Optimization for Data Science Lecture 12: Proximal Gradient Methods

Kimon Fountoulakis

School of Computer Science
University of Waterloo

24/10/2019

Outline of this lecture

- Motivation
- Proximal mapping
- Optimality conditions for composite problems
- Proximal gradient method with fixed step-size

Modelling Motivation

- We are interested in minimizing

$$\textbf{minimize}_{x \in \mathbb{R}^n} \; g(x) + f(x)$$

- $f(x)$ is smooth (differentiable)

- $g(x)$ is convex with a so-called inexpensive proximal operator (will be defined later). This function is not-necessarily smooth.

Modelling Motivation

- This optimization problem with two components is often called **composite** optimization problem.

- Composite problems are very popular in machine learning because

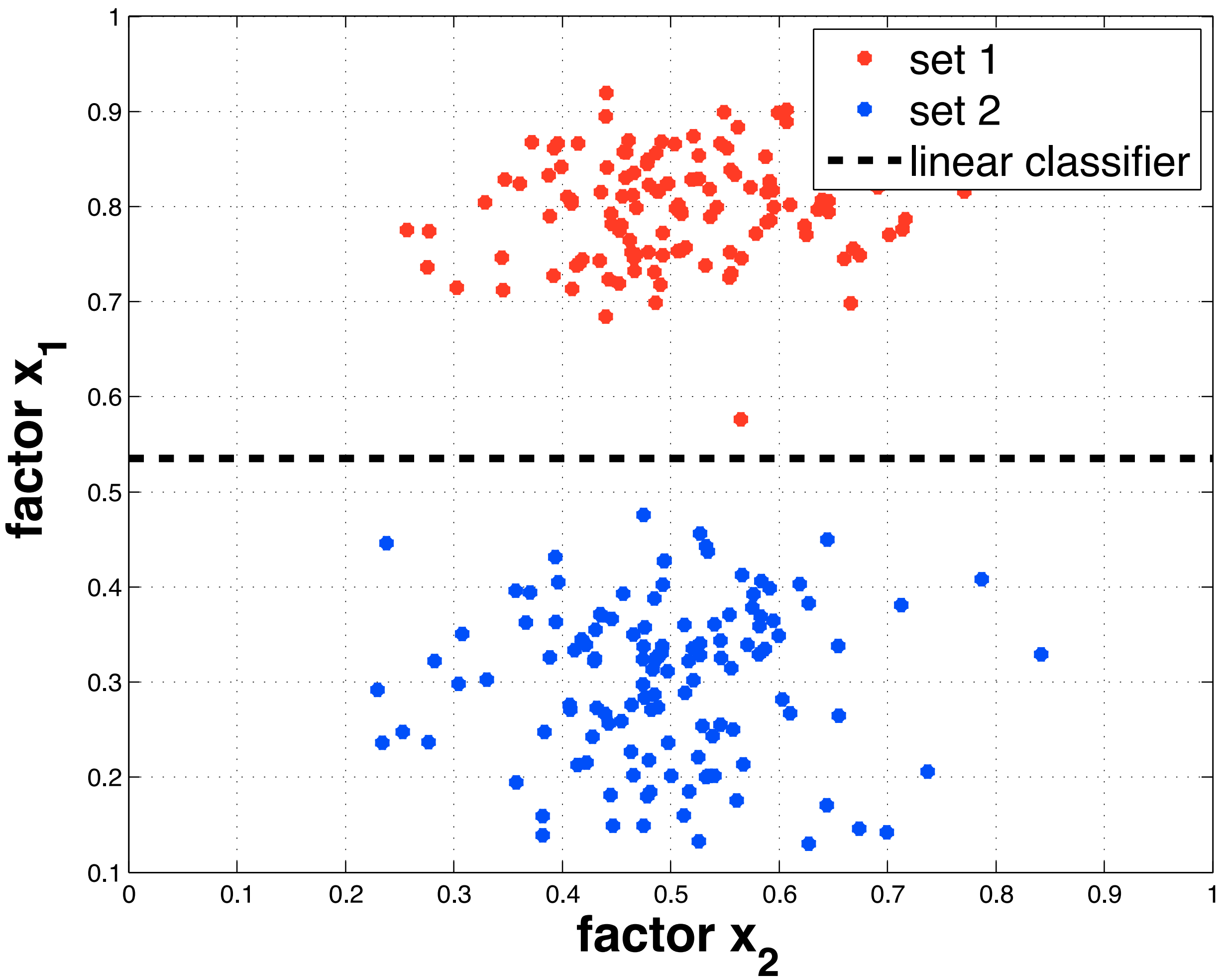
    - f represents a loss function.

    - g represents a regularizer, i.e., $\|x\|_2^2$, $\|x\|_1$.

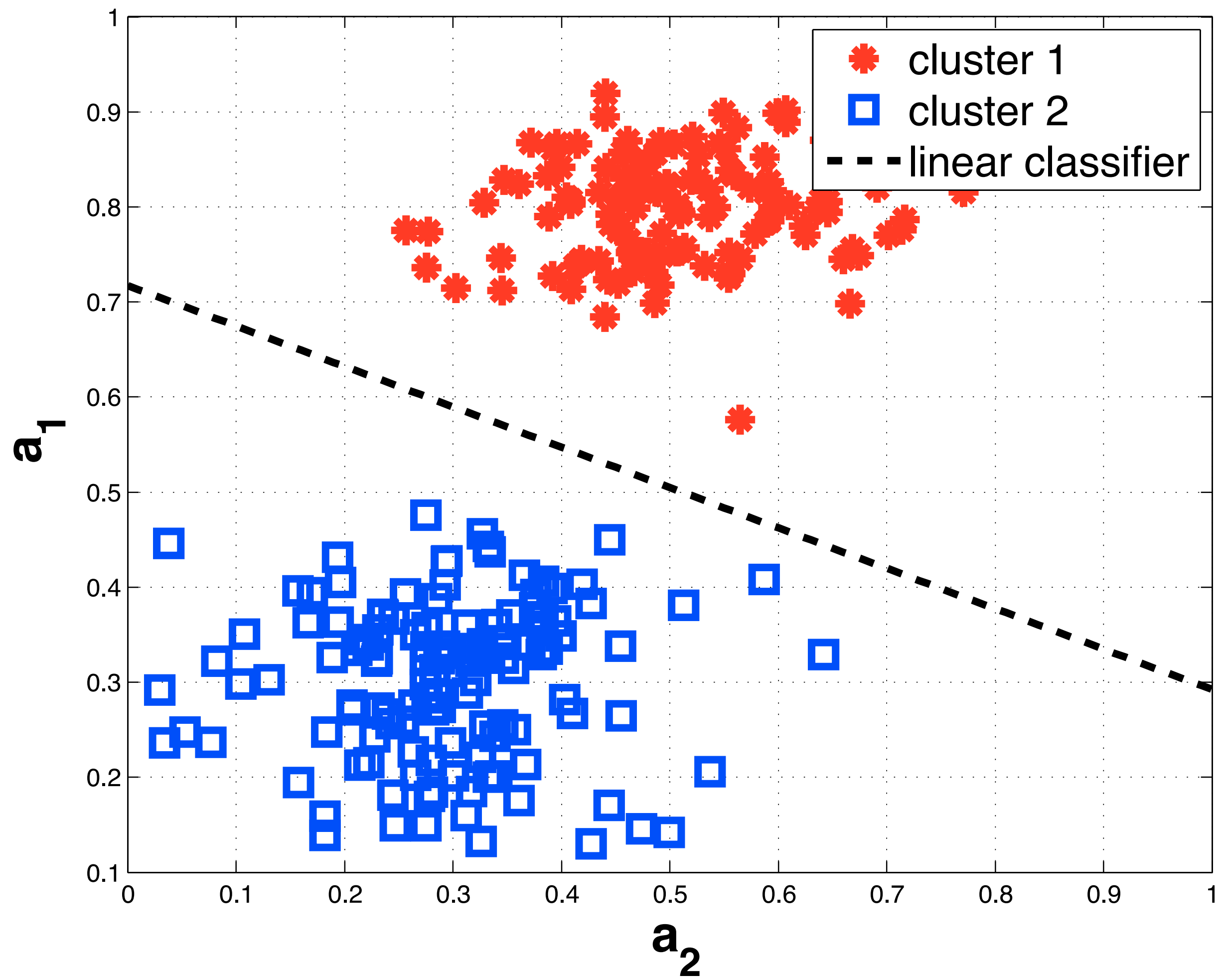
- Different regularizers often represent different prior information about the optimal solution.

Machine Learning Motivation

Logistic regression with L1-norm as the regularizer

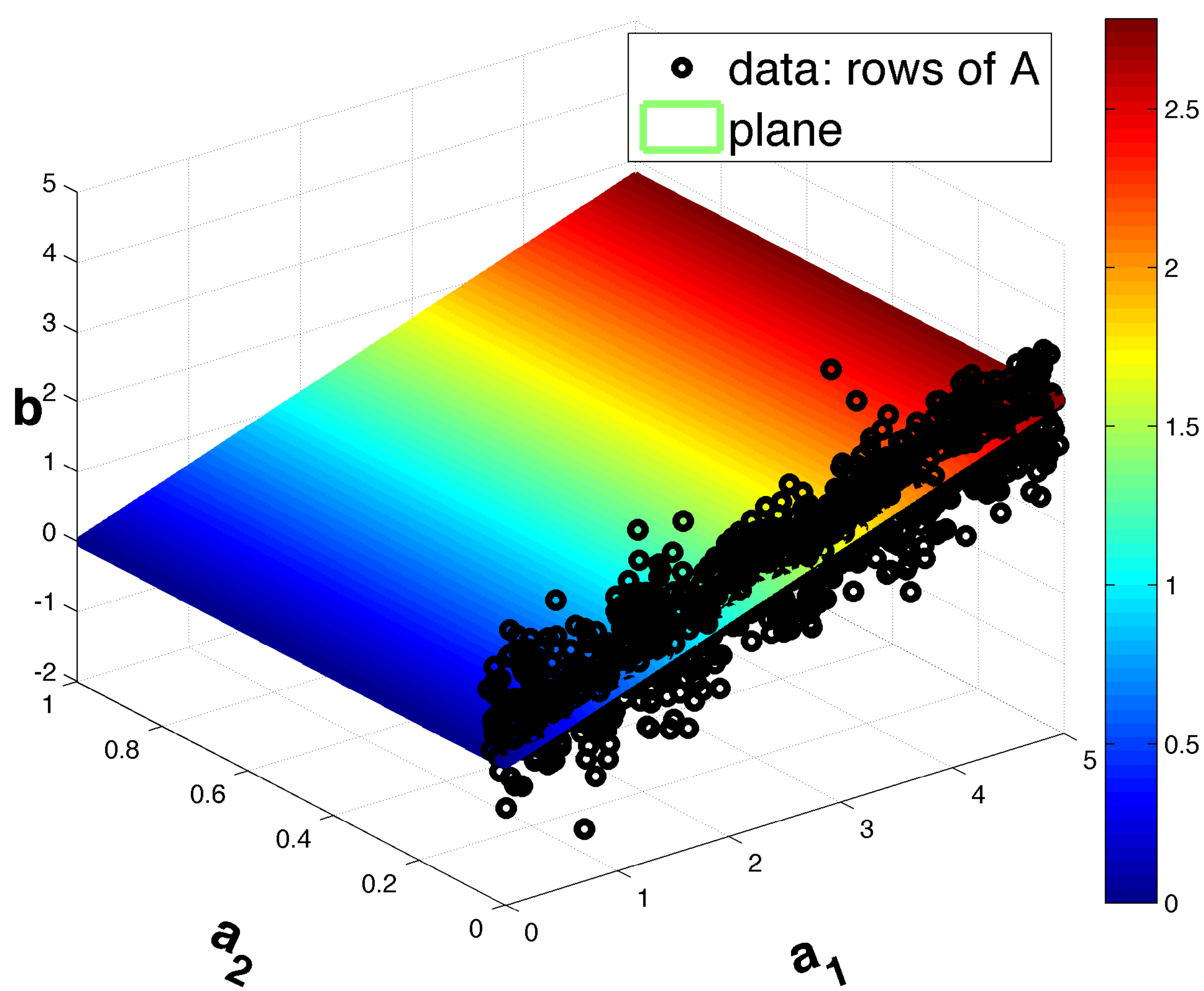


Logistic regression with L2-norm as the regularizer

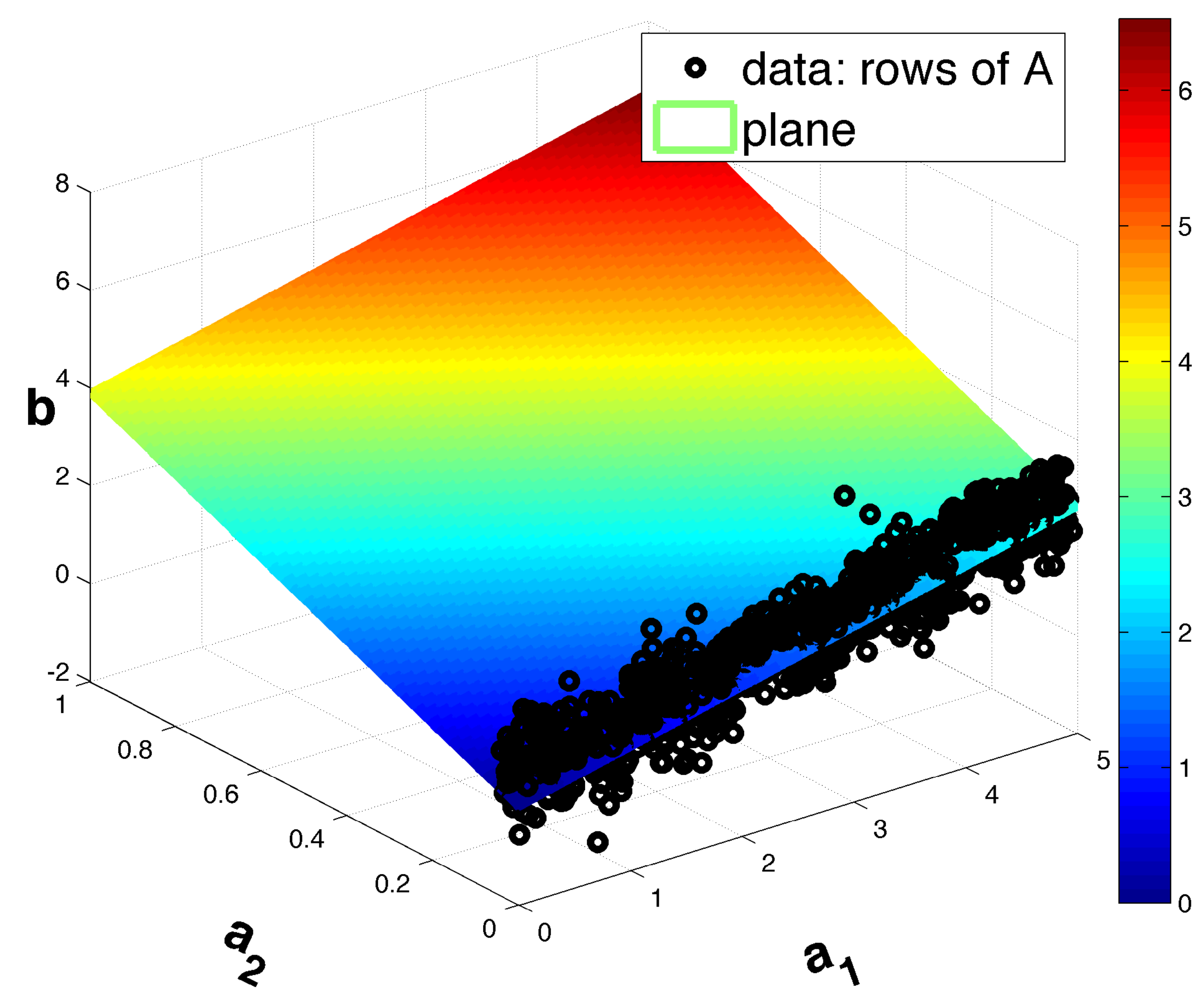


Machine Learning Motivation

Regression with L1-norm as the regularizer



Regression with L2-norm as the regularizer



Algorithmic Motivation

- So far we have seen two ways to solve non-smooth problems:
  - Smooth the objective function and apply a gradient-type method
  - Use a sub-gradient method on the non-smooth objective function

Algorithmic Motivation

- Smoothing makes the problem differentiable, but iteration complexity of gradient methods takes a hit.

- Sub-gradient methods are very slow and they require a lot of parameter tuning.

- There exists a **very** popular class of non-smooth problems for which we can apply a specialized gradient method without smoothing or using sub-gradients. Also, the rate is worse than the rate for smooth objective functions.

Proximal Mapping

- The **proximal mapping** or **proximal operator** of a convex function g is defined as

  $\text{prox}_g(x) = \text{argmin}_{u \in \mathbb{R}^n} \; g(u) + \dfrac{1}{2}\|u - x\|_2^2$

Proximal Mapping

- Examples
  - $g(x) = 0$, then $\text{prox}_g(x) = x$
  - $g(x)$ is an indicator function to a convex set C, then $\text{prox}_g(x) = \text{argmin}_{u \in C} \frac{1}{2} \|u - x\|_2^2$, which is the definition of projection of point x to set C.

Proximal Mapping

- Examples

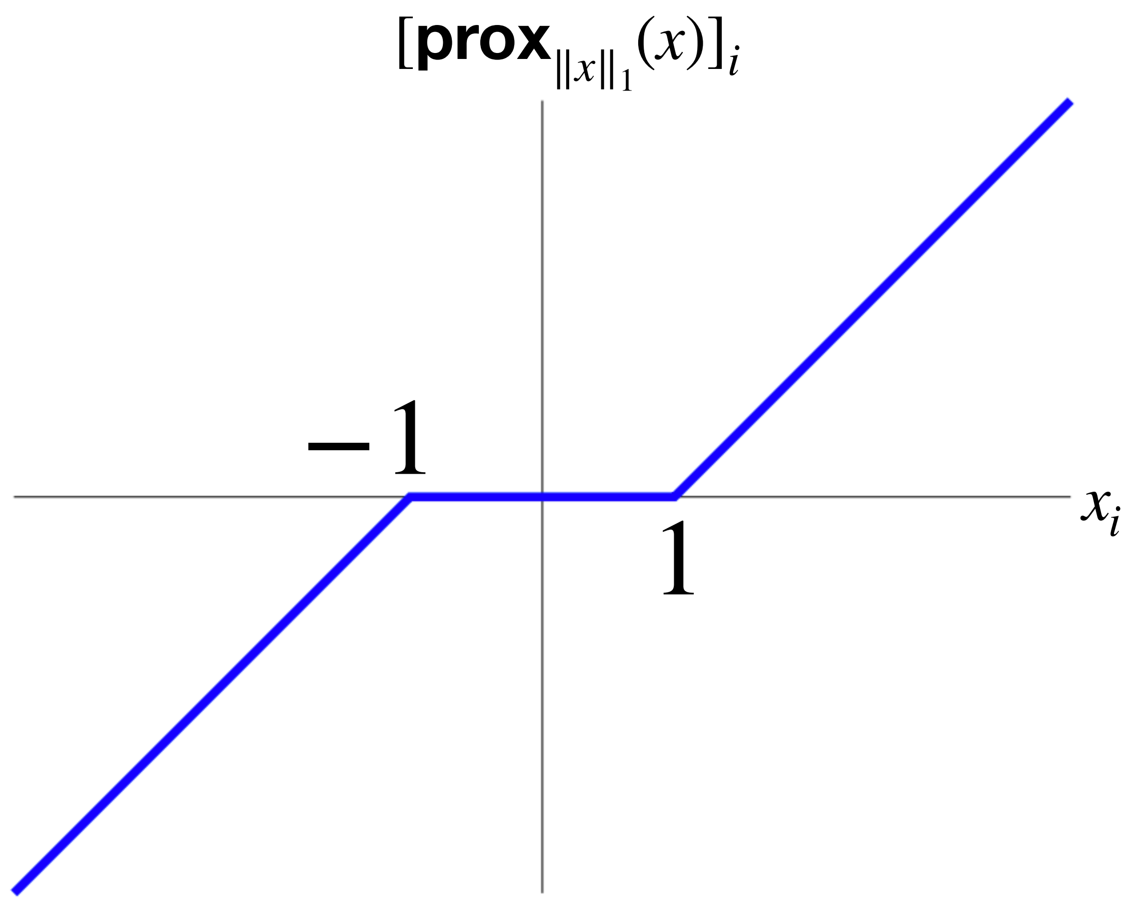
  - $g(x) = \|x\|_1$, then
    $$[\text{prox}_g(x)]_i = \begin{cases} x_i - 1 & \text{if } x_i \geq 1 \\ 0 & \text{if } |x_i| \leq 1 \\ x_i + 1 & \text{if } x_i \leq -1 \end{cases} \quad \forall i$$

  - This is also called the soft-threshold or shrinkage operator.

  - This operator is applied coordinate-wise.

Proximal Mapping

- Visualization of the soft-thresholding operator.



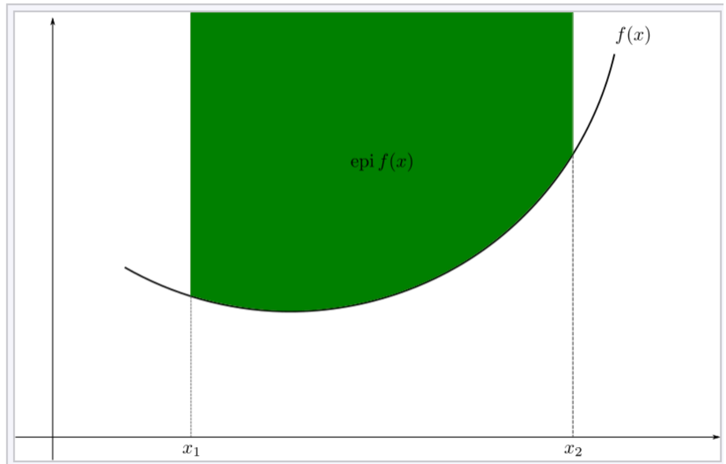
Proximal Mapping

- Examples
  - $g(x) = \|x\|_2$, then
    $$\text{prox}_g(x) = \begin{cases} 1 - \frac{1}{\|x\|_2}x & \text{if } \|x\|_2 \geq t \\ 0 & \text{otherwise} \end{cases}$$
  - $g(x) = \frac{1}{2}x^T A x + b^T x + c$, then
    $$\text{prox}_g(x) = (I + A)^{-1}(x - b)$$

Some definitions

- A function h is called proper if there exists at least one x such that $h(x) < +\infty$ and for any x we have that $h(x) > -\infty$. The first condition implies that the domain is non-empty.

- Let $C \cap \operatorname{dom} h \neq \emptyset$. We define the epigraph of a function $h : C \to [-\infty, +\infty]$ as
epi$(h) = \{(x, w) \in \operatorname{dom} h \times \mathbb{R} \mid x \in C, w \in \mathbb{R}, h(x) \leq w\}$



A function (in black) is convex if and only if the region above its graph (in green) is a convex set. This region is the function's epigraph.

Some definitions

- A function $h : C \to [-\infty, +\infty]$ is said to be closed if it's epi(h) is a closed set.

Assumptions

- We are interested in solving: minimize$_{x \in \mathbb{R}^n}$ $g(x) + f(x)$
- Function g is proper, closed, convex and with bounded domain.
- Function f is proper, closed and with bounded domain.
- dom $g \cap$ dom $f \neq \emptyset$

Proximal Mapping: properties

- The **proximal mapping** or **proximal operator** of a convex function g is defined as

  $\text{prox}_g(x) = \text{argmin}_{u \in \mathbb{R}^n}\ g(u) + \frac{1}{2}\|u - x\|_2^2$

  - We have that $g(u) + \frac{1}{2}\|x - u\|_2^2$ is strongly convex.
  - This implies that the proximal point $\text{prox}_g(x)$ is **unique**.
  - (We did not really prove this, but it is a standard result in optimization based on the assumptions that we made about g, so I will claim it).

Proximal Mapping: properties

- For any $x, u \in \text{dom } g$ the following three are equivalent

    - $u = \text{prox}_g(x)$

    - $x - u \in \partial g(u)$

    - $(x-u)^T(y-u) \leq g(y) - g(u) \ \forall y \in \text{dom } g$

Optimality Conditions of the Composite Problem

- Let's introduce the gradient mapping
  $G(x) := \frac{1}{\alpha}(x - x^+) = \frac{1}{\alpha}(x - \text{prox}_{\alpha g}(x - \alpha \nabla f(x)))$.

- where $\alpha > 0$

- For $x^* \in \text{dom } g \cap \text{dom } f$ it holds that $G(x^*) = 0$ if and only if x^* is a stationary point of the composite problem.

Optimality Conditions of the Composite Problem

- If f is convex, then $G(x^*) = 0$ is a necessary and sufficient condition.

Proximal Gradient Method

- $x_{k+1} = \text{prox}_{\alpha_k g}(x_k - \alpha_k \nabla f(x_k))$

- $\alpha_k g := \alpha_k g(x)$

- The step-size can be constant or it can be computed using line-search.

- Remember that for sub-gradient method is: $x_{k+1} = x_k - \alpha_k(\nabla f(x_k) + h(x_k))$ where $h(x_k) \in \partial g(x_k)$.

- Therefore, instead of using a sub-gradient of g, we use the proximal operator.

Proximal Gradient Method

- $x_{k+1} = \text{prox}_{\alpha_k g}(x_k - \alpha_k \nabla f(x_k))$

- is equivalent to

$$x_{k+1} = \text{argmin}_{x \in \mathbb{R}^n}\ g(x) + \frac{1}{2\alpha_k}\|x - x_k + \alpha_k \nabla f(x_k)\|_2^2$$

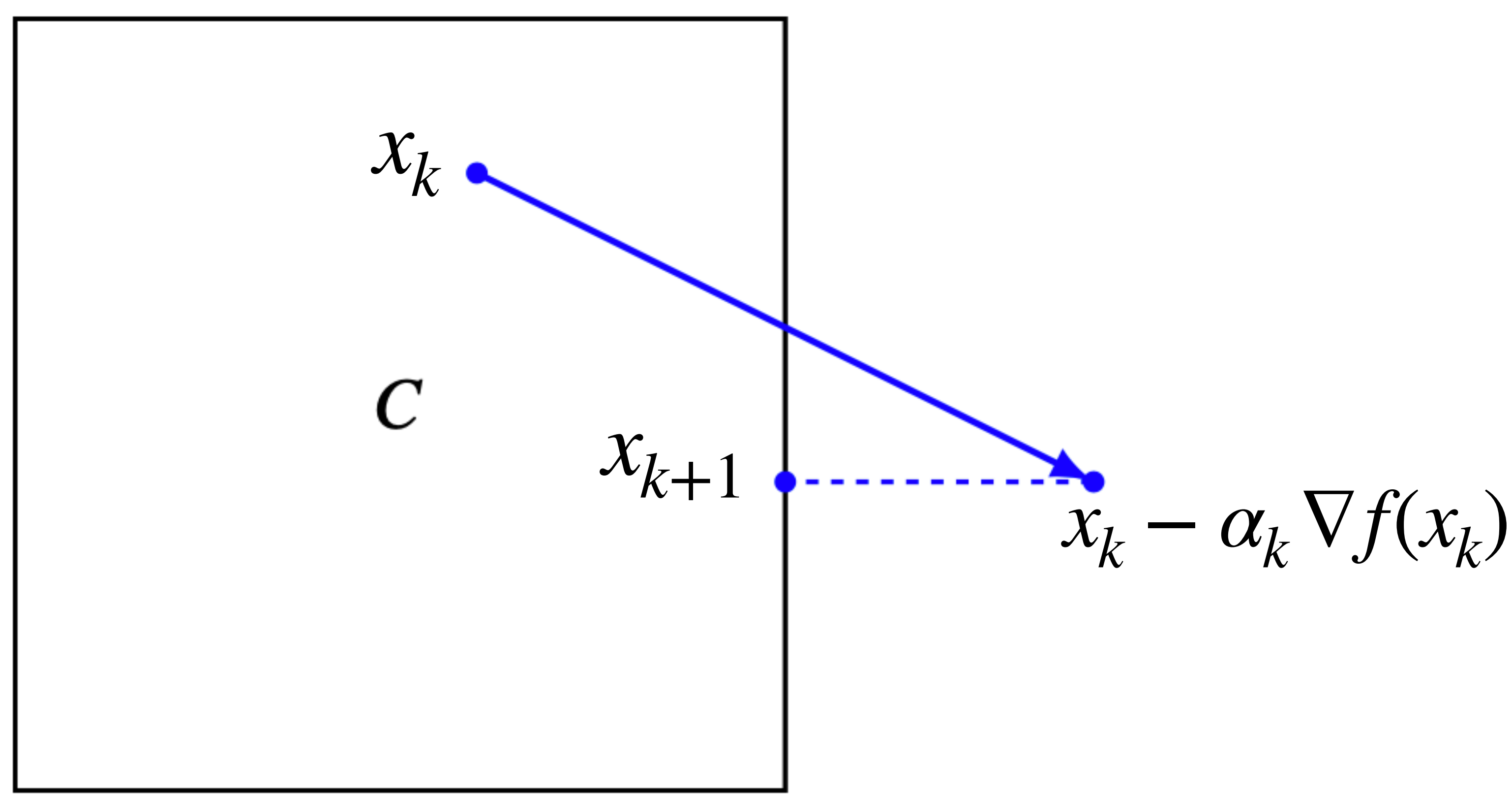
$$= \text{argmin}_{x \in \mathbb{R}^n} \quad \underbrace{g(x)}_{\text{new term}} \quad \underbrace{+ f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{\alpha_k 2}\|x - x_k\|_2^2}_{\text{similar to gradient descent for smooth } f}$$

Examples

- If $g(x) = 0$, then
- $x_{k+1} = \text{prox}_{\alpha_k g}(x_k - \alpha_k \nabla f(x_k)) = x_k - \alpha_k \nabla f(x_k)$
- Proximal gradient method becomes gradient descent for a smooth function f.

Examples

- If $g(x)$ is an indicator function to a convex set C, then
- $x_{k+1} = \text{prox}_{\alpha_k g}(x_k - \alpha_k \nabla f(x_k)) = \text{Projection}_C(x_k - \alpha_k \nabla f(x_k))$



Examples

- If $g(x) = \|x\|_1$, then
- Set $u = x_k - \alpha_k \nabla f(x_k)$
- $$[x_{k+1}]_j = [\text{prox}_{\alpha_k g}(u)]_j = \begin{cases} u_j - \alpha_k & \text{if } u_j \geq \alpha_k \\ 0 & \text{if } |u_j| \leq \alpha_k \\ u_j + \alpha_k & \text{if } u_j \leq -\alpha_k \end{cases} \quad \forall j$$

Convergence rate

- For non-convex and convex functions, the convergence rate of proximal gradient is $\mathcal{O}(L/t)$

- For δ-strongly convex functions, the convergence rate of proximal gradient is $\mathcal{O}((1-\delta/L)^t)$.

Iteration Complexity

	Smoothing + Gradient Descent	Smoothing + Accelerated Gradient	Stochastic Sub-Gradient	Proximal Gradient
Non-convex	$\mathcal{O}\left(\frac{D}{\epsilon^2}\right)$	**??**	$\mathcal{O}\left(\frac{1}{\epsilon^4}\right)$	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$
Convex	$\mathcal{O}\left(\frac{D}{\epsilon^2}\right)$	$\mathcal{O}\left(\frac{\sqrt{D}}{\epsilon}\right)$	$\mathcal{O}\left(e^{\frac{\sigma^2}{\epsilon}}\right)$	$\mathcal{O}\left(\frac{L}{\epsilon}\right)$
Strongly convex	$\mathcal{O}\left(\frac{D}{\delta\epsilon}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\sqrt{\frac{D}{\delta\epsilon}}\log\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{G\sigma^2}{\delta^2}\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{L}{\delta}\log\frac{1}{\epsilon}\right)$

- Some constants might be different, but roughly they are of the same order.
- Proximal gradient beats by far gradient any non-accelerated gradient or sub-gradient method.
- In later lectures we will see that we can accelerated proximal gradient as well, and this will gives us the fastest methods.

References

- Book: First-order Methods in Optimization by A. Beck