

Faculty of Sciences
University of Novi Sad



Department of Mathematics and
Informatics

Master's Thesis

Submitted to

Faculty of Sciences

Department of Mathematics and Informatics

By

Dunja Dragomanović

Convex Optimization for Big Data

Supervisor: Prof. dr. Dušan Jakovetić

December 2025

Acknowledgments

I am thoroughly thankful to Dr. Dušan Jakovetić for his mentorship and dedication to the students throughout the master's studies. I would also like to express my sincere gratitude to Dr. Maja Jolić and Dr. Nataša Krklec Jerinkić for their kind review of this thesis, valuable advice, and for serving on the thesis committee.

I am extremely thankful to all of my colleagues in both the bachelor's and master's programs.

Last but not least, I want to thank my family, especially my husband Danilo, for his huge support and belief in me throughout the studies. I dedicate this thesis to them.

Table of Contents

List of Figures.....	1
List of tables.....	2
1 Introduction.....	3
2 Convexity	5
2.1 Convex sets.....	5
<i>2.1.1 Definition of convex set.....</i>	<i>5</i>
<i>2.1.2 Some significant examples of convex sets</i>	<i>6</i>
<i>2.1.3 Operations that preserve convexity of sets</i>	<i>7</i>
2.2 Convex functions.....	12
<i>2.2.1 Definition of convex function.....</i>	<i>12</i>
<i>2.2.2 Jensen's inequality and extensions</i>	<i>13</i>
<i>2.2.3 Zero-order methods</i>	<i>13</i>
<i>2.2.4 First-order methods.....</i>	<i>14</i>
<i>2.2.5 Second-order methods</i>	<i>17</i>
<i>2.2.6 Epigraph</i>	<i>20</i>
<i>2.2.7 Examples of convex and concave functions</i>	<i>21</i>
<i>2.2.8 Operations that preserve convexity of functions</i>	<i>23</i>
3 Convex optimization problems.....	33
3.1 Optimization problems	33
<i>3.1.1 Fundamental terms</i>	<i>33</i>
<i>3.1.2 Optimal and locally optimal points.....</i>	<i>34</i>
3.2 Convex optimization problems.....	34
<i>3.2.1 Convex optimization problems in standard form</i>	<i>34</i>
<i>3.2.2 Abstract form convex optimization problem</i>	<i>35</i>
<i>3.2.3 Optima.....</i>	<i>35</i>
4 First-Order Methods for Smooth and Non-Smooth Convex Optimization	37
4.1 Smooth objectives	37
<i>4.1.1 Gradient Method</i>	<i>37</i>
<i>4.1.2 Accelerated gradient method</i>	<i>44</i>
4.2 Composite objectives.....	45

4.3 Proximal objectives	50
5 Big Data scaling via randomization	53
5.1 Coordinate descent methods	53
5.2 Stochastic gradient methods	56
5.3 Randomized linear algebra.....	61
6 The role of parallel and distributed computation.....	63
6.1 Embarrassingly parallel first-order methods.....	64
6.2 First-order methods with reduced or decentralized communications.....	66
6.3 Asynchronous first-order methods with decentralized communications	70
7 Conclusion	71

List of Figures

1. The square which contains some boundary points but not others(source:[1])	5
2. Banana-shaped set(source:[1]).....	5
3. Non-convex set	5
4. Convex hull of banana-shaped set from Figure 2(source:[1])	5
5. The convex hull of a set of fifteen points (dots) is the pentagon(source:[1])	6
6. <u>Left.</u> Set $S \subseteq \mathbb{R}^2$. Figure of the domain of the linear-fractional function	11
<u>Right.</u> Image of S under f . Figure of the domain of f^{-1} (source:[1])	
7. Graph of a convex function.....	12
8. An illustration of the restriction of function $f(x_1, x_2)$ to a line(source:[7])	14
9. Illustration of inequality (2.3) (source:[1])	15
10. Epigraph of a function(shaded)(source:[1])	20
11. Progression of function $F(x^k)$ for solving (right) LASSO and (left) LS formulation for four methods with practical enhancements(source:[6])	48
12. The convergence plot of three methods, where objective is heteroskedastic LASSO.....	50

List of tables

1. *Total number of iterations to reach an accuracy ϵ for the different cases (source:[6])*48

1 Introduction

In various applied fields today, it has become standard practice to address challenges through data analysis, particularly by employing statistical and machine learning techniques on large datasets. This approach, commonly known as "Big Data," has gained significant traction in industrial applications. While these challenges emerge across a wide range of domains, they share several common features. First, the datasets tend to be extremely large, often containing hundreds of millions or even billions of data points. Second, the data is frequently high-dimensional, as advanced measurement and storage technologies allow for the collection of highly detailed information about each data point. Third, due to the scale of many modern applications, data is often stored or gathered in a distributed fashion. Consequently, it has become increasingly crucial to develop algorithms that are not only sufficiently sophisticated to capture the intricacies of contemporary data but also scalable enough to handle vast datasets in a parallel or decentralized manner. Many such problems can be posed in the framework of convex optimization.

In this master thesis the advantages in *convex optimization algorithms for Big Data* will be discussed. Big Data problems require a fundamental repair of how convex optimization algorithms are designed. The goal is to reduce the computational, storage, and communications problems which appear with a large amount of data.

Mathematical optimization (or just optimization) or mathematical programming is a field of mathematics that deals with selecting the best element from a set of feasible solutions, taking into account certain criteria. Finding the *optimal solution* is something that we encounter in our everyday life. Indeed, mathematical optimization has become an important tool in many areas. It plays a significant role in engineering disciplines, including electronic design automation and automatic control systems, as well as in optimal design challenges found in civil, chemical, mechanical, and aerospace engineering. Additionally, optimization techniques are applied to problems in network design and operation, finance, supply chain management, and scheduling. The range of applications continues to grow steadily.

In general, solving the optimization problem consists in maximizing or minimizing the objective function by systematic selection of values from the set of feasible solutions and by calculating the value of the function in (selected) optimal point.

Convex optimization problem is optimization problem where objective function is *convex*. Convex optimization is applied in machine learning, statistics, signal processing, control systems, finance, operations research, image processing, robotics, etc. There are constrained and unconstrained convex optimization. In this thesis the unconstrained problems will be analyzed.

The following section will define convex sets, provide examples, and discuss operations that preserve the convexity of sets. Convex sets are important because convex functions are defined on them. This section will also cover convex functions, their definitions, examples, and operations that preserve their convexity. Additionally, it will present zero, first, and second-order methods. In this context, zero, first, and second-order methods refer to different ways of analyzing the

convexity of functions based on derivatives. The third section discusses fundamental terms related to optimization problems, including their optimal and locally optimal points. It also covers convex optimization problems, both in standard and abstract forms, and their optimal points. The fourth section introduces first-order methods for both smooth and non-smooth convex optimization. For smooth problems, the Gradient Method and Accelerated Gradient Method are presented. It can be seen how effective these methods are when the objective function is convex or strongly convex with a Lipschitz continuous gradient. When the objective function F consists of a differentiable convex function f and a non-smooth convex function g (in composite form), the Proximal-Gradient Method and Accelerated Proximal-Gradient Method are introduced. This section also presents a reformulation of the composite form that can handle non-smooth and non-Lipschitz objective functions. To solve this reformulation, the ADMM (Alternating Direction Method of Multipliers) algorithm is used. The fifth section explores various randomization techniques designed to enhance the scalability of first-order optimization methods. It includes Coordinate Descent Methods and Stochastic Gradient Methods, highlighting key concepts such as the use of random partial updates for optimization variables, replacing deterministic gradient and proximal operations with cost-effective statistical estimators, and accelerating fundamental linear algebra procedures through randomization. The sixth section focuses on the influence of parallel and distributed computation on first order methods. The last section is conclusion.

Notation

In this thesis the following notation will be used:

$C^1(\Omega)$ - the class of functions that are continuously differentiable on Ω ; that is, functions whose first derivatives exist and are continuous on Ω .

$C^2(\Omega)$ - the space of functions that are twice continuously differentiable on Ω ; that is, functions whose first and second derivatives exist and are continuous on Ω .

∇f - gradient of function f ; that is, the vector of all first-order partial derivatives of f .

$\nabla^2 f$ - Hessian matrix of function f ; that is, the matrix of all second-order partial derivatives of function f .

2 Convexity

For this chapter reference [1] and [15] are consulted.

2.1 Convex sets

2.1.1 Definition of convex set

Definition 2.1.1 A set $S \subseteq \mathbb{R}^n$ is **convex** if for any $x, y \in S$ and any $\theta \in [0, 1]$, there holds that

$$\theta x + (1 - \theta) y \in S. \quad (2.1)$$

In words, S is convex if the line segment between any two points in S remains in S . It can be said that a set is convex if every point in the set can be seen by every other point, along an undisturbed straight path between them (undisturbed means being in the set).

A point of form $\theta_1 x_1 + \dots + \theta_k x_k$, where $\theta_1 + \dots + \theta_k = 1$ and $\theta_i \geq 0$, $i = 1, \dots, k$ is convex combination of points x_1, \dots, x_k from S . It can be proved that a set is convex if and only if it contains every convex combination of its points.

The convex hull of a set S , denoted by $\text{conv } S$, is the set of all convex combinations of points in S : $\text{conv } S = \{\theta_1 x_1 + \dots + \theta_k x_k \mid x_i \in S, \theta_i \geq 0, i = 1, \dots, k, \theta_1 + \dots + \theta_k = 1\}$. The convex hull $\text{conv } S$ is always convex.



Figure 1 The square which contains some boundary points but not others, is not convex. (source: [1])



Figure 2 Banana-shaped set (source: [1])

This set is not convex because the line segment between the two points in the set shown as dots is not in the set.

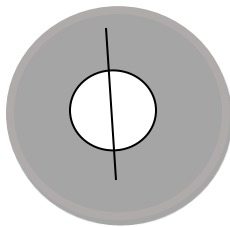


Figure 3 Non-convex set (source: [1])



Figure 4 Convex hull of banana-shaped set from Figure 2 (source: [1])

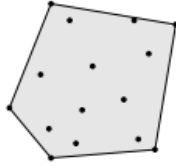


Figure 5 The convex hull of a set of fifteen points (dots) is the pentagon (shown tinted). (source: [1])

2.1.2 Some significant examples of convex sets

- The whole space \mathbb{R}^n , any single point, and the empty set \emptyset , are convex.
- A line segment is convex.
- Any subspace of \mathbb{R}^n is convex.
- A ray, which has the form $C = \{x_0 + \theta z \mid \theta \geq 0\}$, where $x_0 \in \mathbb{R}^n, z \in \mathbb{R}^n, z \neq 0$, is convex.

Proof:

Let's suppose that x_1 and x_2 belong to C , i.e. $x_1 \in C, x_2 \in C$. In other words, $x_1 = x_0 + \theta_1 z$ and $x_2 = x_0 + \theta_2 z$, $\theta_1 \geq 0, \theta_2 \geq 0$. According to the definition of convex sets, we have to prove that $\alpha x_1 + (1-\alpha) x_2 \in C$, where $0 \leq \alpha \leq 1$.

$$\alpha x_1 + (1-\alpha) x_2 = \alpha x_0 + \alpha \theta_1 z + (1-\alpha) x_0 + (1-\alpha) \theta_2 z = \alpha \theta_1 z + x_0 + (1-\alpha) \theta_2 z = x_0 + (\alpha \theta_1 + (1-\alpha) \theta_2) z.$$

From $0 \leq \alpha \leq 1$ and $\theta_1 \geq 0, \theta_2 \geq 0$ follows that $\alpha \theta_1 + (1-\alpha) \theta_2 \geq 0$, so $\alpha x_1 + (1-\alpha) x_2 \in C$.

- A ball is convex set.

A (Euclidean) ball in \mathbb{R}^n has the form:

$$B(x_c, r) = \{x \mid \|x - x_c\|_2 \leq r\}, \text{ where } r > 0, \text{ and } \|\cdot\|_2 \text{ denotes the Euclidean norm.}$$

Proof:

Suppose that x_1 and x_2 belong to ball B , so $\|x_1 - x_c\|_2 \leq r, \|x_2 - x_c\|_2 \leq r$, where $\theta \in [0,1]$, then

$$\begin{aligned} \|\theta x_1 + (1-\theta) x_2 - x_c\|_2 &= \|\theta(x_1 - x_c) + (1-\theta)(x_2 - x_c)\|_2 \leq \\ &\leq \theta \|x_1 - x_c\|_2 + (1-\theta) \|x_2 - x_c\|_2 \leq \text{(Follows from homogeneity and triangle inequality of Euclidean norm)} \\ &\leq \theta r + (1-\theta) r = r \text{ (Follows from assumption that } x_1 \text{ and } x_2 \text{ belong to ball } B) \end{aligned}$$

So, by definition of ball follows that $\theta x_1 + (1-\theta) x_2$ belongs to ball B , so B is convex set.

2.1.3 Operations that preserve convexity of sets

Intersection

Convexity is preserved under intersection.

Theorem 2.1.1 Suppose that $\{S_i\}_{i=1}^n$ is a family of sets which are convex. Then intersection of them $\bigcap_i S_i$ is convex.

Proof:

If intersection is empty set or consists of a single point then it is convex. Otherwise, if we take arbitrary $x, y \in \bigcap_i S_i$ and $\theta \in [0,1]$, then $x, y \in S_i \forall i$ (because x, y are in intersection). Since S_i is convex for $\forall i$ we have that $\theta x + (1-\theta)y \in S_i \forall i$, so $\theta x + (1-\theta)y \in \bigcap_i S_i$. From definition of convex set follows that $\bigcap_i S_i$ is convex. ■

Affine functions

Definition 2.1.2 A set $S \subseteq \mathbb{R}^n$ is **affine** if for any $x, y \in S$ and any $\theta \in \mathbb{R}$, there holds that

$$\theta x + (1-\theta)y \in S. \quad (2.2)$$

Definition 2.1.3 A function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ is **affine** if it has the form $f(x) = Ax + b$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Theorem 2.1.2 Suppose $S \subseteq \mathbb{R}^n$ is convex and $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ is an affine function. Then the image of S under f , $f(S) = \{f(x) \mid x \in S\}$, is also convex.

Proof:

Let's take arbitrary $f(x)$ and $f(y)$ from $f(S)$, ($x, y \in S$), and $\theta \in [0,1]$.

$$\begin{aligned} \theta f(x) + (1-\theta)f(y) &= \theta (Ax + b) + (1-\theta) (Ay + b) = \\ &= \theta Ax + \theta b + (1-\theta) Ay + (1-\theta) b = \\ &= \theta Ax + \theta b + (1-\theta) Ay + b - \theta b = \\ &= \theta Ax + (1-\theta) Ay + b = \\ &= A(\theta x + (1-\theta)y) + b \end{aligned}$$

Since S is convex follows that $\theta x + (1-\theta)y \in S$.

$\theta f(x) + (1-\theta)f(y)$ has a form of affine function and $\theta x + (1-\theta)y \in S$, so $\theta f(x) + (1-\theta)f(y) \in f(S)$. By definition we have that $f(S)$ is also convex set. ■

Theorem 2.1.3 Suppose that $f: \mathbb{R}^k \rightarrow \mathbb{R}^n$ is an affine function, $f^{-1}(S) = \{x \mid f(x) \in S\}$ (the inverse image of S under f), is convex.

Proof:

Let's take arbitrary $x, y \in f^{-1}(S)$ and $\theta \in [0,1]$. It is necessary to show that $\theta x + (1-\theta)y \in f^{-1}(S)$, so we will prove that $f(\theta x + (1-\theta)y) \in S$.

$$\begin{aligned} f(\theta x + (1-\theta)y) &= A(\theta x + (1-\theta)y) + b = \text{(This is because of assumption that } f \text{ is affine function)} \\ &= \theta Ax + (1-\theta)Ay + b + \theta b - \theta b = \\ &= \theta(Ax + b) + (1-\theta)(Ay + b) = \\ &= \theta f(x) + (1-\theta)f(y) \end{aligned}$$

Because $x, y \in f^{-1}(S)$, we know that $f(x), f(y) \in S$. From assumption that S is convex set follows that $\theta f(x) + (1-\theta)f(y) \in S$, so we proved the required. ■

Scaling

Theorem 2.1.4 If $S \subseteq \mathbb{R}^n$ is convex and $\alpha \in \mathbb{R}$ then set $\alpha S = \{\alpha x \mid x \in S\}$ is convex.

Proof:

Let's take arbitrary x_1 and x_2 from αS and $\theta \in [0,1]$. There are x and y from S such that $x_1 = \alpha x$ and $x_2 = \alpha y$.

$$\theta x_1 + (1-\theta)x_2 = \theta \alpha x + (1-\theta)\alpha y = \alpha(\theta x + (1-\theta)y)$$

Because S is convex set it follows that $\theta x + (1-\theta)y \in S$ by definition. Hence, $\alpha(\theta x + (1-\theta)y) \in \alpha S$. So, $\theta x_1 + (1-\theta)x_2 \in \alpha S$, which means that αS is convex set. ■

Translation

Theorem 2.1.5 If $S \subseteq \mathbb{R}^n$ is convex and $a \in \mathbb{R}^n$ then set $a + S = \{a + x \mid x \in S\}$ is convex.

Proof:

Let's take arbitrary x_1 and x_2 from $a + S$ and $\theta \in [0,1]$. There are x and y from S such that $x_1 = a + x$ and $x_2 = a + y$.

$$\theta x_1 + (1-\theta)x_2 = \theta(a + x) + (1-\theta)(a + y) = \theta a + \theta x + a + y - \theta a - \theta y = a + \theta x + (1-\theta)y$$

Because S is convex set it follows that $\theta x + (1-\theta)y \in S$ by definition. Hence, $a + \theta x + (1-\theta)y \in a + S$. So, $\theta x_1 + (1-\theta)x_2 \in a + S$, which means that $a + S$ is convex set. ■

Minkowski sum

Theorem 2.1.6 $S_1 + S_2$ is convex set if S_1 and S_2 are convex.

Proof:

*Recall that sum of two sets is defined as $S_1 + S_2 = \{x + y: x \in S_1 \text{ and } y \in S_2\}$

Suppose that $e, f \in S_1 + S_2$. Let $a, b \in S_1$ and $c, d \in S_2$ such that $e = a + c$ and $f = b + d$.

$$\begin{aligned}\theta e + (1 - \theta)f &= \theta(a + c) + (1 - \theta)(b + d) = \\ &= \theta a + \theta c + (1 - \theta)b + (1 - \theta)d = \\ &= (\theta a + (1 - \theta)b) + (\theta c + (1 - \theta)d)\end{aligned}$$

Since $a, b \in S_1$ and S_1 is convex set, from definition it follows that $\theta a + (1 - \theta)b \in S_1$. Similarly, since $c, d \in S_2$ and S_2 is convex set, from definition it follows that $\theta c + (1 - \theta)d \in S_2$.

To conclude, $(\theta a + (1 - \theta)b) + (\theta c + (1 - \theta)d) \in S_1 + S_2$, so $\theta e + (1 - \theta)f \in S_1 + S_2$, so it follows that $S_1 + S_2$ is convex set by definition. ■

Linear-fractional and perspective function

Perspective function

Definition 2.1.4 A perspective function is a function $P: \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$, $P(z, t) = \frac{z}{t}$, where $z \in \mathbb{R}^n$ and $t > 0$.

Theorem 2.1.8 Let P be perspective function. If $S \subseteq \text{dom } P$ is convex, then its image $P(S) = \{P(x) \mid x \in S\}$ is convex.

Proof:

Let's note that $P(S) = \{y \in \mathbb{R}^n : \exists (z, t) \in S, \text{ such that } y = \frac{z}{t}\}$. Consider any $y_1, y_2 \in P(S)$ where $y_1 = \frac{z_1}{t_1}$ and $y_2 = \frac{z_2}{t_2}$. We have to prove that for any $\theta \in [0, 1]$, $y = \theta y_1 + (1 - \theta) y_2 \in P(S)$. In other words, we have to prove that there exist $(z_0, t_0) \in S$ such that $y = \frac{z_0}{t_0}$.

Since $(z_1, t_1), (z_2, t_2)$ are two points in the convex set S , so their convex combinations are contained in S . Let's assume that $z_0 = \lambda z_1 + (1 - \lambda) z_2$ and $t_0 = \lambda t_1 + (1 - \lambda) t_2$.

The aim is then to find a $\lambda \in [0, 1]$ such that the following equality holds:

$$\theta \frac{z_1}{t_1} + (1 - \theta) \frac{z_2}{t_2} = \frac{\lambda z_1 + (1 - \lambda) z_2}{\lambda t_1 + (1 - \lambda) t_2}$$

It can be checked that previous equality holds when:

$$\lambda = \frac{\theta t_2}{(1-\theta)t_1 + \theta t_2}$$

Since $\theta \in [0, 1]$, it follows that $\lambda \in [0, 1]$.

So, we proved that given any two points $y_1, y_2 \in P(S)$, and any $\theta \in [0, 1]$, their convex combination y may be represented as $\frac{z_0}{t_0}$ where $(z_0, t_0) \in S$. This means that $y \in P(S)$. By definition, $P(S)$ is convex. ■

Theorem 2.1.9 If $S \subseteq \mathbb{R}^n$ is convex, then $P^{-1}(S)$ is convex where P is a perspective function.

Proof:

First note that $P^{-1}(S)$ can be written as:

$$P^{-1}(S) = \{(z, t) \in \mathbb{R}^{n+1} \mid \frac{z}{t} \in S, t > 0\}.$$

Let's assume that $(z_1, t_1), (z_2, t_2) \in P^{-1}(S)$ and $\theta \in [0, 1]$. We have to show that

$$\theta(z_1, t_1) + (1-\theta)(z_2, t_2) \in P^{-1}(S), \text{ i.e.}$$

$$y = \frac{\theta z_1 + (1-\theta)z_2}{\theta t_1 + (1-\theta)t_2} \in S$$

Let's suppose that y may be represented as: $y = \mu \frac{z_1}{t_1} + (1-\mu) \frac{z_2}{t_2}$, where $\mu \in [0, 1]$. Then we have to find μ such that:

$$\frac{\theta}{\theta t_1 + (1-\theta)t_2} z_1 + \frac{1-\theta}{\theta t_1 + (1-\theta)t_2} z_2 = \frac{\mu}{t_1} z_1 + \frac{1-\mu}{t_2} z_2$$

It can be showed that previous equality holds when:

$$\mu = \frac{\theta t_1}{\theta t_1 + (1-\theta)t_2}.$$

Because $\theta \in [0, 1]$ and $t_1, t_2 > 0$, it follows that $\mu \in [0, 1]$.

To conclude, y is a convex combination of $\frac{z_1}{t_1}$ and $\frac{z_2}{t_2}$, which are in S . From assumption that S is convex, it follows that $y \in S$, and $P^{-1}(S)$ is a convex set. ■

Linear-fractional functions

Definition 2.1.5 A linear-fractional function is composition of the perspective function and an affine function. Assume that $g: \mathbb{R}^n \rightarrow \mathbb{R}^{m+1}$ is affine, i.e.,

$$g(x) = \begin{bmatrix} A \\ c^T \end{bmatrix} x + \begin{bmatrix} b \\ d \end{bmatrix}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $d \in \mathbb{R}$. Then a linear-fractional function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ is defined by $f = P \circ g$, i.e.,

$$f(x) = \frac{Ax + b}{c^T x + d}, \text{ dom } f = \{x \mid c^T x + d > 0\}.$$

Theorem 2.1.10 If set $S \subseteq \text{dom } f$ is convex and f is linear-fractional function, then $f(S)$ is also convex set.

Proof:

Let's suppose that f is defined as above. As we proved, the image of convex set under affine mapping is convex (Affine functions preserve convexity), so $g(S)$ is convex. Furthermore, the image of $g(S)$ under the perspective function P , which is $f(S)$, is convex by Theorem 2.1.8. ■

Theorem 2.1.11 If set $S \subseteq \mathbb{R}^m$ is convex and f is linear-fractional function, then $f^{-1}(S)$ is convex set.

Proof:

Let's suppose that f is defined as in Definition 2.1.3. So, $f^{-1}(S) = g^{-1} \circ P^{-1}(S)$. From Theorem 2.1.9 we know that the inverse image of convex set under perspective function is convex, so $P^{-1}(S)$ is convex. Furthermore, the inverse image of $P^{-1}(S)$ under affine mapping is convex from Theorem 2.1.3. Therefore, $f^{-1}(S)$ is convex set. ■

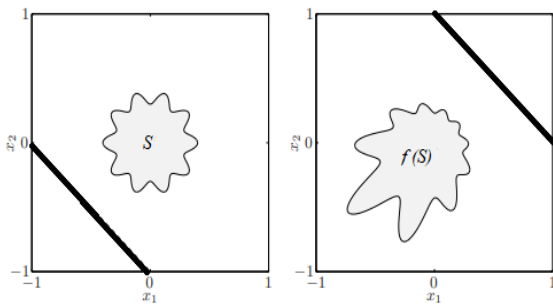


Figure 6 (source:[1])

Left. Set $S \subseteq \mathbb{R}^2$. The bold line shows the boundary of the domain of the linear-fractional function

$$f(x) = \frac{x}{x_1 + x_2 + 1} \text{ with} \\ \text{dom } f = \left\{ (x_1, x_2) \mid x_1 + x_2 + 1 > 0 \right\}.$$

Right. Image of S under f . The bold line shows the boundary of the domain of f^{-1} .

2.2 Convex functions

2.2.1 Definition of convex function

Definition 2.2.1 Let S be a convex set. A function $f: S \rightarrow \mathbb{R}$ is convex on S if for any $x, y \in S$ and any $\theta \in [0, 1]$, there holds that

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y). \quad (2.3)$$

In words, this definition indicates that if we take any two points x, y from S , then f evaluated at any convex combination of these two points should be no larger than the same convex combination of $f(x)$ and $f(y)$.

From the geometric aspect, the line segment connecting $(x, f(x))$ to $(y, f(y))$ must be above the graph of f , that can be seen in following figure.

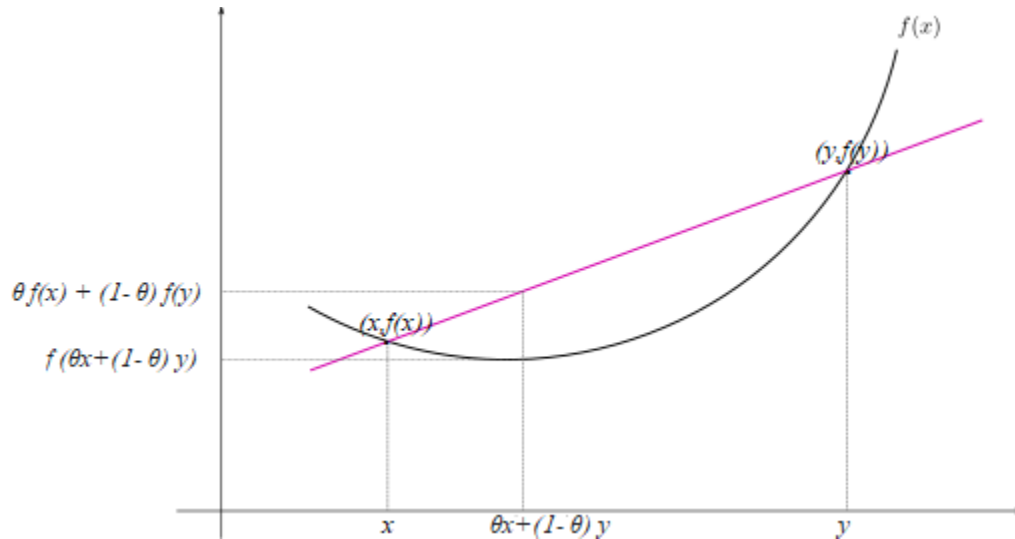


Figure 7 Graph of a convex function

The line segment between any two points on the graph lies above the graph.

The function is strictly convex if the previous inequality (2.3) is strict for all $x \neq y$ and $\theta \in (0, 1)$.

For example, function $f(x) = x^2$ is strictly convex.

The function is concave (strictly concave) if the inequalities in the previous definition are conversely. Namely, we put \geq ($>$) instead of \leq ($<$). By way of explanation, if the function $-f$ is convex, then f is concave and vice versa.

If in (2.3) is equality then f is an affine function, so all affine (thus also linear) functions are both convex and concave. On the other hand, any function that is convex or concave is affine.

2.2.2 Jensen's inequality and extensions

The inequality (2.2), i.e., $f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$, is called Jensen's inequality. It may be extended to convex combinations of more than two points:

Let's suppose that f is convex, $x_1, \dots, x_k \in \text{dom } f$, and $\theta_1, \dots, \theta_k \geq 0$ with $\theta_1 + \dots + \theta_k = 1$, then $f(\theta_1 x_1 + \dots + \theta_k x_k) \leq \theta_1 f(x_1) + \dots + \theta_k f(x_k)$.

2.2.3 Zero-order methods

Pay attention that a function is convex if and only if it is convex when restricted to any line that intersects its domain. Next theorem is about this statement.

Theorem 2.2.1 Let's suppose that S is convex set. Function f is convex on S if and only if for all $x \in S$ and all z , the function $g(t) = f(x + tz)$ is convex (on its domain, $\{t \mid x + tz \in S\}$).

Proof:

(\Rightarrow) Let's assume that S is convex set and f is convex function on S . Then for any $\theta \in [0, 1]$ and $t_1, t_2 \in \text{dom } g$:

$$\begin{aligned} g(\theta t_1 + (1 - \theta)t_2) &= f(x + (\theta t_1 + (1 - \theta)t_2)z) = \\ &= f(x + \theta t_1 z + (1 - \theta)t_2 z + \theta x - \theta x) = \\ &= f(\theta(x + t_1 z) + (1 - \theta)(x + t_2 z)) \leq \text{(This follows from assumption that } f \text{ is convex)} \\ &\leq \theta f(x + t_1 z) + (1 - \theta)f(x + t_2 z) = \\ &= \theta g(t_1) + (1 - \theta)g(t_2) \end{aligned}$$

We have $g(\theta t_1 + (1 - \theta)t_2) \leq \theta g(t_1) + (1 - \theta)g(t_2)$. By definition it follows that g is convex function on its domain.

(\Leftarrow) Let's suppose that S is convex set and g is convex function on its domain $\text{dom } g = \{t \mid x + tz \in S\}$.

Let us take $x_1, x_2 \in S$.

We need to show that for every $\theta \in [0, 1]$: $f(\theta x_1 + (1 - \theta)x_2) \leq \theta f(x_1) + (1 - \theta)f(x_2)$.

Since $g(t) = f(x + tz)$ is convex for all $x \in S$ and all z , for every $\theta \in [0, 1]$ we have:

$$g(\theta t_1 + (1 - \theta)t_2) \leq \theta g(t_1) + (1 - \theta)g(t_2)$$

So, by definition of g it follows:

$$f(x + z(\theta t_1 + (1 - \theta)t_2)) \leq \theta f(x + t_1 z) + (1 - \theta)f(x + t_2 z)$$

Let's take $x = x_1, z = x_2 - x_1, t_1 = 0, t_2 = 1$ and put these values in the previous inequality:

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

So, we have $f(\theta x_1 + (1 - \theta) x_2) \leq \theta f(x_1) + (1 - \theta) f(x_2)$. By definition it follows that f is convex on S .
 ■

This property is very useful, because it allows us to verify whether a function is convex by restricting it to a line.

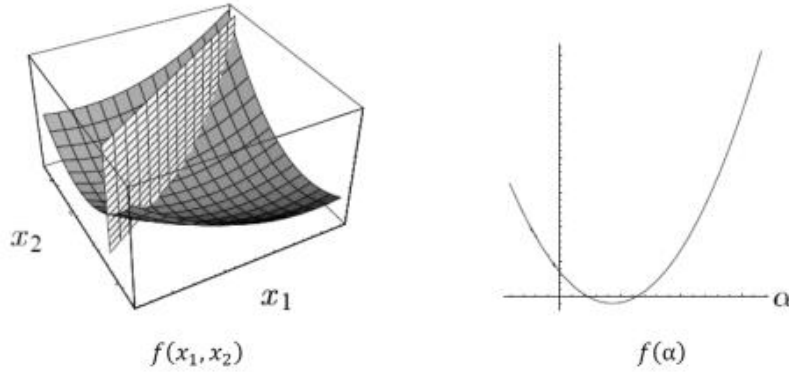


Figure 8 An illustration of the restriction of function $f(x_1, x_2)$ to a line (source: [7])

2.2.4 First-order methods

Theorem 2.2.2 Suppose that $S \subseteq \mathbb{R}^n$ is a convex set and $f \in C^1(S)$. Then the function f is convex on S if and only if the next holds for all $x, y \in S$.

$$f(y) \geq f(x) + \nabla^T f(x)(y-x). \quad (2.4)$$

Proof: First, suppose that f is convex on S . Let us take arbitrary $x, y \in S$ and $\theta \in (0, 1)$. Let's denote $z(\theta) := \theta y + (1 - \theta) x$. Because S is convex, that indicates that $z(\theta) \in S$, for all $\theta \in [0, 1]$.

From convexity of function f follows that

$$f(z(\theta)) \leq \theta f(y) + (1 - \theta) f(x). \quad (2.5)$$

When we add $-f(x)$ to both sides of the previous inequality we get

$$f(z(\theta)) - f(x) \leq \theta f(y) - \theta f(x).$$

Take notice that we can also interpret $z(\theta)$ like $z(\theta) := \theta(y-x) + x$.

If we divide the previous inequality by θ we obtain

$$\frac{f(x + \theta(y-x)) - f(x)}{\theta} \leq f(y) - f(x)$$

If we let $\theta \rightarrow 0^+$ then we obtain the directional derivate on the left side, thus

$$\nabla^T f(x)(y-x) \leq f(y) - f(x)$$

Which is equivalent to (2.4).

Now, let us suppose that (2.4) holds. Taking arbitrary $x, y \in S$, defining $z(\theta)$ as above and applying (2.4) two times we get

$$f(y) \geq f(z(\theta)) + \nabla^T f(z(\theta)) (y - z(\theta)) \quad (2.6)$$

and

$$f(x) \geq f(z(\theta)) + \nabla^T f(z(\theta)) (x - z(\theta)). \quad (2.7)$$

Pay attention that

$$x - z(\theta) = x - (\theta y + (1 - \theta) x) = x - (\theta y + x - \theta x) = x - \theta y - x + \theta x = \theta (x - y) \text{ and}$$

$$y - z(\theta) = y - (\theta y + (1 - \theta) x) = y - (\theta y + x - \theta x) = y - \theta y - x + \theta x = \theta (x - y) + y - x = (y - x) (1 - \theta).$$

If we multiply (2.7) with $1 - \theta$ and (2.6) with θ we would get:

$$\theta f(y) \geq \theta f(z(\theta)) + \nabla^T f(z(\theta)) \theta (y - x) (1 - \theta) \quad (2.8)$$

and

$$(1 - \theta) f(x) \geq (1 - \theta) f(z(\theta)) + \nabla^T f(z(\theta)) (1 - \theta) \theta (x - y) \quad (2.9)$$

Adding (2.8) and (2.9) together, we obtain

$$\theta f(y) + (1 - \theta) f(x) \geq f(z(\theta)) = f(\theta y + (1 - \theta) x)$$

This is exactly (2.5). That is inequality which holds for convex functions, so we prove that f is convex on set S . ■

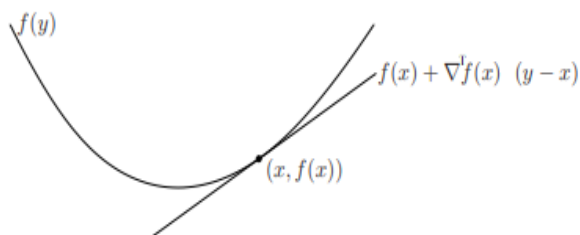


Figure 9 Illustration of inequality (2.4) (source:[1])

The inequality (2.4) highlights that local information about a convex function (such as its value and derivative at a specific point) can be used to derive global information about the function, like constructing a global underestimator. This is one of the key properties of convex functions. For instance, the inequality (2.4) shows that if $\nabla f(x) = 0$, then for all $y \in S, f(y) \geq f(x)$. This means that x is a global minimizer of the function f .

Theorem 2.2.3 Suppose that $S \subseteq \mathbb{R}^n$ is a convex set and $f \in C^1(S)$. If

$$f(y) > f(x) + \nabla^T f(x)(y-x) \quad \forall x, y \in S, x \neq y \quad (2.10)$$

the function f is strictly convex on S .

Proof: Let us take arbitrary $x, y \in S$ and $\theta \in (0,1)$. Form $z(\theta) := \theta y + (1-\theta)x$. From convexity of S follows that $z(\theta) \in S$, for all $\theta \in [0,1]$. Applying (2.9) two times, we get

$$f(y) > f(z(\theta)) + \nabla^T f(z(\theta))(y-z(\theta)) \quad (2.11) \text{ and}$$

$$f(x) > f(z(\theta)) + \nabla^T f(z(\theta))(x-z(\theta)) \quad (2.12)$$

Take notice that

$$x - z(\theta) = x - (\theta y + (1-\theta)x) = x - (\theta y + x - \theta x) = x - \theta y - x + \theta x = \theta(x-y) \text{ and}$$

$$y - z(\theta) = y - (\theta y + (1-\theta)x) = y - (\theta y + x - \theta x) = y - \theta y - x + \theta x = \theta(x-y) + y-x = (y-x)(1-\theta).$$

If we multiply (2.12) with $1-\theta$ and (2.11) with θ we would obtain:

$$\theta f(y) > \theta f(z(\theta)) + \nabla^T f(z(\theta)) \theta (y-x)(1-\theta) \quad (2.13)$$

and

$$(1-\theta)f(x) > (1-\theta)f(z(\theta)) + \nabla^T f(z(\theta))(1-\theta)\theta(x-y) \quad (2.14)$$

Adding (2.13) and (2.14) together, we get

$$\theta f(y) + (1-\theta)f(x) > f(z(\theta)) = f(\theta y + (1-\theta)x),$$

so, it follows that f is strictly convex on set S . ■

2.2.5 Second-order methods

Theorem 2.2.4 Suppose that $S \subseteq \mathbb{R}^n$ is a convex set and $f \in C^2(S)$. Then, the next statements hold.

1. If $\nabla^2 f(x) \geq 0$ (Hessian matrix is positive semi-definite) for every $x \in S$, then f is convex on S .
2. If $\nabla^2 f(x) > 0$ (Hessian matrix is positive-definite) for every $x \in S$, then f is strictly convex on S .
3. If S is open and f is convex on S , then $\nabla^2 f(x) \geq 0$ for every $x \in S$.

Proof:

1. Assume that $\nabla^2 f(x) \geq 0, \forall x \in S$. Let's take arbitrary $x, y \in S$. From Taylor's expansion follows that there exists $z \in S$ such that

$$f(y) = f(x) + \nabla^T f(x)(y-x) + \frac{1}{2}(y-x)^T \nabla^2 f(z)(y-x)$$

Because $\nabla^2 f(z) \geq 0$, follows that $f(y) \geq f(x) + \nabla^T f(x)(y-x)$, so f is convex on S , by Theorem 2.2.2.

2. Assume that $\nabla^2 f(x) > 0, \forall x \in S$. Let's take arbitrary $x, y \in S$. From Taylor's expansion follows that there exists $z \in S$ such that

$$f(y) = f(x) + \nabla^T f(x)(y-x) + \frac{1}{2}(y-x)^T \nabla^2 f(z)(y-x)$$

Because $\nabla^2 f(z) > 0$, follows that $f(y) > f(x) + \nabla^T f(x)(y-x)$. Theorem 2.2.3 implies that the function f is strictly convex on S .

3. Assume that S is open set and f is convex function on S . Let's take arbitrary $x \in S$ and arbitrary $d \in \mathbb{R}^n$. Because S is open, there exists $h' > 0$ such that $x + hd$ remains in S for every $0 \leq h \leq h'$. Further,

$$f(x+hd) = f(x) + h\nabla^T f(x)d + \frac{1}{2}h^2 d^T \nabla^2 f(x)d + o(\|hd\|^2)$$

Since it is assumed that f is convex function, then from Theorem 2.2.2 follows that

$$f(x+hd) \geq f(x) + h\nabla^T f(x)d, \text{ so}$$

$$\frac{1}{2}h^2 d^T \nabla^2 f(x)d + o(\|hd\|^2) \geq 0$$

This holds for any h small enough, so if we divide the previous inequality by h^2 and letting $h \rightarrow 0$, we obtain that $d^T \nabla^2 f(x)d \geq 0$. Because the vector d was arbitrary taken, it follows

that $\nabla^2 f(x) \geq 0$. Eventually, since the point x was arbitrary taken from S , we come to the conclusion that $\nabla^2 f(x), \forall x \in S$. ■

Pay attention:

- If we assume that f is twice continuously differentiable and $\text{dom } f$ (set S) is convex and open then f is convex if and only if its Hessian is positive semidefinite, i.e. for all $x \in \text{dom } f$, $\nabla^2 f(x) \geq 0$.
- If we assume that f is twice continuously differentiable and $\text{dom } f$ (set S) is convex and open then f is concave if and only if its Hessian is negative semidefinite, i.e. for all $x \in \text{dom } f$, $\nabla^2 f(x) \leq 0$.

From the geometric aspect the condition $\nabla^2 f(x) \geq 0$ can be interpreted as the requirement that the graph of the function have positive (upward) curvature at x .

Theorem 2.2.5 Suppose that S is convex set and f is convex function on S . Then every local minimizer of the function f is also the global minimizer.

Proof:

This theorem will be proved by contradiction. Assume that x^* is local minimizer, but not global minimizer of f . So, there exists y^* such that $f(y^*) < f(x^*)$. From convexity we have that for any $\theta \in (0,1)$ there holds following:

$$f(x^* + \theta(y^* - x^*)) = f(\theta y^* + (1 - \theta)x^*) \leq \theta f(y^*) + (1 - \theta)f(x^*) < \theta f(x^*) + (1 - \theta)f(x^*) = f(x^*)$$

So, we can always find small enough θ , i.e., there can be always founded a point $z = x^* + \theta(y^* - x^*)$ in arbitrary small vicinity of the point x^* such that $f(z) < f(x^*)$. This is contradiction with the assumption that x^* is local minimizer. So, x^* is also the global minimizer. ■

Definition 2.2.2 A function f is **strongly convex** with parameter $M > 0$ on a convex set S if for any $x, y \in S$ and any $\theta \in [0,1]$ there holds following:

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) - \frac{M}{2}\theta(1 - \theta)\|x - y\|^2$$

The parameter M from the previous definition is often called strong convexity parameter.

-If the function is differentiable on S then the function f is strongly convex with parameter $M > 0$ on a convex set S if the following holds:

$$f(y) \geq f(x) + \nabla^T f(x) (y-x) + \frac{M}{2} \|x - y\|^2, \forall x, y \in S.$$

-In case when $f \in C^2(S)$ the function is strongly convex on S with parameter $M > 0$ if $\nabla^2 f(x) \geq MI$, $\forall x \in S$.

*Notice that if function f is strongly convex with parameter $M > 0$ on a convex set S then f is also strictly convex on S because following holds:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) - \frac{M}{2} \theta(1-\theta) \|x - y\|^2 < \theta f(x) + (1-\theta)f(y).$$

Definition 2.2.3 A function f is **quasi-convex** on a convex set S if for any $x, y \in S$ and any $\theta \in [0, 1]$ there holds following:

$$f(\theta x + (1-\theta)y) \leq \max\{f(x), f(y)\}.$$

*Pay attention that every convex function is also quasi-convex:

Let's suppose that f is convex function and $f(x) > f(y)$, so $\max\{f(x), f(y)\} = f(x)$, then:

$$\begin{aligned} f(\theta x + (1-\theta)y) &\leq \theta f(x) + (1-\theta)f(y) < \text{(This inequality holds because we assumed that } f \text{ is convex)} \\ &< \theta f(x) + (1-\theta)f(x) = \text{(This follows from assumption that } f(x) > f(y)) \\ &= f(x) = \max\{f(x), f(y)\}, \end{aligned}$$

this means that definition of quasi-convex function is satisfied.

On the other hand, concave function can be quasi-convex. ((for example, $\ln(x)$)).

Definition 2.2.4 A function f is **quasi-concave** on a convex set S if for any $x, y \in S$ and any $\theta \in [0, 1]$ there holds following:

$$f(\theta x + (1-\theta)y) \geq \min\{f(x), f(y)\}.$$

Pay attention that a monotone function is both quasi-convex and quasi-concave.

2.2.6 Epigraph

Definition 2.2.5 The graph of a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is interpreted as

$$\{(x, f(x)) \mid x \in \text{dom } f\},$$

which is a subset of \mathbb{R}^{n+1} .

Definition 2.2.6 The epigraph ('Epi' means 'above', thus epigraph implies 'above the graph'.) of a function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is interpreted as

$$\text{epi } f = \{(x, t) \mid x \in \text{dom } f, f(x) \leq t\},$$

which is a subset of \mathbb{R}^{n+1} .

The definition is shown in following figure.

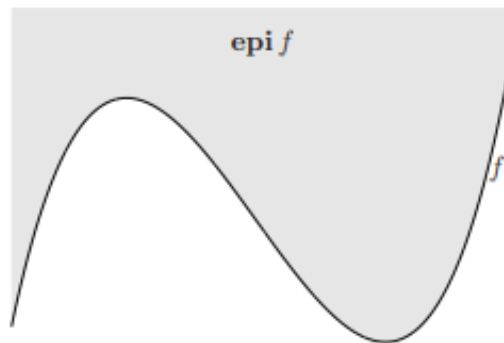


Figure 10 Epigraph of function f (source:[1])

Epigraph is illustrated shaded.

The lower boundary is the graph of function f .

The connection between convex sets and convex functions is via the epigraph.

Theorem 2.2.6 A function is convex if and only if its epigraph is a convex set.

Proof:

(\Rightarrow) Let's suppose that f is convex function and $(x_1, t_1), (x_2, t_2)$ belong to $epi f$. (of course $x_1, x_2 \in \text{dom } f$). We need to show that $f(\theta x_1 + (1 - \theta) x_2) \leq \theta t_1 + (1 - \theta) t_2, \forall \theta \in [0, 1]$, i.e. we will prove that line segment joining $(x_1, t_1), (x_2, t_2)$ belongs to $epi f$.

$$\begin{aligned} f(\theta x_1 + (1 - \theta) x_2) &\leq \theta f(x_1) + (1 - \theta) f(x_2) \leq \text{(This follows from convexity of } f) \\ &\leq \theta t_1 + (1 - \theta) t_2 \text{ (This follows from definition of } epi f \text{ as } f(x_1) \leq t_1 \text{ and } f(x_2) \leq t_2) \end{aligned}$$

So, we showed requested.

(\Leftarrow) Let's assume that $epi f$ is convex. Consider $(x_1, f(x_1)), (x_2, f(x_2))$. Clearly, we have $(x_1, f(x_1)), (x_2, f(x_2)) \in epi f$. Because $epi f$ is convex, the line segment joining $(x_1, f(x_1)), (x_2, f(x_2))$ belongs to $epi f$, i.e.

$$(\theta x_1 + (1 - \theta) x_2, \theta f(x_1) + (1 - \theta) f(x_2)) \in epi f.$$

By definition of epigraph it follows:

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

So, f is convex function. ■

2.2.7 Examples of convex and concave functions

Powers

$f(x) = x^a$ is convex on the interval $0 < x < \infty$ when $a \geq 1$ or $a \leq 0$. ($f''(x) = a(a-1)x^{a-2} \geq 0$ for $a \geq 1$ or $a \leq 0$)

Logarithm

$f(x) = \log x$ is concave on the interval $0 < x < \infty$. ($f''(x) = -\frac{1}{x^2} < 0$).

Exponential functions

Function $f(x) = e^{ax}$ is convex on \mathbb{R} ($f''(x) = a^2 e^{ax} \geq 0, \forall a, x \in \mathbb{R}$).

Norms

If $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is a norm, and $0 \leq \theta \leq 1$, then

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

From the triangle inequality of norms follows inequality, and the equality follows from homogeneity property of a norms.

Affine functions

$$f(x) = a^T x + b \text{ (for any } a \in \mathbb{R}^n, b \in \mathbb{R}\text{)}.$$

They are both convex and concave functions:

$$f(\theta x + (1 - \theta) y) = a^T (\theta x + (1 - \theta) y) + b = \theta a^T x + (1 - \theta) a^T y + \theta b + (1 - \theta) b = \theta f(x) + (1 - \theta) f(y), \forall \theta \in [0, 1].$$

Quadratic functions

Let's assume that $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is quadratic function given by $f(x) = \frac{1}{2}x^T P x + q^T x + r$, with $P \in S^n$, $q \in \mathbb{R}^n$, and $r \in \mathbb{R}$. Since $\nabla^2 f(x) = P \forall x$, f is convex if and only if $P \geq 0$ (and concave if and only if $P \leq 0$).

Quadratic-over-linear function

The function $f(x, y) = \frac{x^2}{y}$, with $dom f = \{(x, y) \in \mathbb{R}^2 \mid y > 0\}$, is convex because

$$\nabla^2 f(x, y) = \frac{2}{y^3} \begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix} = \frac{2}{y^3} \begin{bmatrix} y \\ -x \end{bmatrix} \begin{bmatrix} y \\ -x \end{bmatrix}^T \geq 0.$$

Max function

The function $f(x) = \max_i x_i$ is convex because it satisfies following:

$$f(\theta x + (1 - \theta) y) = \max_i (\theta x_i + (1 - \theta) y_i) \leq \theta \max_i x_i + (1 - \theta) \max_i y_i = \theta f(x) + (1 - \theta) f(y), \forall \theta \in [0, 1].$$

Geometric mean

The geometric mean $f(x) = \left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}}$ is concave on $\text{dom } f = \mathbb{R}^n_{++} (x_i > 0)$. It's Hessian matrix $\nabla^2 f(x)$ is given by:

$$\frac{\partial^2 f(x)}{\partial x_k^2} = - (n-1) \frac{\left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}}}{n^2 x_k^2}, \quad \frac{\partial^2 f(x)}{\partial x_k \partial x_j} = \frac{\left(\prod_{i=1}^n x_i\right)^{\frac{1}{n}}}{n^2 x_k x_j} \text{ for } k \neq j, \text{ and may be interpreted as:}$$

$$\nabla^2 f(x) = - \frac{\prod_{i=1}^n x_i^{\frac{1}{n}}}{n^2} \left(n \cdot \text{diag} \left(\frac{1}{x_1^2}, \dots, \frac{1}{x_n^2} \right) - qq^T \right) \text{ where } q_i = \frac{1}{x_i}.$$

$$v^T \nabla^2 f(x) v = - \frac{\prod_{i=1}^n x_i^{\frac{1}{n}}}{n^2} \left(n \cdot \sum_{i=1}^n \frac{v_i^2}{x_i^2} - \left(\sum_{i=1}^n \frac{v_i}{x_i} \right)^2 \right) \leq 0 \text{ for all } v. \text{ This follows from the Cauchy-}$$

Schwarz inequality $(a^T a)(b^T b) \geq (a^T b)^2$, applied to the vectors $a = \mathbf{1}$ and $b_i = \frac{v_i}{x_i}$.

2.2.8 Operations that preserve convexity of functions

Nonnegative scaling

Theorem 2.2.7 Assume that S is convex set. If $f: S \rightarrow \mathbb{R}$ is convex and $\alpha \geq 0$ then function αf is also convex.

Proof:

Let's suppose that $\theta \in [0,1]$ and $x, y \in S$. Let's define $F(x) := \alpha f(x)$. Domain of $F(x)$ is αS , which is convex, because scaling preserve convexity of sets.

$$\begin{aligned} F(\theta x + (1-\theta)y) &= \alpha f(\theta x + (1-\theta)y) \leq \\ &\leq \alpha [\theta f(x) + (1-\theta)f(y)] = \text{(This is because of assumption that } f \text{ is convex function)} \\ &= \theta \alpha f(x) + (1-\theta) \alpha f(y) = \\ &= \theta F(x) + (1-\theta) F(y) \end{aligned}$$

So, from definition it follows that F is convex because $F(\theta x + (1-\theta)y) \leq \theta F(x) + (1-\theta) F(y)$ and its domain is convex set. ■

Addition

Theorem 2.2.8 Let f_1 and f_2 be convex functions on convex set S . Then sum $f = f_1 + f_2$ is convex function on S .

Proof:

Let's suppose that $\theta \in [0,1]$ and $x, y \in S$.

$$\begin{aligned} f(\theta x + (1-\theta)y) &= (f_1 + f_2)(\theta x + (1-\theta)y) = \\ &= f_1(\theta x + (1-\theta)y) + f_2(\theta x + (1-\theta)y) \leq \\ &\leq \theta f_1(x) + (1-\theta)f_1(y) + \theta f_2(x) + (1-\theta)f_2(y) = \\ &= \theta (f_1 + f_2)(x) + (1-\theta)(f_1 + f_2)(y) = \\ &= \theta f(x) + (1-\theta)f(y) \end{aligned} \quad \text{(This follows from assumption that } f_1 \text{ and } f_2 \text{ are convex functions)}$$

By definition it follows that f is convex function on S . ■

Nonnegative weighted sum

Theorem 2.2.9 Let's suppose that S is convex set, f_1, \dots, f_m are convex functions on S and $\mu_1, \dots, \mu_m \geq 0$. Then function $f = \mu_1 f_1 + \dots + \mu_m f_m$ is convex on S .

Proof:

This will be proved by induction.

Basis of induction: For $m=1$ $f = \mu_1 f_1$ is convex function because we know that nonnegative scaling preserve convexity.

Hypothesis of induction: Let's assume that claim is valid for m .

Induction step: Let's show that claim is valid for $m+1$.

From hypothesis we know that $\mu_1 f_1 + \dots + \mu_m f_m$ is convex. Since nonnegative scaling preserve convexity $\mu_{m+1} f_{m+1}$ is convex. So, function $f = \mu_1 f_1 + \dots + \mu_m f_m + \mu_{m+1} f_{m+1}$ is convex because it is sum of two convex functions (Addition preserve convexity). ■

Also, a nonnegative weighted sum of concave functions is concave. A nonnegative, nonzero weighted sum of strictly convex (strictly concave) functions is strictly convex (strictly concave).

Composition with an affine mapping

Theorem 2.2.10 Assume that $f: \mathbb{R}^n \rightarrow \mathbb{R}$, $A \in \mathbb{R}^{n \times m}$, and $b \in \mathbb{R}^n$. Define $g: \mathbb{R}^m \rightarrow \mathbb{R}$ by $g(x) = f(Ax + b)$, with $\text{dom } g = \{x \mid Ax + b \in \text{dom } f\}$. Then if f is convex, it follows that g is convex; if f is concave, it follows that g is concave.

Proof:

Let's suppose that $\theta \in [0,1]$ and $x, y \in \text{dom } g$.

Case 1: f is convex function

$$\begin{aligned} g(\theta x + (1-\theta)y) &= f(A(\theta x + (1-\theta)y) + b) = \text{(This follows by definition of } g\text{)} \\ &= f(\theta Ax + (1-\theta)Ay + b + \theta b - \theta b) = \\ &= f(\theta(Ax + b) + (1-\theta)(Ay + b)) \leq \text{(This follows by assumption that } f \text{ is convex)} \\ &\leq \theta f(Ax + b) + (1-\theta)f(Ay + b) = \\ &= \theta g(x) + (1-\theta)g(y). \end{aligned}$$

So, $g(\theta x + (1-\theta)y) \leq \theta g(x) + (1-\theta)g(y)$. By definition it follows that g is convex.

Case 2: f is concave function

$$\begin{aligned} g(\theta x + (1-\theta)y) &= f(A(\theta x + (1-\theta)y) + b) = \text{(This follows by definition of } g\text{)} \\ &= f(\theta Ax + (1-\theta)Ay + b + \theta b - \theta b) = \\ &= f(\theta(Ax + b) + (1-\theta)(Ay + b)) \leq \text{(This follows by assumption that } f \text{ is concave)} \\ &\geq \theta f(Ax + b) + (1-\theta)f(Ay + b) = \\ &= \theta g(x) + (1-\theta)g(y). \end{aligned}$$

So, $g(\theta x + (1-\theta)y) \geq \theta g(x) + (1-\theta)g(y)$. By definition it follows that g is concave. ■

Pointwise maximum

Theorem 2.2.11 Let's assume that f_1 and f_2 are convex functions. Then, their pointwise maximum defined as: $f(x) = \max \{f_1(x), f_2(x)\}$ with $\text{dom } f = \text{dom } f_1 \cap \text{dom } f_2$ is also convex function.

Proof:

Let's suppose that $\theta \in [0,1]$ and $x, y \in \text{dom } f$.

$$\begin{aligned} f(\theta x + (1-\theta)y) &= \max \{f_1(\theta x + (1-\theta)y), f_2(\theta x + (1-\theta)y)\} \leq && \text{(This follows from assumption that } f_1 \text{ and } \\ &\leq \max \{\theta f_1(x) + (1-\theta)f_1(y), \theta f_2(x) + (1-\theta)f_2(y)\} \leq && f_2 \text{ are convex functions)} \\ &\leq \theta \max \{f_1(x), f_2(x)\} + (1-\theta) \max \{f_1(y), f_2(y)\} = \\ &= \theta f(x) + (1-\theta)f(y) \end{aligned}$$

So, we have $f(\theta x + (1 - \theta) y) \leq \theta f(x) + (1 - \theta) f(y)$. By definition, it follows that f is convex function. ■

Similarly, if f_1, f_2, \dots, f_m are convex functions, then their pointwise maximum defined as: $f(x) = \max \{f_1(x), f_2(x), \dots, f_m(x)\}$ with $\text{dom } f = \text{dom } f_1 \cap \text{dom } f_2 \cap \dots \cap \text{dom } f_m$ is also convex function.

The pointwise maximum property may be expanded to the pointwise supremum over an infinite set of convex functions.

Composition

Here we will look into conditions on $h: \mathbb{R}^k \rightarrow \mathbb{R}$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}^k$ that guarantee convexity or concavity of their composition $f = h \circ g: \mathbb{R}^n \rightarrow \mathbb{R}$, defined by $f(x) = h(g(x))$, $\text{dom } f = \{x \in \text{dom } g \mid g(x) \in \text{dom } h\}$.

Scalar composition

Theorem 2.2.12 Let's suppose that $h: \mathbb{R} \rightarrow \mathbb{R}$ and $g: \mathbb{R} \rightarrow \mathbb{R}$ (case when $k = 1$ and $n = 1$). Further assume that h and g are twice differentiable functions. Composition of these functions f is defined as above. Then:

- a) if h is convex and nondecreasing and g is convex, it follows that f is convex
- b) if h is convex and nonincreasing, and g is concave, it follows that f is convex
- c) if h is concave and nondecreasing, and g is concave, it follows that f is concave
- d) if h is concave and nonincreasing, and g is convex, it follows that f is concave.

Proof:

In this case, f is convex (concave) if and only if $f''(x) \geq 0$ ($f''(x) \leq 0$) for all $x \in \mathbb{R}$.

The second derivative of the composition function f is given by:

$$f''(x) = h''(g(x)) g'(x)^2 + h'(g(x)) g''(x)$$

- a) When h is convex on \mathbb{R} we know that $h''(x) \geq 0 \forall x \in \mathbb{R}$.

When h is nondecreasing we know that $h'(x) \geq 0 \forall x \in \mathbb{R}$.

When g is convex on \mathbb{R} we know that $g''(x) \geq 0 \forall x \in \mathbb{R}$.

So, $f''(x) = \underbrace{h''(g(x))}_{\geq 0} \underbrace{g'(x)^2}_{\geq 0} + \underbrace{h'(g(x))}_{\geq 0} \underbrace{g''(x)}_{\geq 0} \geq 0 \forall x \in \mathbb{R}$. Thus, f is convex function on \mathbb{R} .

- b) When h is convex on \mathbb{R} we know that $h''(x) \geq 0 \forall x \in \mathbb{R}$.

When h is nonincreasing we know that $h'(x) \leq 0 \forall x \in \mathbb{R}$.

When g is concave on \mathbb{R} we know that $g''(x) \leq 0 \forall x \in \mathbb{R}$.

$$\text{So, } f''(x) = \underbrace{h''(g(x))}_{\geq 0} \underbrace{g'(x)^2}_{\geq 0} + \underbrace{h'(g(x))}_{\leq 0} \underbrace{g''(x)}_{\leq 0} \geq 0 \forall x \in \mathbb{R}. \text{ Thus, } f \text{ is convex function on } \mathbb{R}.$$

c) When h is concave on \mathbb{R} we know that $h''(x) \leq 0 \forall x \in \mathbb{R}$.

When h is nondecreasing we know that $h'(x) \geq 0 \forall x \in \mathbb{R}$.

When g is concave on \mathbb{R} we know that $g''(x) \leq 0 \forall x \in \mathbb{R}$.

$$\text{So, } f''(x) = \underbrace{h''(g(x))}_{\leq 0} \underbrace{g'(x)^2}_{\geq 0} + \underbrace{h'(g(x))}_{\geq 0} \underbrace{g''(x)}_{\leq 0} \leq 0 \forall x \in \mathbb{R}. \text{ Thus, } f \text{ is concave function on } \mathbb{R}.$$

d) When h is concave on \mathbb{R} we know that $h''(x) \leq 0 \forall x \in \mathbb{R}$.

When h is nonincreasing we know that $h'(x) \leq 0 \forall x \in \mathbb{R}$.

When g is convex on \mathbb{R} we know that $g''(x) \geq 0 \forall x \in \mathbb{R}$.

$$\text{So, } f''(x) = \underbrace{h''(g(x))}_{\leq 0} \underbrace{g'(x)^2}_{\geq 0} + \underbrace{h'(g(x))}_{\leq 0} \underbrace{g''(x)}_{\geq 0} \leq 0 \forall x \in \mathbb{R}. \text{ Thus, } f \text{ is concave function on } \mathbb{R}.$$

■

Theorem 2.2.13 Let's suppose that $h: \mathbb{R} \rightarrow \mathbb{R}$ and $g: \mathbb{R}^n \rightarrow \mathbb{R}$ (case when $k = 1$ and $n > 1$). Composition of these functions f is defined as before. Then:

- a) if h is convex, h^* is nondecreasing, and g is convex, it follows that f is convex
- b) if h is convex, h^* is nonincreasing, and g is concave, it follows that f is convex
- c) if h is concave, h^* is nondecreasing, and g is concave, it follows that f is concave
- d) if h is concave, h^* is nonincreasing, and g is convex, it follows that f is concave.

In this case h^* denotes the extended-value extension of the function h , which assigns the value ∞ ($-\infty$) to points not in $\text{dom } h$ for h convex (concave).

Proof:

Let's suppose that $x, y \in \text{dom } f$ and $\theta \in [0,1]$. Since, $x, y \in \text{dom } f$ it follows that $x, y \in \text{dom } g$ and $g(x), g(y) \in \text{dom } h$. Since $\text{dom } g$ is convex set by definition it follows that $\theta x + (1 - \theta) y \in \text{dom } g$.

a) From convexity of g it follows that:

$$g(\theta x + (1 - \theta) y) \leq \theta g(x) + (1 - \theta) g(y) \quad (2.15)$$

Because $\text{dom } h$ is convex set, we can conclude that $\theta g(x) + (1-\theta) g(y) \in \text{dom } h$.

Since h^* is nondecreasing we know that for any $x_1, x_2 \in \mathbb{R}$, with $x_1 < x_2$, we have $h^*(x_1) \leq h^*(x_2)$. On the other hand, this means that if $x_2 \in \text{dom } h$, then $x_1 \in \text{dom } h$.

So, from this it follows that $g(\theta x + (1-\theta) y) \in \text{dom } h$. Thus, $\theta x + (1-\theta) y \in \text{dom } f$. So, we have shown that $\text{dom } f$ is convex set.

Now, combining (2.15) and fact that h^* is nondecreasing we obtain:

$$h(g(\theta x + (1-\theta) y)) \leq h(\theta g(x) + (1-\theta) g(y))$$

From convexity of h we know that $h(\theta g(x) + (1-\theta) g(y)) \leq \theta h(g(x)) + (1-\theta) h(g(y))$.

Thus, $h(g(\theta x + (1-\theta) y)) \leq \theta h(g(x)) + (1-\theta) h(g(y))$.

Furthermore, using definition of f we obtain: $f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y)$. So, it follows that f is convex function.

b) From concavity of g it follows that:

$$g(\theta x + (1-\theta) y) \geq \theta g(x) + (1-\theta) g(y) \quad (2.16)$$

Because $\text{dom } h$ is convex set we can conclude that $\theta g(x) + (1-\theta) g(y) \in \text{dom } h$.

Since h^* is nonincreasing we know that for any $x_1, x_2 \in \mathbb{R}$, with $x_1 < x_2$, we have $h^*(x_1) \geq h^*(x_2)$. On the other hand, this means that if $x_1 \in \text{dom } h$, then $x_2 \in \text{dom } h$.

So, from this it follows that $g(\theta x + (1-\theta) y) \in \text{dom } h$. Thus, $\theta x + (1-\theta) y \in \text{dom } f$. So, we have shown that $\text{dom } f$ is convex set.

Now, combining (2.16) and fact that h^* is nonincreasing we obtain:

$$h(g(\theta x + (1-\theta) y)) \leq h(\theta g(x) + (1-\theta) g(y))$$

From convexity of h we know that $h(\theta g(x) + (1-\theta) g(y)) \leq \theta h(g(x)) + (1-\theta) h(g(y))$.

Thus, $h(g(\theta x + (1-\theta) y)) \leq \theta h(g(x)) + (1-\theta) h(g(y))$.

Furthermore, using definition of f we obtain: $f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y)$. So, it follows that f is convex function.

c) From concavity of g it follows that:

$$g(\theta x + (1-\theta) y) \geq \theta g(x) + (1-\theta) g(y) \quad (2.16)$$

Because $\text{dom } h$ is convex set we can conclude that $\theta g(x) + (1-\theta) g(y) \in \text{dom } h$.

Since h^* is nondecreasing we know that for any $x_1, x_2 \in \mathbb{R}$, with $x_1 < x_2$, we have $h^*(x_1) \leq h^*(x_2)$. On the other hand, this means that if $x_1 \in \text{dom } h$, then $x_2 \in \text{dom } h$.

So, from this it follows that $g(\theta x + (1-\theta) y) \in \text{dom } h$. Thus, $\theta x + (1-\theta) y \in \text{dom } f$. So, we have shown that $\text{dom } f$ is convex set.

Now, combining (2.16) and fact that h^* is nondecreasing we obtain:

$$h(g(\theta x + (1-\theta)y)) \geq h(\theta g(x) + (1-\theta)g(y))$$

From concavity of h we know that $h(\theta g(x) + (1-\theta)g(y)) \geq \theta h(g(x)) + (1-\theta)h(g(y))$.

Thus, $h(g(\theta x + (1-\theta)y)) \geq \theta h(g(x)) + (1-\theta)h(g(y))$.

Furthermore, using definition of f we obtain: $f(\theta x + (1-\theta)y) \geq \theta f(x) + (1-\theta)f(y)$. So, it follows that f is concave function.

d) From convexity of g it follows that:

$$g(\theta x + (1-\theta)y) \leq \theta g(x) + (1-\theta)g(y) \quad (2.15)$$

Because $\text{dom } h$ is convex set we can conclude that $\theta g(x) + (1-\theta)g(y) \in \text{dom } h$.

Since h^* is nonincreasing we know that for any $x_1, x_2 \in \mathbb{R}$, with $x_1 < x_2$, we have $h^*(x_1) \geq h^*(x_2)$. On the other hand, this means that if $x_2 \in \text{dom } h$, then $x_1 \in \text{dom } h$.

So, from this it follows that $g(\theta x + (1-\theta)y) \in \text{dom } h$. Thus, $\theta x + (1-\theta)y \in \text{dom } f$. So, we have shown that $\text{dom } f$ is convex set.

Now, combining (2.15) and fact that h^* is nonincreasing we obtain:

$$h(g(\theta x + (1-\theta)y)) \geq h(\theta g(x) + (1-\theta)g(y))$$

From concavity of h we know that $h(\theta g(x) + (1-\theta)g(y)) \geq \theta h(g(x)) + (1-\theta)h(g(y))$.

Thus, $h(g(\theta x + (1-\theta)y)) \geq \theta h(g(x)) + (1-\theta)h(g(y))$.

Furthermore, using definition of f we obtain: $f(\theta x + (1-\theta)y) \geq \theta f(x) + (1-\theta)f(y)$. So, it follows that f is concave function.

■

Vector composition

We now back to the case when $k \geq 1$. Let's suppose that

$$f(x) = h(g(x)) = h(g_1(x), \dots, g_k(x)), \text{ with } h: \mathbb{R}^k \rightarrow \mathbb{R} \text{ and } g_i: \mathbb{R}^n \rightarrow \mathbb{R}.$$

Theorem 2.2.14 Let's assume that $h: \mathbb{R}^k \rightarrow \mathbb{R}$ and $g_i: \mathbb{R} \rightarrow \mathbb{R}$ (case when $n = 1$). Further assume that h and g are twice differentiable functions. Composition of these functions f is defined as above. Then:

- a) if h is convex, h is nondecreasing in each argument, and g_i are convex, it follows that f is convex
- b) if h is convex, h is nonincreasing in each argument, and g_i are concave, it follows that f is convex
- c) if h is concave, h is nondecreasing in each argument, and g_i are concave, it follows that f is concave.

Proof:

In this case, f is convex (concave) if and only if $f''(x) \geq 0$ ($f''(x) \leq 0$) for all $x \in \text{dom } f$.

The second derivative of the composition function f is given by:

$$f''(x) = g'(x)^T \nabla^2 h(g(x)) g'(x) + \nabla h(g(x))^T g''(x)$$

a) By convexity of h , we know that its Hessian matrix is positive semidefinite at each point from $\text{dom } h$, i.e. $\nabla^2 h(x) \geq 0, \forall x \in \text{dom } h$.

Because h is nondecreasing its gradient is nonnegative at every point from its domain, i.e. $\nabla h(x) \geq 0, \forall x \in \text{dom } h$.

Also, from assumption that g_i are convex it follows that $g_i''(x) \geq 0$ for all $x \in \mathbb{R}$. So, $g''(x) \geq 0$ for all $x \in \mathbb{R}$.

To conclude,

$$f''(x) = \underbrace{g'(x)^T \nabla^2 h(g(x)) g'(x)}_{\geq 0} + \underbrace{\nabla h(g(x))}_{\geq 0} \underbrace{T g''(x)}_{\geq 0} \geq 0, \text{ so } f \text{ is convex } \forall x \in \text{dom } f$$

b) By convexity of h , we know that its Hessian matrix is positive semidefinite at each point from $\text{dom } h$, i.e. $\nabla^2 h(x) \geq 0, \forall x \in \text{dom } h$.

Because h is nonincreasing its gradient is nonpositive at every point from its domain, i.e. $\nabla h(x) \leq 0, \forall x \in \text{dom } h$.

Also, from assumption that g_i are concave it follows that $g_i''(x) \leq 0$ for all $x \in \mathbb{R}$. So, $g''(x) \leq 0$ for all $x \in \mathbb{R}$.

To conclude,

$$f''(x) = \underbrace{g'(x)^T \nabla^2 h(g(x)) g'(x)}_{\geq 0} + \underbrace{\nabla h(g(x))}_{\leq 0} \underbrace{T g''(x)}_{\leq 0} \geq 0, \text{ so } f \text{ is convex } \forall x \in \text{dom } f$$

c) By concavity of h , we know that its Hessian matrix is negative semidefinite at each point from $\text{dom } h$, i.e. $\nabla^2 h(x) \leq 0, \forall x \in \text{dom } h$.

Because h is nondecreasing its gradient is nonnegative at every point from its domain, i.e. $\nabla h(x) \geq 0, \forall x \in \text{dom } h$.

Also, from assumption that g_i are concave it follows that $g_i''(x) \leq 0$ for all $x \in \mathbb{R}$. So, $g''(x) \leq 0$ for all $x \in \mathbb{R}$.

To conclude,

$$f''(x) = \underbrace{g'(x)^T \nabla^2 h(g(x)) g'(x)}_{\leq 0} + \underbrace{\nabla h(g(x))}_{\geq 0} \underbrace{T g''(x)}_{\leq 0} \leq 0, \text{ so } f \text{ is concave } \forall x \in \text{dom } f$$

■

Theorem 2.2.15 Let's suppose that $h: \mathbb{R}^k \rightarrow \mathbb{R}$ and $g_i: \mathbb{R}^n \rightarrow \mathbb{R}$ (case when $n > 1$). Composition of these functions f is defined as before. Then:

- a) if h is convex, h^* is nondecreasing in each argument, and g_i are convex, it follows that f is convex
- b) if h is convex, h^* is nonincreasing in each argument, and g_i are concave, it follows that f is concave
- c) if h is concave, h^* is nondecreasing in each argument, and g_i are concave, it follows that f is concave.

This claim may be proved similarly as Theorem 2.2.13. In order to understand the meaning of the condition that h^* be monotonic, let's consider the case where $h: \mathbb{R}^k \rightarrow \mathbb{R}$ is convex, and h^* nondecreasing, i.e., whenever $a \leq b$, we have $h^*(a) \leq h^*(b)$. From this it follows that if $a \in \text{dom } h$, b is also in $\text{dom } h$. To conclude, the domain of h must extend infinitely in the $-\mathbb{R}_+^k$ directions. We can express this as $\text{dom } h^* - \mathbb{R}_+^k = \text{dom } h$.

Minimization

Theorem 2.2.16 Let's suppose that f is convex in (x, y) and S is a convex nonempty set. Then the function

$$g(x) = \inf_{y \in S} f(x, y)$$

is convex in x , provided $g(x) > -\infty$ for every x . The domain of g is the projection of $\text{dom } f$ on its x -coordinates, i.e., $\text{dom } g = \{x \mid (x, y) \in \text{dom } f \text{ for some } y \in S\}$.

Proof:

This will be proved using Jensen's inequality for $x_1, x_2 \in \text{dom } g$. Let $\lambda > 0$. Then there are $y_1, y_2 \in S$ such that $f(x_i, y_i) \leq g(x_i) + \lambda$ for $i = 1, 2$. Now let $\theta \in [0, 1]$. From convexity of set S it follows that $\theta y_1 + (1 - \theta) y_2 \in S$

$$\begin{aligned} g(\theta x_1 + (1 - \theta) x_2) &= \inf_{y \in S} f(\theta x_1 + (1 - \theta) x_2, y) \leq \\ &\leq f(\theta x_1 + (1 - \theta) x_2, \theta y_1 + (1 - \theta) y_2) \leq \text{(This follows by definition of infimum)} \\ &\leq \theta f(x_1, y_1) + (1 - \theta) f(x_2, y_2) \leq \text{(This follows because } f \text{ is convex function)} \\ &\leq \theta g(x_1) + (1 - \theta) g(x_2) + \lambda \end{aligned}$$

This holds for any $\lambda > 0$, so we get $g(\theta x_1 + (1 - \theta) x_2) \leq \theta g(x_1) + (1 - \theta) g(x_2)$. By definition it follows that g is convex function. ■

Perspective of a function

Definition 2.2.7 Let's assume that $f: \mathbb{R}^n \rightarrow \mathbb{R}$. The perspective of f is the function $g: \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ defined by

$$g(x, t) = t f\left(\frac{x}{t}\right),$$

with domain $\text{dom } g = \{(x, t) \mid \frac{x}{t} \in \text{dom } f, t > 0\}$.

Theorem 2.2.17 Let's assume that f is convex function, then its perspective function g is also convex.

Proof:

$$\begin{aligned} (x, t, s) \in \text{epi } g &\Leftrightarrow g(x, t) \leq s \text{ (Definition of epigraph)} \\ &\Leftrightarrow t f\left(\frac{x}{t}\right) \leq s \text{ (Definition of perspective of a function)} \\ &\Leftrightarrow f\left(\frac{x}{t}\right) \leq \frac{s}{t} \\ &\Leftrightarrow \left(\frac{x}{t}, \frac{s}{t}\right) \in \text{epi } f \text{ (Definition of epigraph)} \end{aligned}$$

So, $\text{epi } g$ is the inverse image of $\text{epi } f$ under the perspective mapping that takes (a, b, c) to $(a, b)/c$. Because f is convex function, it follows that its epigraph is convex ($\text{epi } f$ is convex). As we know the inverse image of convex set under the perspective mapping is also convex, so we can conclude that $\text{epi } g$ is convex. Since function is convex if and only if its epigraph is convex, it follows that g is convex function. ■

3 Convex optimization problems

For this chapter literature [1] is consulted.

3.1 Optimization problems

3.1.1 Fundamental terms

Let's consider problems of the form:

$$\begin{aligned} & \text{minimize } F(x) \\ & \text{s.t. } f_i(x) \leq 0, \quad i=1, \dots, m \\ & \quad h_j(x) = 0, \quad j=1, \dots, q \end{aligned} \quad (3.1)$$

This is problem of finding an x that minimizes objective function $F(x)$ among all x that satisfy the conditions $f_i(x) \leq 0, i = 1, \dots, m$, and $h_j(x) = 0, j = 1, \dots, q$.

$x \in \mathbb{R}^p$ is called the *optimization variable* and the function $F: \mathbb{R}^p \rightarrow \mathbb{R}$ is the *objective function* or *cost function*.

The inequalities $f_i(x) \leq 0$ are inequality constraints, and the related functions $f_i: \mathbb{R}^p \rightarrow \mathbb{R}$ are called the inequality constraint functions.

The equations $h_j(x) = 0$ are called the equality constraints, and the related functions $h_j: \mathbb{R}^p \rightarrow \mathbb{R}$ are the equality constraint functions.

In case when there are *no constraints* (i.e., $m = q = 0$) the problem (3.1) is said to be *unconstrained*.

The *domain* of the optimization problem (3.1), denoted as D , is defined as:

$$D = \bigcap_{i=0}^m \text{dom } f_i \cap \bigcap_{j=1}^q \text{dom } h_j.$$

The domain consists of points for which both the objective function and all constraint functions are defined. A point $x \in D$ is called *feasible* if it satisfies all the constraints $f_i(x) \leq 0, i = 1, \dots, m$, and $h_j(x) = 0, j = 1, \dots, q$.

The problem (3.1) is called *feasible* if there is at least one feasible point; otherwise, it is considered *infeasible*. The collection of all feasible points is referred to as the *feasible set* or *constraint set*.

The optimal value F^* for the problem (3.1) is defined as:

$$F^* = \inf \left\{ F(x) \mid f_i(x) \leq 0, \quad i=1, \dots, m, \quad h_j(x) = 0, \quad j=1, \dots, q \right\}.$$

The optimal value F^* can take the extended values $\pm\infty$. If the problem is infeasible, the optimal value is $F^* = \infty$ (The infimum of the empty set is ∞). The problem (3.1) is said to be *unbounded below* if there are feasible points x_k such that $F(x_k) \rightarrow -\infty$ as $k \rightarrow \infty$, so $F^* = -\infty$.

3.1.2 Optimal and locally optimal points

We define x^* as an *optimal point* or as a solution to the problem (3.1) if x^* is feasible and satisfies $F(x^*) = F^*$.

The collection of all optimal points is called the optimal set, denoted by:

$$X_{\text{opt}} = \{x \mid f_i(x) \leq 0, i = 1, \dots, m, h_j(x) = 0, j = 1, \dots, q, F(x) = F^*\}.$$

In case when there exists at least one optimal point for problem (3.1), the optimal value has been attained or achieved, meaning the problem is solvable. On the other hand, if X_{opt} is empty, we conclude that the optimal value is not attained or achieved.

A feasible point x is *locally optimal* if there is $R > 0$ such that

$$F(x) = \inf\{F(z) \mid f_i(z) \leq 0, i = 1, \dots, m, h_j(z) = 0, j = 1, \dots, q, \|z-x\|_2 \leq R\},$$

or to clarify x solves the optimization problem:

$$\begin{aligned} & \text{minimize } F(z) \\ & \text{s.t. } f_i(z) \leq 0, \quad i = 1, \dots, m \\ & \quad h_j(z) = 0, \quad j = 1, \dots, q \\ & \quad \|z - x\|_2 < R \end{aligned} \quad (3.2)$$

with variable z . In simpler terms, this means that x minimizes F over points close to it within the feasible set. The phrase ‘*globally optimal*’ is sometimes used instead of ‘optimal’ to differentiate it from ‘locally optimal’.

3.2 Convex optimization problems

3.2.1 Convex optimization problems in standard form

A convex optimization problem has the form:

$$\begin{aligned} & \text{minimize } F(x) \\ & \text{s.t. } f_i(x) \leq 0, \quad i = 1, \dots, m \\ & \quad a_j^T x = b_j, \quad j = 1, \dots, q \end{aligned} \quad (3.3)$$

where F, f_1, \dots, f_m are convex functions.

When comparing problem (3.3) to the general standard form problem in equation (3.1), the convex problem has three extra conditions: the objective function must be convex, the inequality constraint functions must be convex, the equality constraint functions $h_j(x) = a_j^T x - b_j$ must be affine.

The feasible set of a convex optimization problem is convex because it is the intersection of the problem's domain $D = \text{dom } F \cap (\bigcap_{i=1}^m \text{dom } f_i)$, which is convex (intersection of convex sets is

convex), with m convex sublevel sets $\{x | f_i(x) \leq 0\}$ and q hyperplanes $\{x | a_j^T x = b_j\}$. To conclude, in a convex optimization problem, we aim to minimize a convex objective function over a convex feasible set.

3.2.2 Abstract form convex optimization problem

Let's consider an example with $x \in \mathbb{R}^2$ which is in the standard form (3.1):

$$\begin{aligned} \text{minimize } F(x) &= x_1^2 + x_2^2 \\ \text{s.t. } f_1(x) &= \frac{x_1}{1+x_2^2} \leq 0 \\ h_1(x) &= (x_1 + x_2)^2 = 0 \end{aligned} \quad (3.4)$$

This problem is not convex optimization problem in standard form because f_1 is not convex function and h_1 is not affine. However, the feasible set, defined by $\{x | x_1 \leq 0, x_1 + x_2 = 0\}$ is convex. Although in this problem we are minimizing a convex function F over a convex set, it does not qualify as a convex optimization problem based on our definition.

This problem may be reformulated to obtain convex optimization problem in standard form:

$$\begin{aligned} \text{minimize } F(x) &= x_1^2 + x_2^2 \\ \text{s.t. } \bar{f}_1(x) &= x_1 \leq 0 \\ \bar{h}_1(x) &= x_1 + x_2 = 0 \end{aligned} \quad (3.5),$$

here F and \bar{f}_1 are convex, and \bar{h}_1 is affine.

Some authors use the term *abstract convex optimization problem* for the (abstract) problem of minimizing a convex function over a convex set. Using this terminology, the problem (3.5) is an abstract convex optimization problem.

3.2.3 Optima

Theorem 3.2.1 Any locally optimal point of convex optimization problems is also (globally) optimal.

Proof:

Let's suppose that x is locally optimal for a convex optimization problem, i.e., x is feasible and $F(x) = \inf\{F(z) | z \text{ is feasible, } \|z-x\|_2 \leq R\}$ for some $R > 0$. (3.6)

Now assume that x is not globally optimal, i.e., there is a feasible y such that $F(y) < F(x)$. Obviously, $\|y - x\|_2 > R$.

Let's consider point z :

$$z = \theta y + (1 - \theta) x, \text{ where } \theta = \frac{R}{2\|y-x\|_2}.$$

Now we have:

$$\|z-x\|_2 = \|\theta y + (1 - \theta) x - x\|_2 = \|\theta y + x - \theta x - x\|_2 = \theta \|y - x\|_2 = \frac{R}{2\|y-x\|_2} \|y - x\|_2 = \frac{R}{2} < R$$

and from convexity of the feasible set, z is feasible.

From convexity of F follows: $F(z) = F(\theta y + (1 - \theta) x) \leq (1 - \theta) F(x) + \theta F(y) < F(x)$. This is in contradiction with (3.6). So, there is no feasible y such that $F(y) < F(x)$, i.e., x is globally optimal.

■

4 First-Order Methods for Smooth and Non-Smooth Convex Optimization

4.1 Smooth objectives

For this subsection the literature in [2] and [6] are consulted.

Let us consider an unconstrained optimization problem where the objective function F consists of a *differentiable convex* function f , i.e.

$$F^* = \min_x \{F(x) = f(x): x \in \mathbb{R}^p\}. \quad (4.1)$$

4.1.1 Gradient Method

Gradient method is the basic first-order technique for solving this problem. It is special case of line search methods where $d^k = -\nabla f(x^k)$. This method is descent method which means: $f(x^{k+1}) < f(x^k)$. It utilizes the local gradient and iteratively executes the following update:

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k),$$

where k is number of iteration and α_k is an appropriate step size (or step length) that insures convergence.

Algorithm 1 Gradient descent method

Initialization: Choose starting point $x^0 \in \text{dom } f$

Repeat:

1. $d^k = -\nabla f(x^k)$
2. *Line search:* Choose step size α_k via exact or backtracking line search.
3. *Update:* $x^{k+1} = x^k - \alpha_k \nabla f(x^k)$

Until stopping criterion is satisfied

The form of stopping criterion is usually: $\|\nabla f(x^k)\|_2 \leq \eta$, where η is small and positive. In most cases, this condition is checked after step 1, preferable than after the update.

An advantage of this method is that many gradient iterations may be performed for the cost of a single iteration of more complicated methods, possibly taking a shorter time to achieve the same level of accuracy ε .

For smooth minimization faster algorithms, such as Newton-like methods, can be used. In this context, faster means that these methods use fewer iterations than the gradient method to reach an appropriate accuracy, i.e. $F(x^k) - F^* \leq \varepsilon$. Weakness of these algorithms are: more expensive computations, they require additional information from the function F and they do not generalize easily to constrained and non-smooth problems.

Note that by making simple assumptions about f , it may be analyzed how many iterations the gradient method will need to reach ε -accurate solution.

Gradient Method on convex functions with Lipschitz continuous gradient

Let's assume that the gradient of function f is *Lipschitz continuous*, which means:

$$\forall x, y \in \mathbb{R}^n, \|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x-y\|_2, \text{ for some constant } L.$$

By using the fixed step-size $\alpha = \frac{1}{L}$ or using a value that decreases f the most we get:

$$f(x^k) - f^* \leq \frac{2L}{k+4} d_0^2, \text{ where } d_0 = \|x^0 - x^*\|_2 \text{ is the distance between initial iterate } x^0 \text{ and optimal point } x^*.$$

This follows from next theorem in which gradient method uses fixed step size α .

Theorem 4.1.1 Let's assume that f is convex function, its gradient is Lipschitz continuous and $0 < \alpha < \frac{2}{L}$. Then the Gradient method generates a sequence of points $\{x_k\}$, with function values satisfying the following inequality:

$$f(x^k) - f^* \leq \frac{2(f(x^0) - f^*)d_0^2}{2d_0^2 + k \cdot \alpha(2 - L\alpha)(f(x^0) - f^*)}, \quad k \geq 0 \text{ and } d_0 = \|x^0 - x^*\|_2. \quad (4.2)$$

Proof:

Let's denote $d_k = \|x^k - x^*\|_2$. Then:

$$d_{k+1}^2 = \|x^{k+1} - x^*\|_2^2 =$$

$$= \|x^k - \alpha \nabla f(x^k) - x^*\|_2^2 = (\text{This follows by definition of } k+1\text{-th iteration})$$

$$= d_k^2 - 2\alpha \langle \nabla f(x^k), d_k \rangle + \alpha^2 \|\nabla f(x^k)\|_2^2 \leq$$

$$\leq d_k^2 - 2\alpha \frac{1}{L} \|\nabla f(x^k) - \nabla f(x^*)\|_2^2 + \alpha^2 \|\nabla f(x^k)\|_2^2 =$$

$$= d_k^2 - \alpha \left(\frac{2}{L} - \alpha \right) \|\nabla f(x^k)\|_2^2 \quad (\text{This follows because } \nabla f(x^*) = 0)$$

This follows by next inequality which states for f defined as above:
 $\frac{1}{L} \|\nabla f(x) - \nabla f(y)\|_2^2 \leq \langle \nabla f(x) - \nabla f(y), x - y \rangle, \forall x, y \in \mathbb{R}^n$

Hence, $d_k \leq d_0$.

Following inequalities states for f defined as above:

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2} \|x - y\|_2^2, \quad \forall x, y \in \mathbb{R}^n (*)$$

From this we have:

$$f(x^{k+1}) - f(x^k) - \langle \nabla f(x^k), x^{k+1} - x^k \rangle \leq \frac{L}{2} \|x^k - x^{k+1}\|_2^2$$

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{L}{2} \|x^k - x^{k+1}\|_2^2 =$$

$$\begin{aligned}
&= f(x^k) + \langle \nabla f(x^k), -\alpha \nabla f(x^k) \rangle + \frac{L}{2} \|\alpha \nabla f(x^k)\|_2^2 = \\
&= f(x^k) - W \|\nabla f(x^k)\|_2^2, \text{ where } W = \alpha \left(1 - \frac{L}{2}\alpha\right). (**)
\end{aligned}$$

Let's note: $\pi_k = f(x^k) - f^*$.

Using (*) it follows that:

$$f(x^k) - f^* \leq \langle \nabla f(x^k), x^k - x^* \rangle$$

$$\pi_k \leq \langle \nabla f(x^k), d_k \rangle \leq d_0 \|\nabla f(x^k)\|_2 \text{ (We proved that } d_k \leq d_0)$$

$$\frac{\pi_k}{d_0} \leq \|\nabla f(x^k)\|_2$$

$$\frac{\pi_k^2}{d_0^2} \leq \|\nabla f(x^k)\|_2^2 (***)$$

Keeping in mind (**) and (***) the next inequalities hold:

$$\pi_{k+1} \leq \pi_k - W \|\nabla f(x^k)\|_2^2 \leq \pi_k - W \frac{\pi_k^2}{d_0^2}$$

Furthermore, we have:

$$\frac{1}{\pi_{k+1}} \geq \frac{1}{\pi_k} + \frac{W}{d_0^2} \frac{\pi_k}{\pi_{k+1}} \geq \frac{1}{\pi_k} + \frac{W}{d_0^2}$$

Summing up these inequalities we obtain:

$$\frac{1}{\pi_{k+1}} \geq \frac{1}{\pi_0} + (k+1) \frac{W}{d_0^2}$$

So, for k we have:

$$\frac{1}{\pi_k} \geq \frac{1}{\pi_0} + k \frac{W}{d_0^2}$$

$$\frac{1}{\pi_k} \geq \frac{1}{\pi_0} + \frac{k\alpha \left(1 - \frac{L}{2}\alpha\right)}{d_0^2}$$

$$\frac{1}{\pi_k} \geq \frac{1}{\pi_0} + \frac{k\alpha \frac{2-\alpha L}{2}}{d_0^2}$$

$$\frac{1}{\pi_k} \geq \frac{1}{\pi_0} + \frac{k\alpha(2-\alpha L)}{2d_0^2}$$

$$\frac{1}{\pi_k} \geq \frac{2d_0^2 + k\alpha(2 - \alpha L)\pi_0}{2\pi_0 d_0^2}$$

$$\pi_k \leq \frac{2\pi_0 d_0^2}{2d_0^2 + k\alpha(2 - \alpha L)\pi_0}$$

Therefore, we have showed requested inequality. ■

Putting the fixed step size $\alpha = \frac{1}{L}$ in previous inequality we obtain:

$$\pi_k \leq \frac{2\pi_0 d_0^2}{2d_0^2 + k \frac{1}{L} \left(2 - \frac{1}{L} L\right) \pi_0}$$

$$\pi_k \leq \frac{2\pi_0 d_0^2}{2d_0^2 + \frac{k\pi_0}{L}}$$

$$\pi_k \leq \frac{2\pi_0 d_0^2}{\frac{2d_0^2 L + k\pi_0}{L}}$$

$$\pi_k \leq \frac{2\pi_0 d_0^2 L}{2d_0^2 L + k\pi_0}$$

$$\pi_k \leq 2d_0^2 L \frac{\pi_0}{2d_0^2 L + k\pi_0}$$

$$\pi_k \leq 2d_0^2 L \frac{1}{\frac{d_0^2}{2L} + k}$$

Putting x^0 and x^* in (*) (This inequality is in proof of previous theorem) we will get:

$$f(x^0) - f^* - \langle \nabla f(x^*), x^0 - x^* \rangle \leq \frac{L}{2} \|x^0 - x^*\|_2^2$$

$$f(x^0) - f^* \leq \frac{L}{2} \|x^0 - x^*\|_2^2$$

$$\frac{f(x^0) - f^*}{\|x^0 - x^*\|_2^2} \leq \frac{L}{2}$$

$$\frac{\|x^0 - x^*\|_2^2}{f(x^0) - f^*} \geq \frac{2}{L}$$

$$\text{So, } \frac{d_0^2}{\pi_0} \geq \frac{2}{L}$$

$$\text{Furthermore, } 2L \frac{d_0^2}{\pi_0} + k \geq 2L \cdot \frac{2}{L} + k = 4 + k$$

$$\frac{1}{2L \frac{d_0^2}{\pi_0} + k} \leq \frac{1}{k + 4}$$

$$\text{Hence, } \pi_k \leq \frac{2L}{k + 4} d_0^2.$$

Therefore, in the worst case the gradient method must perform $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ -iterations to obtain ε -accurate solution.

Theorem 4.1.2 For any k , $1 \leq k \leq \frac{n-1}{2}$, and any $x^0 \in \mathbb{R}^n$ there exists a convex function f with Lipschitz continuous gradient such that for any first-order iterative method which generates a sequence of test points $\{x^k\}$ where each point x^k is expressed as a linear combination of the initial point x^0 and the gradients of the function f at previous points $\{x^0, x^1, \dots, x^{k-1}\}$, i.e. $x^k = x^0 + \sum_{i=0}^{k-1} \alpha_i \nabla f(x^i)$ we have:

$$f(x^k) - f^* \geq \frac{3Ld_0^2}{32(k+1)^2} \quad (4.3)$$

Gradient Method on strongly convex functions with Lipschitz continuous gradient

There are additional structures of functions useful for numerical optimization. One of them is **strong convexity** (defined in Definition 2.2.2) which ensures existence of unique minimizer and upgrades optimization efficiency. Note that even non-smooth functions can have strong convexity.

Also, we can transform any convex problem into a strong-convex problem by adding a squared l_2 -regularization term. To show that this is true, let's look at the following lemmas.

Lemma 4.1.1 Let g be a strongly convex function with parameter $M > 0$ on convex set S , and let f be a convex function on S . Then the function $h(x) := f(x) + g(x)$ is also strongly convex function with parameter M on S .

Proof:

We know that g is strongly convex function with parameter $M > 0$ on convex set S , so from definition it follows that:

$$g(\theta x + (1-\theta)y) \leq \theta g(x) + (1-\theta)g(y) - M \frac{1}{2} \theta(1-\theta) \|x - y\|^2 \text{ for any } x, y \in S \text{ and any } \theta \in [0,1].$$

From definition of convexity of function f on set S we have:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \text{ for any } x, y \in S \text{ and any } \theta \in [0,1].$$

So,

$$\begin{aligned} h(\theta x + (1-\theta)y) &= f(\theta x + (1-\theta)y) + g(\theta x + (1-\theta)y) \\ &\leq \theta f(x) + (1-\theta)f(y) + \theta g(x) + (1-\theta)g(y) - M \frac{1}{2} \theta(1-\theta) \|x - y\|^2 \\ &= \theta h(x) + (1-\theta)h(y) - M \frac{1}{2} \theta(1-\theta) \|x - y\|^2 \text{ for any } x, y \in S \text{ and any } \theta \in [0,1]. \end{aligned}$$

To conclude, $h(\theta x + (1-\theta)y) \leq \theta h(x) + (1-\theta)h(y) - M \frac{1}{2} \theta(1-\theta) \|x - y\|^2$ for any $x, y \in S$ and any $\theta \in [0,1]$. By definition of strongly convex function, it follows that $h(x)$ is also strongly convex function with parameter M on S . ■

Lemma 4.1.2 Let f be a convex function on convex set S , and let $M > 0$. Then function $f(x) + \frac{M}{2} \|x\|_2^2$ is strongly convex function on S with parameter $M > 0$.

Proof:

Let $g(x) = \frac{M}{2} \|x\|_2^2$. The Hessian matrix of function $g(x)$ is $\nabla^2 g(x) = MI$.

It holds that $\nabla^2 g(x) \geq MI, \forall x \in S$. From this we can conclude that $g(x)$ is strongly convex function on set S with parameter $M > 0$.

From previous lemma it follows that function $f(x) + \frac{M}{2} \|x\|_2^2$ is strongly convex function on S with parameter $M > 0$. ■

So, if we for example have convex optimization problem $\min_x f(x)$, it may be transformed into strong convex problem $\min_x \{f(x) + \frac{M}{2} \|x\|_2^2\}$.

Theorem 4.1.3 Let's assume that f is strongly convex function, its gradient is Lipschitz continuous and $0 < \alpha \leq \frac{2}{M+L}$. M is strong convexity parameter and L is Lipschitz constant. Then the Gradient method generates a sequence of points $\{x_k\}$ such that

$$\|x^k - x^*\|_2^2 \leq \left(1 - \frac{2\alpha ML}{L+M}\right)^k \|x^0 - x^*\|_2^2 \quad (4.4)$$

If $\alpha = \frac{2}{M+L}$ then:

$$\|x^k - x^*\|_2 \leq \left(\frac{L-M}{L+M}\right)^k \|x^0 - x^*\|_2,$$

$$f(x^k) - f^* \leq \frac{L}{2} \left(\frac{L-M}{L+M}\right)^{2k} \|x^0 - x^*\|_2^2$$

Proof:

Let's denote $d_k = \|x^k - x^*\|_2$. Then:

$$d_{k+1}^2 = \|x^{k+1} - x^*\|_2^2 =$$

$$= \|x^k - \alpha \nabla f(x^k) - x^*\|_2^2 = (\text{This follows by definition of } k+1\text{-th iteration})$$

$$= d_k^2 - 2\alpha \langle \nabla f(x^k), d_k \rangle + \alpha^2 \|\nabla f(x^k)\|_2^2 \leq$$

$$\leq d_k^2 - 2\alpha \left(\frac{ML}{M+L} \|x^k - x^*\|_2^2 + \frac{1}{M+L} \|\nabla f(x^k)\|_2^2 \right) + \alpha^2 \|\nabla f(x^k)\|_2^2 =$$

$$= \left(1 - \frac{2\alpha ML}{M+L}\right) d_k^2 + \alpha \left(\alpha - \frac{2}{M+L}\right) \|\nabla f(x^k)\|_2^2 \leq$$

$$\leq \left(1 - \frac{2\alpha ML}{M+L}\right) d_k^2 \leq$$

$$\leq \left(1 - \frac{2\alpha ML}{M+L}\right)^k d_0^2$$

This follows by next inequality which states for f defined as above:

$$\begin{aligned} & \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \\ & \geq \frac{ML}{M+L} \|x - y\|_2^2 + \frac{1}{M+L} \|\nabla f(x) - \nabla f(y)\|_2^2 \quad \forall x, y \in \mathbb{R}^n \\ & \text{and } \nabla f(x^*) = 0 \end{aligned}$$

The last inequality follows from $0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2} \|x - y\|_2^2$, $\forall x, y \in \mathbb{R}^n$ and (4.4). ■

For $\alpha = \frac{2}{M+L}$ GM converges linearly to a strict local minimum x^* .

Theorem 4.1.4 For any $x^0 \in \mathbb{R}^n$ and any constants $M > 0$ and $L > 0$, such that $\frac{L}{M} > 1$, there exists a smooth strongly convex function f with Lipschitz continuous gradient such that for any first-order iterative method which generates a sequence of test points $\{x^k\}$ where each point x^k is expressed as a linear combination of the initial point x^0 and the gradients of the function f at previous points $\{x^0, x^1, \dots, x^{k-1}\}$, i.e.

$$x^k = x^0 + \sum_{i=0}^{k-1} \alpha_i \nabla f(x^i)$$

we have:

$$f(x^k) - f^* \geq \frac{M}{2} \left(\frac{\sqrt{\frac{L}{M}} - 1}{\sqrt{\frac{L}{M}} + 1} \right)^{2k} \|x^0 - x^*\|_2^2 \quad (4.5)$$

To conclude, if the function is convex and has Lipschitz continuous gradient it is not guaranteed that the iterates x^k are going to converge, while for strongly-convex function with Lipschitz continuous gradient we have both $f(x^k)$ and x^k converge.

Comparing the rate of convergence of the Gradient Method with the lower complexity bounds (Theorem 4.1.2 and Theorem 4.1.4), we can conclude that GM is away from being optimal for the classes of smooth convex and strongly convex functions with Lipschitz continuous gradient. The optimal methods for minimizing smooth convex and strongly convex functions need some global information on the objective function.

4.1.2 Accelerated gradient method

Slovak mathematician Yurii Nesterov designed accelerated gradient method which achieves the best possible worst-case error rate for unconstrained minimization problem. This method upgrades upon the standard gradient descent by introducing momentum into the update process. The momentum term helps accelerate the updates, reducing oscillations and speeding up convergence. Optimal convergence is realized by the simple step-size $\alpha_k = \frac{1}{L}$ and an extra-momentum step with a parameter $\beta_k = \frac{k}{k+3}$. This is referred as an *optimal* first-order method.

Algorithm 2 *Nesterov's accelerated gradient method for unconstrained minimization*

Initialization: Start with initial position x^0 and initial velocity v^0 . ($x^0 = v^0$)

Repeat:

Step 1: Update the parameters

$$x^{k+1} = v^k - \alpha_k \nabla f(v^k)$$

Step 2: Update the velocity

$$v^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k)$$

Until stopping criterion is satisfied

In Step 1 v^k represents the previous velocity. In this step we update the parameters applying the gradient evaluated at the previous velocity. Step 2 updates the velocity based on the new parameter position and the change in position from the previous iteration, scaled by the momentum term β_k .

Theorem 4.1.5 Let's assume that f is convex function and its gradient is Lipschitz continuous. Then the Nesterov's accelerated gradient method generates a sequence of points $\{x_k\}$, with function values satisfying the following inequality:

$$f(x^k) - f^* \leq \frac{8Ld_0^2}{3(k+1)^2}, \quad k \geq 0 \text{ and } d_0 = \|x^0 - x^*\|_2. \quad (4.6)$$

Therefore, in the worst case the Nesterov's accelerated gradient method must perform $\mathcal{O}\left(\frac{1}{\sqrt{\varepsilon}}\right)$ iterations to obtain ε -accurate solution.

Also, the accelerated-gradient method can have utility from strong-convexity with a suitable choice of the momentum term β_k . For instance, this method obtains a near-optimal convergence rate given its assumptions by setting $\beta_k = \frac{L-M}{L+M}$.

By using Nesterov's smoothing technique, the fast gradient algorithms may be also applied to non-smooth minimization optimization problems.

Furthermore, preferably than assuming Lipschitz continuity of the gradient of the objective function, recent work has considered effective gradient methods for smooth self-concordant functions, which naturally appear in graph learning, Poisson imaging, and numerous tomography problems. [8]

4.2 Composite objectives

For this subsection references [5] and [6] are used.

Let us consider an unconstrained optimization problem where the objective function F consists of a *differentiable convex* function f and a ***non-smooth convex*** function g , i.e.

$$F^* = \min_x \{F(x) = f(x) + g(x): x \in \mathbb{R}^p\}. \quad (4.7)$$

In sum, the non-differentiability of g seems to reduce the productivity of first order methods. Generic non-smooth optimization methods, such as bundle methods and subgradient require $\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$ iterations to reach ε -accurate solutions. Although the rate can be improved to $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ under the assumption of strong convexity, the resulting convergence rates are still slower compared to those achieved by first-order methods when the objective function is smooth.

Fortunately, ***proximal-gradient methods*** offer a more suitable approach for this class of problems. These methods preserve the convergence rates of gradient methods applied to smooth problem classes. In fact, proximal gradient methods can be seen as natural extensions of the gradient method when we view the gradient method's iterations as an optimization problem:

$$x^{k+1} = \underset{y \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ f(x^k) + \nabla f(x^k)^T (y - x^k) + \frac{1}{2\alpha_k} \|y - x^k\|_2^2 \right\}.$$

This represents finding the point x^{k+1} that minimizes a quadratic approximation of f around x^k moving in the direction of $-\nabla f(x^k)$ scaled by α_k . Proximal-gradient methods use the same approximation of f , but insert the non-smooth term g in an explicit mode:

$$x^{k+1} = \underset{y \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ f(x^k) + \nabla f(x^k)^T (y - x^k) + \frac{1}{2\alpha_k} \|y - x^k\|_2^2 + g(y) \right\}$$

Algorithm 3 Proximal gradient descent method

Initialization: Choose starting point $x^0 \in \operatorname{dom} F$

Repeat:

$$\text{Update: } x^{k+1} = \operatorname{prox}_{\alpha_k g}(x^k - \alpha_k \nabla f(x^k))$$

Until stopping criterion is satisfied

The *proximal map* or *proximal operator* is defined as:

$$\operatorname{prox}_g(y) = \underset{x}{\operatorname{argmin}} \left\{ g(x) + \frac{1}{2} \|x - y\|_2^2 \right\}$$

The proximal operator of the scaled function $\alpha_k g$ (where $\alpha_k > 0$ is step size) can be expressed as:

$$\operatorname{prox}_{\alpha_k g}(y) = \underset{x}{\operatorname{argmin}} \left\{ g(x) + \frac{1}{2\alpha_k} \|x - y\|_2^2 \right\}$$

Theorem 4.2.1 Let's assume that f is convex differentiable function, its gradient is Lipschitz continuous, $\alpha \leq \frac{1}{L}$, g is convex non-smooth function. Then the Proximal gradient method with fixed step size generates a sequence of points $\{x^k\}$, with function values satisfying the following inequality:

$$F(x^k) - F^* \leq \frac{d_0^2}{2\alpha k}, \quad k \geq 0 \text{ and } d_0 = \|x^0 - x^*\|_2. \quad (4.8)$$

Therefore, in the worst case the proximal gradient method must perform $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ -iterations to obtain ϵ -accurate solution.

The *accelerated proximal-gradient* method is defined as following:

Algorithm 4 Accelerated proximal-gradient method for unconstrained minimization

Initialization: Start with initial position x^0 and initial velocity v^0 . ($x^0 = v^0$)

Repeat:

Step 1: Update the parameters

$$x^{k+1} = \text{prox}_{\alpha_k g} \left(v^k - \alpha_k \nabla f(v^k) \right)$$

Step 2: Update the velocity

$$v^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k)$$

Until stopping criterion is satisfied

Theorem 4.2.2 Let's assume that f is convex differentiable function, its gradient is Lipschitz continuous, $\alpha \leq \frac{1}{L}$, g is convex non-smooth function. Then the Accelerated proximal gradient method generates a sequence of points $\{x_k\}$, with function values satisfying the following inequality:

$$F(x^k) - F^* \leq \frac{2d_0^2}{\alpha(k+1)^2}, \quad k \geq 0 \text{ and } d_0 = \|x^0 - x^*\|_2. \quad (4.9)$$

Therefore, in the worst case the Accelerated proximal gradient method must perform $\mathcal{O}\left(\frac{1}{\sqrt{\epsilon}}\right)$ iterations to obtain ϵ -accurate solution.

It's interesting to mention the special case of the proximal-gradient algorithm when we consider the indicator function on a convex set S , which is way of including constraints into (4.7)

$$g(x) = \begin{cases} 0 & x \in S \\ \infty & x \notin S \end{cases}$$

In this case, the proximal-gradient method incomes the classic projected-gradient method for *constrained optimization*.

When $f = 0$ proximal-gradient method reduces to proximal minimization, and when $g = 0$, it becomes the standard gradient descent method.

In general, when the proximal operator can be computed efficiently, proximal gradient algorithms tend to be computationally efficient as well. Interestingly, even sets with an infinite number of atoms can admit efficient proximal mappings. A notable example is the set of rank-one matrices

with unit Frobenius norm, for which the proximal operator is given by singular value thresholding. In contrast, certain sets with a finite number of atoms-such as the set of rank-one matrices with binary entries-may lead to intractable proximal operators.

In following table is total number of iterations required for the model to reach an error level of ϵ . Here, L and M are Lipschitz and strong convexity constants respectively and $d_0 = \|x^0 - x^*\|_2$.

Algorithm	Convex	Strongly-Convex
(Proximal) Gradient	$\mathcal{O}\left(\frac{Ld_0^2}{\epsilon}\right)$	$\mathcal{O}\left(\frac{L}{M} \log\left(\frac{d_0^2}{\epsilon}\right)\right)$
Accelerated (Proximal) Gradient	$\mathcal{O}\left(\sqrt{\frac{Ld_0^2}{\epsilon}}\right)$	$\mathcal{O}\left(\sqrt{\frac{L}{M}} \log\left(\frac{d_0^2}{\epsilon}\right)\right)$

Table 1(source:[6])

Recent research has introduced various strategies for step-size selection and adaptive restarting of the momentum parameter in first-order optimization methods. [13] These approaches typically incur only a modest additional computational cost and crucially do not rely on prior knowledge of problem-specific constants such as the strong convexity parameter M or the Lipschitz constant L . Although such heuristic enhancements do not lead to provable improvements in the worst-case theoretical convergence rates, they have demonstrated significant gains in empirical performance. These techniques can also be naturally extended to proximal variants of first-order methods. An illustration of the improved practical performance enabled by these enhancements is presented in the following figure.

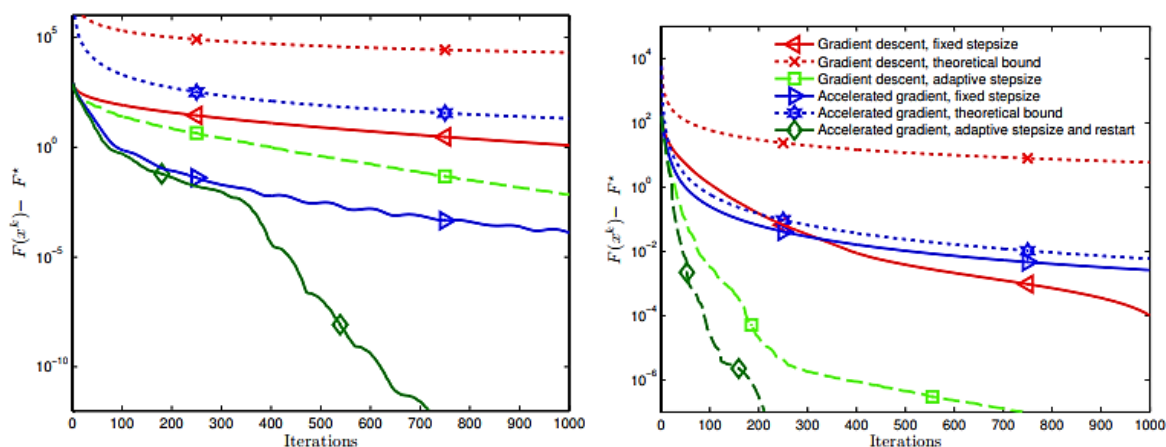


Figure 11(source:[6])

Here is demonstrated how the objective function $F(x^k)$ progress as a function of iterations k for solving (Right) the LASSO formulation:

$$\hat{x}_{LASSO} = \underset{x \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ F(x) := \frac{1}{2} \|y - \phi x\|_2^2 + \lambda \|x\|_1 \right\}$$

and (Left) the LS formulation:

$$\hat{x}_{LS} = \underset{x \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ F(x) := \frac{1}{2} \|y - \phi x\|_2^2 \right\},$$

both with $p = 5000$ (number of features or columns in data matrix) and $n = 2500$ (number of data points or rows in data matrix) for four methods:

- 1) (proximal) gradient descent with adaptive step-size,
- 2) (proximal) gradient descent with fixed step-size $\alpha = \frac{1}{L}$,
- 3) accelerated (proximal) gradient descent with fixed step-size and
- 4) accelerated (proximal) gradient descent with the adaptive step-size and restart scheme in TFOCS (Templates for First-Order Conic Solvers – a software package for solving convex optimization problems using first-order methods, especially focusing on problems that involve conic constraints).

For the least-squares (LS) formulation, the baseline optimization methods exhibit qualitative behavior that aligns well with their theoretical upper-bound predictions. However, when practical enhancements are incorporated these methods show markedly improved performance.

In the LASSO formulation (proximal) gradient descent outperforms the basic accelerated (proximal) method in high-accuracy regime, further it automatically benefits from sparsity of the solution. Accelerated (proximal) method also benefits from sparsity by adding the adaptive restart enhancement.

The figure below illustrates a crucial insight: the numerical efficiency of first-order methods in composite minimization problems heavily depends on selecting an appropriate *smoothness structure for the smooth component f* . This is demonstrated through a convergence plot using the **heteroskedastic LASSO (hLASSO)** model, as described in [8], for the case where $n = 1.5 \times 10^4$ and $p = 5 \times 10^4$. The hLASSO objective includes a smooth part f that is **self-concordant** but does *not possess a Lipschitz continuous gradient*. Despite this, it enables recovery of sparse solutions while simultaneously estimating the unknown noise variance σ^2 in the underlying linear model.

When a basic first-order method is tailored to the correct self-concordant structure of f , it is able to determine optimal step sizes automatically. As a result, it outperforms standard first-order methods that rely on the assumption of a Lipschitz continuous gradient—even when such methods are enhanced with adaptive step-size or restart techniques. Interestingly, the standard gradient descent method converges more quickly than the accelerated variant, as the latter depends heavily on Lipschitz continuity for its momentum updates, which in this case lead to inefficient and costly step-size adaptation.

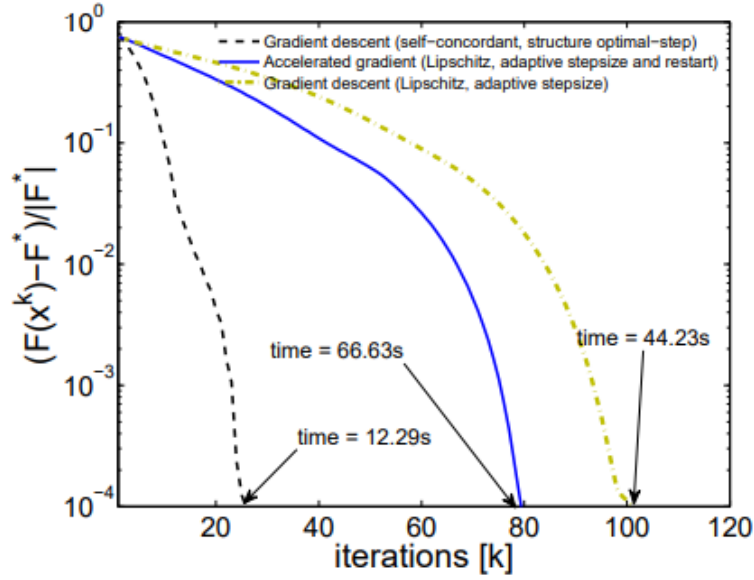


Figure 12

4.3 Proximal objectives

For this subsection references [3] and [6] are used.

The first-order methods we have discussed heretofore are not directly applicable for many applications. For this problem it turned out to be useful to reformulate the composite form into the following form:

$$\min_{x, z \in \mathbb{R}^p} \{F(x, z) \equiv h(x) + g(z) : \phi z = x\}, \quad (4.10)$$

and only suppose that proximity operators of h and g are both efficient.

This form can deal with **non-smooth and non-Lipschitz objective functions** which usually occur in many applications such as graph learning, Poisson imaging, and robust principal component analysis (RPCA). For its solutions it is used ADMM algorithm – alternating direction method of multipliers, which uses powerful *augmented Lagrangian* and *dual decomposition* techniques. ADMM is structured as a decomposition-coordination method, where solutions to smaller local subproblems are integrated to obtain a solution for a larger global problem. This algorithm is a simple but powerful. It is suitable to distributed convex optimization.

First of all, augmented Lagrangian function for (4.10) problem is:

$$L_p(x, z, y) = h(x) + g(z) + y^T(x - \phi z) + \frac{p}{2} \|x - \phi z\|_2^2,$$

where y is the dual variable or Lagrange multiplier and p is positive scalar known as the penalty parameter.

ADMM consists of iterations:

$$\text{Step 1: } x^{k+1} = \underset{x}{\operatorname{argmin}} L_p(x, z^k, y^k) = \underset{x}{\operatorname{argmin}} \left(h(x) + (y^k)^T (x - \phi z^k) + \frac{p}{2} \|x - \phi z^k\|_2^2 \right) \quad (\text{x-minimization step})$$

$$\text{Step 2: } z^{k+1} = \underset{z}{\operatorname{argmin}} L_p(x^{k+1}, z, y^k) = \underset{z}{\operatorname{argmin}} \left(g(z) + (y^k)^T (x^{k+1} - \phi z) + \frac{p}{2} \|x^{k+1} - \phi z\|_2^2 \right) \quad (\text{z-minimization step})$$

$$\text{Step 3: } y^{k+1} = y^k + p(x^{k+1} - \phi z^{k+1}) \quad (\text{dual variable update})$$

The algorithm is very close to dual ascent and the method of multipliers: it consists of an x -minimization step, a z -minimization step and a dual variable update. Similar to the method of multipliers, the update for the dual variable involves using a step size that matches the augmented Lagrangian parameter, p .

Let's see form of method of multipliers for (4.10):

$$\text{Step 1: } (x^{k+1}, z^{k+1}) = \underset{x, z}{\operatorname{argmin}} L_p(x, z, y^k)$$

$$\text{Step 2: } y^{k+1} = y^k + p(x^{k+1} - \phi z^{k+1})$$

In this context the augmented Lagrangian is minimized jointly with respect to the both primal variables.

In contrast, ADMM updates the variables x and z in an alternating or sequential manner, which is the reason for the term "*alternating direction*". In ADMM, the algorithm state is defined by z^k and y^k . This means that (z^{k+1}, y^{k+1}) is a function of (z^k, y^k) . The variable x^k is not included in the state; instead, it is an intermediate result derived from the previous state (z^{k-1}, y^{k-1}) .

Often is more suitably to write ADMM in a little bit different form, by combining the linear and quadratic terms in the augmented Lagrangian and scaling the dual variable.

Defining the residual r as $x - \phi z$, we have:

$$\begin{aligned} y^T r + \frac{p}{2} \|r\|_2^2 &= \frac{p}{2} \|r\|_2^2 + \frac{1}{p} \|y\|_2^2 - \frac{1}{2p} \|y\|_2^2 = \\ &= \frac{p}{2} \|r + u\|_2^2 - \frac{p}{2} \|u\|_2^2, \end{aligned}$$

where $u = \frac{1}{p} y$ is scaled dual variable.

So, scaled form of ADMM follows.

Algorithm 3 Scaled ADMM; $p > 0, x^0 = u^0 = 0$

$$\text{Step 1: } x^{k+1} = \underset{x}{\operatorname{argmin}} \left(h(x) + \frac{p}{2} \|x - \phi z^k + u^k\|_2^2 \right) = \operatorname{prox}_{\frac{1}{p}h}(\phi z^k - u^k)$$

$$\text{Step 2: } z^{k+1} = \underset{z}{\operatorname{argmin}} \left(g(z) + \frac{p}{2} \|x^{k+1} - \phi z + u^k\|_2^2 \right)$$

$$\text{Step 3: } u^{k+1} = u^k + x^{k+1} - \phi z^{k+1}$$

The scaled and unscaled form are clearly equivalent, but the formulas in the scaled form of ADMM are often shorter. The unscaled form is used when we wish to stress the role of the dual variable or to give an interpretation that relies on the (unscaled) dual variable.

This algorithm needs a penalty parameter p as input, generates a sequence of iterates that approach feasibility and produces the optimal objective value in the limit.

It's important to point out two warnings for ADMM.

First, step 2 in Algorithm 3 has to be numerically solved in general except in case when $\phi^T \phi$ is effectively diagonalizable. For this problem, z^{k+1} can be updated by using a single step of the proximal gradient method, which brings to the method shown in Algorithm 4.

Secondly, if we *naïvely* extend ADMM to handle more than two objective terms we might not be able to guarantee convergence. For this problem, *dual decomposition* techniques can be used to treat the multiple terms in the objective of (4.10) as individual problems and simultaneously solve them in *parallel*. This will be discussed in Section 6.

It's interesting to mention the case when $h(x)$ has a difficult proximal operator in Algorithm 3 but also has a Lipschitz continuous gradient. For that matter $h(x)$ in Step 1 may be replaced with its quadratic surrogate to obtain the linearized ADMM. Amazingly, these inexact update-steps can be as fast to converge as the full ADMM in specified applications.

Algorithm 4 Primal-Dual Hybrid Gradient method to solve (4.10) $\lambda > 0$ and $\tau \leq \frac{1}{\|\phi\|^2}$

$$\text{Step 1: } x^{k+1} = \operatorname{prox}_{\lambda h}(\phi z^k - u^k)$$

$$\text{Step 2: } z^{k+1} = \operatorname{prox}_{\lambda \tau g} \left(z^k + \tau \phi^T (x^{k+1} - \phi z^k + u^k) \right)$$

$$\text{Step 3: } u^{k+1} = u^k + x^{k+1} - \phi z^{k+1}$$

5 Big Data scaling via randomization

For this section literature [6] is used.

In theory, first-order methods can effectively tackle very large-scale problems. However, as the dimensions of the problem increase, these straightforward methods can become impractical due to the precise numerical calculations required during their iterations. Fortunately, first-order methods are quite resilient when it comes to using approximations for their optimization components, such as gradients and proximal calculations. This chapter will introduce new *randomized approximations* that enable first-order methods to remain efficient even for very large-scale problems, despite relying on inexact computations.

To illustrate the key concepts, we will focus specifically on smooth and strongly convex functions F , while also noting potential extensions.

For example, consider Google's PageRank problem, which assesses the importance of nodes in a graph based on its incidence matrix, where p may reach tens of billions. Assuming that more important nodes have more connections, the task essentially involves identifying the top singular vector of the stochastic matrix $\phi = A \text{diag}(A^T \mathbf{1}_p)^{-1}$, where $\mathbf{1}_p \in \mathbb{R}^p$ is the vector where every element is 1. The PageRank algorithm employs the *power method* to address this fundamental linear algebra issue (find $x^* \geq 0$ such that $\phi x^* = x^*$ and $\mathbf{1}_p^T x^* = 1$). This goal can be effectively approximated using a least squares problem by relaxing the constraints with a penalty parameter $\lambda > 0$:

$$\min_{x \in \mathbb{R}^p} \left\{ F(x) \equiv \frac{1}{2} \|x - \phi x\|_2^2 + \frac{\lambda}{2} (\mathbf{1}_p^T x - 1)^2 \right\} \quad (5.1)$$

5.1 Coordinate descent methods

For the PageRank problem formulation computing the full gradient demands a matrix-vector operation at each iteration. A low-cost vector-only operation would be to select a coordinate k of vector x and **only transform the variable x_k** to upgrade the objective function (other variables remain fixed). This is essence of *coordinate descent methods*. The advantage of these methods is that each iteration is simple, in generating the search direction and in performing the update of variables.

The general form of coordinate descent methods follows in Algorithm 5, where e_i is the i^{th} canonical coordinate vector (vector in which only one element on coordinate i is non-zero and equals one, while all other elements are zero) and $\nabla_i F(\cdot)$ is the i^{th} coordinate of the gradient.

Algorithm 5 Coordinate descent method to minimize F over \mathbb{R}^p

Step 1: Choose an index $i_k \in \{1, 2, \dots, p\}$

Step 2: Update $x^{k+1} = x^k - \alpha \nabla_{i_k} F(x^k) e_{i_k}$

Example:

To demonstrate the fundamental concept of the Coordinate Descent method, consider a simple quadratic function of two variables: $F(x, y) = (x-2)^2 + (y+3)^2$. This function is convex and continuously differentiable, with a unique global minimum at the point $(x^*, y^*) = (2, -3)$. The objective is to minimize $F(x, y)$. So,

Step 1: Choose an index (For example $i_k = 1$ for x and $i_k = 2$ for y)

Step 2: Update $x^{k+1} = x^k - \alpha \nabla_{i_k} F(x^k) e_{i_k}$, where:

- $x^k = (x_k, y_k)$ is the current point at iteration k

Here we will use $\alpha = 0.1$ for simplicity.

Gradient of function is: $\nabla F(x, y) = [2(x-2), 2(y+3)]$

Iteration 0: Let's for example $x^0 = (0, 0)$.

Iteration 1: Step 1: Choose an index $i_0 = 1$

Calculate partial gradient: $\frac{\partial F}{\partial x} = 2(x-2) = 2(0-2) = -4$

Step 2: Update: $x^1 = x^0 - \alpha \nabla_1 F(x^0) e_1 = (0, 0) - 0.1 \cdot (-4) \cdot (1, 0) = (0.4, 0)$

Iteration 2: Step 1: Choose an index $i_1 = 2$

Calculate partial gradient: $\frac{\partial F}{\partial y} = 2(y+3) = 2(0+3) = 6$

Step 2: Update: $x^2 = x^1 - \alpha \nabla_2 F(x^1) e_2 = (0.4, 0) - 0.1 \cdot 6 \cdot (0, 1) = (0.4, -0.6)$

Iteration 3: Step 1: Choose an index $i_2 = 1$

Calculate partial gradient: $\frac{\partial F}{\partial x} = 2(x-2) = 2(0.4-2) = -3.2$

Step 2: Update: $x^3 = x^2 - \alpha \nabla_1 F(x^2) e_1 = (0.4, -0.6) - 0.1 \cdot (-3.2) \cdot (1, 0) = (0.72, -0.6)$

Iteration 4: Step 1: Choose an index $i_3 = 2$

Calculate partial gradient: $\frac{\partial F}{\partial y} = 2(y+3) = 2(-0.6+3) = 4.8$

Step 2: Update: $x^4 = x^3 - \alpha \nabla_2 F(x^3) e_2 = (0.72, -0.6) - 0.1 \cdot 4.8 \cdot (0, 1) = (0.72, -1.08)$

It should be noted that the iterative process does not terminate here. We can see that the algorithm steadily and gradually converges to the minimum point $(2, -3)$.

In this example, the coordinates were chosen alternately, which is one possible selection strategy (cyclic). There are several approaches to coordinate selection in the coordinate descent method, which will be discussed in more detail below.

One strategy is to select the coordinate with the largest directional derivative $\nabla_i F$. Then coordinate descent method has form:

Choose $x^0 \in \mathbb{R}^p$. For $k \geq 0$ iterate:

Step 1: Choose an index $i_k = \arg \max_{1 \leq i \leq p} |\nabla_i F(x^k)|$

Step 2: Update $x^{k+1} = x^k - \alpha \nabla_{i_k} F(x^k) e_{i_k}$

If we use $\alpha = \frac{1}{L_{max}}$ or optimizing the variable exactly, this choice brings to a convergence rate of

$$F(x^k) - F(x^*) \leq \left(1 - \frac{\mu}{pL_{max}}\right)^k (F(x^0) - F(x^*)), \quad (5.2)$$

where $L_{max} \equiv \max_i L_i$ is the maximum across the Lipschitz constants of $\nabla_i F$.

In this example we can see the primary difficulty in coordinate descent methods. For finding the best coordinate to update, in this case finding the maximum of the gradient element's greatness, we have to calculate the *whole* gradient vector. Thus, *a computational cost is as high as the gradient calculation cost*. It may be proved that convergence rate of this method is worse than the rate of convergence of the gradient method.

The simplest form of the coordinate descent method is based on a *cyclic coordinate search*. This is the cheapest strategy but convergence rate is a really slower.

Randomization of the coordinate selection is a commonly used strategy in coordinate descent methods, where the coordinate i is chosen uniformly at random from the set $\{1, 2, \dots, p\}$. One of the key advantages of this approach is that the cost of selecting a coordinate can be made independent of the problem dimension p . More precisely, the selection can be performed in constant time $\mathcal{O}(1)$ using a direct call to a pseudorandom number generator, without the need to explicitly scan or process all coordinates.

Despite its simplicity and low per-iteration cost, this uniformly random selection achieves the same expected convergence rate as the one described in equation (5.2). This property makes randomized coordinate descent methods particularly efficient and scalable for high-dimensional optimization problems, such as those encountered in large-scale machine learning and data analysis tasks.

The convergence rate of the randomized coordinate descent method can be improved to (5.3) by applying an ***importance sampling*** strategy. Instead of selecting coordinates uniformly at random, coordinates are chosen with probabilities proportional to their "importance" — that is, to how much they influence the value of the objective function. In this context, importance is measured by the Lipschitz constants of the gradient with respect to each coordinate (L_i).

$$F(x^k) - F(x^*) \leq \left(1 - \frac{\mu}{pL_{mean}}\right)^k (F(x^0) - F(x^*)), \quad (5.3)$$

(L_{mean} is the mean across the L_i)

The speed is upgraded by this non-uniform random sampling strategy, appending an $\mathcal{O}(\log(p))$ importance sampling cost to the algorithm.

It's important to highlight that coordinate descent methods are possibly most useful for objectives of the form $F(Ax)$ with $A \in \mathbb{R}^{n \times p}$, where evaluating the (not necessarily smooth) F costs $\mathcal{O}(n)$.

Eventually, there are recent researches about accelerated and composite versions of coordinate descent methods. It's important to mention that accelerated methods often do not preserve the low-cost iteration expense of the non-accelerated versions.

5.2 Stochastic gradient methods

Stochastic gradient methods update ***all coordinates*** at the same time and use approximate gradients. These methods are commonly employed to optimize the sample average of a finite training dataset. The concept behind these methods is fairly straightforward.

Beginning from the problem of minimizing ***decomposable*** objective function:

$$\min_{x \in \mathbb{R}^p} \left\{ F(x) = \frac{1}{n} \sum_{i=1}^n F_i(x) \right\} \quad (5.4)$$

and supposing each component function F_i is differentiable and measures the data misfit for a single data point, the method randomly selects an index j and makes a step in the direction of the negative gradient of F_j .

Algorithm 6 Stochastic gradient descent method to minimize F over \mathbb{R}^p

Step 1: Choose an index $i_k \in \{1, 2, \dots, n\}$ uniformly at random (Each index has an equal probability of being chosen)

Step 2: Update $x^{k+1} = x^k - \alpha_k \nabla F_{i_k}(x^k)$

Example:

To illustrate the fundamental concept of the Stochastic Gradient Descent (SGD) method, consider the following example.

Problem: Binary classification

Model: Use Sigmoid function $\hat{y}_i = \frac{1}{1 + e^{-w^T x_i}} = \sigma(w^T x_i) = \sigma(z_i)$

Error function for one data point:

$$F_i(w) = -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

This is the log-loss (or binary cross-entropy) function — the standard error function for binary classification with sigmoid output.

Objective: Minimize average error across all data points $F(w) = \frac{1}{n} \sum_{i=1}^n F_i(w)$

One training point: - Input vector $x_i = (x_{i1}, x_{i2})$

- Label $y_i \in \{0, 1\}$

- w is weight vector (determines how important each input feature (coordinate) is to the output of the model)

Gradient of function $F_i(w)$ by w : $\nabla_w F_i(w) = \frac{\partial F_i}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w}$ where $z_i = w^T x_i$ and $\hat{y}_i = \sigma(z_i)$

$$\frac{\partial F_i}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i} + \frac{1 - y_i}{1 - \hat{y}_i}$$

$$\frac{\partial \hat{y}_i}{\partial z_i} = \frac{\partial \sigma(z_i)}{\partial z_i} = \frac{1'(1+e^{-z_i}) - 1(1+e^{-z_i})'}{(1+e^{-z_i})^2} = \frac{e^{-z_i}}{(1+e^{-z_i})^2}$$

We know that: $\sigma(z_i) = \frac{1}{1+e^{-z_i}}$ so, $1 - \sigma(z_i) = \frac{e^{-z_i}}{1+e^{-z_i}}$.

Thus, $\sigma(z_i) \cdot (1 - \sigma(z_i)) = \frac{e^{-z_i}}{(1+e^{-z_i})^2}$.

So, $\frac{\partial \hat{y}_i}{\partial z_i} = \hat{y}_i \cdot (1 - \hat{y}_i)$.

$$\frac{\partial z_i}{\partial w} = x_i$$

Thus, $\nabla_w F_i(w) = \left(-\frac{y_i}{\hat{y}_i} + \frac{1-y_i}{1-\hat{y}_i}\right) \cdot \hat{y}_i \cdot (1-\hat{y}_i) \cdot x_i = (\hat{y}_i - y_i) \cdot x_i$.

Example of iterations:

Let's assume:

- $w^0 = [0,0]$

- Learning rate $\alpha = 0.1$

- Training set (n = 3):

i_k	x_{i_k}	y_{i_k}
1	[1,0]	0
2	[0,1]	1
3	[1,1]	1

Iteration 1: Step 1: Choose a random data, for example: $i_l = 3$:

$$x_3 = [1,1], y_3 = 1$$

Calculate: $z_i = w^{0T} x_i = 0 \cdot 1 + 0 \cdot 1 = 0$

$$\hat{y}_i = \sigma(z_i) = \frac{1}{1+e^{-0}} = 0.5$$

Gradient: $\nabla_w F_i(w) = (\hat{y}_i - y_i) \cdot x_i = (0.5 - 1) \cdot [1,1] = [-0.5, -0.5]$

$$\text{Step 2: Update: } w^1 = w^0 - \alpha \nabla_w F_{i_1}(w) = [0, 0] - 0.1 \cdot [-0.5, -0.5] = [0.05, 0.05]$$

Iteration 2: Step 1: Choose a random data, for example: $i_2 = 1$:

$$x_1 = [1, 0], y_1 = 0$$

$$\text{Calculate: } z_{i_2} = w^{1T} x_{i_2} = 0.05 \cdot 1 + 0.05 \cdot 0 = 0.05$$

$$\hat{y}_{i_2} = \sigma(0.05) = \frac{1}{1 + e^{-0.05}} \approx 0.5125$$

$$\text{Gradient: } \nabla_w F_{i_2}(w) = (\hat{y}_{i_2} - y_{i_2}) \cdot x_{i_2} = (0.5125 - 0) \cdot [1, 0] = [0.5125, 0]$$

$$\text{Step 2: Update: } w^2 = w^1 - \alpha \nabla_w F_{i_2}(w) = [0.05, 0.05] - 0.1 \cdot [0.5125, 0] = [-0.00125, 0.05]$$

Iteration 3: Step 1: Choose a random data, for example: $i_3 = 2$:

$$x_2 = [0, 1], y_2 = 1$$

$$\text{Calculate: } z_{i_3} = w^{2T} x_{i_3} = -0.00125 \cdot 0 + 0.05 \cdot 1 = 0.05$$

$$\hat{y}_{i_3} = \sigma(0.05) = \frac{1}{1 + e^{-0.05}} \approx 0.5125$$

$$\text{Gradient: } \nabla_w F_{i_3}(w) = (\hat{y}_{i_3} - y_{i_3}) \cdot x_{i_3} = (0.5125 - 1) \cdot [0, 1] = [0, -0.4875]$$

$$\text{Step 2: Update: } w^3 = w^2 - \alpha \nabla_w F_{i_3}(w) = [-0.00125, 0.05] - 0.1 \cdot [0, -0.4875] = [-0.00125, 0.09875]$$

Iteration 4: Step 1: Choose a random data, for example: $i_4 = 1$:

$$x_1 = [1, 0], y_1 = 0$$

$$\text{Calculate: } z_{i_4} = w^{3T} x_{i_4} = -0.00125 \cdot 1 + 0.09875 \cdot 0 = -0.00125$$

$$\hat{y}_{i_4} = \sigma(-0.00125) = \frac{1}{1 + e^{-0.00125}} \approx 0.4996875$$

$$\text{Gradient: } \nabla_w F_{i_4}(w) = (\hat{y}_{i_4} - y_{i_4}) \cdot x_{i_4} = (0.4996875 - 0) \cdot [1, 0] = [0.4996875, 0]$$

Step 2: Update:

$$w^4 = w^3 - \alpha \nabla_w F_{i_4}(w) = [-0.00125, 0.09875] - 0.1 \cdot [0.4996875, 0] = [-0.05121875, 0.09875]$$

Iteration 5: Step 1: Choose a random data, for example: $i_5 = 2$:

$$x_2 = [0, 1], y_2 = 0$$

Calculate: $z_{i_5} = w^{4T} x_{i_5} = -0.05121875 \cdot 0 + 0.09875 \cdot 1 = 0.09875$

$$\hat{y}_{i_5} = \sigma(0.09875) = \frac{1}{1 + e^{-0.09875}} \approx 0.5247$$

$$\text{Gradient: } \nabla_w F_{i_5}(w) = (\hat{y}_{i_5} - y_{i_5}) \cdot x_{i_5} = (0.5247 - 1) \cdot [0, 1] = [0, -0.4753]$$

Step 2: Update:

$$w^5 = w^4 - \alpha \nabla_w F_{i_5}(w) = [-0.05121875, 0.09875] - 0.1 \cdot [0, -0.4753] = [-0.05121875, 0.14628]$$

It should be noted that the iterative process does not terminate here. The purpose of this example is merely to demonstrate how the algorithm operates over a few iterations.

In a manner similar to coordinate descent methods, the design problem in Stochastic Gradient Methods (SGM) involves choosing one data point i at each iteration. Also, better convergence rates are achieved by selecting i uniformly at random instead of cycling through the data. The cost per iteration is influenced solely by the number of parameters p rather than the number of data points n .

It is important to note that selecting a data point randomly leads to an **unbiased estimate of the gradient**, when equation (5.4) is interpreted as an empirical approximation of an expected risk function that drives the optimization process:

$$\min_{x \in \mathbb{R}^p} \left\{ F(x) \equiv \mathbb{E}_{\xi} [F_{\xi}(x)] \right\} \quad (5.5)$$

Here, the expectation is taken with respect to the sampling distribution of the indices ξ .

Stochastic gradient methods can directly optimize the expected risk minimization problem, especially in cases where we are able to sample from the true underlying data distribution. This provides the foundation for their well-established generalization capabilities in machine learning.

Importantly, this concept of **unbiased gradient estimation** allows stochastic gradient descent to be effective not only for decomposable objectives such as (5.4), but also for more **general convex optimization problems**.

The standard Stochastic Gradient (SG) method typically employs a **decreasing sequence of step sizes** $\{\alpha_k\}$. This approach leads to convergence rates of $\mathcal{O}(1/\epsilon)$ or $\mathcal{O}(1/\sqrt{\epsilon})$, which are comparable to those of the subgradient method and can be relatively slow.

However, when a **constant step size** is used in each iteration, the stochastic gradient method is capable of **rapidly reducing the initial error**.

Historically, tuning stochastic gradient descent methods has been quite challenging. However, recent findings indicate that employing *large step sizes* along with *weighted averaging of the iterates* can lead to optimal convergence rates while remaining resilient to variations in step size and modeling assumptions. For instance, recent research has demonstrated that an averaged stochastic gradient iteration with a constant step size can achieve an $\mathcal{O}(1/\epsilon)$ convergence rate, even in the absence of strong convexity, provided certain joint self-concordance-like and Lipschitz gradient conditions are met. Additionally, there have been noteworthy advancements in stochastic algorithms that attain linear convergence rates for strongly convex problems, in case when the dataset is finite.

5.3 Randomized linear algebra

For this subsection reference [14] is consulted.

Randomized linear algebra encompasses a set of techniques that leverage randomness to simplify and accelerate linear algebra computations. In large-scale data problems, fundamental linear algebra operations such as matrix decompositions (e.g., eigenvalue decomposition, singular value decomposition, and Cholesky decomposition) and matrix-matrix multiplications often become significant computational bottlenecks due to their superlinear complexity growth with respect to matrix dimensions. However, when the corresponding matrix objects possess low-rank structures (For example matrix $M \in \mathbb{R}^{p \times p}$ has low-rank form if exists $r \ll p$ such that $M \approx UV^T$ where $U \in \mathbb{R}^{p \times r}$, $V \in \mathbb{R}^{p \times r}$), the efficiency of these methods improves consistently. By exploiting these low-rank structures, randomized methods can provide fast approximations while substantially reducing computational costs, making them highly valuable for processing large datasets.

Randomized linear algebra methods focus on approximating a matrix $M \approx Q(Q^T M)$ with $Q \in \mathbb{R}^{p \times r}$. Alternatively, these methods can construct a low-rank representation by selecting subsets of columns or rows, which enhances computational efficiency. Employing a *randomized* approach allows for better control over error distribution. This concept is applicable to matrices of any size and takes advantage of well-established computational routines available in most programming languages.

We present three key effects of incorporating randomized linear algebra routines into optimization processes.

- 1) The first impact is the ability to **speed up the computation of proximity operators** for functions that rely on the spectral values of a matrix.
- 2) Secondly, this approach is effective for obtaining **unbiased gradient estimates** for matrix objects when randomization is applied appropriately. As a result, it is relevant to all stochastic gradient algorithms.

- 3) Finally, the randomized approach can be employed to **sketch objective functions**, allowing for their approximation. This leads to significantly cheaper iterations with exact first-order methods while still maintaining accuracy guarantees for the actual objective.

Algorithm 7 Randomized low-rank approximation

Require: $M \in \mathbb{R}^{p \times p}$, integer r

- 1: Draw $A \in \mathbb{R}^{p \times r}$ iid $\mathcal{N}(0, 1)$
- 2: Form the $p \times r$ matrix $B = MA$
- 3: Construct an $p \times r$ matrix Q whose columns form an orthonormal basis for the range of B (e.g. using the QR factorization $QR = B$)
- 4: $C = M^T Q$
- 5: **return** $\hat{M}_{(r)} = QC^T$

Algorithm 7 outlines a method for obtaining a low-rank approximation of a matrix $M \in \mathbb{R}^{p \times p}$ using randomized techniques. The goal is to approximate M with a matrix $\hat{M}_{(r)}$ of rank r . Here's a step-by-step breakdown of the algorithm:

Input Requirements:

- The input matrix $M \in \mathbb{R}^{p \times p}$ (which we want to approximate)
- An integer r specifying the desired rank for the approximation. (i.e., what rank do we want the new, approximated matrix to have).

Step 1: Random Sampling:

A random matrix $A \in \mathbb{R}^{p \times r}$ is generated. The entries of matrix are independent and identically distributed (iid) from a standard normal distribution $\mathcal{N}(0, 1)$

Step 2: Matrix Multiplication:

Compute $B = MA$.

A projection of the matrix M is made onto the random subspace defined by the columns of the matrix A .

The matrix B serves to approximate the action of the original matrix M in a lower-dimensional subspace, preserving its most significant components.

Step 3: QR Decomposition:

Perform a QR decomposition on B to obtain Q and R (with $QR = B$).

Here we make a matrix Q which has orthogonal columns (columns which are normalized and mutually orthogonal) and every vector from the range of B may be represented as a linear combination of the columns of Q .

R is uppertriangular matrix.

Step 4: Matrix Multiplication:

Compute $C = M^T Q$.

This gives a kind of matrix projection M to the space defined by the matrix Q .

Step 5: Construct the Low-Rank Approximation:

Return $\hat{M}_{(r)} = QC^T = Q(Q^T M)$

This matrix $\hat{M}_{(r)}$ is the rank- r approximation of M . It retains the most significant components of M while discarding the less significant ones.

6 The role of parallel and distributed computation

For this section literature [6] is used.

To meet the huge computational and storage needs of Big Data without high power costs, we need to increasingly lean on *parallel* and *distributed* computing.

Although first-order methods are well-suited for achieving significant performance speed-ups, two challenges arise when using distributed and heterogeneous hardware:

1) Communication:

Inefficient or unreliable communication between computers and within the local memory hierarchy can greatly hinder the numerical efficiency of first-order methods. There are two main strategies to tackle these issues. The first involves designing algorithms that reduce the need for communication. The second approach eliminates the need for a master vector x^k , allowing each machine to operate with its own local copy, which converges to a consensus solution x^* .

2) Synchronization:

To effectively execute computations in a distributed manner, first-order methods need to synchronize the activities of various computers that rely on the same vector x^k during each iteration. However, this coordination can become inefficient, especially when one machine

takes significantly longer than the others. To address this fundamental synchronization challenge, *asynchronous algorithms* permit updates to occur using outdated versions of their parameters.

In the continuation will be outlined several important advancements related to first-order methods in this context.

6.1 Embarrassingly parallel first-order methods

First-order methods can greatly enhance their performance through the use of parallel computing. Such systems typically consist of uniform processing nodes located in close proximity to one another, ensuring reliable communication between them. The term *embarrassingly parallel* describes an optimal situation for parallelization, in which a task can be divided into independent calculations that can be executed simultaneously in a consistent manner.

An important example of an embarrassingly parallel computation is the *calculation of the gradient vector*, particularly when the objective function naturally decomposes into independent components, as illustrated in equation (5.4).

In this case, the gradient computation can be distributed across multiple machines. Specifically, we can process each F_i with one of m computers, where i indexes the individual functions or data points. Each machine handles only a subset of the data, with $\mathcal{O}(n/m)$ computations required locally, where n is the total number of data points. Since each function F_i corresponds to a particular data point, each machine stores its assigned data locally-specifically, the subset of with $\mathcal{O}(n/m)$ data samples that are relevant to the functions it is processing. This local storage minimizes the need for data sharing between machines during the computation phase. Once the local gradients are computed, the machines communicate their results to a central coordinator (or master node), which aggregates the individual gradients to compute the final gradient vector. This communication step is relatively simple since it only involves sending the local gradients to the central node. The central node then combines these individual gradient components, to obtain the full gradient. In an ideal scenario, this parallelized approach achieves linear speed-up, meaning the time required to compute the final gradient is reduced by a factor of m assuming negligible communication overhead.

In addition to parallelizing the standard gradient method for smooth problems, a framework that allows for embarrassingly parallel computation in the context of *non-smooth* problems can be derived through an artificial reformulation of equation (5.4). The idea is to express optimization problem as:

$$\min_{x, x_1, \dots, x_n} \left\{ \frac{1}{n} \sum_{i=1}^n F_i(x_{(i)}) : x_{(i)} = x, i=1, \dots, n \right\} \quad (6.1)$$

where $x_{(i)}$ are local variables and x is a common global variable. The constraint is that all the local variables should be equal to the global variable. This is known as the **global consensus problem**, since all the local variables should *agree*, i.e., be equal.

Formulating the problem as shown in expression (6.1) enables the use of decomposition techniques, such as Algorithm 8.

Algorithm 8 Decomposition algorithm to solve (6.1);

Initialization: $\beta > 0$, $x_{(i)}^0 = 0$ for $i = 1, \dots, n$.

Step 1: $x^{k+1} = \frac{1}{n} \sum_{i=1}^n \text{prox}_{\beta F_i}(x_{(i)}^k)$

Step 2: **for** $i = 1$ to n **do**

$$x_{(i)}^{k+1} = 2x^{k+1} - x^k + x_{(i)}^k - \text{prox}_{\beta F_i}(x_{(i)}^k)$$

end for

We use the following notations in the algorithm:

$x_{(i)}^k$ - local variable for node i , in iteration k

x^k - global variable in iteration k

$\text{prox}_{\beta F_i}$ - proximal operator for function F_i with a parameter β

Explanation of algorithm step by step:

Initialization: Set $x_{(i)}^0 = 0$ for $i = 1, \dots, n$. Parameter β is given.

Every iteration k looks like following:

Step 1: *Updating a global variable*

Firstly, every node i calculate proximal operator of its function F_i at the current point $x_{(i)}^k$ - $\text{prox}_{\beta F_i}(x_{(i)}^k)$, then all these results are sent to some central place where the results are added up and divided by $\frac{1}{n}$.

Step 2: *Updating local variables*

For every node $i = 1, \dots, n$, $x_{(i)}^{k+1} = 2x^{k+1} - x^k + x_{(i)}^k - \text{prox}_{\beta F_i}(x_{(i)}^k)$.

This is calculated for each node independently - fully *parallel*.

This update uses:

- the **previous consensus** value x^k ,
- the **current consensus** value x^{k+1} ,
- the **previous local value** $x_{(i)}^k$,
- the **proximal operator** of the local function F_i .

The iterative process continues until a predefined stopping criterion is satisfied. For example, the algorithm can be terminated when the difference between the local variables and the consensus variable becomes sufficiently small, i.e., when a satisfactory level of agreement is achieved. Additionally, termination is possible when the objective function value stabilizes, meaning that changes between consecutive iterations are negligible.

This decomposition concept serves as the foundation for the highly scalable *massively parallel consensus ADMM algorithm*, which is particularly effective for optimization problems when $n > 2$.

Fortunately, there are a variety of powerful computing models and software frameworks available that allow us to quickly implement these concepts. Frameworks like MapReduce, Hadoop, Spark, Mahout, MADlib, SystemML, and BigInsights, along with high-level languages such as Pig, Hive, and Jaql, enable the efficient parallel execution of optimization tasks. These tools handle all aspects of communication and data transfer across the system, ensuring that tasks are distributed and managed effectively. Additionally, they provide built-in redundancy and fault tolerance, ensuring that the system remains robust even in the event of failures. This combination of scalability, parallelism, and fault tolerance makes these frameworks ideal for large-scale optimization problems.

6.2 First-order methods with reduced or decentralized communications

For this subsection literature [4] is consulted.

In large-scale systems, transmitting the gradient or its components to a central location can often lead to a communication bottleneck. For this problem, *coordinate descent methods* offer a structured approach to minimize communication overhead. The core idea is to perform multiple coordinate descent updates simultaneously, in parallel, across different processors. This approach reduces communication by ensuring that each processor only needs to send a *single coordinate update* and receive updates only from the *coordinates that have changed*, rather than the entire gradient.

In case when the objective function is *decomposable*, this parallel approach effectively becomes an embarrassingly parallel extension of the serial coordinate descent algorithm. This method

remains convergent, though it may require a smaller step size compared to the serial version. However, when the objective function is non-separable, this parallelization does not always result in improved performance.

Remarkably, it is possible to decentralize the communication needs of gradient-based methods for decomposable objectives with just a few simple adjustments.

First of all, we consider a consensus optimization problem of the form (5.4) defined over a connected network of n agents. These problems in multi-agent networks appear in various applications, including mobile computing, coordination of self-driving cars, cognitive radio networks, and collaborative data mining. Here each agent i holds a private function F_i , which encodes the agent's data and objective. The agents will work together to collaboratively find the minimizer, while each agent can only communicate with its neighbors in the network. This decentralized computational approach eliminates the need for a central data aggregation node and minimizes long-distance communication, thereby enhancing scalability and improving load distribution across the network. The following algorithms focus on the **synchronous setting**, where all agents perform their updates simultaneously at fixed time intervals.

An example of algorithm with decentralized communication is **Decentralized Gradient Descent (DGD)** algorithm. It is described by iteration:

$$x_{(i)}^{k+1} = \sum_{j=1}^n w_{ij} x_{(j)}^k - \alpha_k \nabla F_i(x_{(i)}^k), \text{ for agent } i = 1, \dots, n.$$

Here represents $x_{(i)}^k \in \mathbb{R}^p$ the local copy of the vector x maintained by agent i at iteration k , $W=[w_{ij}] \in \mathbb{R}^{n \times n}$ is a symmetric mixing matrix, where each element w_{ij} indicates how much agent j 's state influences agent i 's state during an update. Specifically, w_{ij} is the weight of the connection between agent i and agent j . The entries w_{ij} are non-negative, meaning that the influence between agents is non-negative. Furthermore, w_{ij} is nonzero if and only if agents i and j are either neighbors or identical, and zero otherwise (i.e., if there is no direct communication between agents, meaning that agent i does not use agent j 's information). From the symmetry of W ($w_{ij} = w_{ji}$) we can conclude that neighboring agents influence each other. Matrix W satisfies $\text{null}\{I-W\} = \text{span}\{\mathbf{1}\}$ and $\sigma_{\max}\left(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) < 1$. From the condition $\text{null}\{I-W\} = \text{span}\{\mathbf{1}\}$, it follows that W has a single eigenvalue equal to 1, and the corresponding eigenvector is the vector of all ones, $\mathbf{1}$. In the context of consensus problems, the vector $\mathbf{1}$ is a natural candidate for the eigenvector. This is because, in a distributed system, if all agents (represented by the components of a vector v) are in agreement, the state $v = \mathbf{1}$ naturally represents the scenario where all agents have the same value. This condition guarantees the existence of a unique consensus state where all agents converge to the same value. Specifically, the condition $\sigma_{\max}\left(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T\right) < 1$ implies that 1 is the largest eigenvalue of W , and all other eigenvalues of W are strictly smaller than 1 in absolute value. This is crucial because it ensures that the system will eventually converge to the consensus state. In the other words, this condition is a key factor in guaranteeing that decentralized gradient descent leads to global consensus across all agents.

When a fixed step size $\alpha_k \equiv \alpha$ is used, DGD has inexact convergence. Specifically, for each agent i , $x_{(i)}^k$ converges to a point within an $\mathcal{O}(\alpha)$ - neighborhood of a solution to equation (5.4), with these points potentially differing across agents. However, by appropriately reducing the step size α_k , exact convergence can be achieved, meaning that each $x_{(i)}^k$ converges to the same exact solution. The trade-off, though, is that smaller values of α_k lead to slower convergence, both theoretically and in practice.

In case when $\alpha_k = \frac{1}{k^{1/3}}$ (α_k diminishes over time as k increases) and ∇F_i 's are Lipschitz continuous and bounded the objective convergence rate slows down to $\mathcal{O}\left(\frac{1}{k^{2/3}}\right)$ [9]. In terms of the

algorithm's convergence rate, this means that as the algorithm progresses, the decrease in the objective function value becomes slower over time. Specifically, the convergence rate is limited by $\mathcal{O}\left(\frac{1}{k^{2/3}}\right)$, meaning the distance to the optimal solution shrinks at a rate proportional to $k^{-2/3}$.

This rate of convergence is slower compared to other methods where the convergence rate could be $\mathcal{O}(1/k)$, which indicates a faster approach to the optimal solution.

The method is **decentralized** because each agent i updates its local variable $x_{(i)}^{k+1}$ by combining its own local gradient $\nabla F_i(x_{(i)}^k)$ with the weighted average of the states of its neighbors (represented by the mixing matrix W). Therefore, agents only exchange information with their neighbors and do not require a central coordinator to perform the updates.

Another example is **EXTRA** — An **Exact first-order algorithm for decentralized consensus optimization**. "Exact" refers to the ability of method to converge to the exact solution. EXTRA allows the use of a fixed, large step size that remains constant regardless of the network size, and it operates with synchronized iterations. Each agent's local variable converges uniformly and collaboratively to the exact minimizer of the function F .

Algorithm 8 EXTRA- Exact first-order algorithm

Choose $\alpha > 0$ and mixing matrices $W \in \mathbb{R}^{n \times n}$ and $\bar{W} \in \mathbb{R}^{n \times n}$. Pick any $x^0 \in \mathbb{R}^{n \times p}$;

Step 1: $x^1 = Wx^0 - \alpha \nabla F(x^0)$

Step 2: **for** $k = 0, 1, \dots$ **do**

$$x^{k+2} = (I + W)x^{k+1} - \bar{W}x^k - \alpha [\nabla F(x^{k+1}) - \nabla F(x^k)]$$

The expression for Step 1 of the EXTRA algorithm can be written in a more compact form as follows:

$$x_{(i)}^1 = \sum_{j=1}^n w_{ij} x_{(j)}^0 - \alpha \nabla F_i(x_{(i)}^0), i = 1, \dots, n.$$

Also Step 2 can be written in following form:

$$x_{(i)}^{k+2} = x_{(i)}^{k+1} + \sum_{j=1}^n w_{ij} x_{(j)}^{k+1} - \sum_{j=1}^n \bar{w}_{ij} x_{(j)}^k - \alpha [\nabla F_i(x_{(i)}^{k+1}) - \nabla F_i(x_{(i)}^k)], i = 1, \dots, n$$

Each agent calculates the gradient $\nabla F_i(x_{(i)}^k)$ at each iteration k and uses this gradient for two purposes: once to update $x_{(i)}^{k+1}$ and again to update $x_{(i)}^{k+2}$. For the chosen matrix $\bar{W} = \frac{W+I}{2}$, each agent computes the weighted sum $\sum_{j=1}^n w_{ij} x_{(j)}^k$ once per iteration as well. Following suppositions are important to converge analysis.

Assumption 1 Let's consider a connected network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of a set of agents $\mathcal{V} = \{1, 2, \dots, n\}$ and a set of undirected edges \mathcal{E} . The mixing matrices W and \bar{W} satisfy:

1. $w_{ij} = \bar{w}_{ij} = 0$ in case when $i \neq j$ and $(i, j) \notin \mathcal{E}$ (There is no direct communication between agents)
2. $W = W^T, \bar{W} = \bar{W}^T$ (These matrices are symmetric)
3. $\text{null}\{W - \bar{W}\} = \text{span}\{\mathbf{1}\}, \text{span}\{\mathbf{1}\} \subseteq \text{null}\{I - W\}$
4. $\bar{W} > 0$ and $\frac{W+I}{2} \geq \bar{W} \geq W$.

Points 2–4 of Assumption 1 lead to the conclusion that $\text{null}\{I - W\} = \text{span}\{\mathbf{1}\}$ and the eigenvalues of W are within the range $(-1, 1]$, which are standard assumptions for DGD. Consequently, the extra assumptions apply only to \bar{W} . In fact, EXTRA can operate with the same W used in DGD, simply setting $\bar{W} = \frac{W+I}{2}$, which satisfies Point 4.

EXTRA is widely recognized for having the fastest convergence rates among current first-order decentralized algorithms designed for decentralized consensus optimization when objectives are convex and have Lipschitz continuous gradient. In particular, if the functions F_i are convex and have Lipschitz continuous gradients, the EXTRA algorithm exhibits an ergodic convergence rate of $\mathcal{O}(1/k)$ with respect to the first-order optimality residual. In the case where the average function F is additionally (restricted) strongly convex, EXTRA converges to an optimal solution at a linear rate, $\mathcal{O}(C^{-k})$, for some constant $C > 1$.

6.3 Asynchronous first-order methods with decentralized communications

The gradient and decomposition methods mentioned above still require global synchronization to solve decomposable problems like (5.4). For example, the gradient algorithm computes the gradient precisely with respect to one or more examples at x^k , and then the updates for x^{k+1} are synchronized in sequence. On the other hand, stochastic gradient algorithms, when addressing problems like (5.4), only use an approximate gradient. This approximation makes these algorithms more resilient to outdated information, which can occur in *asynchronous* settings.

Recent research has confirmed this robustness. For instance, the work in [10] models a **lock-free shared-memory system** where stochastic gradient updates are performed independently by each processor. Although the system still maintains a global vector x , each processor can update it without coordinating with the others, using its cached version of x . Under certain conditions, this asynchronous procedure can preserve the convergence of stochastic gradient methods and achieve significant speed-ups when multiple cores are available.

This lock-free memory model is also applicable to stochastic parallel coordinate descent methods [11].

Moreover, first-order algorithms with randomization have been shown to be effective in asynchronous and decentralized environments, even when communication failures occur [12].

7 Conclusion

In light of the growing complexity of big data problems, it is evident that traditional methods of designing convex optimization algorithms must undergo a significant transformation. The increasing scale of these problems demands a shift in how we approach computational efficiency, pushing us to reconsider conventional algorithmic choices and adopt more innovative strategies. To address the challenges posed by larger convex optimization problems, this work highlights the importance of recognizing and exploiting *structure-dependent* trade-offs in algorithmic approximations. Such insights are crucial in enabling us to solve larger-scale problems without an overwhelming increase in computational resources.

In this work, we primarily focused on **unconstrained convex optimization problems**. Although explicit constraints were not the central topic, it is important to note that **constraints can be naturally incorporated** within the same optimization framework using **proximal gradient methods**.

A particularly interesting case arises when the **indicator function of a convex set S** is used as the non-smooth component of the objective. This allows us to rewrite a constrained problem of the form: $\min_{x \in S} f(x)$ as an unconstrained one: $\min_x f(x) + g(x)$.

It's important to mention that practical enhancements such as adaptive step-size selection and momentum parameter restarting have proven to be highly effective in the context of first-order optimization methods. Although these heuristic improvements do not guarantee better worst-case theoretical convergence rates, they require minimal additional computational cost and do not rely on prior knowledge of problem-specific constants like the Lipschitz constant or the strong convexity parameter.

Empirical evaluation in the context of the **LASSO formulation** confirmed that accelerated methods, although theoretically superior, often exhibit oscillatory behavior and stagnation in later stages of optimization — particularly when no adaptive restart is applied. In this setting, it was observed that **(proximal) gradient descent with adaptive step-size** outperforms the basic accelerated method without adaptive step-size and restart, particularly in later stages where it more effectively exploits the solution's sparsity. However, the **best performance is achieved by the accelerated method combining adaptive step-size and restart**, which demonstrates the fastest and most stable convergence overall.

This behavior can be explained by the method's ability to more quickly recognize the structure of the solution — namely, sparsity — which non-adaptive accelerated methods fail to exploit efficiently. In other words, although accelerated methods should theoretically be faster, in practice they often show limited progress in later stages, whereas adaptive proximal gradient methods leverage solution information more effectively and converge more robustly.

These results highlight the value of combining theoretical approaches with practical heuristics, and motivate further research into more robust and automatically adaptive algorithms that bring together the best of both worlds — theoretical guarantees and practical efficiency.

Additionally, experiments with the **heteroskedastic LASSO (hLASSO) formulation** revealed that the numerical efficiency of first-order methods also strongly depends on selecting an appropriate **smoothness structure for the function f** . In this case, f is **self-concordant**, but **does not have a Lipschitz-continuous gradient**. When the **first-order method** is tailored to this self-concordant structure, it can determine optimal step sizes automatically, outperforming methods that rely on the Lipschitz assumption — including accelerated variants with adaptive step-size and restart. In other words, efficient optimization in composite problems does not depend solely on algorithmic improvements, but also on the proper formulation of the optimization problem — that is, on how the objective function is decomposed into smooth and non-smooth components, and whether its theoretical properties, such as Lipschitz smoothness or self-concordance, are correctly identified and exploited in the algorithm design.

It turned out to be useful to reformulate the composite form into the proximal form. This form can deal with **non-smooth and non-Lipschitz objective functions** which usually occur in many applications such as graph learning, Poisson imaging, and robust principal component analysis (RPCA). For its solutions it is used ADMM algorithm. There are two warnings about ADMM. First, step 2 in Algorithm 3 has to be *numerically solved* in general except in case when $\phi^T\phi$ is effectively diagonalizable. Numerical solution of subproblems presents a challenge, as it requires additional computations in each iteration, thereby increasing computational complexity and overall execution time. Moreover, such numerical procedures complicate implementation and, if not solved with sufficient accuracy, can negatively affect the stability and convergence of the algorithm. Secondly, if we *naïvely* extend ADMM to handle more than two objective terms we might not be able to guarantee convergence. For this problem, *dual decomposition* techniques can be used to treat the multiple terms in the objective as individual problems and simultaneously solve them in *parallel*.

In theory, first-order methods provide a simple and efficient framework for solving large-scale optimization problems, but their application can become challenging as the dimensionality of the problem increases. These methods often require precise numerical computations at each iteration, which can significantly increase computational complexity in practice. Nevertheless, one of their key advantages lies in their robustness to approximations, making them well-suited for optimizing components such as gradients and proximal operators. This property enables the development of scalable and efficient algorithms that remain applicable even in environments with extremely large datasets. In this thesis, two such algorithmic approaches are explored: **Coordinate Descent Methods** and **Stochastic Gradient Methods**.

We saw that **selection strategy for the coordinate k** plays a critical role in the efficiency of coordinate descent methods. For instance, choosing the coordinate with the largest directional derivative may seem intuitive; however, this approach requires computing the full gradient at each iteration. Consequently, the computational cost becomes comparable to that of traditional gradient-based methods, thereby diminishing the advantages of coordinate descent. Moreover, it can be

shown that the convergence rate of such coordinate-based methods is generally slower than that of full gradient methods. A widely adopted strategy in coordinate descent methods is the **random selection** of coordinates, where at each iteration a coordinate is chosen uniformly at random from the set $\{1, 2, \dots, p\}$. While this uniform sampling approach is simple and easy to implement, the convergence rate of the method can be further enhanced through *importance sampling*.

Stochastic Gradient Methods are commonly employed to optimize the sample average of a finite training dataset and deal with **decomposable** objective function. The standard Stochastic Gradient (SG) method typically employs a **decreasing sequence of step sizes** $\{\alpha_k\}$. This approach leads to convergence rates which are comparable to those of the subgradient method and can be relatively slow. However, recent findings indicate that employing *large step sizes* along with *weighted averaging of the iterates* can lead to optimal convergence rates. Importantly, the concept of **unbiased gradient estimation** allows stochastic gradient descent to be effective not only for decomposable objectives, but also for more **general convex optimization problems**.

In modern large-scale optimization problems, classical linear algebra techniques often become computationally infeasible due to the high cost of matrix operations. The introduction of **randomized linear algebra** offers a powerful alternative by enabling low-rank approximations that significantly reduce complexity without compromising accuracy.

These techniques bring several key advantages:

- They **accelerate proximity operator computations** for functions that depend on the spectral properties of matrices, thus improving the efficiency of many first-order optimization methods;
- They provide **unbiased gradient estimates**, which are essential for the convergence of stochastic optimization algorithms;
- They enable **sketching of objective functions**, allowing for significantly cheaper iterations while maintaining theoretical guarantees on convergence and solution quality.

In big data optimization problems, traditional algorithms face significant limitations in terms of memory usage and computational time. **Parallel and distributed computing**, when combined with efficient **first-order methods**, offers a way to overcome these challenges by distributing both data and computation across multiple processing units.

Furthermore, the reformulation of optimization problems into a **global consensus framework** enables the extension of parallelization techniques to **non-smooth problems**. By allowing each computational unit to work with its own local variable, while enforcing consensus with a shared global variable, this approach facilitates scalable optimization even in more complex problem settings.

One of the key challenges in scaling gradient-based optimization methods is the **communication bottleneck** caused by the need to transmit the full gradient information to a central coordinator. **Coordinate descent methods** offer a natural solution to this problem. By performing multiple coordinate updates in parallel across distributed processors, communication overhead is

significantly reduced — each processor only transmits its local coordinate update and receives updates for the coordinates that have changed, rather than the entire gradient vector. When the objective function is **decomposable**, this parallelization strategy becomes highly efficient and naturally leads to an **embarrassingly parallel** implementation of the serial coordinate descent algorithm. However, in the case of **non-separable** objective functions, such parallelization does not always guarantee performance improvements.

A second approach to reducing communication costs involves the use of **decentralized optimization algorithms**. This work presents two such methods: **Decentralized Gradient Descent (DGD)** and **EXTRA**. These algorithms enable agents within a network to locally update their copies of the solution using their own data, along with information exchanged only with their immediate neighbors. This decentralized framework eliminates the need for a central coordination node, thereby improving **scalability**, **robustness**, and the overall **efficiency of network resource utilization**.

In comparing decentralized optimization methods, this work highlights key differences between the DGD and EXTRA algorithms, particularly regarding their accuracy and convergence speed. While DGD is simple to implement, it only achieves inexact convergence when using a fixed step size—meaning that agents may converge to slightly different solutions. Exact consensus can be obtained by gradually reducing the step size; however, this leads to significantly slower convergence both theoretically and in practice.

On the other hand, the EXTRA algorithm guarantees exact convergence even with a fixed step size, making it a more efficient solution for decentralized consensus optimization problems. When the objective functions are convex with Lipschitz continuous gradients, EXTRA achieves an ergodic convergence rate with respect to the first-order optimality residual. Moreover, when the average objective function is strongly convex, EXTRA converges to the optimal solution at a linear rate, representing the most favorable class of convergence rates among first-order decentralized methods.

By relying on approximate gradients, stochastic gradient algorithms demonstrate a high degree of resilience to stale information, making them particularly suitable for *asynchronous* and decentralized environments. Furthermore, the use of randomization enhances robustness, allowing for effective performance even in the presence of communication failures — a critical property for real-world large-scale distributed systems.

Biography



Dunja Dragomanović was born on the 14th of March in 1998 in Novi Sad. She finished elementary school "Jovan Jovanović Zmaj" in Sremska Kamenica with a diploma "Vuk Stefanović Karadžić" and then enroll in grammar school "Svetozar Marković" in Novi Sad. In 2017 Dunja finished grammar school with a diploma "Vuk Stefanović Karadžić" and started her education at the Faculty of Sciences, University of Novi Sad. In 2020 she received a Bachelor's degree in Applied Mathematics in Finance with an average 8.89 and enrolled in a Data Science program at the same faculty. In 2022 she passed all exams with an average 9.25 and went to maternity leave. Her sphere of interest is data analysis.

Bibliography

- [1] Stephen Boyd and Lieven Vandenberghe, “*Convex optimization*”, “Cambridge University Press”, 2009. eprint: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
- [2] Yurii Nesterov, “*Lectures on Convex optimization*”, “Springer International Publishing”, 2018. eprint: <https://shuyuej.com/books/Lectures%20on%20Convex%20Optimization.pdf>
- [3] Stephen Boyd, Eric Chu, Jonathan Eckstein, Neal Parikh and Borja Peleato, “*Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*”, In: “*Foundations and Trends in Machine Learning*”, vol. 3, no. 1, pp. 1–122, 2011.
- [4] Qing Ling, Wei Shi, Gang Wu, and Wotao Yin, “*Extra: An exact first-order algorithm for decentralized consensus optimization*”, In: “*SIAM Journal on Optimization*”, vol. 25, no. 2, pp. 944–966, 2015., arXiv:1404.6264v4 [math.OC]
- [5] Stephen Boyd and Neal Parikh, “*Proximal algorithms*”, In: “*Foundations and Trends in Machine Learning*”, vol. 1, no. 3, pp. 123-231, 2013.
- [6] Stephen Becker, Volkan Cevher, and Mark Schmidt, “*Convex Optimization for Big Data*”, In: “*IEEE Signal Processing Magazine*”, vol. 31, no. 5, pp. 32-43, 2014., arXiv:1411.0972v1 [math.OC]
- [7] Jonathan Richard Shewchuk, “*An Introduction to the Conjugate Gradient Method without the Agonizing Pain*”, “Carnegie Mellon University, School of Computer Science”, 1994.
- [8] Volkan Cevher, Anastasios Kyrillidis, Quoc Tran-Dinh, “*Composit Self-Concordant Minimization*”, In: “*Journal of Machine Learning Research*”, vol. 16, no. 12, pp. 371-416, 2015., arXiv:1308.2867v2[stat.ML]
- [9] Dusan Jakovetic, Jose M. F. Moura, Joao Xavier, “*Fast Distributed Gradient Methods*”, In: “*IEEE Transactions on Automatic Control*”, vol. 59, no. 5, pp. 1131–1146, 2014., arXiv: 1112.2972v4[cs.IT]
- [10] Feng Niu, Christopher Re, Benjamin Recht, Stephen J. Wright, “*Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent*”, In: “*Advances in Neural Information Processing Systems*”, vol. 24, pp. 693–701, 2011.
- [11] Peter Richtárik, Martin Takáč, “*Parallel coordinate descent methods for big data optimization*”, In: “*Mathematical Programming: Series A and B*”, vol. 156, no. 1-2, pp. 433-484, 2016., arXiv: 1212.0873v2[math.OC]
- [12] Alekh Agarwal, John C. Duchi, “*Distributed delayed stochastic optimization*”, In: “*Advances in Neural Information Processing Systems*”, vol. 24, pp. 873-881, 2011., arXiv: 1104.5525v1[math.OC]
- [13] Emmanuel Candes, Brendan O’ Donoghue, “*Adaptive Restart for Accelerated Gradient Schemes*”, In: “*Foundations of Computational Mathematics*”, vol.15, no. 3, 715-732, 2015., arXiv: 1204.3982[math.OC]

[14] Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp, “*Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*”, In: “*SIAM review*”, vol. 53, no. 2, pp. 217–288, 2011., arXiv: 0909.4061[math.NA]

[15] Ana Friedlander, Nataša Krejić, Nataša Krklec Jerinkić, “*Lectures on Fundamentals of Numerical Optimization*”, “*Faculty of Sciences, University of Novi Sad*”, ISBN: 978-86-7031-474-0, 2019.

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:

RBR

Identifikacioni broj:

IBR

Tip dokumentacije: Monografska dokumentacija

TD

Tip zapisa: Tekstualni štampani materijal

TZ

Vrsta rada: Master rad

VR

Autor: Dunja Dragomanović

AU

Mentor: dr Dušan Jakovetić

MN

Naslov rada: Konveksna optimizacija za velike podatke

NR

Jezik publikacije: Engleski

JP

Jezik izvoda: Engleski

JI

Zemlja publikovanja: Republika Srbija

ZP

Uže geografsko područje: Vojvodina

UGP

Godina: 2025.

GO

Izdavač: Autorski reprint

IZ

Mesto i adresa: Novi Sad, Prirodno-matematički fakultet, Departman za matematiku i informatiku, Trg Dositeja Obradovića 4

MA

Fizički opis rada: 7 poglavlja, 74 strane, 15 lit. citata, 12 slika, jedna tabela

FO

Naučna oblast: Matematika

NO

Naučna disciplina: Primenjena matematika

ND

Ključne reči: Konveksni skupovi, konveksne funkcije, jako konveksne funkcije, konveksna optimizacija, velika količina podataka, glatke ciljne funkcije, neglatke ciljne funkcije, gradijentni metod, Lipšic neprekidan gradijent, Nesterov ubrzani gradijentni metod, proksimalni gradijentni metod, ubrzani proksimalni gradijentni metod, gornja granica brzine konvergencije, donja granica brzine konvergencije, ADMM algoritam, randomizovane aproksimacije, metoda spuštanja po koordinatama, stohastičke gradijentne metode, randomizovana linearna algebra, randomizovana aproksimacija niskog ranga, paralelno računanje, distribuirano računanje, sinhronizacija, embarrassingly parallel tehnika, decentralizovana komunikacija, decentralizovani gradijentni spust, EXTRA algoritam, konsenzus optimizacija, asinhrono okruženje.

KR

Univerzalna decimalna klasifikacija:

UDK

Čuva se: Biblioteka Departmana za matematiku i informatiku, Prirodno-matematičkog fakulteta, u Novom Sadu

ČU

Važna napomena:

VN

Izvod: U ovom master radu proučavaju se savremeni algoritmi konveksne optimizacije primenljivi na probleme velikih razmera, sa posebnim fokusom na smanjenje računskih, memorijskih i komunikacionih troškova prilikom obrade velike količine podataka. Nakon uvodnog pregleda konveksnih skupova, konveksnih funkcija i osnovnih pojmova optimizacije, analiziraju se metode prvog reda za glatku i kompozitnu konveksnu optimizaciju. Razmatrani su gradijentni metod, Nesterov ubrzani gradijent, kao i proksimalni i ubrzani proksimalni metodi, uz osvrt na njihova teorijska svojstva i praktična poboljšanja, kao što su adaptivni koraci i restart tehnike. Pokazalo se korisno reformulisati kompozitni oblik u proksimalni oblik. Ova formulacija može da obrađuje neglatke i ne-Lipšic neprekidne ciljne funkcije, koje se često javljaju u brojnim primenama. U

ovom slučaju koristi se ADMM algoritam koji kombinuje moćne tehnike proširenog Lagranžijana i dualne dekompozicije. Razmatrane su i randomizovane aproksimacije gradijenta, koje omogućavaju skalabilnost metoda prvog reda u problemima veoma velikih dimenzija. U tom okviru analizirani su stohastički gradijentni metodi, metode spuštanja po koordinatama i tehnike randomizovane linearne algebre, uključujući slučajne niskorangirane aproksimacije. Takođe je razmotrena i embarrassingly parallel tehnika, Termin embarrassingly parallel opisuje idealnu situaciju za paralelizaciju u kojoj se zadatak može podeliti na nezavisne proračune koji se mogu izvršavati istovremeno na više računara, što značajno smanjuje ukupno vreme računanja. Rad obuhvata i algoritme za distribuiranu i decentralizovanu optimizaciju, među kojima su decentralizovani gradijentni spust i EXTRA metod, pogodni za okruženja sa smanjenom komunikacijom. Prikazana analiza naglašava kako pravilna formulacija problema i izbor odgovarajućih algoritama — uključujući randomizovane, paralelne i distribuirane pristupe — mogu značajno poboljšati efikasnost rešavanja velikodimenzionalnih konveksnih optimizacionih problema.

IZ

Datum prihvatanja teme od strane NN veka: 21.11.2025.

DP

Datum odbrane:

DO

Članovi komisije:

KO

Predsednik: dr Maja Jolić, docent, Prirodno-matematički fakultet, Univerzitet u Novom Sadu

Mentor: dr Dušan Jakovetić, redovni profesor, Prirodno-matematički fakultet, Univerzitet u Novom Sadu

Član: dr Nataša Krklec Jerinkić, redovni profesor, Prirodno-matematički fakultet, Univerzitet u Novom Sadu

UNIVERSITY OF NOVISAD
FACULTY OF SCIENCES
KEY WORDS DOCUMENTATION

Accession number:

ANO

Identification number:

INO

Document type: Monograph type

DT

Type of record: Printed text

TR

Contents code: Master's thesis

CC

Author: Dunja Dragomanović

AU

Mentor: Dr. Dušan Jakovetić

MN

Title: Convex optimization for big data

TL

Language of text: English

LT

Language of abstract: English

LA

Country of publication: Republic of Serbia

CP

Locality of publication: Vojvodina

LP

Publication year: 2025.

PY

Publisher: Author's reprint

PU

Publ. place: Novi Sad, Department of Mathematics and Informatics, Faculty of Science and Mathematics, University of Novi Sad, Trg Dositeja Obradovića 4

PP

Physical description: 7 chapters, 74 pages, 15 references, 12 figures, one table

PD

Scientific field: Mathematics

SF

Scientific discipline: Applied mathematics

SD

Key words: Convex sets, convex functions, strongly convex functions, convex optimization, big data, smooth objective functions, nonsmooth objective functions, gradient method, Lipschitz-continuous gradient, Nesterov's accelerated gradient method, proximal gradient method, accelerated proximal gradient method, upper bound on convergence rate, lower bound of convergence rate, ADMM algorithm, randomized approximations, coordinate descent method, stochastic gradient methods, randomized linear algebra, randomized low-rank approximation, parallel computing, distributed computing, synchronization, embarrassingly parallel technique, decentralized communication, decentralized gradient descent, EXTRA algorithm, consensus optimization, asynchronous environment.

KW

Universal decimal classification:

UDC

Holding data: The Library of the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad

HD

Note:

N

Abstract: This master's thesis investigates modern convex optimization algorithms applicable to large-scale problems, with a particular focus on reducing computational, memory, and communication costs when processing large amount of data. After an introductory overview of convex sets, convex functions, and basic optimization concepts, first-order methods for smooth and composite convex optimization are analyzed. The gradient method, Nesterov's accelerated gradient, as well as proximal and accelerated proximal methods are considered, along with a

discussion of their theoretical properties and practical improvements, such as adaptive step sizes and restart techniques. It has been shown to be useful to reformulate the composite form into a proximal form. This formulation can handle nonsmooth and non-Lipschitz continuous objective functions, which frequently appear in many applications. In this context, the ADMM algorithm is employed, combining powerful techniques from the augmented Lagrangian and dual decomposition frameworks. Randomized gradient approximations are also considered, enabling the scalability of first-order methods for high-dimensional problems. Within this framework, stochastic gradient methods, coordinate descent methods, and randomized linear algebra techniques, including randomized low-rank approximation, are analyzed. Furthermore, the embarrassingly parallel technique is discussed. The term embarrassingly parallel refers to an ideal parallelization scenario in which a task can be decomposed into independent computations that can be executed simultaneously on multiple machines, thereby significantly reducing the overall computation time. The thesis also covers algorithms for distributed and decentralized optimization, including decentralized gradient descent and the EXTRA method, suitable for environments with reduced communication. The presented analysis emphasizes how proper problem formulation and the choice of appropriate algorithms — including randomized, parallel, and distributed approaches — can significantly improve the efficiency of solving large-scale convex optimization problems.

AB

Accepted by the Scientific Board on: 21.11.2025.

ASB

Defended on:

DE

Thesis defend board:

DB

Chairperson: Dr Maja Jolić, assistant professor, Faculty of Sciences, University of Novi Sad

Mentor: Dr Dušan Jakovetić, full professor, Faculty of Sciences, University of Novi Sad

Member: Dr Nataša Krklec Jerinkić, full professor, Faculty of Sciences, University of Novi Sad