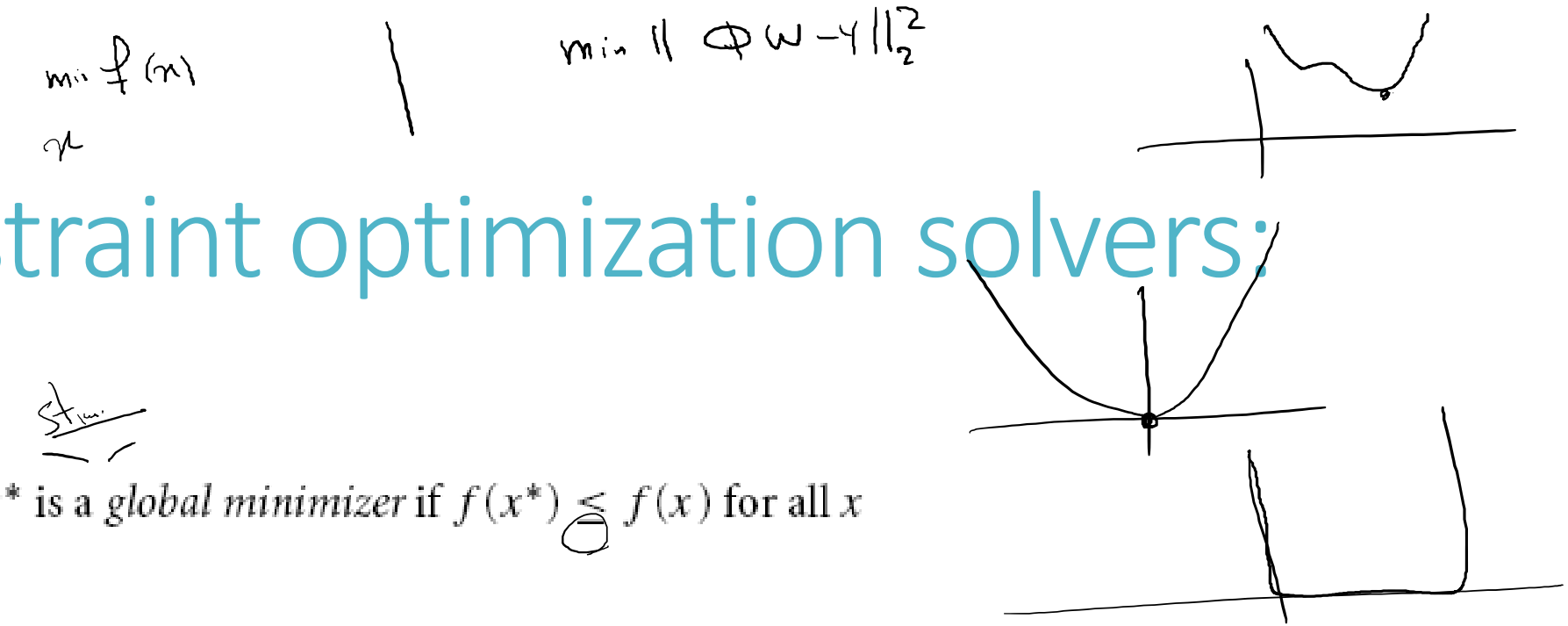
A nonnegative weighted sum of convex functions, $f = w_1 f_1 + \cdots + w_m f_m$ is convex. Boyd chap3

$$\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

$$\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

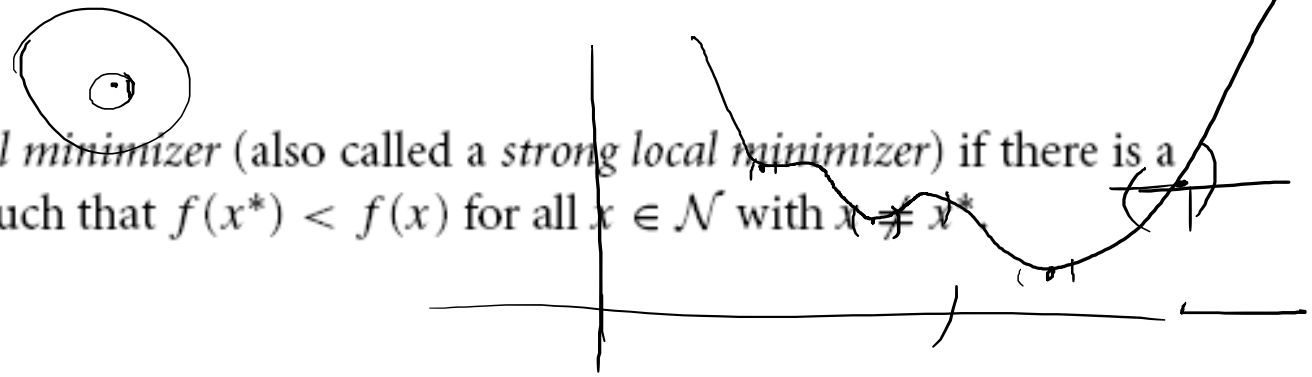
Un-Constraint Optimization

Unconstraint optimization solvers:



A point x^* is a *global minimizer* if $f(x^*) \leq f(x)$ for all x

A point x^* is a *local minimizer* if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{N}$.



A point x^* is a *strict local minimizer* (also called a *strong local minimizer*) if there is a neighborhood $\mathcal{N}$ of x^* such that $f(x^*) < f(x)$ for all $x \in \mathcal{N}$ with $x \neq x^*$

$y^T \Phi w = z^T w \qquad z = \Phi^T y$

Recognizing a Local Minimum

Theorem (Second-Order Necessary Conditions).

If x^* is a local minimizer of f then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

Theorem (Second-Order Sufficient Conditions).

Suppose that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite.

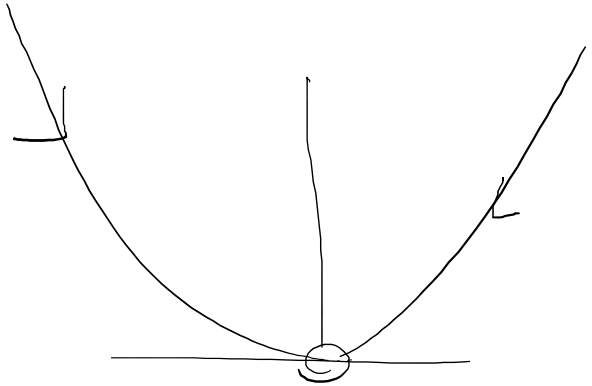
Then x^* is a strict local minimizer of f.

$f(w) = \|\Phi w - y\|_2^2 = w^T \Phi^T \Phi w - 2 y^T \Phi w + y^T y$

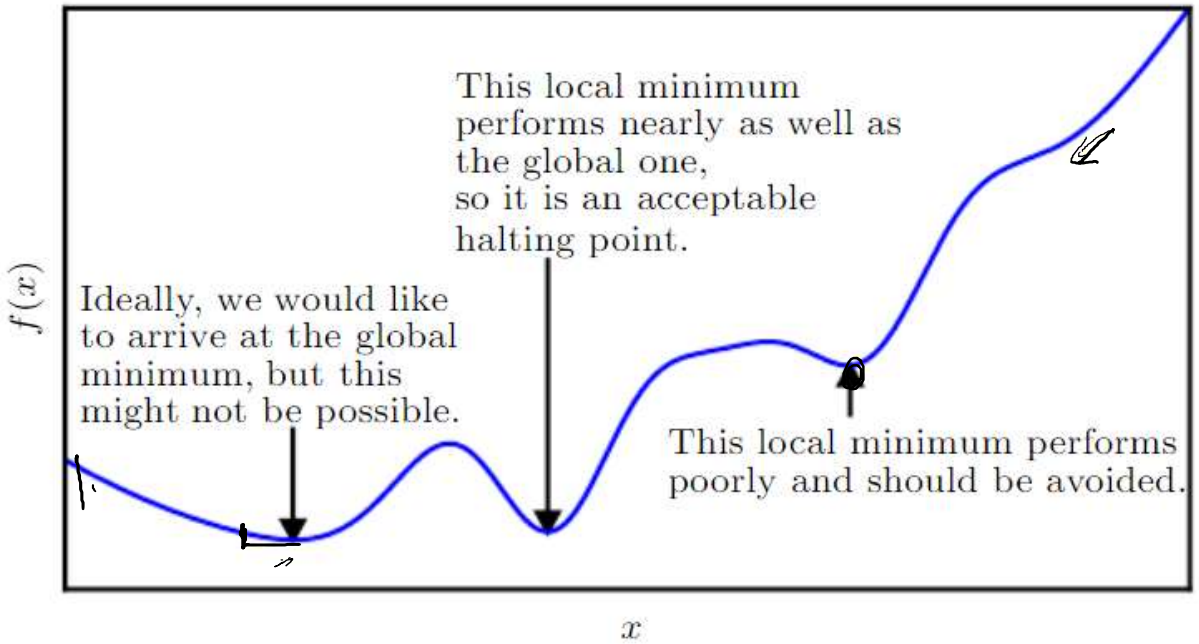
When f is convex, any local minimizer x^* is a global minimizer of f.

$\nabla f = 2\Phi^T \Phi w - 2\Phi^T y = 0 \Rightarrow \Phi^T \Phi w = \Phi^T y \Rightarrow w = (\Phi^T \Phi)^{-1} \Phi^T y$

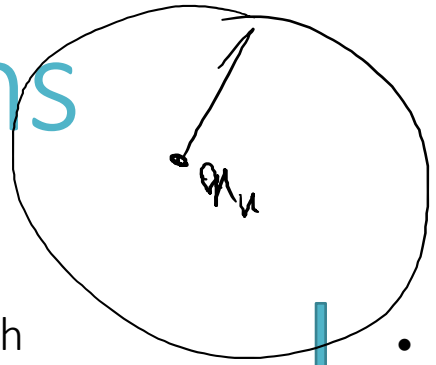
$\nabla^2 f = 2\Phi^T \Phi \qquad \Phi = [\phi_1 \; \cdots \; \phi_n] \qquad z^T \nabla^2 f z = z^T \Phi^T \Phi z$



Nonconvex

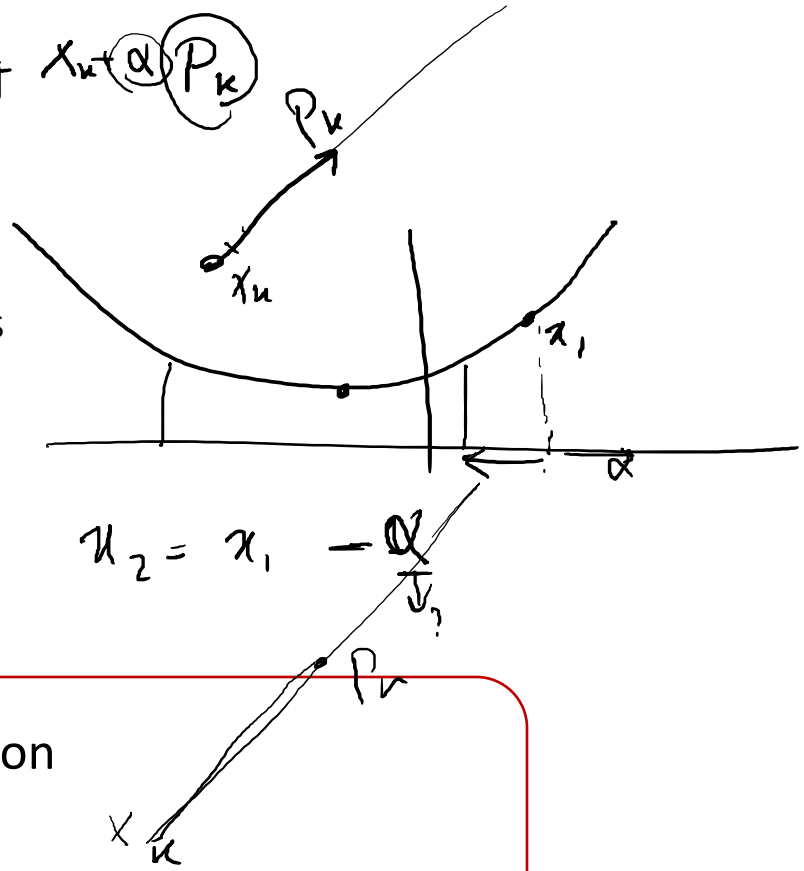

This local minimum
performs nearly as well as
the global one,
so it is an acceptable
halting point.
Ideally, we would like
to arrive at the global
minimum, but this
might not be possible.
This local minimum performs
poorly and should be avoided.
$f(x)$
x

Algorithms



- Line Search
- Trust Region
- Quasi Newton
- Conjugate gradient
- …

- First order methods
- Second order-methods



1. Line search

✓choose a direction p_k *and search along* this direction

$$\min_{\alpha>0} f(x_k + \alpha p_k).$$

Line S- → descent method → Steepest descent, Gradient descent

SEARCH DIRECTIONS FOR LINE SEARCH METHODS

Theorem (Taylor's Theorem).

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $p \in \mathbb{R}^n$. Then we have that

$$f(x+p) = f(x) + \nabla f(x)^T p + \tfrac{1}{2} p^T \nabla^2 f(x+tp) p,$$

(1) $f: \mathbb{R}^n \to \mathbb{R}$

$f(x_k + \alpha p) \simeq f(x_k) + \alpha p^T \nabla f_k + \frac{1}{2}\alpha^2 p^T \nabla^2 f(x_k + tp) p,$

$f(x_{k+1}) \simeq f(x_k) + \alpha p^T \nabla f_k + \frac{L}{2}\alpha^2 p^T \nabla^2 f(x_k + tp) p$

$f_{k+1} = f_k + \alpha p^T \nabla f_k$ if $p^T \nabla f_k < 0$

$f_{k+1} < f_k$

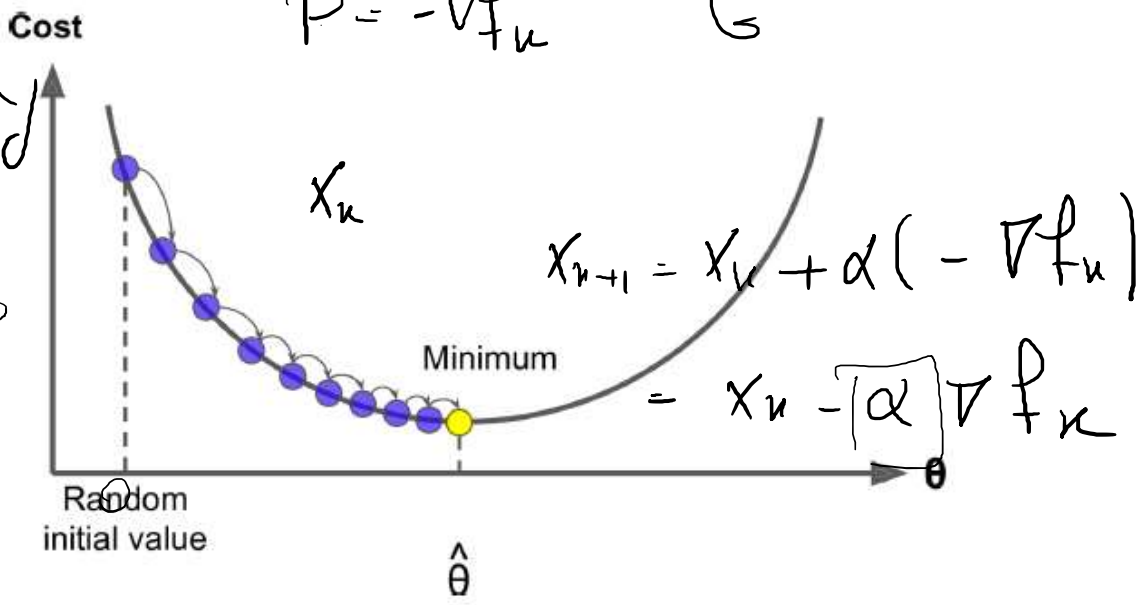
Descent Methods

Descent direction $\mathbf{p}$: $\nabla f_k^T \mathbf{p} < 0$

Descent direction is guaranteed to produce a decrease in f , provided that the step length is sufficiently small

Steepest descent(Gradient descent)

$$\mathbf{p} = -\nabla f_k \qquad \mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f_k$$



Newton direction

Second-order Taylor series approximation

$$f(x_k + p) \approx f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p \stackrel{\text{def}}{=} m_k(p).$$

$$p_k^N = -\left(\nabla^2 f_k\right)^{-1} \nabla f_k.$$

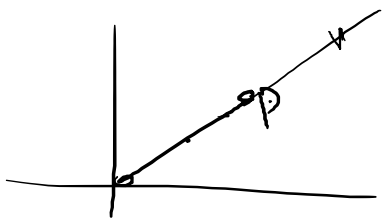
The Newton direction is reliable when the difference between the true function f ($x_k + p$) and its quadratic model $m_k(p)$ is not too large.

$$\min_p \; f(x_k+p) \simeq f(x_k) + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p$$

$$\min_p \; f_k + p^T \nabla f_k + \frac{1}{2} p^T \nabla^2 f_k p$$

$$m_k(p)$$

$$\frac{\partial m_k(p)}{\partial p} = 0 \Rightarrow \nabla f_k + \nabla^2 f_k p = 0 \Rightarrow p = -\left(\nabla^2 f_k\right)^{-1} \nabla f_k$$



$$f(x_k + \alpha p_k)$$

$x_{k+1} = x_k + \alpha p$ if $p = -\nabla f_k \Rightarrow x_{k+1} = x_k - \alpha \nabla f_k$ GD SD

$\min f(x)$ ML

loss

$\hat{y}_i = f(x_i; \theta)$

$\frac{1}{N} \sum loss(\hat{y}_i, y_i)$

$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} loss(x_i, y_i, \theta)$ $g(\theta)$

Train بستگی بر loss

Early — Stopping → Regularization

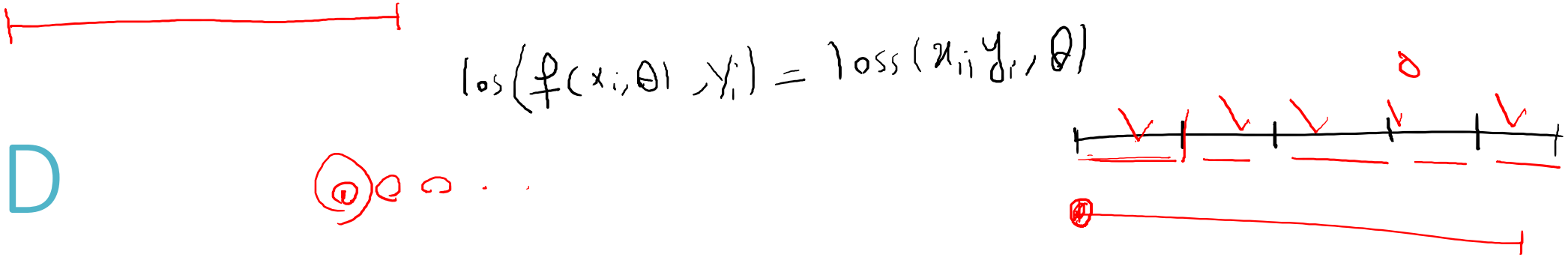
$\theta_{k+1} = \theta_k + \frac{\gamma g(\theta)}{\gamma \theta_k}$

$\frac{\partial g_k}{\partial \theta} = \frac{1}{N} \sum_{i=1}^{N} \frac{\gamma loss(x_i, y_i, \theta)}{\gamma \theta}$

$= \frac{1}{N} \sum_{i=1}^{N'} \frac{\gamma loss(x_{i+1}, y_{i+1}, \theta)}{\gamma \theta}$

$N' < N$

GD



$$Loss(\theta) = \frac{1}{N}\sum_{i=1}^{N} Loss(x_i, y_i, \theta) \Rightarrow \frac{\partial Loss(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial Loss(x_i, y_i, \theta)}{\partial \theta}$$

- **Batch:** Uses the whole training set to compute the gradients at every step, which makes it very slow when the training set is large

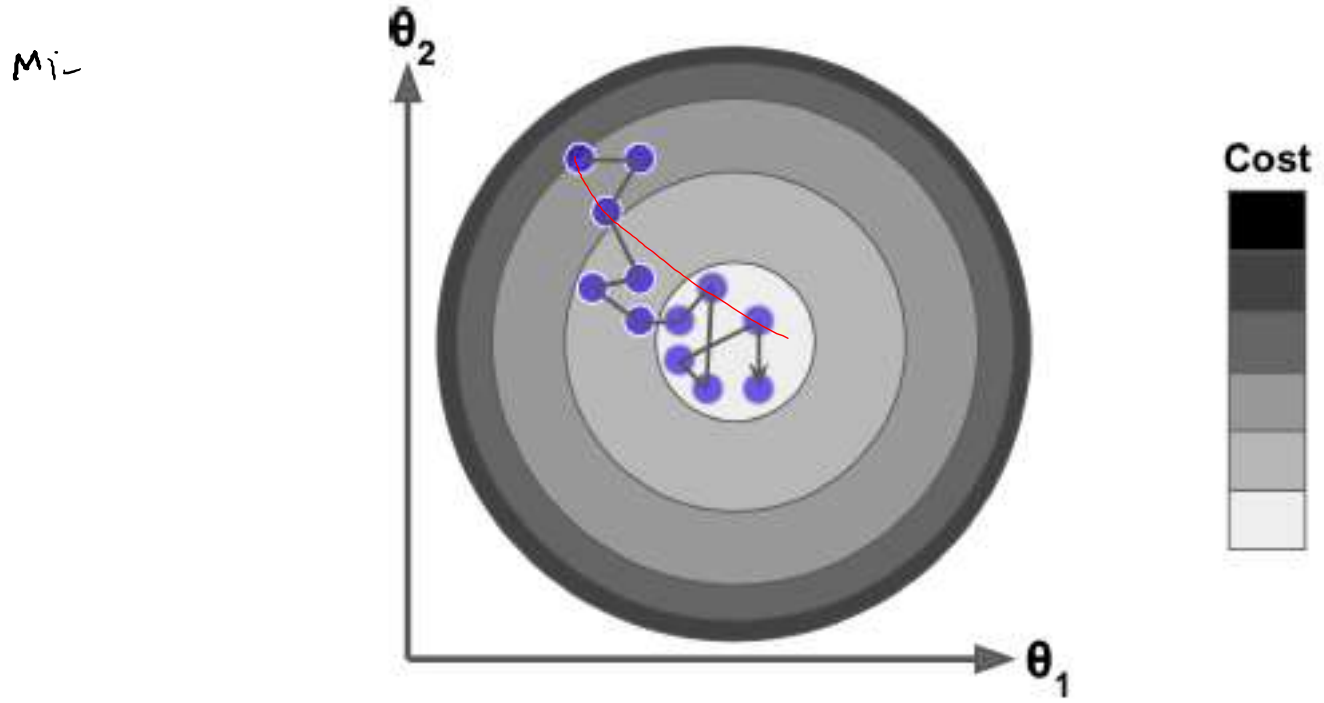
- **Stochastic GD(SGD):** Picks a random instance in the training set at every step and computes the gradients based only on that single instance.

  - SGD is much less regular than Batch Gradient Descent
  - Instead of gently decreasing until it reaches the minimum, the cost function will bounce up and down, decreasing only on average.

- Mini- Batch Gradient Descent

Mini-batch GD computes the gradients on small random sets of instances called mini-batches

Stochastic Gradient Descent



final parameter values are good, but not optimal

SGD

$$J(\boldsymbol{\theta}) = \mathbb{E}_{,y\sim\hat{p}} \quad L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}).$$

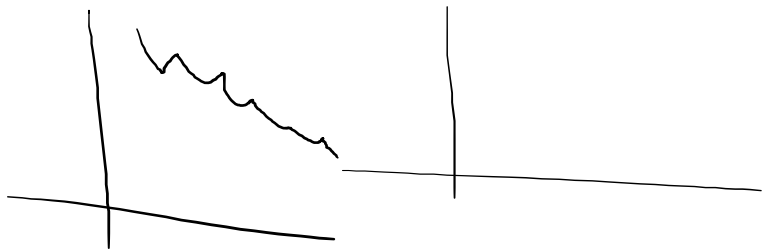
$$\boldsymbol{g} = \frac{1}{m'}\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

Estimation of gradient

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g},$$

$\min \|Xw - y\|_2^2 + \lambda \Omega(w)$

Early Stopping



Stochastic Gradient

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.

Require: Initial parameter $\boldsymbol{\theta}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$

end while

Condition for Convergence

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$$

$\epsilon_k = \frac{1}{k}$

In Practice (One example):

$$\epsilon_k = (1 - \alpha)\epsilon_0 + \alpha \epsilon_\tau \qquad \alpha = \frac{k}{\tau}$$

After iteration τ, it is common to leave ϵ constant

Unconstraint Examples in Learning

Least squares

$$E = \sum_{n=1}^{N} (\mathbf{w}^{\mathrm{T}}\phi_n - t_n)^2$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} \phi_n \phi_n^{\mathrm{T}} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} \qquad \nabla E(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^{\mathrm{T}}\phi_n - t_n)\phi_n = \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{w} - \mathbf{\Phi}^{\mathrm{T}}\mathbf{t}$$

$$\begin{aligned} \mathbf{w}^{(\mathrm{new})} &= \mathbf{w}^{(\mathrm{old})} - (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\left\{\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{w}^{(\mathrm{old})} - \mathbf{\Phi}^{\mathrm{T}}\mathbf{t}\right\} \\ &= (\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \end{aligned}$$

$$f(w) = w^T \Phi^T \Phi w - 2y^T \Phi w + y^T y + \lambda w^T w$$

$$\frac{\partial f}{\partial w} = 2\Phi^T \Phi w - 2\Phi^T y + 2\lambda w = 0$$

$$(\Phi^T \Phi + \lambda I) w = \Phi^T y$$

$$w = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$$

$$\frac{\partial^2 f}{\partial w^2} = 2\Phi^T \Phi + 2\lambda I = 2(\Phi^T \Phi + \lambda I)$$

$$\min_{\mathbf{w}} \|\Phi \mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

GD

$$w_{k+1} = w_k - \alpha \left(2\Phi^T \Phi w_k - 2\Phi^T y + 2\lambda w_k\right)$$

Newton

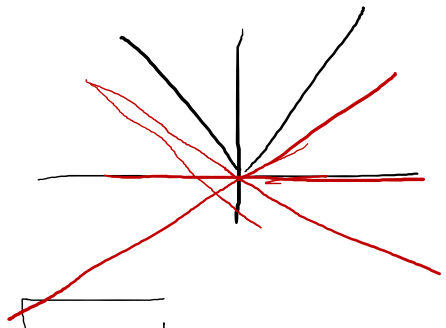
$$w_{k+1} = w_k + p$$

$$p = -\nabla^2 f^{-1} \nabla f \Rightarrow p = -(\Phi^T \Phi + \lambda I)^{-1} \left[2\Phi^T \Phi w_k - 2\Phi^T y + \lambda w_k\right]$$

$$= -(\Phi^T \Phi + \lambda I)^{-1} \left[(\Phi^T \Phi + \lambda I) w_k - \Phi^T y\right]$$

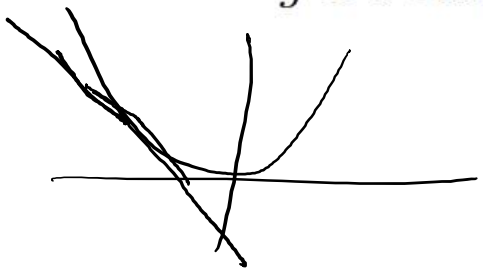
$$w_{k+1} = w_k - (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y$$

SGD for Lasso



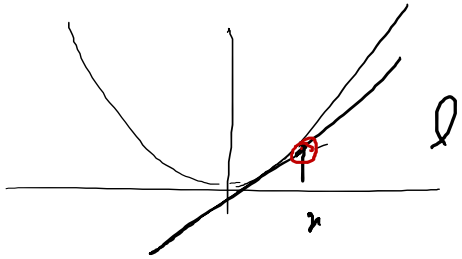
$$\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

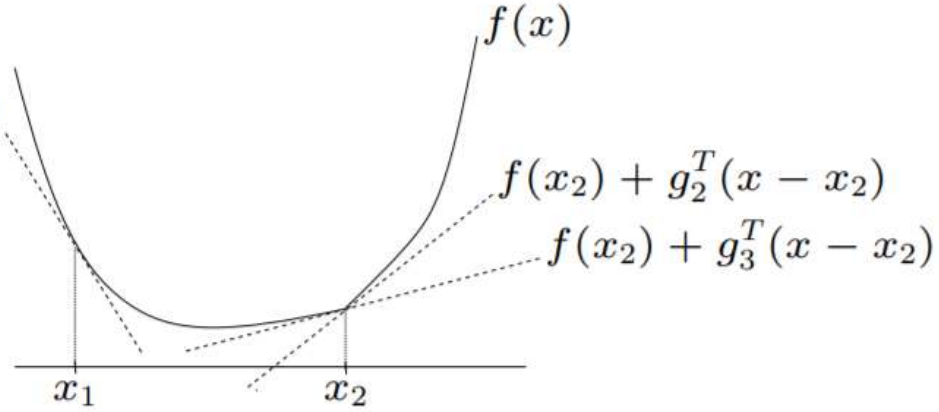
Boyd-
Murphy-13.3.2-13.3.5



g is a **subgradient** of f (not necessarily convex) at x if

$$f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y$$





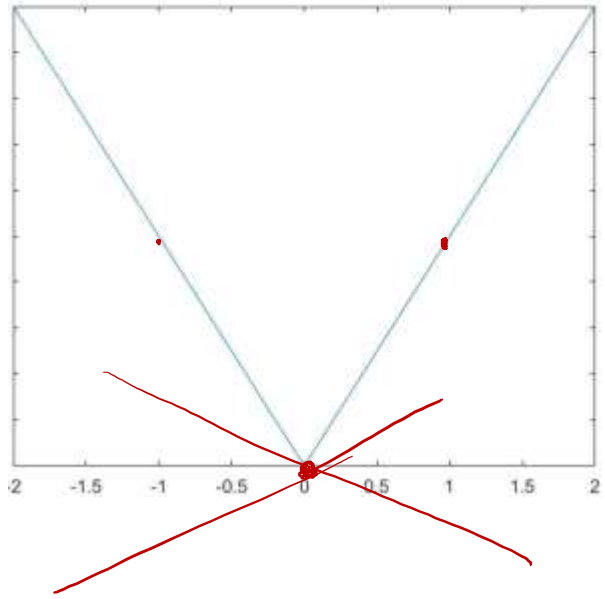
if f is convex and differentiable, $\nabla f(x)$ is a subgradient of f at x

Subgradient: Example

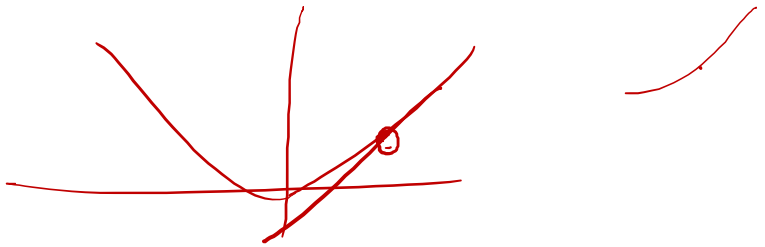
$$f(\theta) = |\theta| \quad \Longrightarrow \quad \partial f(\theta) = \begin{cases} \{-1\} & \text{if } \theta < 0 \\ [-1, 1] & \text{if } \theta = 0 \\ \{+1\} & \text{if } \theta > 0 \end{cases}$$

subgradient

1



$f(\theta) = |\theta|$



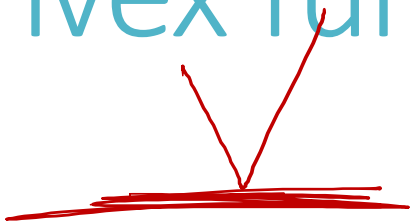
if f is convex,

- $\partial f(x)$ is nonempty, for $x \in \mathbf{relint}\,\mathbf{dom}\, f$
- $\partial f(x) = \{\nabla f(x)\}$, if f is differentiable at x
- if $\partial f(x) = \{g\}$, then f is differentiable at x and $g = \nabla f(x)$

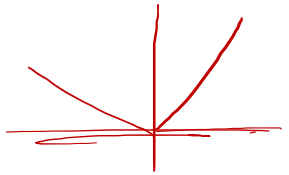
Optimality condition for convex function

recall for f convex, differentiable,

$$f(x^\star) = \inf_x f(x) \iff 0 = \nabla f(x^\star)$$



generalization to nondifferentiable convex f:



$$f(x^\star) = \inf_x f(x) \iff 0 \in \partial f(x^\star)$$

Boyd

Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is convex. To minimize f, the subgradient method uses the iteration

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}.$$

Here $x^{(k)}$ is the kth iterate, $g^{(k)}$ is *any* subgradient of f at $x^{(k)}$, and $\alpha_k > 0$ is the kth step size.

Coordinate descent for lasso

-RSS(w)

$$\sum_{i=1}^{N} (y_i - (w_0 + \mathbf{w}^T \mathbf{x}_i))^2 + \lambda ||\mathbf{w}||_1$$

$$\frac{\partial}{\partial w_j} \text{RSS}(\mathbf{w}) = a_j w_j - c_j$$

$$a_j = 2 \sum_{i=1}^{n} x_{ij}^2$$

$$c_j = 2 \sum_{i=1}^{n} x_{ij} (y_i - \mathbf{w}_{-j}^T \mathbf{x}_{i,-j})$$

$$\partial_{w_j} f(\mathbf{w}) = (a_j w_j - c_j) + \lambda \partial_{w_j} ||\mathbf{w}||_1 = \begin{cases} \{a_j w_j - c_j - \lambda\} & \text{if } w_j < 0 \\ [-c_j - \lambda, -c_j + \lambda] & \text{if } w_j = 0 \\ \{a_j w_j - c_j + \lambda\} & \text{if } w_j > 0 \end{cases}$$

$$\hat{w}_j(c_j) = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } c_j \in [-\lambda, \lambda] \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$\hat{w}_j = \text{soft}(\frac{c_j}{a_j}; \frac{\lambda}{a_j})$$

$$\text{soft}(a; \delta) \triangleq \text{sign}(a) (|a| - \delta)_+$$

$x_+ = \max(x, 0)$ is the positive part of x.

- LARS
- Nesterov
- Proximal Gradient method
- ADMM
- FISTA