Non-smooth Optimization

C^1 C^2 C^3 $\cdots$ C^∞

Optimization Techniques (ENGG*6140)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh

Winter 2023

$\underbrace{f \in C^2}$ $\downarrow$ gradient, Hessian

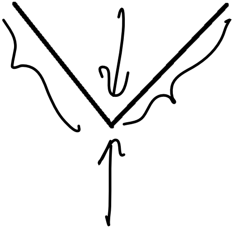
$f \notin C^\infty$

$\underbrace{f \notin C^1}$

Non-smooth Function

Non-smooth function

- When we have non-smooth function, the gradient is not defined at the non-smooth point(s).



- In these cases, we need non-smooth optimization.

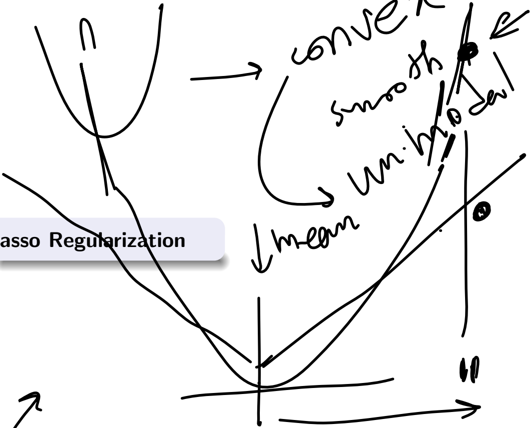
Lasso Regularization

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

$f = x$

x^2

x



Lasso Regularization

- The ℓ_1 norm can be used for sparsity [1]. Sparsity is very useful and effective because of betting on sparsity principal [2] and the Occam's razor [3].
- If $\boldsymbol{x} = [x_1, \dots, x_d]^\top$, for having sparsity, we should use **subset selection** for the regularization of a cost function $\Omega_0(\boldsymbol{x})$:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \Omega(\boldsymbol{x}) := \Omega_0(\boldsymbol{x}) + \lambda ||\boldsymbol{x}||_0, \qquad (1)$$

"ℓ_0"

where:

$$||\boldsymbol{x}||_0 := \sum_{j=1}^{d} \mathbb{I}(x_j \neq 0) = \begin{cases} 0 & \text{if } x_j = 0, \\ 1 & \text{if } x_j \neq 0, \end{cases} \qquad (2)$$

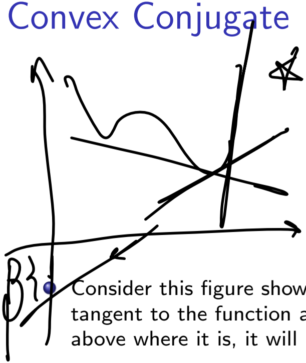
is the "ℓ_0" norm, which is not a norm (so we use "." for it) because it does not satisfy the norm properties [4]. The "ℓ_0" norm counts the number of non-zero elements so when we penalize it, it means that we want to have sparser solutions with many zero entries.
- According to [5], the convex relaxation of "ℓ_0" norm (subset selection) is ℓ_1 norm. Therefore, we write the regularized optimization as:

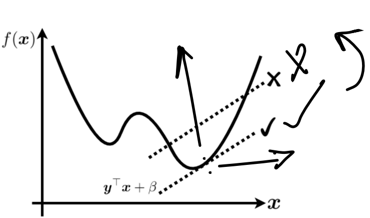
$$\boxed{\underset{\boldsymbol{x}}{\text{minimize}} \quad \Omega(\boldsymbol{x}) := \Omega_0(\boldsymbol{x}) + \lambda ||\boldsymbol{x}||_1.} \qquad (3)$$

- The ℓ_1 regularization is also referred to as **lasso (least absolute shrinkage and selection operator)** regularization [6, 7]. Different methods exist for solving optimization having ℓ_1 norm, such as its **approximation by Huber function** [8], **proximal algorithm** and **soft thresholding** [9], **coordinate descent** [10, 11], and **subgradients**. In the following, we explain these methods.

Convex Conjugate and the Huber function

Convex Conjugate





- Consider this figure showing a line which supports the function f meaning that it is tangent to the function and the function upper-bounds it. In other words, if the line goes above where it is, it will intersect the function in more than a point.
- Now let the support line be multi-dimensional to be a support hyperplane. For having this tangent support hyperplane with slope $\boldsymbol{y} \in \mathbb{R}^d$ and intercept $\beta \in \mathbb{R}$, we should have:

$$\boxed{\boldsymbol{y}^\top \boldsymbol{x} + \beta} = f(\boldsymbol{x}) \implies \beta = f(\boldsymbol{x}) - \boldsymbol{y}^\top \boldsymbol{x}.$$

- We want the smallest intercept for the support hyperplane:

$$\boxed{\beta^* = \min_{\boldsymbol{x} \in \mathbb{R}^d} \big(f(\boldsymbol{x}) - \boldsymbol{y}^\top \boldsymbol{x}\big)} = -\max_{\boldsymbol{x} \in \mathbb{R}^d} \big(\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\big).$$

Convex Conjugate

- We found:

$$\boxed{\beta^* = \min_{\boldsymbol{x} \in \mathbb{R}^d} \big(f(\boldsymbol{x}) - \boldsymbol{y}^\top \boldsymbol{x}\big) = -\max_{\boldsymbol{x} \in \mathbb{R}^d} \big(\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\big).}$$

Definition (Convex conjugate of function)

The conjugate gradient of function $f(.)$ is defined as:

$$\boxed{f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \big(\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\big).} \tag{4}$$

- Therefore, we define $f^*(\boldsymbol{y}) := -\beta^*$ to have the convex conjugate defined above.
- The convex conjugate of a function is always convex, even if the function itself is not convex, because it is point-wise maximum of affine functions.

Convex Conjugate

Recall the convex conjugate: $f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \big(\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\big)$. ★

Lemma (Conjugate of convex conjugate)

The conjugate of convex conjugate of a function is:

$$\underline{f^{**}(\boldsymbol{x})} = \underbrace{\sup_{\boldsymbol{y} \in \mathbf{dom}(f^*)} \big(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y})\big)}. \tag{5}$$

It is always a <u>lower-bound for the function</u>, i.e., $f^{**}(\boldsymbol{x}) \leq f(\boldsymbol{x})$.
If the function $f(.)$ is convex, we have $f^{**}(\boldsymbol{x}) = f(\boldsymbol{x})$; hence, for a convex function, we have:

$$★ \quad \underbrace{\boxed{f(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in \mathbf{dom}(f^*)} \big(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y})\big).}}_{f^{**}(x)} \tag{6}$$

Huber Function: Smoothing L1 Norm by Convex Conjugate

Lemma (The convex conjugate of ℓ_1 norm)

The convex conjugate of $f(.) = \|.\|_1$ is:

$f(x) = \|x\|_1$

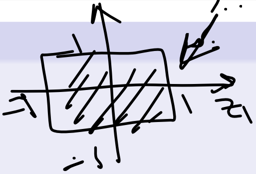
$$\boxed{f^*(\boldsymbol{y}) = \begin{cases} 0 & \textit{if } \|\boldsymbol{y}\|_\infty \leq 1 \\ \infty & \textit{Otherwise.} \end{cases}} \tag{7}$$

$z = \begin{bmatrix}1\\0\end{bmatrix}$

Proof.

We can write ℓ_1 norm as:

$$\boxed{f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 = \max_{\|\boldsymbol{z}\|_\infty \leq 1} \boldsymbol{x}^\top \boldsymbol{z}.}$$



Using this in Eq. (4), $f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \big(\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\big)$, results in Eq. (7) because:

$$f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \Big(\boldsymbol{y}^\top \boldsymbol{x} - \underbrace{\max_{\|\boldsymbol{z}\|_\infty \leq 1} \boldsymbol{x}^\top \boldsymbol{z}}_{f(x)}\Big) = \sup_{\boldsymbol{x} \in \mathbb{R}^d} \Big(\underbrace{\boldsymbol{y}^\top \boldsymbol{x}}_{0} - \max_{\|\boldsymbol{z}\|_\infty \leq 1} \boldsymbol{z}^\top \boldsymbol{x}\Big) = \begin{cases} 0 & \text{if } \|\boldsymbol{y}\|_\infty \leq 1 \\ \infty & \text{Otherwise.} \end{cases}$$

$\|y\|_\infty \leq 1$

□

Huber Function: Smoothing L1 Norm by Convex Conjugate

0

Lemma (Gradient in terms of convex conjugate)

For any function $f(.)$, we have:

$$\boxed{\nabla f(\boldsymbol{x}) = \arg \max_{\boldsymbol{y} \in \mathbf{dom}(f^*)} \big(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y})\big).} \tag{8}$$

- We saw that the convex conjugate of $f(.) = \|.\|_1$ is:

$$f^*(\boldsymbol{y}) = \begin{cases} 0 & \text{if } \|\boldsymbol{y}\|_\infty \leq 1 \\ \infty & \text{Otherwise.} \end{cases}$$

- According to Eq. (8), we have $\boxed{\nabla f(\boldsymbol{x}) = \arg\max_{\|\boldsymbol{y}\|_\infty \leq 1} \boldsymbol{x}^\top \boldsymbol{y}}$ for $f(.) = \|.\|_1$.
- For $\boldsymbol{x} = \mathbf{0}$, we have $\nabla f(\boldsymbol{x}) = \arg\max_{\|\boldsymbol{y}\|_\infty \leq 1} \mathbf{0}$ which has many solutions. Therefore, at $\boldsymbol{x} = \mathbf{0}$, the function $\|.\|_1$ norm is not differentiable and not smooth because the gradient at that point is not unique.

Huber Function: Smoothing L1 Norm by Convex Conjugate

- We can smooth the ℓ_1 norm at $\boldsymbol{x} = \boldsymbol{0}$ using convex conjugate.
- We saw that the convex conjugate of $f(.) = \|.\|_1$ is:

$$\boxed{f^*(\boldsymbol{y}) = \begin{cases} 0 & \text{if } \|\boldsymbol{y}\|_\infty \leq 1 \\ \infty & \text{Otherwise.} \end{cases}}$$

- Let $\boldsymbol{x} = [x_1, \dots, x_d]^\top$. As we have $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 = \sum_{j=1}^d |x_j|$, we can use the convex conjugate for every dimension $f(x_j) = |x_j|$:

$$f^*(y_j) = \begin{cases} 0 & \text{if } |y_j| \leq 1 \\ \infty & \text{Otherwise.} \end{cases} \quad (9)$$

- According to Eq. (6), $f(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in \mathbf{dom}(f^*)} \left(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y})\right)$, we have:

$$|x_j| = \sup_{y \in \mathbb{R}} \left(x_j y_j - \cancel{f^*(y_j)}\right) \overset{(9)}{=} \max_{|y_j| \leq 1} x_j y_j. \quad = \max_{|y_j| \leq 1} \odot$$

- This is not unique for $x_j = 0$. Hence, we add a μ-strongly convex function to the above equation to make the solution unique at $x_j = 0$ also.
- This added term is named the **proximity function** defined below.

Huber Function: Smoothing L1 Norm by Convex Conjugate

Definition (Proximity function [12])

A proximity function $p(\boldsymbol{y})$ for a closed convex set $\mathcal{S} \in \textbf{dom}(p)$ is a function which is continuous and strongly convex. We can change Eq. (6), $f(\boldsymbol{x}) = \sup_{\boldsymbol{y} \in \textbf{dom}(f^*)} \big(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y})\big)$, to:

$$f(\boldsymbol{x}) \approx f_\mu(\boldsymbol{x}) := \sup_{\boldsymbol{y} \in \textbf{dom}(f^*)} \big(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y}) - \mu\, p(\boldsymbol{y})\big), \tag{10}$$

where $\mu > 0$.

- Recall Eq. (9): $f^*(y_j) = \begin{cases} 0 & \text{if } |y_j| \leq 1 \\ \infty & \text{Otherwise.} \end{cases}$
- Using Eq. (10), we can have:

$$|x_j| \approx \sup_{y \in \mathbb{R}} \big(x_j y_j - f^*(y_j) - \frac{\mu}{2} y_j^2\big) \overset{(9)}{=} \max_{|y_j| \leq 1} \big(x_j y_j - \frac{\mu}{2} y_j^2\big) = \boxed{\begin{cases} \frac{x_j^2}{2\mu} & \text{if } |x_j| \leq \mu \\ |x_j| - \frac{\mu}{2} & \text{if } |x_j| > \mu. \end{cases}}$$

- This approximation to ℓ_1 norm, which is differentiable everywhere, including at $x_j = 0$, is named the **Huber function**.
- Note that the Huber function is the Moreau envelope of absolute value.

Huber Function: Smoothing L1 Norm by Convex Conjugate

Definition (Huber and pseudo-Huber functions (1992) [8])

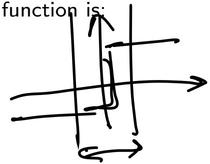
The Huber function and pseudo-Huber functions are:

$$\boxed{h_\mu(x) = \begin{cases} \frac{x^2}{2\mu} & \text{if } |x| \le \mu \\ |x| - \frac{\mu}{2} & \text{if } |x| > \mu, \end{cases}} \tag{11}$$

$$\widehat{h}_\mu(x) = \sqrt{\left(\frac{x}{\mu}\right)^2 + 1} - 1, \tag{12}$$

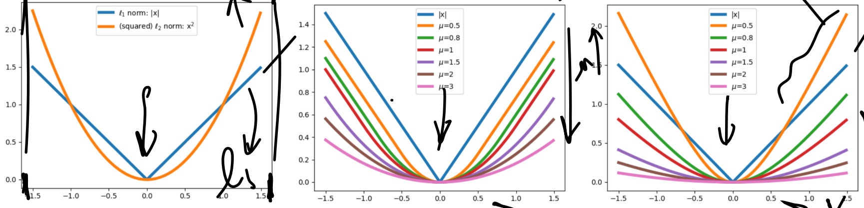
respectively, where $\mu > 0$.

The derivative of these functions is easily calculated. For example, the derivative of Huber function is:



$$\boxed{\nabla h_\mu(x) = \begin{cases} \frac{x}{\mu} & \text{if } |x| \le \mu \\ \mathbf{sign}(x) & \text{if } |x| > \mu. \end{cases}} \longrightarrow \boxed{\mu \approx 0}$$

Huber and pseudo-Huber Functions



(a) (b) (c)

- (a) Comparison of ℓ_1 and ℓ_2 norms in $\mathbb{R}^1$, (b) comparison of ℓ_1 norm (i.e., absolute value in $\mathbb{R}^1$) and the Huber function, and (c) comparison of ℓ_1 norm (i.e., absolute value in $\mathbb{R}^1$) and the pseudo-Huber function.
- In contrast to ℓ_1 norm or absolute value, these two functions are smooth so they approximate the ℓ_1 norm smoothly. This figure also shows that the Huber function is always upper-bounded by absolute value (ℓ_1 norm); however, this does not hold for pseudo-Huber function.
- We can also see that the approximation of Huber function is better than the approximation of pseudo-Huber function; however, its calculation is harder than pseudo-Huber function because it is a piece-wise function (compare Eqs. (11) and (12)).
- Moreover, the figure shows a smaller positive value μ give better approximations, although it makes calculation of the Huber and pseudo-Huber functions harder.

Soft-thresholding and Proximal Methods

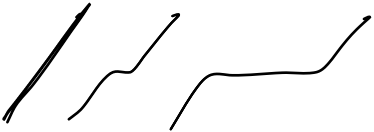
Soft-thresholding and Proximal Methods

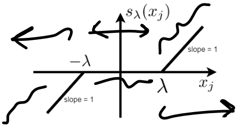
- Proximal mapping was introduced before:

$$\boxed{\mathbf{prox}_{\lambda g}(\boldsymbol{x}) := \arg\min_{\boldsymbol{u}} \Big(g(\boldsymbol{u}) + \frac{1}{2\lambda}\|\boldsymbol{u} - \boldsymbol{x}\|_2^2\Big).} \tag{13}$$

- We can use proximal mapping of non-smooth functions to solve non-smooth optimization by proximal methods introduced before.
- For example, we can solve an optimization problem containing ℓ_1 norm in its objective function using the proximal mapping of ℓ_1 norm (soft-thresholding):

$$[\mathbf{prox}_{\lambda\|.\|_1}(\boldsymbol{x})]_j = \max(0, |x_j| - \lambda)\, \mathbf{sign}(x_j) = s_\lambda(x_j) := \begin{cases} x_j - \lambda & \text{if } x_j \geq \lambda \\ 0 & \text{if } |x_j| < \lambda \\ x_j + \lambda & \text{if } x_j \leq -\lambda, \end{cases} \tag{14}$$





- Then, we can use any of the proximal methods such as proximal point method and proximal gradient method.
- For solving the regularized problem (3), $\boxed{\text{minimize}_{\boldsymbol{x}}\ \Omega(\boldsymbol{x}) := \Omega_0(\boldsymbol{x}) + \lambda\,||\boldsymbol{x}||_1}$ which is optimizing a composite function, we can use the proximal gradient method introduced before.

Coordinate Descent

Coordinate Method $P(x_1, \cdots, x_d)$

- Assume $\boldsymbol{x} = [x_1, \dots, x_d]^\top$. For solving $\text{minimize}_{\boldsymbol{x}}\ f(\boldsymbol{x})$, **coordinate method** [10] updates the dimensions (coordinates) of solution one-by-one and not all dimensions together at once:

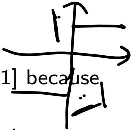
$$
\left.
\begin{aligned}
x_1^{(k+1)} &:= \arg\min_{x_1} f(x_1, x_2^{(k)}, x_3^{(k)}, \dots, x_d^{(k)}), \\
x_2^{(k+1)} &:= \arg\min_{x_2} f(x_1^{(k+1)}, x_2, x_2^{(k)}, \dots, x_d^{(k)}), \\
&\ \vdots \\
x_d^{(k+1)} &:= \arg\min_{x_d} f(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k+1)}, \dots, x_d),
\end{aligned}
\right\}
\tag{15}
$$

  until convergence of all dimensions of solution.
- Note that the update of every dimension uses the latest update of previously updated dimensions.
- The order of updates for the dimensions does not matter.
- The idea of coordinate descent algorithm is similar to the idea of Gibbs sampling [13, 14] where we work on the dimensions of the variable one by one.

Coordinate Descent

- If we use a step of gradient descent, $\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta \nabla f(\boldsymbol{x}^{(k)})$, for every of the above updates, the method is named **coordinate descent**.
- If we use proximal gradient method, $\boldsymbol{x}^{(k+1)} := \textbf{prox}_{\eta^{(k)} g}\big(\boldsymbol{x}^{(k)} - \eta^{(k)} \nabla f(\boldsymbol{x}^{(k)})\big)$, for every update in coordinate method, the method is named the **proximal coordinate descent**.
- We can group some of the dimensions (features) together and alternate between updating the blocks (groups) of features. That method is named **block coordinate descent**.
- The convergence analysis of coordinate descent and block coordinate descent methods can be found in [15, 16] and [17], respectively. They show that if the function $f(.)$ is continuous, proper, and closed, the coordinate descent method converges to a stationary point.
- There exist some other faster variants of coordinate descent named **accelerated coordinate descent** [18, 19].
- Similar to SGD, the full gradient is not available in coordinate descent to use for checking convergence. So, we should use other convergence criteria such as maximum number of iterations or checking convergence for each of the variables.
- Although coordinate descent methods are very simple and have shown to work properly for ℓ_1 norm optimization [11], they have not sufficiently attracted the attention of researchers in the field of optimization [10].

L1 Norm Optimization



- Coordinate descent method can be used for ℓ_1 norm (lasso) optimization [11] because every coordinate of the ℓ_1 norm is an absolute value ($\|\boldsymbol{x}\|_1 = \sum_{j=1}^{d} |x_j|$ for $\boldsymbol{x} = [x_1, \dots, x_d]^\top$) and the derivative of absolute value is a simple sign function.
- One of the well-known ℓ_1 optimization methods is the lasso regression (1996) [6, 2, 7]:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{16}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ are the labels, $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_d] \in \mathbb{R}^{n \times d}$ are the observations, $\boldsymbol{\beta} = [\beta_1, \dots, \beta_d]^\top \in \mathbb{R}^d$ are the regression coefficients, and λ is the regularization parameter. The lasso regression is sparse which is effective as explained before.

- Let c denote the objective function in Eq. (16). The objective function can be simplified as:

$$\|x\|_2^2 = x^\top x$$

$$C = 0.5(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{y} + \boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1.$$

- We can write the j-th element of this objective, denoted by c_j, as:

$$c_j = \frac{1}{2}(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{x}_j^\top\boldsymbol{y}\beta_j + \beta_j\boldsymbol{x}_j^\top\boldsymbol{x}_j\beta_j + \beta_j\boldsymbol{x}_j^\top\boldsymbol{X}_{-j}\boldsymbol{\beta}_{-j}) + \lambda|\beta_j|,$$

where $\boldsymbol{X}_{-j} := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_{j-1}, \boldsymbol{x}_{j+1}, \dots, \boldsymbol{x}_d]$ and $\boldsymbol{\beta}_{-j} := [\beta_1, \dots, \beta_{j-1}, \beta_{j+1}, \dots, \beta_d]^\top$.

L1 Norm Optimization

- We wrote the j-th element of this objective, denoted by c_j, as: $c_j = \frac{1}{2}(\boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{x}_j^\top \boldsymbol{y}\beta_j + \beta_j \boldsymbol{x}_j^\top \boldsymbol{x}_j \beta_j + \beta_j \boldsymbol{x}_j^\top \boldsymbol{X}_{-j}\boldsymbol{\beta}_{-j}) + \lambda|\beta_j|$.
- For coordinate descent, we need gradient of objective function w.r.t. every coordinate. The derivatives of other coordinates of objective w.r.t. β_j are zero so we need c_j for derivative w.r.t. β_j.
- Taking derivative of c_j w.r.t. β_j and setting it to zero gives:

$\partial / \partial \beta_j$

$$\frac{\partial c}{\partial \beta_j} = \frac{\partial c_j}{\partial \beta_j} = \boldsymbol{x}_j^\top \boldsymbol{x}_j \beta_j + \boldsymbol{x}_j^\top (\boldsymbol{X}_{-j}\boldsymbol{\beta}_{-j} - \boldsymbol{y}) + \lambda\, \mathbf{sign}(\beta_j) \overset{\text{set}}{=} 0$$

$$\implies \boxed{\beta_j = s_{\frac{\lambda}{\|\boldsymbol{x}_j\|_2^2}}\Big(\frac{\boldsymbol{x}_j^\top(\boldsymbol{y} - \boldsymbol{X}_{-i}\boldsymbol{\beta}_{-i})}{\boldsymbol{x}_j^\top \boldsymbol{x}_j}\Big)}$$

$$= \begin{cases} \frac{\boldsymbol{x}_j^\top(\boldsymbol{y}-\boldsymbol{X}_{-i}\boldsymbol{\beta}_{-i})}{\|\boldsymbol{x}_j\|_2^2} - \frac{\lambda}{\|\boldsymbol{x}_j\|_2^2} & \text{if } \frac{\boldsymbol{x}_j^\top(\boldsymbol{y}-\boldsymbol{X}_{-i}\boldsymbol{\beta}_{-i})}{\boldsymbol{x}_j^\top \boldsymbol{x}_j} \geq \frac{\lambda}{\|\boldsymbol{x}_j\|_2^2} \\ 0 & \text{if } |\frac{\boldsymbol{x}_j^\top(\boldsymbol{y}-\boldsymbol{X}_{-i}\boldsymbol{\beta}_{-i})}{\boldsymbol{x}_j^\top \boldsymbol{x}_j}| < \frac{\lambda}{\|\boldsymbol{x}_j\|_2^2} \\ \frac{\boldsymbol{x}_j^\top(\boldsymbol{y}-\boldsymbol{X}_{-i}\boldsymbol{\beta}_{-i})}{\|\boldsymbol{x}_j\|_2^2} + \frac{\lambda}{\|\boldsymbol{x}_j\|_2^2} & \text{if } \frac{\boldsymbol{x}_j^\top(\boldsymbol{y}-\boldsymbol{X}_{-i}\boldsymbol{\beta}_{-i})}{\boldsymbol{x}_j^\top \boldsymbol{x}_j} \leq -\frac{\lambda}{\|\boldsymbol{x}_j\|_2^2}, \end{cases}$$

which is a soft-thresholding function (see Eq. (14)).
- Therefore, coordinate descent for ℓ_1 optimization finds the soft-thresholding solution, the same as the proximal mapping. We can use this soft-thresholding in coordinate descent where we use β_j's in Eq. (15) rather than x_j's.

Subgradient Methods

Subgradient

- We know that the convex conjugate $f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbb{R}^d} \big(\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x})\big)$ is always convex.
- If the convex conjugate $f^*(\boldsymbol{y})$ is strongly convex, then we have only one gradient according to Eq. (8), $\nabla f(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \textbf{dom}(f^*)} \big(\boldsymbol{x}^\top \boldsymbol{y} - f^*(\boldsymbol{y})\big)$.
- However, if the convex conjugate is only convex and not strongly convex, Eq. (8) might have several solutions so the gradient may not be unique.
- For the points in which the function does not have a unique gradient, we can have a set of subgradients, defined below.

Definition (Subgradient)

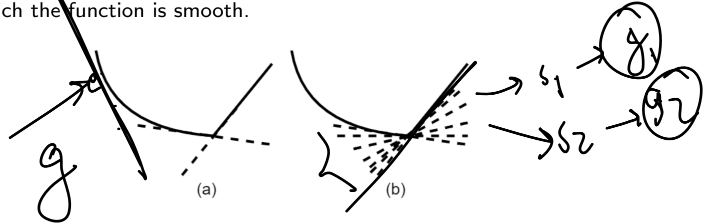
Consider a convex function $f(.)$ with domain $\mathcal{D}$. The vector $\boldsymbol{g} \in \mathbb{R}^d$ is a **subgradient** of $f(.)$ at $\boldsymbol{x} \in \mathcal{D}$ if it satisfies:

$$\boxed{f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^\top (\boldsymbol{y} - \boldsymbol{x}),} \quad \forall \boldsymbol{y} \in \mathcal{D}. \tag{17}$$

(a) (b)

Subdifferential

- As this figure shows, if the function is not smooth at a point, it has multiple subgradients at that point. This is while there is only one subgradient (which is the gradient) for a point at which the function is smooth.



(a) (b)

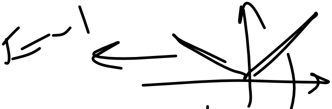
Definition (subdifferential)

The subdifferential of a convex function $f(.)$, with domain $\mathcal{D}$, at a point $\boldsymbol{x} \in \mathcal{D}$ is the set of all subgradients at that point:

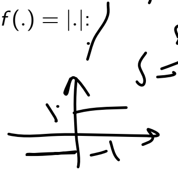
$$\partial f(\boldsymbol{x}) = \{\boldsymbol{g} \,|\, \boldsymbol{g}^\top(\boldsymbol{y} - \boldsymbol{x}) \overset{(17)}{\leq} f(\boldsymbol{y}) - f(\boldsymbol{x}), \forall \boldsymbol{y} \in \mathcal{D}\}. \tag{18}$$

- The subdifferential is a closed convex set.
- Every subgradient is a member of the subdifferential, i.e., $\boldsymbol{g} \in \partial f(\boldsymbol{x})$. An example subdifferential is shown in the above figure.

Subdifferential for ℓ_1 norm



- An example of subgradient is the subdifferential of absolute value, $f(.) = |.|$:



$$\partial f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } x > 0 \\ \in [-1, 1] & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases} \tag{19}$$

- The subgradient of absolute value is equal to the gradient for $x < 0$ and $x > 0$ while there exists a set of subgradients at $x = 0$ because absolute value is not smooth at that point.
- We can also compute the subgradient of ℓ_1 norm because we have:

$$\boxed{f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 = \sum_{i=1}^{d} |x_i| = \sum_{i=1}^{d} f_i(\boldsymbol{x}_i),}$$

for $\boldsymbol{x} = [x_1, \dots, x_d]^\top$.

- We take Eq. (19) as the subdifferential of the i-th dimension, denoted by $\partial f_i(x_i)$. Hence, for $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$, we have $\partial f(\boldsymbol{x}) = \partial f_1(x_1) \times \cdots \times \partial f_d(x_d)$ where $\times$ denotes the Cartesian product of sets.

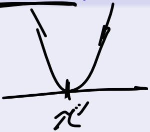
Subgradient

- We can have the first-order optimality condition using subgradients by generalizing $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$ as follows.

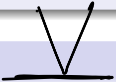
Lemma (First-order optimality condition with subgradient)

If $\boldsymbol{x}^*$ is a local minimizer for a function $f(.)$, then:

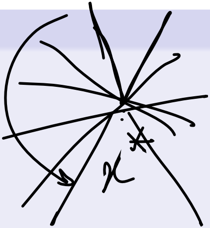
$$\boxed{\mathbf{0} \in \partial f(\boldsymbol{x}^*).} \tag{20}$$



Note that if $f(.)$ is convex, this equation is a necessary and sufficient condition for a minimizer.



Proof.



According to Eq. (17) in the definition of subgradient, we have:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}^*) + \boldsymbol{g}^\top(\boldsymbol{y} - \boldsymbol{x}^*), \forall \boldsymbol{y}.$$

If we have $\boldsymbol{g} = \mathbf{0} \in \partial f(\boldsymbol{x}^*)$, we have:

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}^*) + \mathbf{0}^\top(\boldsymbol{y} - \boldsymbol{x}^*) = f(\boldsymbol{x}^*),$$

which means that $\boldsymbol{x}^*$ is a minimizer. □

Subgradient in Some Example Functions

- The following lemma can be useful for calculation of subdifferential of functions.

Lemma

Some useful properties for calculation of subdifferential of functions:

- *For a smooth function or at points where the function is smooth, subdifferential has only one member which is the gradient:* $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.
- *Linear combination: If* $f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i f_i(\boldsymbol{x})$ *with* $a_i \geq 0$*, then* $\partial f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i \partial f_i(\boldsymbol{x})$.
- *Affine transformation: If* $f(\boldsymbol{x}) = f_0(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$*, then* $\partial f(\boldsymbol{x}) = \boldsymbol{A}^\top \partial f_0(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$.
- *Point-wise maximum: Suppose* $f(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), \dots, f_n(\boldsymbol{x})\}$ *where* f_i*'s are differentiable. Let* $I(\boldsymbol{x}) := \{i | f_i = f(\boldsymbol{x})\}$ *states which function has the maximum value for the point* $\boldsymbol{x}$*. At any point other than the intersection point of functions (which is smooth), the subgradient is* $g = \nabla f_i(\boldsymbol{x})$ *for* $i \in I(\boldsymbol{x})$*. At the intersection point of two functions (which is not smooth), e.g.* $f_i(\boldsymbol{x}) = f_{i+1}(\boldsymbol{x})$*, we have:*

$$\boxed{\partial f(\boldsymbol{x}) = \{\boldsymbol{g} \mid t\nabla f_i(\boldsymbol{x}) + (1-t)\nabla f_{i+1}(\boldsymbol{x}), \forall t \in [0,1]\}.}$$

Subgradient Method

- The **subgradient method**, first proposed in [20], is used for solving the unconstrained optimization problem, $\text{minimize}_{\boldsymbol{x}}\ f(\boldsymbol{x})$, where the function $f(.)$ is not smooth, i.e., not differentiable, everywhere in its domain.
- It iteratively updates the solution as:

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta^{(k)} \boldsymbol{g}^{(k)},} \tag{21}$$

∇f

where $\boldsymbol{g}^{(k)}$ is **any** subgradient of function $f(.)$ in point $\boldsymbol{x}$ at iteration k, i.e. $\boldsymbol{g}^{(k)} \in \partial f(\boldsymbol{x}^{(k)})$, and $\eta^{(k)}$ is the step size at iteration k.
- Comparing this update with the update in gradient descent:

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta \nabla f(\boldsymbol{x}^{(k)}),}$$

shows that gradient descent is a special case of the subgradient method because for a smooth function, gradient is the only member of the subdifferential set (see Lemma in the previous slide); hence, the only subgradient is the gradient.

Stochastic Subgradient Method

- Consider the optimization problem $\text{minimize}_{\boldsymbol{x}}\ f(\boldsymbol{x})$ where at least one of the $f_i(.)$ functions is not smooth.
- **Stochastic subgradient method** [21] randomly samples one of the points to update the solution in every iteration:

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta^{(k)} \boldsymbol{g}_i^{(k)},} \tag{22}$$

∇f_i

where $\boldsymbol{g}_i^{(k)} \in \partial f_i(\boldsymbol{x}^{(k)})$.

- Comparing this with the update in stochastic gradient descent (SGD):

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta^{(k)} \nabla f_i(\boldsymbol{x}^{(k)}),}$$

shows that stochastic gradient descent is a special case of stochastic gradient descent because for a smooth function, gradient is the only member of the subdifferential set (see Lemma in two previous slides).

- We can have **mini-batch stochastic subgradient method** which is a generalization of mini-batch SGD for non-smooth functions. In this case, the update of solution is:

$$\boxed{\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} - \eta^{(k)} \frac{1}{b} \sum_{i \in \mathcal{B}_{k'}} \boldsymbol{g}_i^{(k)}.} \tag{23}$$

- If the function is not smooth, we can also use subgradient instead of gradient in other stochastic methods such as SAG and SVRG, which were introduced before. For this, we need to use $\boldsymbol{g}_i^{(k)}$ rather than $\nabla f(\boldsymbol{x}^{(k)})$ in these methods.

Projected Subgradient Method

- Consider the constrained optimization problem:

$$\boxed{\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{S}, \end{aligned}} \tag{24}$$

where $\mathcal{S}$ is the feasible set of constraints.

- If the function $f(.)$ is not smooth, we can use he **projected subgradient method** [22] which generalizes the projected gradient method introduced before.
- Similar to the update in projected gradient method:

$$\boxed{\boldsymbol{x}^{(k+1)} := \Pi_{\mathcal{S}}\big(\boldsymbol{x}^{(k)} - \eta^{(k)} \nabla f(\boldsymbol{x}^{(k)})\big),}$$

projected subgradient method iteratively updates the solution as:

$$\boxed{\boldsymbol{x}^{(k+1)} = \Pi_{\mathcal{S}}\big(\boldsymbol{x}^{(k)} - \eta^{(k)} \boldsymbol{g}^{(k)}\big),} \tag{25}$$

until convergence of the solution.

Acknowledgement

- Some slides of this slide deck are inspired by the lectures of Prof. Stephen Boyd at the Stanford University.
- Some slides of this slide deck are inspired by the teaching of Prof. Mu Zhu at the University of Waterloo.
- Our tutorial also has the materials of this slide deck: [23]

References

[1] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019.

[2] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[3] P. Domingos, "The role of Occam's razor in knowledge discovery," *Data mining and knowledge discovery*, vol. 3, no. 4, pp. 409–425, 1999.

[4] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[5] D. L. Donoho, "For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 6, pp. 797–829, 2006.

[6] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2019.

References (cont.)

[8] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*, pp. 492–518, Springer, 1992.

[9] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[10] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[11] T. T. Wu and K. Lange, "Coordinate descent algorithms for lasso penalized regression," *The Annals of Applied Statistics*, vol. 2, no. 1, pp. 224–244, 2008.

[12] B. Banaschewski and J.-M. Maranda, "Proximity functions," *Mathematische Nachrichten*, vol. 23, no. 1, pp. 1–37, 1961.

[13] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

[14] B. Ghojogh, H. Nekoei, A. Ghojogh, F. Karray, and M. Crowley, "Sampling algorithms, from survey sampling to Monte Carlo methods: Tutorial and literature review," *arXiv preprint arXiv:2011.00901*, 2020.

[15] Z.-Q. Luo and P. Tseng, "On the convergence of the coordinate descent method for convex differentiable minimization," *Journal of Optimization Theory and Applications*, vol. 72, no. 1, pp. 7–35, 1992.

References (cont.)

[16] Z.-Q. Luo and P. Tseng, “Error bounds and convergence analysis of feasible descent methods: a general approach,” *Annals of Operations Research*, vol. 46, no. 1, pp. 157–178, 1993.

[17] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[18] Y. T. Lee and A. Sidford, “Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems,” in *2013 ieee 54th annual symposium on foundations of computer science*, pp. 147–156, IEEE, 2013.

[19] O. Fercoq and P. Richtárik, “Accelerated, parallel, and proximal coordinate descent,” *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 1997–2023, 2015.

[20] N. Z. Shor, *Minimization methods for non-differentiable functions*, vol. 3. Springer Science & Business Media, 2012.

[21] N. Z. Shor, *Nondifferentiable optimization and polynomial problems*, vol. 24. Springer Science & Business Media, 1998.

[22] Y. I. Alber, A. N. Iusem, and M. V. Solodov, “On the projected subgradient method for nonsmooth convex optimization in a Hilbert space,” *Mathematical Programming*, vol. 81, no. 1, pp. 23–35, 1998.

References (cont.)

[23] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "KKT conditions, first-order and second-order optimization, and distributed optimization: Tutorial and survey," *arXiv preprint arXiv:2110.01858*, 2021.