Imperial College London

Imperial College London

Department of Computing

Mirror Descent and Interacting Mirror Descent

Almost Dimension-Free Convex Optimisation For Non-Euclidean Spaces

Author:
George Yiasemis

Supervisor:
Dr Panos Parpas

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

September 2020

Abstract

Many fundamental optimisation problems in machine learning and artificial intelligence have at the core of their frameworks the search of an optimal set parameters that lead to the minimisation or maximisation of a differentiable convex or non-convex objective function over either constrained or unconstrained feasible sets. A plethora of these problems utilise iterative methods that compute the gradient of the objective function at each iteration and take a step towards the steepest ascent or descent. A widely used optimisation method is Stochastic Gradient Descent. SGD is typically a well-performing algorithm for optimisation over Euclidean vector spaces. However, SGD does not always manage to converge to the optimum due to scaling of the dimensionality of the problem on non-Euclidean spaces, or due to noisy computations of the gradient and frequently proves to be expensive if the optimisation is constrained. In this project we study an other almost dimension-free iterative optimisation algorithm called Mirror Descent and its variant Interacting particles Mirror Descent on six convex frameworks with convex objective functions. Our aim is to observe the performance and convergence properties of these methods and also compare them with the standard Stochastic Gradient Descent method. We show that the mirror descent algorithms can outperform gradient descent in cases of optimisation over constrained sets with certain geometry, such as the probability simplex or the matrix spectrahedron. Additionally, we demonstrate the benefits in terms of convergence when allowing particles to interact in the case of Mirror Descent.

Acknowledgements

Undertaking this MSc individual project has been an unremarkable and challenging experience and it would not have been possible without the inspiration, support and guidance that I received.

First, I would like to thank my supervisor Dr Panos Parpas of the Department of Computing at Imperial College London for taking over the supervision of my proposal and for the support, counsel and recommendations he gave me over the past few months.

Additionally, I would like to thank my family and friends for their continuous help, encouragement and support during this difficult and challenging period.

Finally, I would like to thank both, Dr Robert A Craven, the MSc in Artificial Intelligence coordinator, and Dr Oana Cocarascu, Research Associate at the Department of Computing, for their valuable help and guidance throughout the duration of my studies.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI	Artificial Intelligence
AUC	Area Under the roc Curve
DL	Deep Learning
GD	Gradient Descent
IMD	Interacting Mirror Descent
IMLD	Interacting Mirror Langevin Diffusion
INLA	Interacting Newton Langevin Algorithm
INLD	Interacting Newton Langevin Diffusion
LD	Langevin Diffusions
MCMC	Markov Chain Monte Carlo
MD	Mirror Descent
ML	Machine Learning
MLD	Mirror Langevin Diffusion
NLA	Newton Langevin Algorithm
NLD	Newton Langevin Diffusion
PGD	Projected Gradient Descent
PSD	Positive Definite
SDE	Stochastic Differential Equation
SDP	Semidefinite Progtamming
SGD	Stochastic Gradient Descent
SIMD	Stochastic Interacting Mirror Descent
SL	Statistical Learning
SMD	Stochastic Mirror Descent
SNR	Signal-to-Noise Ratio
tr	Trace

Contents

Chapter 1

Introduction and Outline

1.1 Introduction

The term"optimisation" in Mathematics usually refers to the process of selecting the "best" parameters out of a set of feasible parameters, for example $\mathcal{X}$, which optimise (maximise or minimise) an objective function $f : \mathcal{X} \longrightarrow \mathbb{R}$. A basic optimisation problem has usually one of the two forms:

$$\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \quad \min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}),$$

where $\mathcal{X}$ can be bounded (e.g. a bounded subset of $\mathbb{R}^n$, such as a hyperball or a hypercube) or not bounded (e.g. $\mathbb{R}^n$). If $\mathcal{X}$ is bounded then the optimisation is called "constrained", otherwise it is called "unconstrained". Constrained optimisation techniques can be useful in a plethora of applications, not only in Mathematics, but also in Engineering, Mechanics, Economics, Finance etc. A special case of optimisation is convex optimisation which aims to optimise a convex function f over a convex set $\mathcal{X}$. Otherwise, we call it non-convex optimisation. See Figure 1.1 for an illustrative example.

Optimisation algorithms are key factors in almost every Machine Learning, Statistical Learning, and Deep Learning application and the optimiser is one of the most important contributors to a well-performed model. Aiming to minimise or maximise an objective function, the optimiser plays the role of the decision-making factor who decides how the values of parameters of a model should be updated, i.e. increased, decreased or remain unchanged, and thereby make the model we are training more accurate.

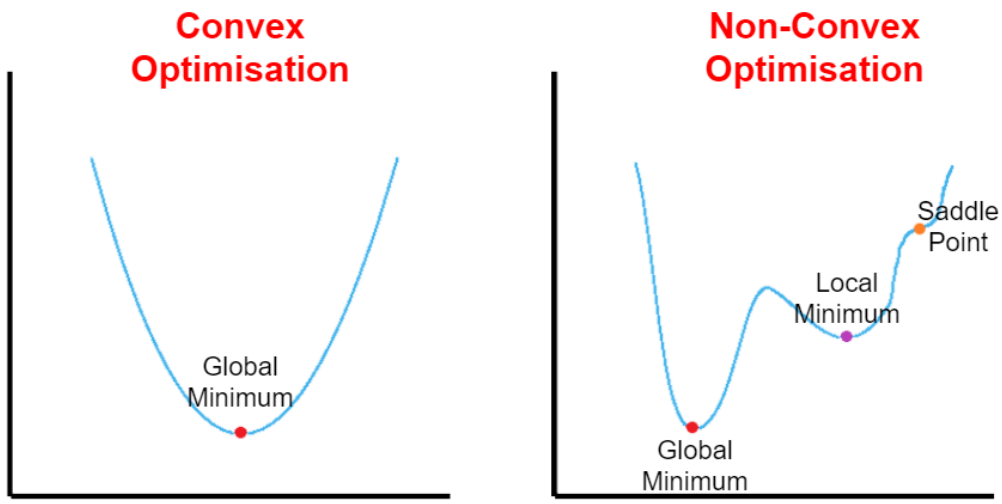

Figure 1.1: Convex Optimisation (Left) and Non-convex Optimisation (Right).

Such parameters can be coefficients $\boldsymbol{m} \in \mathbb{R}^n$ in simple problems such as Linear Regression

$$f(\boldsymbol{m}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{m}^T\boldsymbol{x}_i)^2,$$

or weights or biases $(\boldsymbol{W}, \boldsymbol{b})$ of a layer of an artificial neural network

$$f(\boldsymbol{W}, \boldsymbol{B}) = ||\boldsymbol{Y} - g(\boldsymbol{W}^T\boldsymbol{X} - \boldsymbol{B})||_2^2$$

where g is an activation function such as ReLu (rectified linear unit), the sigmoid or the hyperbolic tangent, etc.

Despite the fact that the vast majority of problems arising in Artificial Intelligence and Statistical Learning are non-convex, in this project we put our focus on convex and strictly convex cases since there is a guarantee of existence of a global (unique) minimum and therefore are simpler to solve. In contrast, an optimisation algorithm in a non-convex case can get stuck on local optima or saddle points (see Figure 1.1 for an illustration).

In AI, ML and DL contexts, Stochastic Gradient Descent is a first-order optimisation algorithm exploited to minimise an objective function by iteratively taking steps in the direction of the steepest descent utilising the negative of the function's gradient (vector of partial derivatives). Gradient Descent then updates the parameters of the model such that at each iteration the value of the objective function is decreasing. However, the size of the update is often controlled by the step size (or learning rate) which is usually treated as a hyperparameter. It should be mentioned that Stochastic Gradient Descent and all of its variants are usually the first choice for an optimiser in many applications of AI.

For an illustration of the performance of Gradient Descent with constant learning rate on a convex problem in $\mathbb{R}^2$ see Figure 1.2.

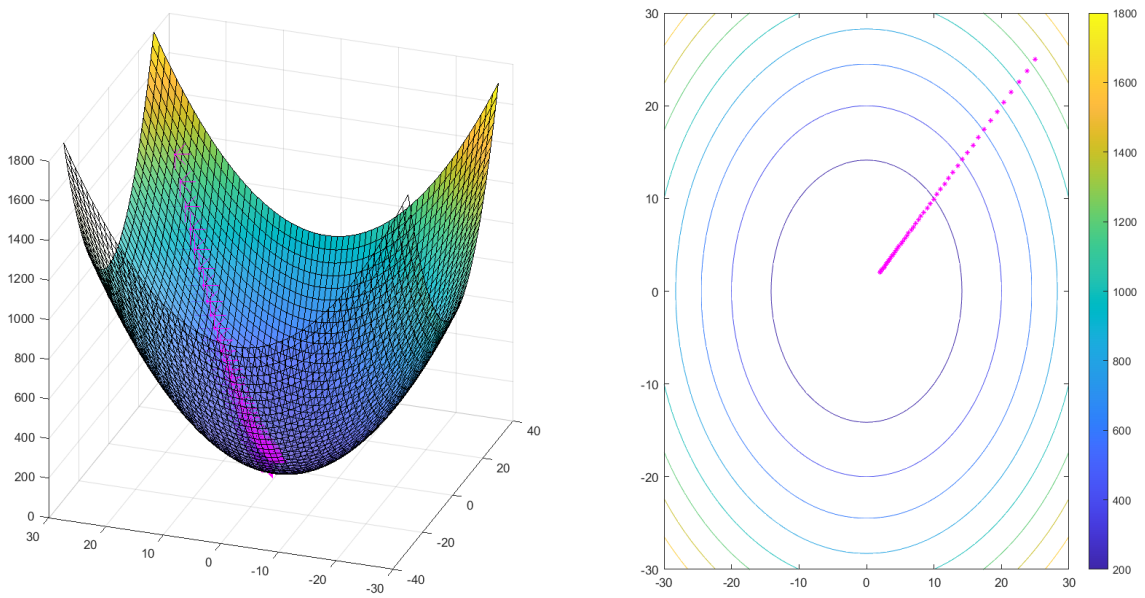

Figure 1.2: Steps of Gradient Descent towards the global minimum for a simple convex function in $\mathbb{R}^2$, Graph of the function (Left) and Contour levels (Right).

In the case of constrained optimisation over a constraint set $\mathcal{X}$, Projected Gradient Descent (PGD) can be exploited. In PGD at each iteration the next point is projected onto $\mathcal{X}$ utilising the negative gradient of the objective function at the previous point. However, projection can be computationally expensive as it requires at each iteration one computation of the gradient and one projection onto $\mathcal{X}$ ([45]) and it is not always feasible.

One of the goals of this individual project is the investigation of another first order optimisation algorithm called Mirror Descent. Mirror Descent is a generalization of PGD that works on both, Euclidean and non-Euclidean geometries. Nemirovskii and Yudin in their work [40] introduced the idea of a "mirror map" function. As it will be explicitly described later in Chapter 4, Mirror Descent converts a constrained problem into an unconstrained problem utilising the mirror map that fits the problem and its geometry, and thus, it removes the requirements of projecting points back onto the constraints set. We also provide illustrations for both methods in Chapter 4.

In their works [5, 13] the authors provide some theoretical results concerning the convergence of both, MD and PGD. In our work we present numerical results (Chapter 5) for six problems (five are constrained and one is unconstrained) which verify their theoretical results.

Another aim of this work is the analysis of an interacting variant of MD, called Interacting Mirror Descent initially proposed in the works [9, 10]. We study the properties of the convergence of the method when it is used to optimise six problems in Chapter 5. We also make comparisons with the case of single particle optimisation (Mirror Descent and Projected Gradient Descent) which are averaged over a number of independent and identical replicas.

1.2 Outline

The rest of this paper is structured into five chapters, Chapter 2 (Related Literature), Chapter 3 (Contribution), Chapter 4 (Preliminaries), Chapter 5 (Experimental Results) and Chapter 6 (Conclusion and Future Work). Here we provide a short description for each of the next chapters.

- **Related Literature.** In this chapter we provide a summary of previous research performed on the topics of this individual project (Gradient Descent, Mirror Descent, Interacting Mirror Descent) as found in the work of related literature.
- **Contribution.** In this chapter we provide a synopsis of the main contributions of this project to the literature.
- **Preliminaries.** In this chapter we provide preliminaries, background notes and some theory on the optimisation methods used in this individual project. In the case of Mirror Descent we provide some examples of frameworks (optimisation on the Simplex and on the Spectrahedron) we are going to be using for our Experiments. Along with the above, we provide step-by-step algorithms for the implementation of (Projected) Gradient Descent, Mirror Descent and its variant Interacting Mirror Descent.
- **Experimental Results.** In this chapter we give insight on the formulation of the six problems we are concerned with. We define each one in detail and provide their formulations. Furthermore, we provide numerical experimental results on all six problems. We additionally, discuss and compare the performance of the three methods we use in this project.
- **Conclusion and Future Work.** In the last chapter we provide some conclusions as well as possible future directions.

Chapter 2

Related Literature

2.1 Gradient Descent

As afformentioned in 1.1 of Chapter 1, Gradient Descent is the most widely used optimisation framework in many Machine Learning and Deep Learning tasks. Zhang in his work [58] collocates some applications of Gradient Descent and its variants (Stochastic Gradient Descent, Mini-batch Gradient Descent, etc) for finding the optimal parameters of Deep Learning models.

Additionally, in the literature there have been proposed many first-order variants of Stochastic Gradient Descent used to optimise objective functions. Some of them are namely Adam, AdaGrad, RMSProp (see Chapter 8 of [22] for a complete guide) which are widely used and seem to work well in cases of unconstrained convex or non-convex optimisation.

As we stated in the previous Chapter, in the case of constrained optimisation over a constraint set $\mathcal{X}$, Projected Gradient Descent is widely used. In PGD at each iteration the next point is projected onto $\mathcal{X}$ utilising the gradient of the objective function at the previous point. PGD has a variety of applications. Some interesting works found in the Machine and Deep Learning literature that utilise PGD as an optimiser are [3, 25, 31, 33, 34, 49, 56, 48, 19, 52, 23, 8, 32]. However, the operation of projecting a single point onto a constraint set can be computationally expensive and not always feasible [45].

One problem we are concerned with in this project in particular, are the works of [51] who propose a new algorithm for clustering called The Laplacian k-modes for clustering. The authors propose an efficient method based on Projected Gradient Descent in [50] for solving this problem over the probability simplex. We will later solve this problem using the Mirror Descent and Interacting Mirror Descent setups.

2.2 Mirror Descent

The algorithm of Mirror Descent, proposed initially by Nemirovskii and Yudin in [40], has become one of the most popular optimisation frameworks in ML and SL. MD and its variants have been used for optimising various convex and non-convex tasks by many authors in the related literature which includes the works in [5, 15, 18, 46, 39, 38, 36, 61, 55, 37, 30, 29, 53].

As mentioned in the introduction, Mirror Descent can be used to transform a constrained problem into an unconstrained one. This property of MD was utilised by the authors who used as a mirror map the "entropy" function to solve optimisation problems on simplex setups. We report here some of the works in the bibliography which we are interested in.

In [28], the authors make use of mirror descent for first-order sampling from a constrained -on the simplex- distribution and show that their method Mirrored Langevin Dynamics outperformed the previous state-of-the art method that utilised Stochastic Gradient Descent. Additionally, Mertikopoulos and Staudigl in [37] solve a version of the traffic assignment problem where the feasible space is again the probability simplex by employing MD dynamics.

Another work we are concerned with is Connected Subgraph Elevated Mean Detection on semi-definite positive programs which includes matrix constrained, convex optimisation. The authors in [1], exploiting Mirror Descent, solve this problem over the Spectrahedron (space of semi-definite positive with fixed trace, analogous to the probability simplex but in $\mathbb{R}^{n\times n}$) and they show that the performance of their Mirror Descent algorithm on Elevated Mean Detection surpasses the performance of the methods previously proposed.

2.3 Interacting Mirror Descent

In the past years there has been a great deal of research in Statistical Theory around interacting particle systems, which they have found applications in many areas. We report some works in the related literature. The authors in [7] use n interacting particles of diffusions for continuous and discrete sampling. In [20], the authors also allow particles to interact in a system of diffusion processes and they show that an increase in interaction results in an increase of the systemic risk. Moreover, in [16], they use a mean interaction field (which we will use here as well) and they propose a maximum likelihood based inference to estimate the parameters of the interaction function induced by a potential energy in an n interacting particles system. Another interesting application of interacting particles in Finance, was presented in [14], where the authors introduce the use of interacting particle systems in the computation of probabilities of simultaneous defaults in large credit portfolios.

In our work here, we also utilise interacting particles in our optimisation setup. We adopt a method from [10], where the authors extend Stochastic Mirror Descent, by proposing a new optimisation framework that estimates the optimum of a function in which they allow N_p particles to interact. The optimisation update rule they propose is given by the following Itô Stochastic Differential Equation in continuous time:

$$\begin{aligned} d\boldsymbol{y}_t^i &= -\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)dt + \sum_{j=1}^{N_p} A_{ij}(\boldsymbol{y}_t^j - \boldsymbol{y}_t^i) + \sigma d\boldsymbol{B}_t^i, \\ \boldsymbol{x}_t^i &= \boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_i^t), \quad i = 1, ..., N_p, \end{aligned}$$

where each particle is characterised by independent Brownian motions $\boldsymbol{B}_t^i$ and $\boldsymbol{A}$ denotes a matrix that contains the interaction weight of each particle. Without further analysis of the equation (see Chapter 4 for more on this equation), we mention some of their findings in the next paragraph.

Among with some theoretical results, they exhibit results on three optimisation problems. The authors using a mean interacting field ($A_{ij} = \frac{1}{N_p}$) show that interaction can speed up optimisation convergence and helps in variance reduction. Even in the case of mini-batch optimisation, they show that for large number of interacting particles the variance of the the algorithm around the optimum can be comparable to that when optimising over a full-batch. In this project we reproduce and verify some of the results of [10].

Chapter 3

Contribution

This MSc individual project aims at researching two optimisation algorithms, defined later on, (Stochastic) Mirror Descent (SMD/MD) and its variant (Stochastic) Interacting Mirror Descent (SIMD/IMD) proposed in [10]. We investigate their convergence properties when applied on convex optimisation frameworks and we make comparisons against the traditional optimisation technique, Projected (Stochastic) Gradient Descent (SGD/GD). We present numerical experimental results on six problems appearing in the related literature from which, for the last three, this project is the first to report results of the application of the discretised version of Interacting Mirror Descent algorithm.

We provide a synopsis of the contributions of this project:

- We provide the reader with a complete collection of the basic background (in Chapter 4) for optimisation with Mirror Descent and its variant Interacting Mirror Descent.

- We reproduce experiments on three constrained optimisation schemes on the probability simplex; for a Linear System, Mini-batch optimisation on a Linear System and the Traffic Assignment problem (see Sections 5.2, 5.3, 5.1 respectively). We compare the performance of (S)GD with that of MD and its interacting variant.

- We demonstrate how interaction in Interacting Mirror Descent can help in variance/noise reduction which is present in cases such us batch optimisation or when samples used for optimisation are drawn from a noisy distribution. Similarly to [10], we show that an increase in interacting particles can lead to a decline of noise/variance.

- Using Mirror Descent and Interacting Mirror Descent we solve numerically two clustering problems (Section 5.4) for random generated data on the probability simplex proposed in [50, 51] which they solve with Stochastic Gradient Descent. We show experimentally that the former, surpasses the latter in terms of performance, convergence and speed of convergence.

- We also solve Elevated Mean detection on two problems (of which one is a real case scenario and one is obtained from randomly generated data) on positive semi-definite matrices on the Spectrahedron previously proposed and solved in [1] with Mirror Descent (see Section 5.5 for the formulation). Here we utilise the Interacting particle Mirror Descent framework. Our numerical results show that the performance of Interacting Mirror Descent surpasses that of the original version of Mirror Descent.

- Lastly, in Section 5.6 we propose a novel algorithm called Interacting Newton Langevin Algorithm (INLA) used to sample from a target distribution. Our method allows N_p particles to interact and attains better performance than the original algorithm Newton Langevin Algorithm (NLA) proposed in the related literature (see [17]). Our algorithm outperforms NLA in convergence speed and we also show that interaction helps in reducing the variance.

Chapter 4

Preliminaries

4.1 Notation

For any two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$:

- $< \cdot, \cdot >_2 : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \mathbb{R}$ will denote the Euclidean inner product in $\mathbb{R}^n$, such that $< \boldsymbol{x}, \boldsymbol{y} >_2 = \boldsymbol{x}^T \boldsymbol{y}$.
- $|| \cdot ||_2 : \mathbb{R}^n \longrightarrow \mathbb{R}_0^+$ will denote the Euclidean $\mathcal{L}2$-norm on $\mathbb{R}^n$, such that
  $|| \boldsymbol{x} ||_2 = \sqrt{< \boldsymbol{x}, \boldsymbol{x} >_2} = \sqrt{\boldsymbol{x}^T \boldsymbol{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}.$
- $|| \cdot ||_1 : \mathbb{R}^n \longrightarrow \mathbb{R}_0^+$ will denote the $\mathcal{L}1$-norm on $\mathbb{R}^n$, such that
  $|| \boldsymbol{x} ||_1 = \sum_{i=1}^{n} |x_i|.$

For any two real matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$:

- $< \cdot, \cdot >_F : \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times n} \longrightarrow \mathbb{R}$ will denote the Frobenius inner product in $\mathbb{R}^{n \times n}$, such that $< \boldsymbol{A}, \boldsymbol{B} >_F = \operatorname{tr}(\boldsymbol{A}^T \boldsymbol{B}) = \boldsymbol{A} \bullet \boldsymbol{B} = \sum_{i,j=1}^{n} A_{ij} B_{ij}$.
- $|| \cdot ||_F : \mathbb{R}^{n \times n} \longrightarrow \mathbb{R}_0^+$ will denote the Frobenius norm on $\mathbb{R}^{n \times n}$, such that
  $$||\boldsymbol{A}||_F = \sqrt{< \boldsymbol{A}, \boldsymbol{A} >_F} = \sqrt{\operatorname{tr}(\boldsymbol{A}^T \boldsymbol{A})} = \sqrt{\sum_{i=1}^{n} \lambda_i(\boldsymbol{A})^2},$$
  where $\lambda_i(\boldsymbol{A})$ denotes the i-th eigenvalue of $\boldsymbol{A}$.
- $|| \cdot ||_1 : \mathbb{R}^{n \times n} \longrightarrow \mathbb{R}_0^+$ will denote the nuclear norm or Shatten-1 norm on $\mathbb{R}^{n \times n}$, such that
  $$||\boldsymbol{A}||_1 = \operatorname{tr}(\boldsymbol{A}^T \boldsymbol{A})^{\frac{1}{2}}.$$

- $\kappa(\cdot) : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}$ will denote the condition number of matrices in $\mathbb{R}^{n\times n}$, such that

$$\kappa(\boldsymbol{A}) = \frac{\lambda_{\max}(\boldsymbol{A})}{\lambda_{\min}(\boldsymbol{A})},$$

where $\lambda_{\max}(\boldsymbol{A})$ and $\lambda_{\min}(\boldsymbol{A})$ denote the maximum and minimum eigenvalues of matrix $\boldsymbol{A}$.

4.2 Background and Problem Setup

4.2.1 Background and Basic Definitions

First, we provide some definitions that are needed to define our main problem in this project. We also provide all necessary definitions essential to outline the optimisation algorithms that we use.

Definition 1
(Convex Set)
A set $\mathcal{X}$ is said to be convex if every point on the line connecting any two points $\boldsymbol{x}$ and $\boldsymbol{z}$ in $\mathcal{X}$ is itself in $\mathcal{X}$. More formally

$$\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{z} \in \mathcal{X} \quad \forall\alpha \in (0, 1). \tag{4.1}$$

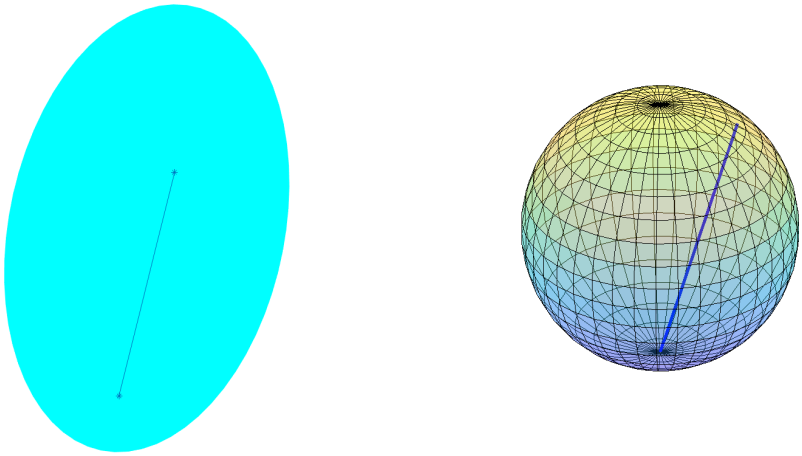

Figure 4.1: Examples of convex sets: An ellipsoid in $\mathbb{R}^2$ (Left) and a sphere in $\mathbb{R}^3$ (Right).

Definition 2
(Convex Function)
A function $f : \mathcal{X} \longrightarrow \mathbb{R}$, defined on a convex set $\mathcal{X}$, is called convex on $\mathcal{X}$ if the line

segment connecting $f(\boldsymbol{x})$ and $f(\boldsymbol{z})$ at any two points $\boldsymbol{x}$, $\boldsymbol{z} \in \mathcal{X}$ lies above the function between $\boldsymbol{x}$ and $\boldsymbol{z}$, i.e.

$$f(\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{z}) \leq \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{z}), \quad \forall\alpha \in (0, 1). \tag{4.2}$$

Now, if

$$f(\alpha\boldsymbol{x} + (1 - \alpha)\boldsymbol{z}) < \alpha f(\boldsymbol{x}) + (1 - \alpha)f(\boldsymbol{z}), \quad \forall\alpha \in (0, 1), \tag{4.3}$$

then f is called strictly convex on $\mathcal{X}$.

Definition 3
(Strong Convexity)
A differentiable function $f : \mathcal{X} \longrightarrow \mathbb{R}$, defined on a convex set $\mathcal{X}$, is called strongly convex with constant $\mu > 0$ on $\mathcal{X}$ with respect to an arbitrary norm defined on $\mathcal{X}$, if for any two points $\boldsymbol{x}$, $\boldsymbol{z} \in \mathcal{X}$ it holds:

$$f(\boldsymbol{z}) \geq f(\boldsymbol{x}) + \boldsymbol{\nabla} f(\boldsymbol{x})^T(\boldsymbol{z} - \boldsymbol{x}) + \frac{\mu}{2} \| \boldsymbol{z} - \boldsymbol{x} \|^2. \tag{4.4}$$

Definition 4
(Lipschitz function)
We say that f is L_f-Lipschitz with respect to $\| \cdot \| : \mathbb{R}^n \longrightarrow \mathbb{R}_0^+$ an arbitrary norm on $\mathbb{R}^n$ if:

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L_f \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Definition 5
(Dual Norm)
Let $\| \cdot \| : \mathbb{R}^n \longrightarrow \mathbb{R}_0^+$ be an arbitrary norm on $\mathbb{R}^n$. We will denote with $\| \cdot \|_*$ the dual norm defined as:

$$\|\boldsymbol{g}\|_* = \max_{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x}\| \leq 1} < \boldsymbol{g}, \boldsymbol{x} >_2 = \max_{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} \neq \boldsymbol{0}} \frac{1}{\|\boldsymbol{x}\|} < \boldsymbol{g}, \boldsymbol{x} >_2 .$$

We will say that $\boldsymbol{\nabla} f$ is $L_{\boldsymbol{\nabla} f}$-Lipschitz on $\mathcal{X}$ with constant $L_{\boldsymbol{\nabla} f} > 0$ if $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$:

$$\|\boldsymbol{\nabla} f(\boldsymbol{x}) - \boldsymbol{\nabla} f(\boldsymbol{y})\|_* \leq L_{\boldsymbol{\nabla} f} \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Definition 6
(n-Simplex)
We define as the n-Simplex the set

$$\Delta^n = \left\{ \boldsymbol{x} \in \mathbb{R}^n \,|\, \sum_{i=1}^{n} x_i = \|\boldsymbol{x}\|_1 = 1,\, x_i \geq 0,\, i = 1, 2, .., n \right\}.$$

Δ^n is usually called the probability simplex.

Definition 7
(n-Spectrahedron)
Let $\mathcal{S}_+^{n \times n} = \left\{ \boldsymbol{X} \in \mathbb{R}^{n \times n} \,|\, \boldsymbol{X} \succeq \boldsymbol{0},\, \boldsymbol{X}^T = \boldsymbol{X} \right\}$ be the set of all positive semi-definite (PSD) matrices in $\mathbb{R}^{n \times n}$.

We define as the n-Spectrahedron or the set of density matrices the set

$$\Delta^{n\times n} = \left\{ \boldsymbol{X} \in \mathcal{S}_+^{n\times n} \mid \mathrm{tr}(\boldsymbol{X}) = 1 \right\},$$

i.e. the set of all positive semi-definite $n \times n$ matrices with unit trace.

Definition 8
(Convex Conjugate)
Let $\Psi : \mathcal{D} \longrightarrow \mathbb{R}$ be a convex, continuous and differentiable function. The convex conjugate of Ψ is the function $\Psi^* : \mathbb{R}^n \longrightarrow \mathbb{R}$ defined by:

$$\Psi^*(\boldsymbol{z}) = \max_{\boldsymbol{y}\in\mathcal{D}} \left(\boldsymbol{y}^T \boldsymbol{z} - \Psi(\boldsymbol{y}) \right).$$

Some properties of Ψ^* are:

$$\nabla\Psi^*(\boldsymbol{z}) = \arg\max_{\boldsymbol{y}\in\mathcal{D}} \left(\boldsymbol{y}^T \boldsymbol{z} - \Psi(\boldsymbol{y}) \right),$$

$$\nabla\Psi^*(\mathbb{R}^n) = \mathcal{D},$$
$$\nabla\Psi \circ \nabla\Psi^*(\boldsymbol{z}) = \boldsymbol{z},$$
$$\Psi^{**} = \Psi.$$

Additionally, if Ψ is $\frac{1}{L}$-strongly convex ($L > 0$) with respect to a norm $||\cdot||$ defined on $\mathcal{D}$ it holds:

$$||\nabla\Psi^*(\boldsymbol{z}) - \nabla\Psi^*(\boldsymbol{y})|| \leq L||\boldsymbol{z} - \boldsymbol{y}||_*, \quad \forall \boldsymbol{z}, \boldsymbol{y} \in \mathbb{R}^n$$

i.e. Ψ^* is Lipschitz with constant L [26].

We now provide the following definitions which will be important to define the Mirror Descent optimisation later on.

Definition 9
(Mirror Map)
Assume that $\mathcal{D} \subset \mathbb{R}^n$ is a non-empty, open and convex set such that its closure contains the constraint set $\mathcal{X}$ and their intersection is not the empy set, i.e. $\mathcal{X} \subseteq \overline{\mathcal{D}}$ and $\mathcal{X} \cap \mathcal{D} \neq \emptyset$. We say that a function $\Psi : \mathcal{D} \longrightarrow \mathbb{R}$ is a mirror map under the assumptions:

- *Ψ is strictly convex on $\mathcal{D}$, i.e.*

$$\forall\, \boldsymbol{z} \neq \boldsymbol{y} \in \mathcal{D} \quad \forall\, \alpha \in (0,\, 1) \quad \Psi(\alpha\boldsymbol{z} + (1-\alpha)\boldsymbol{y}) < \alpha\Psi(\boldsymbol{z}) + (1-\alpha)\Psi(\boldsymbol{y}).$$

- *Ψ is continuous on and differentiable on $\mathcal{D}$.*
- *The dual space is all of $\mathbb{R}^n$, i.e.*

$$\nabla\Psi(\mathcal{D}) = \left\{ \nabla\Psi(\boldsymbol{z}) \,|\, \boldsymbol{z} \in \mathcal{D} \right\} = \mathbb{R}^n.$$

- *The gradient of* Ψ, $\boldsymbol{\nabla}\Psi : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ *diverges on the boundary of* $\mathcal{D}$, *i.e.*

$$\lim_{\boldsymbol{z} \longrightarrow \partial\mathcal{D}} ||\boldsymbol{\nabla}\Psi(\boldsymbol{z})|| \; = \; +\infty.$$

  Note that by equivalence of norms in finite spaces, the definition does not depend on the choice of norm.

Definition 10
(Bregman divergence)
Let Ψ *be a mirror map on* $\mathcal{D}$. *Then the Bregman divergence associated to* Ψ *is the function* $D_\Psi : \mathcal{D} \times \mathcal{D} \longrightarrow \mathbb{R}$ *given by:*

$$\begin{aligned} D_\Psi(\boldsymbol{x}, \boldsymbol{y}) \; &= \; \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{y}) - \; < \nabla\Psi(\boldsymbol{y}), \, \boldsymbol{x} - \boldsymbol{y} > \\ &= \; \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{y}) \; - \; \nabla\Psi(\boldsymbol{y})^T \, (\boldsymbol{x} - \boldsymbol{y}), \quad (\boldsymbol{x}, \boldsymbol{y}) \; \in \; \mathcal{D} \times \mathcal{D}. \end{aligned} \tag{4.5}$$

Some properties of the Bregman divergence are:

- $$\boldsymbol{\nabla}_{\boldsymbol{x}} D_\Psi(\boldsymbol{x}, \, \boldsymbol{y}) \; = \; \boldsymbol{\nabla}\Psi(\boldsymbol{x}) \; - \; \boldsymbol{\nabla}\Psi(\boldsymbol{y})$$

- *For any* $\boldsymbol{x}, \, \boldsymbol{y} \; \in \mathcal{D}$ *define* $\hat{\boldsymbol{x}} \; = \; \boldsymbol{\nabla}\Psi(\boldsymbol{x})$ *and* $\hat{\boldsymbol{y}} \; = \; \boldsymbol{\nabla}\Psi(\boldsymbol{y})$. *Then it holds:*

$$D_\Psi(\boldsymbol{x}, \, \boldsymbol{y}) \; = \; D_{\Psi^*}(\hat{\boldsymbol{x}}, \, \hat{\boldsymbol{y}}),$$

  where Ψ^* *denotes the convex conjugate of* Ψ.

  Note that we define as the dual space of $\mathcal{D}$ *(under* Ψ*) the space* $\mathcal{D}^* \; = \; \boldsymbol{\nabla}\Psi(\mathcal{D})$. *The last equation explains how the the convex conjugate of* Ψ *behaves on the primal space* $\mathcal{D}$. *The gradient* $\boldsymbol{\nabla}\Psi$ *maps the primal space onto the dual space* $\mathcal{D}^*$ *and* $\boldsymbol{\nabla}\Psi^*$ *maps the dual space onto the primal.*

- *For* $\boldsymbol{x}, \, \boldsymbol{y}, \, \boldsymbol{z} \; \in \mathcal{D}$ *it holds:*

$$D_\Psi(\boldsymbol{x}, \, \boldsymbol{y}) \; + \; D_\Psi(\boldsymbol{z}, \, \boldsymbol{x}) \; - \; D_\Psi(\boldsymbol{z}, \, \boldsymbol{y}) \; = \; \big(\boldsymbol{\nabla}\Psi(\boldsymbol{x}) \; - \; \boldsymbol{\nabla}\Psi(\boldsymbol{y})\big)^T (\boldsymbol{x} \; - \; \boldsymbol{z}).$$

Definition 11
(Projection)
The projection of a point $\boldsymbol{y} \in \mathcal{D}$, *onto a feasible set* $\mathcal{X} \; \subseteq \; \mathcal{D}$ *is defined as*

$$\Pi_{\mathcal{X}}(\boldsymbol{y}) \; = \; \underset{\boldsymbol{x} \in \mathcal{X}}{\arg\min} \, \frac{1}{2} ||\boldsymbol{x} \; - \; \boldsymbol{y}||_2^2 \, ,$$

where $\Pi_{\mathcal{X}} \; : \; \mathcal{D} \; \longrightarrow \; \mathcal{X}$ *is called the projection operator on* $\mathcal{X}$.
Note that in the case that $\boldsymbol{y} \; \in \; \mathcal{X}$*:*

$$\Pi_{\mathcal{X}}(\boldsymbol{y}) \; = \; \boldsymbol{y}.$$

Definition 12
(Bregman projection)
We define as the Bregman projection associated to Ψ *of a point* $\boldsymbol{y} \in \mathcal{D}$ *onto* $\mathcal{X}$ *as*

$$\Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{y}) \;=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} D_{\Psi}(\boldsymbol{x}\,,\,\boldsymbol{y}).$$

Lemma 1
Let Ψ *be a mirror map and* D_{Ψ} *its associated Bregman divergence function. Then if we restrict* Ψ^* *on* $\mathcal{X}$*, i.e.*

$$\Psi^*(\boldsymbol{z}) \;=\; \max_{\boldsymbol{y}\in\mathcal{X}} \big(\boldsymbol{y}^T\boldsymbol{z} \;-\; \Psi(\boldsymbol{y})\big),$$

it holds:

$$\boldsymbol{\nabla}\Psi^*(\boldsymbol{y}) \;=\; \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{y}) \;=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} D_{\Psi}(\boldsymbol{x},\, \boldsymbol{\nabla}\Psi^*(\boldsymbol{y})) \tag{4.6}$$

Proof 1

$$\begin{aligned}
\boldsymbol{\nabla}\Psi^*(\boldsymbol{y}) \;&=\; \arg\max_{\boldsymbol{x}\in\mathcal{X}} \big(\boldsymbol{x}^T\boldsymbol{y} \;-\; \Psi(\boldsymbol{x})\big)\\
&=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big(\Psi(\boldsymbol{x}) \;-\; \boldsymbol{x}^T\boldsymbol{y}\big)\\
&=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big(\Psi(\boldsymbol{x}) \;-\; \Psi(\boldsymbol{y}) \;-\; \boldsymbol{y}^T\boldsymbol{x}\big)\\
&=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big(\Psi(\boldsymbol{x}) \;-\; \Psi(\boldsymbol{y}) \;-\; \boldsymbol{\nabla}\Psi\circ\boldsymbol{\nabla}\Psi^*(\boldsymbol{y})^T\boldsymbol{x}\big)\\
&=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big(\Psi(\boldsymbol{x}) \;-\; \Psi(\boldsymbol{y}) \;-\; \boldsymbol{\nabla}\Psi\circ\boldsymbol{\nabla}\Psi^*(\boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{\nabla}\Psi^*(\boldsymbol{y}))\big)\\
&=\; \arg\min_{\boldsymbol{x}\in\mathcal{X}} D_{\Psi}(\boldsymbol{x},\, \boldsymbol{\nabla}\Psi^*(\boldsymbol{y}))
\end{aligned}$$

For the third and fifth equalities we added constant terms with respect to $\boldsymbol{x}$*. For the fourth equality we used* $\boldsymbol{\nabla}\Psi\circ\boldsymbol{\nabla}\Psi^*(\boldsymbol{z}) = \boldsymbol{z}$*.*

■

4.2.2 Main Problem Setup

Throughout this paper we focus on convex and constrained optimisation problems as outlined below:

For $\mathcal{X}$ a non-empty, compact and convex set (*feasible/ constraint* set), and for $f : \mathcal{X} \longrightarrow \mathbb{R}$ a convex and at least twice differentiable smooth function we are interested in solving the convex optimisation problem

$$f^* = \min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \quad \boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \tag{4.7}$$

We say that $\boldsymbol{x}^*$ is the *optimum/ minimum* of f on $\mathcal{X}$ and $f^* = f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ the *optimum/ minimum* value of f on $\mathcal{X}$.

4.2.3 Examples

Below are presented some examples of mirror maps and their associated Bregman divergence which we will be using later on in this project.

Example 1
(Euclidean distance function)
Let $\mathcal{X} \subseteq \mathbb{R}^n$ be any compact and convex set in $\mathbb{R}^n$ and $\mathcal{D} = \mathbb{R}^n$. $\mathcal{D}$ is clearly open and convex, $\mathcal{X} \subseteq \overline{\mathcal{D}}$ and $\mathcal{X} \cap \mathcal{D} \neq \emptyset$.

The Euclidean distance function or $\mathcal{L}2$-norm is defined as:

$$\Psi(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{x}||_2^2 = \frac{1}{2}\sum_{i=1}^{n} x_i^2, \quad \forall \boldsymbol{x} \in \mathbb{R}^n. \tag{4.8}$$

The $\mathcal{L}2$-norm is continuous and infinitely-differentiable ($\mathcal{C}^\infty$) on $\mathbb{R}^n$. Ψ is strictly convex on $\mathbb{R}^n$ since for any $\boldsymbol{x} \neq \boldsymbol{y}$, $\alpha \in (0,1)$:

$$\begin{aligned}\Psi(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) &= \frac{1}{2}||\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}||_2^2 \\ &< \frac{1}{2}\alpha||\boldsymbol{x}||_2^2 + \frac{1}{2}(1-\alpha)||\boldsymbol{y}||_2^2 \\ &= \alpha\Psi(\boldsymbol{x}) + (1-\alpha)\Psi(\boldsymbol{y})\end{aligned}$$

We have that

$$\boldsymbol{\nabla}\Psi(\boldsymbol{x}) = \boldsymbol{x},$$

and

$$\boldsymbol{\nabla}^2\Psi(\boldsymbol{x}) = I_n.$$

The $\mathcal{L}2$-norm satisfies all assumptions of Definition 9. The Bregman Divergence induced by Ψ is given by:

$$\begin{aligned} D_\Psi(\boldsymbol{x}, \boldsymbol{y}) &= \frac{1}{2}||\boldsymbol{x}||_2^2 - \frac{1}{2}||\boldsymbol{y}||_2^2 - \boldsymbol{y}^T(\boldsymbol{x} - \boldsymbol{y}) \\ &= \frac{1}{2}||\boldsymbol{x} - \boldsymbol{y}||_2^2 \end{aligned} \tag{4.9}$$

i.e. the Euclidean-distance between $\boldsymbol{x}$ and $\boldsymbol{y}$.

The convex conjugate of Ψ on $\mathcal{D}$ is given by:

$$\Psi^*(\boldsymbol{z}) = \frac{1}{2}||\boldsymbol{z}||_2^2 = \Psi(\boldsymbol{z}),$$

and

$$\boldsymbol{\nabla}\Psi^*(\boldsymbol{z}) = \boldsymbol{z}.$$

Ψ is 1 strongly convex with respect to the L2-norm over $\mathcal{D}$ (see 4 of [57]).

Example 2
(Negative Entropy function)
Let $\mathcal{X} = \Delta^n$ the n-Simplex and $\mathcal{D} = \mathbb{R}^n_{0^+}$. $\mathcal{X}$ is compact and convex, $\mathcal{D}$ is open and also convex, $\mathcal{X} \subseteq \overline{\mathcal{D}}$ and $\mathcal{X} \cap \mathcal{D} \neq \emptyset$. The negative entropy function on $\mathcal{D}$ is defined as:

$$\Psi(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \log(x_i) - x_i, \quad \forall \boldsymbol{x} \in \mathbb{R}^n_{0^+}, \tag{4.10}$$

where we define $0 \cdot \log(0) := 0$.

It's clear that Ψ is continuous and it is doubly differentiable on $\mathcal{D}$ with first and second derivatives:

$$\boldsymbol{\nabla}\Psi(\boldsymbol{x}) = \mathbf{log}(\boldsymbol{x}) = \begin{bmatrix} \log(x_1) \\ \log(x_2) \\ \vdots \\ \log(x_n) \end{bmatrix}, \boldsymbol{\nabla}^2\Psi(\boldsymbol{x}) = \boldsymbol{diag}(\boldsymbol{x})^{-1} = \begin{bmatrix} \frac{1}{x_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{x_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{x_n} \end{bmatrix}. \tag{4.11}$$

It is easy to see that $\boldsymbol{\nabla}^2\Psi(\boldsymbol{x}) \succ \mathbf{0}$ for all $\boldsymbol{x} \in \mathbb{R}^n_{0^+}$ and therefore Ψ is strictly convex on $\mathcal{D}$. Also, note that

$$\log(\mathbb{R}_{0^+}) = \mathbb{R}$$

and therefore,

$$\boldsymbol{\nabla}\Psi(\mathcal{D}) = \mathbf{log}(\mathcal{D}) = \mathbf{log}(\mathbb{R}^n_{0^+}) = \mathbb{R}^n.$$

The gradient of Ψ diverges on $\partial\mathcal{D} = \left\{\mathbf{0} = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^T\right\}$, i.e.

$$\lim_{\boldsymbol{x} \longrightarrow \mathbf{0}} ||\mathbf{log}(\boldsymbol{x})||_1 = +\infty$$

since $\lim_{z \longrightarrow 0} |\log(z)| = +\infty$.

Thus, the negative entropy function can serve as a mirror map as it satisfies all assumptions of Definition 9.

The convex conjugate of Ψ *on* $\mathcal{D}$ *is given by:*

$$\Psi^*(\boldsymbol{z}) = \mathbf{1}_n^T \mathbf{exp}(\boldsymbol{z}) = \sum_{i=1}^{n} \exp(z_i), \quad \boldsymbol{\nabla}\Psi^*(\boldsymbol{z}) = \mathbf{exp}(\boldsymbol{z}), \quad \forall \boldsymbol{z} \in \mathbb{R}^n.$$

Note that the convex conjugate of Ψ *constrained on* $\mathcal{X}$ *becomes:*

$$\Psi^*(\boldsymbol{z}) = \log\big(\sum_{i=1}^{n} \exp(z_i)\big) + 1, \quad \boldsymbol{\nabla}\Psi^*(\boldsymbol{z}) = \frac{1}{\sum_{i=1}^{n} \exp(z_i)} \mathbf{exp}(\boldsymbol{z}), \quad \forall \boldsymbol{z} \in \mathbb{R}^n$$

and $\boldsymbol{\nabla}\Psi^*(\mathbb{R}^n) = \mathcal{X}$.

Proof 2

$$\boldsymbol{\nabla}\Psi^*(\boldsymbol{z}) = \max_{\boldsymbol{y} \in \mathcal{X}} \big(\boldsymbol{y}^T \boldsymbol{z} - \Psi(\boldsymbol{y})\big) = \max_{\boldsymbol{y} \in \mathcal{X}} \big(\boldsymbol{y}^T \boldsymbol{z} - \boldsymbol{y}^T(\mathbf{log}(\boldsymbol{y}) - \mathbf{1}_n)\big)$$

The Lagrangian of the problem is given by:

$$L(\boldsymbol{y}, \lambda, \boldsymbol{\mu}) = \boldsymbol{y}^T \boldsymbol{z} - \boldsymbol{y}^T(\mathbf{log}(\boldsymbol{y}) - \mathbf{1}_n) - \lambda(\boldsymbol{y}^T \mathbf{1}_n - 1) - \boldsymbol{\mu}^T \boldsymbol{y}.$$

Setting the gradient equal to $\mathbf{0}$*:*

$$\boldsymbol{\nabla}_{y_i} L(\boldsymbol{y}, \lambda, \boldsymbol{\mu}) = z_i - \log(y_i) - \lambda - \mu_i = 0 \Longrightarrow y_i = \exp(z_i - \mu_i - \lambda).$$

Utilising the KKT conditions we find that $\mu_i = 0$ *and that* $e^{\lambda} = \sum_{i=1}^{n} \exp(z_i)$.

Therefore $y_i^* = \frac{\exp(z_i)}{\sum_{i=1}^{n} \exp(z_i)}$ *and* $\Psi(\boldsymbol{y}^*) = \log\big(\mathbf{exp}(\boldsymbol{z})^T \mathbf{1}_n\big) + 1$ *and*

$$\boldsymbol{\nabla}\Psi^*(\boldsymbol{z}) = \frac{\mathbf{exp}(\boldsymbol{z})}{\mathbf{exp}(\boldsymbol{z})^T \mathbf{1}_n}.$$

■

The Bregman Divergence induced by Ψ *is given by:*

$$\begin{aligned} D_{\Psi}(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{i=1}^{n} \big(x_i \log x_i - x_i\big) - \sum_{i=1}^{n} \big(y_i \log y_i - y_i\big) - \mathbf{log}(\boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y}) \\ &= \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i} - \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i = D_{KL}(\boldsymbol{x} \,||\, \boldsymbol{y}), \end{aligned} \tag{4.12}$$

i.e. the generalized Kullback–Leibler divergence between $\boldsymbol{x}$ *and* $\boldsymbol{y}$.
The Negative Entropy function is 1 *strongly convex with respect to the* $L1$*-norm (Pinsker's Inequality) over* $\mathcal{D}$ *[57].*

Example 3
(Negative Von Neumann Entropy function)
Let $\mathcal{X} = \Delta^{n\times n}$ *the n-Spectrahedron and* $\mathcal{D} = \mathcal{S}^{n\times n} = \left\{\boldsymbol{X} \in \mathbb{R}^{n\times n} | \boldsymbol{X} \succ \mathbf{0},\, \boldsymbol{X}^T = \boldsymbol{X}\right\}$.
$\mathcal{X}$ *is compact and convex,* $\mathcal{D}$ *is open and also convex,* $\mathcal{X} \subseteq \overline{\mathcal{D}}$ *and* $\mathcal{X} \cap \mathcal{D} \neq \emptyset$.

Proof 3 *(Convexity of* $\mathcal{X}$ *and* $\mathcal{D}$*).*

Let $\boldsymbol{A}, \boldsymbol{B}, \in \mathcal{D}, \quad \boldsymbol{Z}, \boldsymbol{Y} \in \mathcal{X}, \quad \alpha \in (0,1)$*:*

$$\begin{aligned} \alpha\,\boldsymbol{A} \succ \mathbf{0}, \quad (1-\alpha)\,\boldsymbol{B} \succ \mathbf{0} \quad &\Longrightarrow \alpha\,\boldsymbol{A} + (1-\alpha)\,\boldsymbol{B} \succ \mathbf{0} \\ &\Longrightarrow \Big(\alpha\,\boldsymbol{A} + (1-\alpha)\,\boldsymbol{B}\Big) \in \mathcal{D} \end{aligned}$$

$$\begin{aligned} \operatorname{tr}\big(\alpha\,\boldsymbol{Z}\big) = \alpha, \quad \operatorname{tr}\big((1-\alpha)\,\boldsymbol{Y}\big) = 1-\alpha \quad &\Longrightarrow \operatorname{tr}\big(\alpha\,\boldsymbol{Z} + (1-\alpha)\,\boldsymbol{Y}\big) = 1 \\ &\Longrightarrow \Big(\alpha\,\boldsymbol{Z} + (1-\alpha)\,\boldsymbol{Y}\Big) \in \mathcal{X}. \end{aligned}$$

■

The negative Von Neumann Entropy function on $\mathcal{D}$ *is defined as:*

$$\Psi(\boldsymbol{X}) = \operatorname{tr}(\boldsymbol{X}\mathbf{log}\boldsymbol{X} - \boldsymbol{X}) = \sum_{i=1}^{n}(\lambda_i(\boldsymbol{X})\log(\lambda_i(\boldsymbol{X})) - \lambda_i(\boldsymbol{X})), \quad \forall \boldsymbol{X} \in \mathcal{D}, \tag{4.13}$$

where we define $0 \cdot \log 0 := 0$.

Ψ *is continuous and differentiable on* $\mathcal{D}$*, with gradient:*

$$\boldsymbol{\nabla}\Psi(\boldsymbol{X}) = \mathbf{log}\boldsymbol{X} = \begin{bmatrix} \log X_{11} & \log X_{12} & \dots & \log X_{1n} \\ \log X_{21} & \log X_{22} & \dots & \log X_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \log X_{n1} & \log X_{n2} & \dots & \log X_{nn} \end{bmatrix}. \tag{4.14}$$

Ψ *is strictly convex on* $\mathcal{D}$*. For the proof of the convexity of* Ψ *refer to [2]. The Bregman Divergence induced by* Ψ *is given by:*

$$\begin{aligned} D_\Psi(\boldsymbol{X}, \boldsymbol{Y}) &= \operatorname{tr}(\boldsymbol{X}\mathbf{log}\boldsymbol{X} - \boldsymbol{X}) - \operatorname{tr}(\boldsymbol{Y}\mathbf{log}\boldsymbol{Y} - \boldsymbol{Y}) - \mathbf{log}(\boldsymbol{Y}) \bullet (\boldsymbol{X} - \boldsymbol{Y}) \\ &= \operatorname{tr}\big(\boldsymbol{X}(\mathbf{log}\boldsymbol{X} - \mathbf{log}\boldsymbol{Y}) - \boldsymbol{X} + \boldsymbol{Y}\big) \end{aligned} \tag{4.15}$$

For the convex conjugate of Ψ *on* $\mathcal{D}$*,* $\Psi^*(\boldsymbol{Z}) = \max_{\boldsymbol{Y}\in\mathcal{D}}(\boldsymbol{Y} \bullet \boldsymbol{Z} - \Psi(\boldsymbol{Y}))$*, we have:*

$$\boldsymbol{\nabla}\Psi^*(\boldsymbol{Z}) = \mathbf{exp}(\boldsymbol{Z}).$$

For $\Psi^*(\boldsymbol{Z}) = \max_{\boldsymbol{Y}\in\mathcal{X}}(\boldsymbol{Y} \bullet \boldsymbol{Z} - \Psi(\boldsymbol{Y}))$ *we have,*

$$\boldsymbol{\nabla}\Psi^*(\boldsymbol{Z}) = \frac{\mathbf{exp}(\boldsymbol{Z})}{\operatorname{tr}(\mathbf{exp}(\boldsymbol{Z}))}.$$

The Negative Von Neumann entropy function can be proved to be $\frac{1}{2}$ *strongly convex with respect to the nuclear norm (Matrix Pinsker's Inequality) over* $\mathcal{D}$ *[13, 57].*

4.3 Mirror Descent

4.3.1 The (Projected) Gradient Descent Algorithm

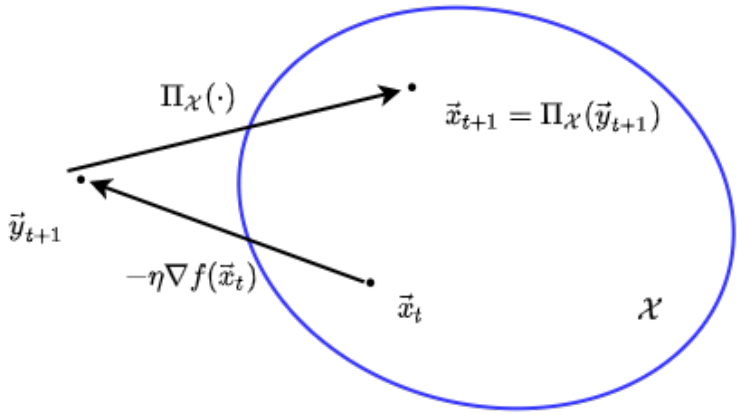

Figure 4.2: Illustration of Projected Gradient Descent.

The (Stochastic) Gradient Descent is the simplest and one of the most widely used methods for unconstrained and constrained optimisation of a differentiable function. One simply takes a step in the direction of the negative gradient of f (steepest descent) and projects it in the feasible set if the optimisation is constrained using the projection operator.

The Projected (Stochastic) Gradient Descent update rule for one iteration consists of two steps and it is illustrated in Figure 4.2 and is given by:

$$\begin{aligned} \boldsymbol{y}_{t+1} &= \boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t) + \sigma \mathcal{N}(0, \boldsymbol{I}), \quad \sigma \geq 0 \\ \boldsymbol{x}_{t+1} &= \Pi_{\mathcal{X}}(\boldsymbol{y}_{t+1}), \quad t \geq 0. \end{aligned} \tag{4.16}$$

Note that if we set $\sigma > 0$ we say that the optimisation is stochastic and deterministic otherwise.

However, for high dimensionality problems, the projection operator can slow down the convergence of projected gradient descent [4] (Chapter 9). Moreover, there are cases in which gradient descent may never converge to the true minimum. Such is the scenario when using fixed (non-adaptive) learning rate η_t in cases with noisy gradient calculations (e.g. Section 5.2 of [10]). For instance, in mini-batch optimisation, the gradient is calculated over a subset of the training samples set and it is usually noisy. In Chapter 5 we provide a relevant experiment. We will also see later on that Projected Gradient Descent is a special case of a "broader" optimisation algorithm called Mirror Descent. Below, we provide the reader with the PGD algorithm averaged over N_{iid} runs.

Algorithm 1 Projected Gradient Descent Algorithm

Input: $\boldsymbol{X}_0 = [\boldsymbol{x}_0^1, ..., \boldsymbol{x}_0^{N_{iid}}] : \{\boldsymbol{x}_0^N\}_{N=1}^{N_{iid}} \in \mathcal{X}$, $\eta > 0$ learning rate
$\sigma \geq 0$ noise parameter, $T > 0$ iterations
Output: $\boldsymbol{x}^*$, f^*
for N = 1,...,N_{iid} **do**
$\boldsymbol{x}_0 = \boldsymbol{x}_0^N$
$\eta_0 = \eta$
for t = 1,...,T **do**
$\boldsymbol{y}_{t+1} = \boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t) + \sigma \mathcal{N}(0, \boldsymbol{I})$

$\boldsymbol{x}_{t+1} = \Pi_{\mathcal{X}}(\boldsymbol{y}_{t+1}) = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2} ||\boldsymbol{x} - \boldsymbol{y}_{t+1}||_2^2$

if learning rate is adaptive **then**
Update η_t
end if
end for
$\boldsymbol{x}_N^* = \frac{1}{T+1} \sum_{s=0}^{T} \boldsymbol{x}_s$ or $\boldsymbol{x}_N^* = \arg\min_{\boldsymbol{x} \in \{\boldsymbol{x}_s\}_0^T} f(\boldsymbol{x})$,
$f_N^* = f(\boldsymbol{x}_N^*)$
end for
$\boldsymbol{x}^* = \frac{1}{N_{iid}} \sum_{N=1}^{N_{iid}} \boldsymbol{x}_N^*$
$f^* = \frac{1}{N_{iid}} \sum_{N=1}^{N_{iid}} f_N^*$

4.3.2 The Mirror Descent Algorithm

The motivations of Mirror Descent are:

- Projected Gradient Descent does not discriminate between the primal and dual spaces. While at iteration t $\boldsymbol{x}_t$ lies in a subset of $\mathbb{R}^n$ (the primal space), the gradient $\nabla f(\boldsymbol{x}_t)$ (a linear function of $\boldsymbol{x}_t$) lies in the dual space. Despite the fact that $\boldsymbol{x}_t$ and $\nabla f(\boldsymbol{x}_t)$ live in different vector spaces, GD at each iteration calculates $x_t - \eta_t \nabla f(\boldsymbol{x}_t)$ as if they both lie in the same vector space. In contrast, as it will be explained in detail in the next paragraph, Mirror Descent discriminates between these two spaces by utilising as a bijection between the two spaces a mirror map function Ψ and its convex conjugate Ψ^*.

- In addition, the convergence analysis on projected Gradient Descent utilises L-Lipschitz objective functions with respect to the $L2$-norm. However, take for example a case where the objective function is 1-Lipschitz w.r.t. the $L\infty$-norm. Then, supposing that the dimensions of the domain of f is equal to N, then f is $\sqrt{N}$-Lipschitz w.r.t to the $L2$-norm and thus, for large N the convergence of the method can be really slow.

For a more detailed discussion of the motivations of Mirror Descent we refer the reader to [13, 24] .

In continuous time dynamics, the update schema of Stochastic Mirror Descent is given by the Itô SDE (see [44]):

$$d\boldsymbol{y}_t = \eta_t \boldsymbol{\nabla} f(\boldsymbol{x}_t)dt + \sigma d\boldsymbol{B}_t, \quad \boldsymbol{x}_t = \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_t)), \quad \sigma \geq 0, t \geq 0,$$

where σ denotes a noise parameter and $\boldsymbol{B}_t$ denotes a Brownian motion. From the above equation we obtain the Euler time-discretisation:

$$\begin{aligned} \boldsymbol{y}_{t+1} &= \boldsymbol{y}_t - \epsilon\, \eta_t\, \boldsymbol{\nabla} f(\boldsymbol{x}_t) + \sqrt{\epsilon}\, \sigma\, \mathcal{N}(0, \boldsymbol{I}), \quad \sigma \geq 0 \\ \boldsymbol{x}_{t+1} &= \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_{t+1})), \quad t \geq 0, \end{aligned}$$

where ϵ is a discretisation parameter which we will set equal to 1 for the rest of this project.

The update schema of one iteration of (Stochastic) Mirror Descent can be described in four steps for an initial point $\boldsymbol{x}_0 \in \mathcal{X} \cap \mathcal{D}$:

$$\hat{\boldsymbol{x}}_t = \boldsymbol{\nabla}\Psi(\boldsymbol{x}_t) \tag{4.17a}$$

$$\hat{\boldsymbol{y}_{t+1}} = \hat{\boldsymbol{x}}_t - \eta_t \boldsymbol{\nabla} f(\boldsymbol{x}_t) + \sigma\, \mathcal{N}(0, \boldsymbol{I}), \quad \sigma \geq 0 \tag{4.17b}$$

$$\boldsymbol{y}_{t+1} = \boldsymbol{\nabla}\Psi^*(\hat{\boldsymbol{y}_{t+1}}) \tag{4.17c}$$

$$\boldsymbol{x}_{t+1} = \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{y}_{t+1}) = \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathcal{X}} D_{\Psi}(\boldsymbol{x}\,,\, \boldsymbol{y}_{t+1}) \tag{4.17d}$$

where $\Psi : \mathcal{D} \longrightarrow \mathbb{R}$ is a function that meets all assumptions of a mirror map (see Definition 9) and $\Psi^*(\boldsymbol{z}) = \max_{\boldsymbol{y}\in\mathcal{D}} \big(\boldsymbol{y}^T\boldsymbol{z} - \Psi(\boldsymbol{y})\big)$ is its convex conjugate as described in Definition 8.

The geometric intuition behind the Mirror Descent algorithm is that in order to perform the optimization in the primal space $\mathcal{D}$, we first map the point $\boldsymbol{x}_t \in \mathcal{X} \subset \mathcal{D}$ to its dual space $\mathcal{D}^*$ (4.17a). Then the negative gradient step is performed in the dual space to obtain $\hat{\boldsymbol{y}_{t+1}}$ (4.17b) which is then mapped back to the primal space via the convex conjugate function of the mirror map (4.17c).

Note that at each update step, the new point in the primal space $\boldsymbol{y}_{t+1} \in \mathcal{D}$ might be outside of the feasible set $\mathcal{X}$. In that case it should be projected onto $\mathcal{X}$. The projection used to project $\boldsymbol{y}_{t+1}$ onto $\boldsymbol{x}_{t+1} \in \mathcal{X}$ is the Bregman Projection $D_{\mathcal{X}}^{\Psi}$ associated with the mirror map Ψ (4.17d).

Note also that from Lemma 1, if we take

$$\Psi^*(\boldsymbol{z}) = \max_{\boldsymbol{y}\in\mathcal{X}} \big(\boldsymbol{y}^T\boldsymbol{z} - \Psi(\boldsymbol{y})\big)$$

then the last step can be rewritten into $\boldsymbol{x}_{t+1} = \boldsymbol{y}_{t+1}$.

In that case the algorithm of the Mirror Descent (without using the Bregman Divergence) is given by:

$$\begin{aligned} \boldsymbol{y}_{t+1} &= \boldsymbol{y}_t - \eta_t \nabla f(\boldsymbol{x}_t) + \sigma \mathcal{N}(0, \boldsymbol{I}), \quad \sigma \geq 0 \\ \boldsymbol{x}_{t+1} &= \nabla \Psi^*(\boldsymbol{y}_{t+1}), \quad t \geq 0, \end{aligned}$$

which is taken as the definition in [10]. However, the two definitions are equivalent. An illustration of the above schema is provided below on Figure 4.3 and the step-by-step algorithm of Mirror Descent is provided in Algorithm 2.

The update rule of (Stochastic) Mirror Descent is usually written in two steps which are often taken as the definition of the method (see [13]):

$$\begin{aligned} \nabla \Psi(\boldsymbol{y}_{t+1}) &= \nabla \Psi(\boldsymbol{x}_t) - \eta_t \nabla f(\boldsymbol{x}_t) + \sigma \mathcal{N}(0, \boldsymbol{I}), \quad \sigma \geq 0 \\ \boldsymbol{x}_{t+1} &= \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{y}_{t+1}) = \arg\min_{\boldsymbol{x} \in \mathcal{X}} D_{\Psi}(\boldsymbol{x}, \boldsymbol{y}_{t+1}). \end{aligned} \tag{4.18}$$

Note that, in the case of Mirror Descent we will be using the above scheme in this project.

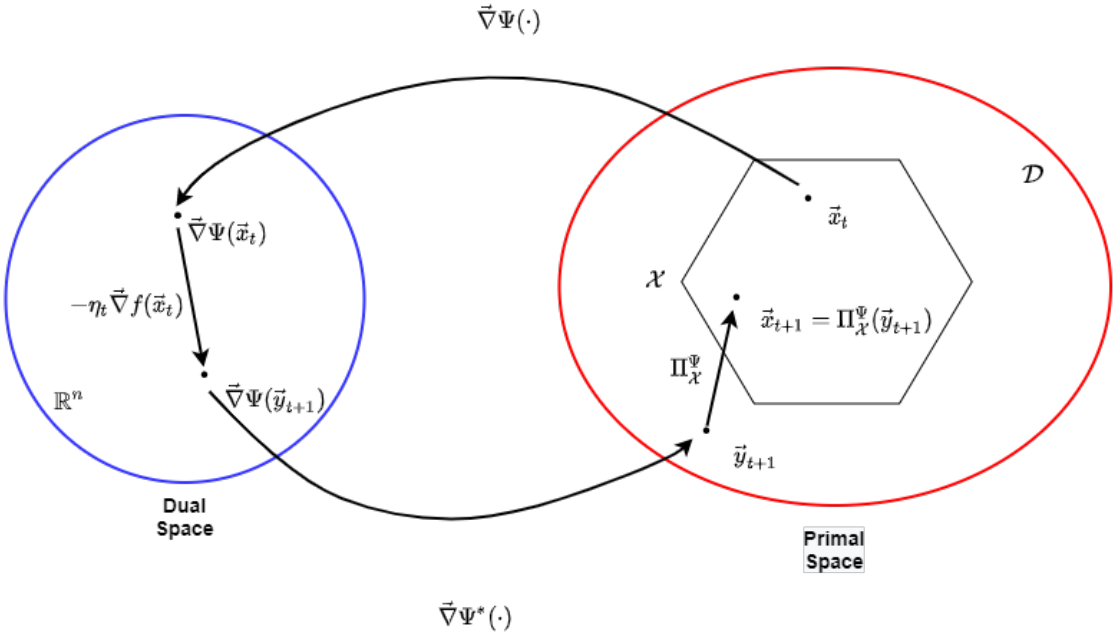

Figure 4.3: Illustration of Mirror Descent in accordance with Equations 4.17a-4.17d.

We point out that the update rule of $\boldsymbol{x}_{t+1}$ in Equation 4.18 can be combined in a single step and be rewritten as:

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ (\eta_t \nabla f(\boldsymbol{x}_t) - \sigma \mathcal{N}(0, \boldsymbol{I}))^T \boldsymbol{x} + D_\Psi(\boldsymbol{x}, \boldsymbol{x}_t) \right\}. \tag{4.19}$$

Proof 4 *Let* $\sigma = 0$:

$$\begin{aligned}
\boldsymbol{x}_{t+1} &= \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{y}_{t+1}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} D_\Psi(\boldsymbol{x}, \boldsymbol{y}_{t+1}) \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{y}_{t+1}) - \nabla\Psi(\boldsymbol{y}_{t+1})^T (\boldsymbol{x} - \boldsymbol{y}_{t+1}) \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{y}_{t+1}) - \nabla\Psi(\boldsymbol{y}_{t+1})^T \boldsymbol{x} + \nabla\Psi(\boldsymbol{y}_{t+1})^T \boldsymbol{y}_{t+1} \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - \nabla\Psi(\boldsymbol{y}_{t+1})^T \boldsymbol{x} \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - \left(\nabla\Psi(\boldsymbol{x}_t) - \eta_t \nabla f(\boldsymbol{x}_t) \right)^T \boldsymbol{x} \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \eta_t \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + \Psi(\boldsymbol{x}) - \nabla\Psi(\boldsymbol{x}_t)^T \boldsymbol{x} \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \eta_t \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + \Psi(\boldsymbol{x}) + \Psi(\boldsymbol{x}_t) - \nabla\Psi(\boldsymbol{x}_t)^T \boldsymbol{x} + \nabla\Psi(\boldsymbol{x}_t)^T \boldsymbol{x}_t \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \eta_t \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + D_\Psi(\boldsymbol{x}, \boldsymbol{x}_t) \right\}.
\end{aligned}$$

Note that for the first equality we expanded the update rule of Mirror Descent as portrayed by 4.18. For the fifth and eighth equalities we added/ removed constants w.r.t. to the optimisation parameter $\boldsymbol{x}$.

■

As stated above, Projected Gradient Descent is a special case of Mirror Descent. One could use as a mirror map the Euclidean $L2$-norm (see Example 1). In that case the update rule becomes:

$$\begin{aligned}
\boldsymbol{x}_{t+1} &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ (\eta_t \nabla f(\boldsymbol{x}_t) - \sigma \mathcal{N}(0, \boldsymbol{I}))^T \boldsymbol{x} + D_\Psi(\boldsymbol{x}, \boldsymbol{x}_t) \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ (\eta_t \nabla f(\boldsymbol{x}_t) - \sigma \mathcal{N}(0, \boldsymbol{I}))^T \boldsymbol{x} + \frac{1}{2} ||\boldsymbol{x} - \boldsymbol{x}_t||_2^2 \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \frac{1}{2} \left|\left| \boldsymbol{x} - (\boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t) + \sigma \mathcal{N}(0, \boldsymbol{I})) \right|\right|_2^2 \\
&= \Pi_{\mathcal{X}}(\boldsymbol{y}_{t+1})
\end{aligned}$$

which is exactly the Projected Gradient Descent update schema.

Algorithm 2 (Stochastic) Mirror Descent Algorithm

Input: $\boldsymbol{X}_0 = [\boldsymbol{x}_0^1, ..., \boldsymbol{x}_0^{N_{iid}}] : \{\boldsymbol{x}_0^N\}_{N=1}^{N_{iid}} \in \mathcal{X}$, $\eta > 0$ learning rate
$\sigma \geq 0$ noise parameter, $T > 0$ iterations

Output: $\boldsymbol{x}^*$, f^*

for $N = 1,...,N_{iid}$ **do**
$\boldsymbol{x}_0 = \boldsymbol{x}_0^N$
$\eta_0 = \eta$
for $t = 1,...,T$ **do**

$$\nabla\Psi(\boldsymbol{y}_{t+1}) = \nabla\Psi(\boldsymbol{x}_t) - \eta_t \nabla f(\boldsymbol{x}_t) + \sigma\mathcal{N}(0, \boldsymbol{I})$$

$$\boldsymbol{x}_{t+1} = \Pi_{\mathcal{X}}^{\Psi}(\boldsymbol{y}_{t+1}) = \arg\min_{\boldsymbol{x}\in\mathcal{X}} D_\Psi(\boldsymbol{x}\,, \boldsymbol{y}_{t+1}).$$

if learning rate is adaptive **then**

Update η_t

end if
end for

$$\boldsymbol{x}_N^* = \frac{1}{T+1}\sum_{s=0}^{T} \boldsymbol{x}_s \text{ or } \boldsymbol{x}_N^* = \arg\min_{\boldsymbol{x}\in\{\boldsymbol{x}_s\}_0^T} f(\boldsymbol{x})$$
$$f_N^* = f(\boldsymbol{x}_N^*)$$
end for
$$\boldsymbol{x}^* = \frac{1}{N_{iid}}\sum_{N=1}^{N_{iid}} \boldsymbol{x}_N^*$$
$$f^* = \frac{1}{N_{iid}}\sum_{N=1}^{N_{iid}} f_N^*$$

As aforementioned, our study here is rather experimental rather than theoretical. For more elaborate theoretical results on the convergence of Gradient Descent and Mirror Descent please refer to Sections 3.2 and 4 of [13], respectively.

Note that if we set $\sigma > 0$ we say that the optimisation is stochastic (Stochastic Mirror Descent) and deterministic (Mirror Descent) otherwise.

4.4 Interacting Mirror Descent

Optimisation algorithms such as Projected Gradient Descent may converge to non optimum solutions, such as local minima, saddle points or to neighbourhoods of the real optimum. This is due various factors. For instance, noisy data and or computing the gradient over a subset of the sample data can cause this. To combat this, $N_{iid} \geq 1$ independent and identical versions of the same algorithm are ran in parallel (see [62, 27]) and the final result is the average of the result of all runs (see Algorithms 1 and 2), in other words:

$$f^* = \frac{1}{N_{iid}} \sum_{N=1}^{N_{iid}} f_N^*.$$

A variation of the (Stochastic) Mirror Descent Algorithm is the (Stochastic) Interacting Mirror Descent Algorithm proposed in [10] where they suggest instead of running N_{iid} replicas of the Mirror Descent algorithm, running one replica of the MD Algorithm in which they allow $N_p \geq 1$ particles to "interact". Assuming that $\Psi^*(\boldsymbol{z}) = \max_{\boldsymbol{y} \in \mathcal{X}} \big(\boldsymbol{y}^T \boldsymbol{z} - \Psi(\boldsymbol{y})\big)$, the optimisation update rule they propose is inspired by the following Itô Stochastic Differential Equation in continuous time for :

$$\begin{aligned} d\boldsymbol{y}_t^i &= -\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)dt + \sum_{j=1}^{N_p} A_{ij}(\boldsymbol{y}_t^j - \boldsymbol{y}_t^i) + \sigma d\boldsymbol{B}_t^i, \\ \boldsymbol{x}_t^i &= \boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_i^t), \quad i = 1, ..., N_p, \quad t \geq 0, \end{aligned}$$

where each particle is characterised by independent Brownian motions $\boldsymbol{B}_t^i$ and $\boldsymbol{x}_t^i$ denotes the i-th particle at time-step t, where $i = 1, ..., N_p$. The method is described in detail in the rest of this section.

We will be concerned with the descritised version of the Stochastic Interacting Mirror Descent Algorithm:

$$\boldsymbol{y}_{t+1}^i = \boldsymbol{y}_t^i - \eta_t \cdot \epsilon\, \boldsymbol{\nabla} f(\boldsymbol{x}_t^i) + \epsilon \sum_{j=1}^{N_p} A_{ij}(\boldsymbol{y}_t^j - \boldsymbol{y}_t^i) + \sigma\sqrt{\epsilon}\mathcal{N}(0, 1) \tag{4.20a}$$

$$\boldsymbol{x}_{t+1}^i = \boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_{t+1}^i), \quad i = 1, ..., N_p, \quad t \geq 0. \tag{4.20b}$$

Note that, if we use $\Psi^*(\boldsymbol{z}) = \max_{\boldsymbol{y} \in \mathcal{D}} \big(\boldsymbol{y}^T \boldsymbol{z} - \Psi(\boldsymbol{y})\big)$, then the last step becomes

$$\boldsymbol{x}_{t+1}^i = \Pi_{\mathcal{X}}^{\Psi}\Big(\boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_{t+1}^i)\Big), \quad i = 1, ..., N_p, \quad t \geq 0.$$

We remark that $\boldsymbol{A}$ represents an $N_p \times N_p$ doubly stochastic ($\sum_{i=1}^{N_p} A_{ij} = \sum_{j=1}^{N_p} A_{ij} = 1$) matrix constituting of the interaction weights A_{ij} between the particles i and j and $\epsilon > 0$ is called the discretisation parameter. We remark that for a particle p only j particles $1 \leq j \leq N_p$ have impact on such that $A_{pj} \neq 0$. In case that $\boldsymbol{A} = \boldsymbol{0}$ (i.e.

there is no interaction), then the problem is equivalent to the MD framework averaged over N_p iid copies.

We also remark that if $\sigma > 0$ the optimisation is called Stochastic Interacting Mirror Descent, and Interacting Mirror Descent otherwise.

Let $\epsilon = 1$. The update rule of $\boldsymbol{x}_{t+1}^i$ in the previous update schema is equivalent to

$$\boldsymbol{x}_{t+1}^i = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ (\eta_t \nabla f(\boldsymbol{x}_t^i) - \sigma \mathcal{N}(0, \boldsymbol{I}))^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} D_\Psi(\boldsymbol{x}, \boldsymbol{x}_t^j) \right\} \tag{4.21}$$

Proof 5 *Let $i \in \{1, ..., N_p\}$. For $\sigma = 0$ we have that*

$$\hat{\boldsymbol{y}_{t+1}}^i = -\eta_t \nabla f(\boldsymbol{x}_t^i) + \sum_{j=1}^{N_p} A_{ij} \boldsymbol{y}_t^j.$$

Then,

$$\begin{aligned}
\boldsymbol{x}_{t+1}^i &= \nabla\Psi^*(\boldsymbol{y}_{t+1}^i) \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ D_\Psi(\boldsymbol{x}, \nabla\Psi^*(\boldsymbol{y}_{t+1}^i)) \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - \Psi \circ \nabla\Psi^*(\boldsymbol{y}_{t+1}^i) - (\boldsymbol{y}_{t+1}^i)^T(\boldsymbol{x} - \nabla\Psi^*(\boldsymbol{y}_{t+1}^i)) \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - (\boldsymbol{y}_{t+1}^i)^T \boldsymbol{x} \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x}_{t+1}^i) + \eta_t \nabla f(\boldsymbol{x}_t^i)^T \boldsymbol{x} - \sum_{j=1}^{N_p} A_{ij} (\boldsymbol{y}_t^j)^T \boldsymbol{x} \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \eta_t \nabla f(\boldsymbol{x}_t^i)^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} \big(\Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x}_{t+1}^i) - (\boldsymbol{y}_t^j)^T \boldsymbol{x} \big) \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \eta_t \nabla f(\boldsymbol{x}_t^j)^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} \big(\Psi(\boldsymbol{x}) - \Psi(\boldsymbol{x}_t^j) - \nabla\Psi(\boldsymbol{x}_t^j)^T (\boldsymbol{x} - \boldsymbol{x}_t^i) \big) \right\} \\
&= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left\{ \eta_t \nabla f(\boldsymbol{x}_t^i)^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} D_\Psi(\boldsymbol{x}, \boldsymbol{x}_t^j) \right\}
\end{aligned}$$

For the second equality we utilise Lemma 1. For the third equality we use $\nabla\Psi \circ \nabla\Psi^*(\boldsymbol{z}) = \boldsymbol{z}$ and for the fourth and fifth equality we add or remove constants since the optimisation is w.r.t. to $\boldsymbol{x}$. Additionally, in the fifth equality we expanded $\boldsymbol{y}_{t+1}^i$.

■

Algorithm 3 (Stochastic) Interacting Mirror Descent Algorithm

Input: $\boldsymbol{X}_0 = [\boldsymbol{x}_0^1, ..., \boldsymbol{x}_0^{N_p}]$, $\eta > 0$ learning rate, $\sigma \geq 0$ noise parameter, $\boldsymbol{A}$ an $N_p \times N_p$ doubly stochastic matrix, $T > 0$ iterations
Output: $\boldsymbol{X}^*$, f^*
$\eta_0 = \eta$
for t = 1,...,T **do**
  for $i = 1, ..., N_p$ **do**

$$\boldsymbol{x}_{t+1}^i = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \left\{ (\eta_t \cdot \boldsymbol{\nabla} f(\boldsymbol{x}_i^t) - \sigma \mathcal{N}(0, \boldsymbol{I}))^T \boldsymbol{x} + \sum_{j=1}^{N_p} \boldsymbol{A}_{ij} D_\Psi(\boldsymbol{x}, \boldsymbol{x}_{t+1}^j) \right\}$$

    if learning rate is adaptive **then**
      Update η_t
    else
      $\eta_t = \eta$
    end if
  end for
end for
$\boldsymbol{X}^* = \boldsymbol{X}_T = [\boldsymbol{x}_T^1, ..., \boldsymbol{x}_T^{N_p}]$
$f^* = \frac{1}{N_p} \sum_{i=1}^{N_p} f(\boldsymbol{x}_T^i)$

4.5 Applications

In Chapter 5 we present results for the optimisation of six convex problems, out of which the five are constrained. We are concerned with two specific constrained optimisation frameworks: one for optimising a convex function

$$f : \mathcal{X} \subseteq \mathbb{R}^n \longrightarrow \mathbb{R},$$

where $\mathcal{X}$ is the n-Simplex and one for optimising a convex function

$$f : \mathcal{X} \subseteq \mathbb{R}^{n \times n} \longrightarrow \mathbb{R},$$

where $\mathcal{X}$ is the n-Spectrahedron. In the rest of this Chapter we provide the update rules for the Mirror Descent and Interacting Mirror Descent algorithms for these two cases.

4.5.1 Optimisation on the Probability Simplex

First, we consider:

$$\min_{\boldsymbol{x} \in \Delta^n} f(\boldsymbol{x})$$

and let Ψ be such that $\Psi(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \log(x_i) - x_i$ (i.e. the negative entropy function). As described in Example 2, Ψ attains all assumptions to be a mirror map.

Lemma 2
The Mirror Descent update rule for $\sigma = 0$ at time-step $t+1$ takes the form:

$$\begin{aligned} y_i^{t+1} &= x_i^t \exp\Big(-\eta_t \big(\nabla f(\boldsymbol{x}_t)\big)_i \Big) \\ x_i^{t+1} &= \frac{y_{t+1}^i}{\sum_{j=1}^{n} y_i^{t+1}}, \quad i = 1, ..., n, \end{aligned} \tag{4.22}$$

where i represents the i-th component of each vector.

Proof 6 *We remind that the associated with Ψ Bregman divergence is given by $D_\Psi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i} - \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i$. From Equation 4 for the update rule at time-step $t+1$ we have:*

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big\{ \eta_t \cdot \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + D_\Psi(\boldsymbol{x}, \boldsymbol{x}_t) \big\} \\ &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big\{ \eta_t \cdot \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + \sum_{i=1}^{n} x_i \log \frac{x_i}{x_i^t} - \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} x_i^t \big\} \\ &= \arg\min_{\boldsymbol{x}\in\mathcal{X}} \big\{ \eta_t \cdot \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + \sum_{i=1}^{n} x_i \log \frac{x_i}{x_i^t} - \sum_{i=1}^{n} x_i \big\} \end{aligned}$$

. The Lagrangian of this optimisation problem is given by:

$$L(\boldsymbol{x}, \lambda, \boldsymbol{\mu}) = \eta_t \cdot \nabla f(\boldsymbol{x}_t)^T \boldsymbol{x} + \sum_{i=1}^{n} x_i \log \frac{x_i}{x_i^t} - \sum_{i=1}^{n} x_i - \lambda \cdot (\sum_{i=1}^{n} x_i - 1) + \boldsymbol{\mu}^T \boldsymbol{x},$$

where due to KKT conditions it holds:

$$\boldsymbol{\mu} \geq 0 \tag{4.23a}$$

$$\mu_i \cdot x_i = 0, \quad i = 1, ..., n \tag{4.23b}$$

$$\sum_{i=1}^{n} x_i = 1 \tag{4.23c}$$

$$\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \lambda, \boldsymbol{\mu}) = \mathbf{0}. \tag{4.23d}$$

We take the derivative of the Lagrangian with respect to the i-th component of $\boldsymbol{x}$ and set it equal to 0:

$$\begin{aligned} \nabla_{x_i} L(\boldsymbol{x}, \lambda, \boldsymbol{\mu}) &= \eta_t \cdot \big(\nabla f(\boldsymbol{x}_t)\big)_i + \log \frac{x_i}{x_i^t} - \lambda + \mu_i = 0 \\ \implies \log(x_i) &= \log(x_i^t) - \eta_t \cdot \big(\nabla f(\boldsymbol{x}_t)\big)_i + \lambda - \mu_i \\ \implies x_i &= e^\lambda \exp\Big(\log(x_i^t) - \eta_t \cdot \big(\nabla f(\boldsymbol{x}_t)\big)_i - \mu_i \Big) \\ \implies x_i &> 0 \quad \implies \mu_i = 0 \end{aligned}$$

Utilising 4.23c we get that:

$$e^{\lambda} = \frac{1}{\sum_{i=1}^{n} x_i^t \exp(-\eta_t \cdot \big(\boldsymbol{\nabla} f(\boldsymbol{x}_t)\big)_i)}$$

and therefore:

$$x_i^{t+1} = \frac{x_i^t \exp(-\eta_t \cdot \big(\boldsymbol{\nabla} f(\boldsymbol{x}_t)\big)_i)}{\sum_{j=1}^{n} x_t^j \exp(-\eta_t \cdot \big(\boldsymbol{\nabla} f(\boldsymbol{x}_t)\big)_j)}. \tag{4.24}$$

■

Lemma 3
The Interacting Mirror Descent update rule for $\sigma = 0$ for the i-th particle at time-step $t+1$ takes the form:

$$(x_{t+1}^i)_m = \frac{B_m^i \exp\Big(- \cdot \eta_t \cdot \big(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\big)_m \Big)}{\sum_{k=1}^{n} B_k^i \exp\Big(- \cdot \eta_t \cdot \big(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\big)_k \Big)}, \quad m = 1, ..., n, \tag{4.25}$$

where $B_m^i = \prod_{j=1}^{N_p} \big((x_t^j)_m\big)^{A_{ij}}$ and m represents the m-th component of each vector.

Proof 7 *Let $\eta_t = 1$:*

$$\begin{aligned} \boldsymbol{x}_{t+1}^i &= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \big\{ \boldsymbol{\nabla} f(\boldsymbol{x}_t^i)^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} D_{\Psi}(\boldsymbol{x}, \boldsymbol{x}_t^j) \big\} \\ &= \arg\min_{\boldsymbol{x} \in \mathcal{X}} \Big\{ \boldsymbol{\nabla} f(\boldsymbol{x}_t^i)^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} \Big(\sum_{m=1}^{n} x_m \log\big(\frac{x_m}{(x_t^j)_m}\big) - \sum_{m=1}^{n} x_m \Big) \Big\} \end{aligned} \tag{4.26}$$

The Lagrangian of this optimisation problem is given by:

$$L(\boldsymbol{x}, \lambda, \boldsymbol{\mu}) = \boldsymbol{\nabla} f(\boldsymbol{x}_t^i)^T \boldsymbol{x} + \sum_{j=1}^{N_p} A_{ij} \Big(\sum_{m=1}^{n} x_m \log\big(\frac{x_m}{(x_t^j)_m}\big) - \sum_{m=1}^{n} x_m \Big) - \lambda \cdot (\sum_{m=1}^{n} x_m - 1) + \boldsymbol{\mu}^T \boldsymbol{x},$$

where due to KKT conditions it holds:

$$\boldsymbol{\mu} \geq 0 \tag{4.27a}$$

$$\mu_m \cdot x_m = 0, \quad m = 1, ..., n \tag{4.27b}$$

$$\sum_{m=1}^{n} x_m = 1 \tag{4.27c}$$

$$\boldsymbol{\nabla}_{\boldsymbol{x}} L(\boldsymbol{x},\, \lambda,\, \boldsymbol{\mu}) \,=\, \mathbf{0}. \tag{4.27d}$$

We take the derivative of the Lagrangian with respect to the m-th, $m = 1, ..., n$ component of $\boldsymbol{x}$ and set it equal to 0:

$$\begin{aligned}
\nabla_{x_m} L(\boldsymbol{x},\, \lambda,\, \boldsymbol{\mu}) \,&=\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \,+\, \sum_{j=1}^{N_p} \left(\log \frac{x_m}{(x_t^j)_m} \right) \,-\, \lambda \,-\, \mu_m \\
&=\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \,+\, \log(x_m) \,+\, \sum_{j=1}^{N_p} A_{ij} \left(\,-\log(x_t^j)_m \right) \,-\, \lambda \,+\, \mu_m \\
&=\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \,+\, \log(x_m) \,-\, \sum_{j=1}^{N_p} \left(\log \left((x_t^j)_m\right)^{A_{ij}} \right) \,-\, \lambda \,+\, \mu_m \\
&=\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \,+\, \log(x_m) \,-\, \log \left(\prod_{j=1}^{N_p} \left((x_t^j)_m\right)^{A_{ij}} \right) \,-\, \lambda \,+\, \mu_m \,=\, 0
\end{aligned}$$

Similarly to the MD case, we derive

$$x_m \,=\, \prod_{j=1}^{N_p} \left((x_t^j)_m\right)^{A_{ij}} \, \exp \left\{ \,-\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \,-\, \mu_m \,+\, \lambda \right\}$$

and therefore $\mu_m = 0$. Let $B_m^i = \prod_{j=1}^{N_p} \left((x_t^j)_m\right)^{A_{ij}}$, utilising the third KKT condition:

$$e^{\lambda} \,=\, \frac{1}{\sum_{m=1}^{n} B_m^i \, \exp \left(\,-\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \right)}.$$

Hence, for $\eta_t = 1$:

$$(x_{t+1}^i)_m \,=\, x_m \,=\, \frac{B_m^i \, \exp \left(\,-\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \right)}{\sum_{k=1}^{n} B_k^i \, \exp \left(\,-\, \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_k \right)}, \quad m \,=\, 1, ..., n.$$

In the general case:

$$(x_{t+1}^i)_m \,=\, \frac{\prod_{j=1}^{N_p} \left((x_t^j)_m\right)^{A_{ij}} \, \exp \left(\,-\, \eta_t \cdot \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_m \right)}{\sum_{k=1}^{n} \left(\prod_{j=1}^{N_p} \left((x_t^j)_k\right)^{A_{ij}} \, \exp \left(\,-\, \eta_t \cdot \left(\boldsymbol{\nabla} f(\boldsymbol{x}_t^i)\right)_k \right) \right)}, \quad m \,=\, 1, ..., n. \tag{4.28}$$

■

4.5.2 Optimisation on the Spectrahedron

Now consider:

$$\min_{\boldsymbol{X} \in \Delta^{n\times n}} f(\boldsymbol{X})$$

and let Ψ be such that $\Psi(\boldsymbol{X}) = \operatorname{tr}\big(\boldsymbol{X}\mathbf{log}(\boldsymbol{X})\big)$ (i.e. the negative von Neumann entropy function). As described in Example 3 Ψ attains all assumptions to be a mirror map.

Lemma 4
The mirror descent update rule for $\boldsymbol{X}_0 \in S^{n\times n}$ for $\sigma = 0$ at time-step $t+1$ takes the form:

$$\begin{aligned} \boldsymbol{Y}_{t+1} &= \boldsymbol{X}_t \circ \mathbf{exp}(-\eta_t \boldsymbol{\nabla} f(\boldsymbol{X}_t)) \\ \boldsymbol{X}_{t+1} &= \frac{1}{\operatorname{tr}(\boldsymbol{Y}_{t+1})} \boldsymbol{Y}_{t+1}, \end{aligned} \tag{4.29}$$

where the operator $\circ$ represents the element-wise product, i.e.

$$(\boldsymbol{A} \circ \boldsymbol{B})_{ij} = A_{ij} \cdot B_{ij}.$$

Lemma 5
The Interacting Mirror Descent update rule of $\boldsymbol{X}_i^{t+1}$ for $\boldsymbol{X}_i^0 \in S^{n\times n}$, $i = 1, ..., N_p$ for $\sigma = 0$ at time-step $t+1$ takes the form:

$$\begin{aligned} \boldsymbol{Y}^i_{t+1} &= \prod_{j=1}^{N_p} \circ \left((\boldsymbol{X}_t^j)^{\circ A_{ij}} \right) \circ \mathbf{exp}(-\eta_t \boldsymbol{\nabla} f(\boldsymbol{X}_t^i)) \\ \boldsymbol{X}^i_{t+1} &= \frac{1}{\operatorname{tr}(\boldsymbol{Y}^i_{t+1})} \boldsymbol{Y}^i_{t+1}, \end{aligned} \tag{4.30}$$

where the operator $\prod_{j=1}^{N_p} \circ$ denotes the elementwise product of matrices and the power $\circ A_{ij}$ is an elementwise exponentiation.

Chapter 5

Experimental Results

In this section we present experimental results of six problems which fit our framework:

- five (convex) constrained optimisation problems (four on the probability simplex and one on the Spectrahedron) (Sections 5.1 - 5.5), and
- one convex unconstrained problem (Section 5.6).

Each problem is explained and formulated in detail below. We solve these problems using three optimisation methods; the standard method, (Stochastic) Gradient Descent, (Stochastic) Mirror Descent and (Stochastic) Interacting Mirror Descent. We discuss and compare our numerical results for the three methods. In case of the latter the interaction matrix A is assumed to be a mean-field interaction matrix, i.e. $A_{ij} = \frac{1}{N_p}$ where N_p denotes the number of particles used in each case. Note that it holds that $\boldsymbol{A}$ is doubly stochastic. In the case of GD/ SGD or MD/ SMD the number of independent and identical runs N_{iid} is stated for each problem.

5.1 The Traffic Assignment problem

The first application we consider is the Traffic Assignment problem, which is concerned with selecting the best path in a traffic system between a given starting point and a destination. The best path is defined as the path which incurs the minimum cost of transporting loads from the origin to the destination. Below, we define the problem more formally. For a more detailed discussion of the formulation of the problem refer to [47].

5.1.1 Formulation of the traffic assignment Problem

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a directed, weighted graph, where $\mathcal{V}$ denotes a set of n vertices (nodes) and $\mathcal{E}$ the set of edges connecting the vertices. Assuming that $\mathcal{V}$ contains n vertices ($|\mathcal{V}| = n$), then $|\mathcal{E}| \leq n \times n$. We denote with $\mathcal{W} \in \mathbb{R}^{n \times n}$ the weights-matrix, where $w_{ij} \neq 0$ if and only if nodes n_i, $n_j \in \mathcal{V}$ are connected with an edge,i.e. n_i, $n_j \in \mathcal{V}$ are not connected if and only if $w_{ij} = 0$. We further denote with $\mathcal{O} \in \mathcal{V}$ the starting point and with $\mathcal{T} \in \mathcal{V}$ the destination. Let also $\mathcal{R}$ be the set of simple

routes (paths) connecting $\mathcal{O}$ and $\mathcal{T}$ within the graph and let $|\mathcal{R}| = n_{\mathcal{R}}$. We are concerned with sending μ units of traffic from $\mathcal{O}$ to $\mathcal{T}$ via $r \in \mathcal{R}$. Note that for the experiments performed in this project we set $\mu = 1$ for simplicity. Let $\mathcal{X}$ be the feasible set defined as:

$$\mathcal{X} = \left\{ \boldsymbol{x} \in \mathbb{R}^{n_{\mathcal{R}}} \,|\, x_r \geq 0, \forall r = 1, .., n_{\mathcal{R}}, \quad \sum_{r=1}^{n_{\mathcal{R}}} x_r = \mu = 1 \right\} = \Delta^{n_{\mathcal{R}}} \subset \mathbb{R}^{n_{\mathcal{R}}}.$$

A vector $[x_1, .., x_{n_{\mathcal{R}}}]^T = \boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{x}$ is called a routing flow. For a routing flow $\boldsymbol{x} \in \mathcal{X}$, the load on an edge $\epsilon \in \mathcal{E}$ is given by $l_\epsilon(\boldsymbol{x}) = \sum_{r \subset \mathcal{R} : \epsilon \in r} x_r$. The cost (delay) occurring on an edge ϵ is denoted as $c_\epsilon(\boldsymbol{x})$ where $c_\epsilon : \mathbb{R} \longrightarrow \mathbb{R}_+$ is a non decreasing convex function in $\mathbb{R}$.

For the purposes of this project we assume that

$$c_\epsilon(\boldsymbol{x}) = \frac{1}{2} w_\epsilon \, (1 + l_\epsilon(\boldsymbol{x}))^2, \quad \forall \epsilon \in r \subset \mathcal{R},$$

where w_ϵ denotes the weight of the two vertices of edge ϵ. Note that c_ϵ is convex on $\mathbb{R}^{n_{\mathcal{R}}}$:

Proof 8 *Let* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{n_{\mathcal{R}}}, \quad \alpha \in (0, 1)$:

$$\begin{aligned}
c_\epsilon(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y}) &= \frac{1}{2} w_\epsilon \, (1 + l_\epsilon(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y}))^2 \\
&= \frac{1}{2} w_\epsilon \, (1 + \alpha l_\epsilon(\boldsymbol{x}) + (1 - \alpha) l_\epsilon(\boldsymbol{y}))^2 \\
&= \frac{1}{2} w_\epsilon \, \big(1 + \alpha l_\epsilon(\boldsymbol{x}) + (1 - \alpha) l_\epsilon(\boldsymbol{y}) + \alpha - \alpha\big)^2 \\
&= \frac{1}{2} w_\epsilon \, \big(\alpha(1 + l_\epsilon(\boldsymbol{x})) + (1 - \alpha)(1 + l_\epsilon(\boldsymbol{y}))\big)^2 \\
&\leq \frac{1}{2} w_\epsilon \, \alpha^2 \big(1 + l_\epsilon(\boldsymbol{x})\big)^2 + \frac{1}{2} w_\epsilon \, (1 - \alpha)^2 \big(1 + l_\epsilon(\boldsymbol{y})\big)^2 \\
&\leq \alpha c_\epsilon(\boldsymbol{x}) + (1 - \alpha) c_\epsilon(\boldsymbol{y}).
\end{aligned}$$

■

The cost through route $r \in \mathcal{R}$ is defined as the sum of costs over all edges on the path, i.e.:

$$c_r(\boldsymbol{x}) = \sum_{\epsilon \in r} c_\epsilon(\boldsymbol{x})$$

and the mean cost of the network is obtained by

$$f(\boldsymbol{x}) = \sum_{r \in \mathcal{R}} x_r \, c_r(\boldsymbol{x}) = \sum_{r \in \mathcal{R}} \sum_{\epsilon \in \mathcal{E}} x_r \, c_\epsilon(\boldsymbol{x}) = \sum_{\epsilon \in \mathcal{E}} l_\epsilon(\boldsymbol{x}) \, c_\epsilon(\boldsymbol{x}).$$

For our setting the loss function is given by:

$$f(\boldsymbol{x}) = \frac{1}{2} \sum_{\epsilon \in \mathcal{E}} l_\epsilon(\boldsymbol{x}) \, w_\epsilon \, (1 + l_\epsilon(\boldsymbol{x}))^2, \quad \boldsymbol{x} \in \Delta^{n_{\mathcal{R}}}. \tag{5.1}$$

The gradient of f as obtained by the chain rule is given by:

$$\frac{\partial f}{\partial x_r}(\boldsymbol{x}) = \sum_{\epsilon \in r} \frac{d\big(l_\epsilon(\boldsymbol{x})\, c_\epsilon(\boldsymbol{x})\big)}{dl_\epsilon(\boldsymbol{x})} = \sum_{\epsilon \in r} c_\epsilon(\boldsymbol{x}) + l_\epsilon(\boldsymbol{x})\, c'_\epsilon(l_\epsilon(\boldsymbol{x}))$$
$$= \sum_{\epsilon \in r} \left(\frac{1}{2}\, w_\epsilon\, (1 + l_\epsilon(\boldsymbol{x}))^2 + l_\epsilon(\boldsymbol{x})\, w_\epsilon\, (1 + l_\epsilon(\boldsymbol{x})) \right), \quad r = 1, .., n_{\mathcal{R}}. \tag{5.2}$$

5.1.2 Numerical Results

For this problem we generate two geographical threshold graphs consisting of $|\mathcal{V}| = 50, 100$ vertices placed uniformly at random in a rectangular domain. The resulting generated graph consisting of 50 vertices is illustrated in Figure 5.1. Each vertex is assigned a random weight generated from the exponential distribution. Given a threshold θ, the vertices are joined by an edge as proposed in [35, 12]. Origin and destination nodes $\mathcal{O}\,(= 46, 8)$ and $\mathcal{T}\,(= 32, 74)$ respectively, are chosen randomly.

We then calculate all simple paths between the origin and destination of each graph with maximum length (henceforth called cutoff and denoted $r_{\max}$). We choose $r_{\max} = 5$ or 6 in the case of $|\mathcal{V}| = 50$ and $r_{\max} = 7$ in the case of $|\mathcal{V}| = 100$, which amount to a total of 70 and 1169 , and 1647 distinct paths respectively. Therefore, the dimension of each problem is equal to $n_{\mathcal{R}} = 70$, 1169 and $n_{\mathcal{R}} = 1647$. We perform experiments on both problems and plot the convergence of the loss function for the three methods.

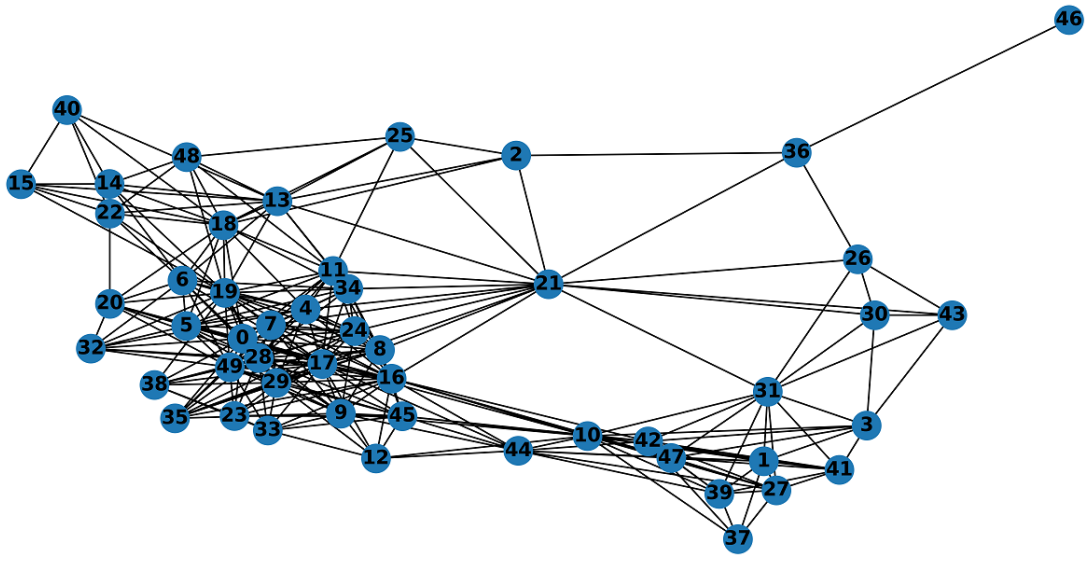

Figure 5.1: Traffic Assignment Problem: Random Geographical Threshold Graph, $|\mathcal{V}| = 50$ $(\mathcal{O}, \mathcal{T}) = (46, 32)$.

Experiments for random graph with $|\mathcal{V}| = 50$

We begin by demonstrating results on the random geographical threshold graph illustrated in Figure 5.1 above which contains 50 vertices. In Figure 5.2 (a) and (b) we plot the convergence of the loss function of the problem resulting when $r_{\max} = 5$ ($n_{\mathcal{R}} = 70$) and $r_{\max} = 6$ ($n_{\mathcal{R}} = 1169$) respectively, after running the three algorithms for $T = 500$ iterations.

For the experiment in (a) we use fixed learning rate $\eta = 0.01$ for the deterministic Gradient Descent method and fixed learning rate $\eta = 0.2$ for deterministic Mirror Descent and its variant IMD. For GD and MD we run 5 identical and independent copies, whilst for IMD we use 5 interacting particles. We can observe that all three methods perform comparably. As expected, since the dimensionality of the problem is low ($n_{\mathcal{R}} = 70$), all methods converge to the minimum of the objective function in less than 50 iterations.

For the experiment in (b) we set $\sigma = 0.01$ and run 10 iid copies of SGD and SMD and use 5 interacting particles in the SIMD case. All other choices of hyperparameters are the same as (a). All three algorithms manage to converge to the optimum even with the added noise. However, we can notice that SIMD with 5 interacting particles and SMD outperform the SGD as they converge faster to the optimum.

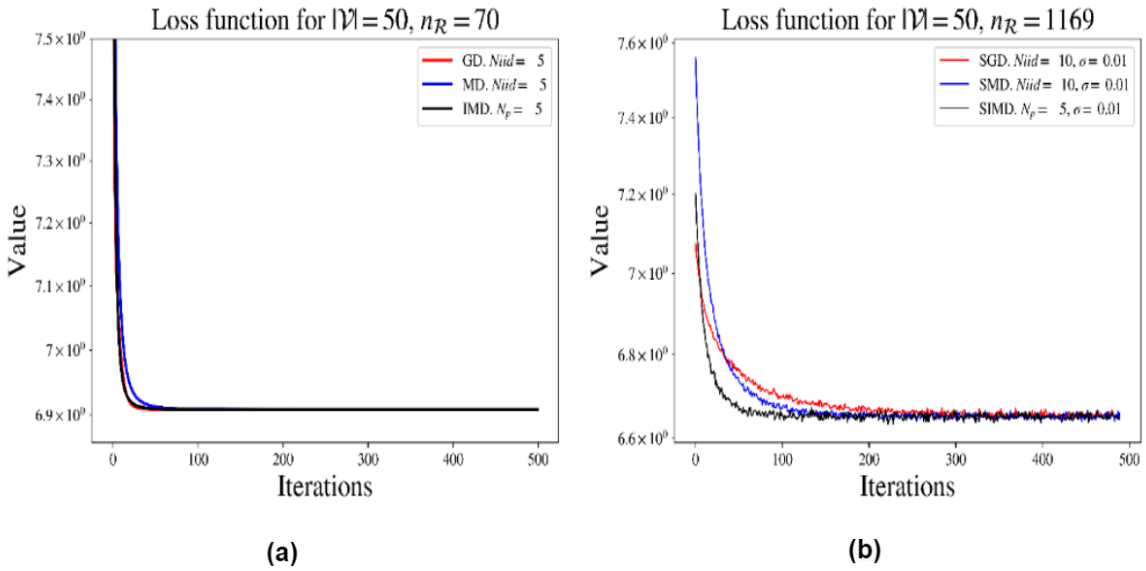

Figure 5.2: Traffic Assignment Problem: **(a)** Convergence of deterministic GD, MD and IMD in the case of $n_{\mathcal{R}} = 70$, **(b)** Convergence of SGD, SMD and SIMD in the case of $n_{\mathcal{R}} = 1169$.

Experiments for random graph with $|\mathcal{V}| = 100$

In this paragraph we provide the results of our experiments for a graph consisting of 100 vertices and cutoff parameter $r_{\max} = 7$. Note that the dimension of the problem is now $n_{\mathcal{R}} = 1647$.

We run all three methods in the deterministic case with noise parameter $\sigma = 0.1$ for $T = 500$ iterations and compare their performance. In the deterministic case we set a fixed learning rate $\eta = 0.005$ for GD and, $\eta = 0.05$ for MD and IMD. In the stochastic case we use a decreasing learning rate $\eta_t = \frac{0.01}{\sqrt{t}}$ for SGD and, $\eta_t = \frac{0.5}{\sqrt{t}}$ for SMD and SIMD.

In Figure 5.3 (a), we plot the convergence of the loss function w.r.t. the iterations in the stochastic case ($\sigma = 0.1$) with a decreasing learning rate. Despite the fact that SIMD and SMD converge faster than SGD, all three methods, seem to converge to a neighbourhood of the solution (the fluctuation term is governed by the noise parameter σ).

However, as shown in Figure 5.4, when using fixed learning rates in the deterministic case, IMD outruns the other two methods again as it seems to converge faster to a minimum solution. In contrast, as anticipated due to the high dimensionality of the problem, GD performs much worse than MD and IMD, as it never converges to the optimum. Moreover, the samples of the loss function in the case of GD are noisier and have higher variance.

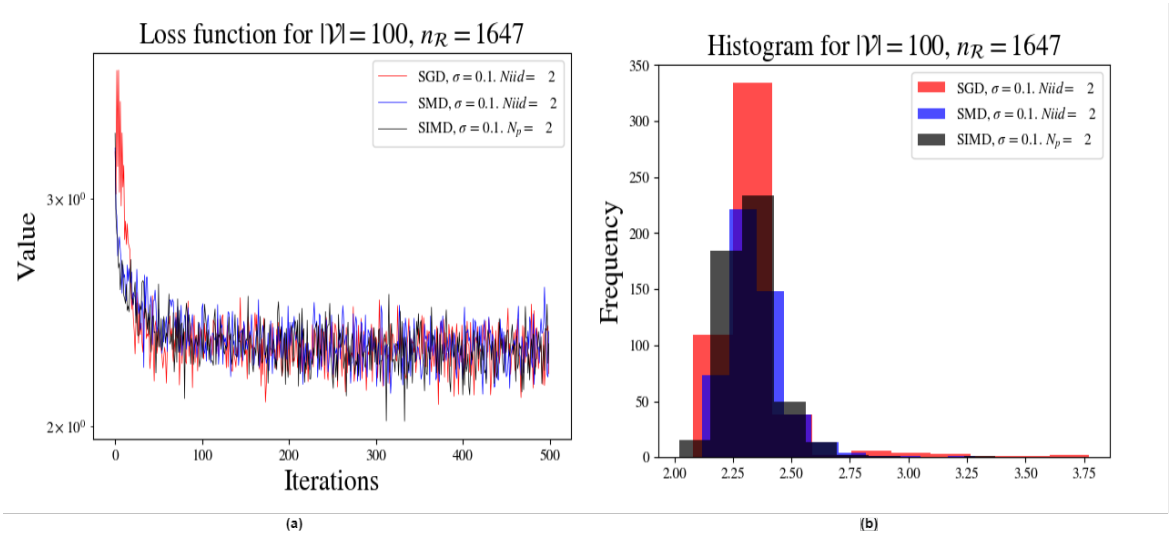

Figure 5.3: Traffic Assignment Problem: Convergence of stochastic optimisation with decreasing learning rate: SGD, SMD and SIMD in the case of higher dimensionaliy, $n_{\mathcal{R}} = 1647$.

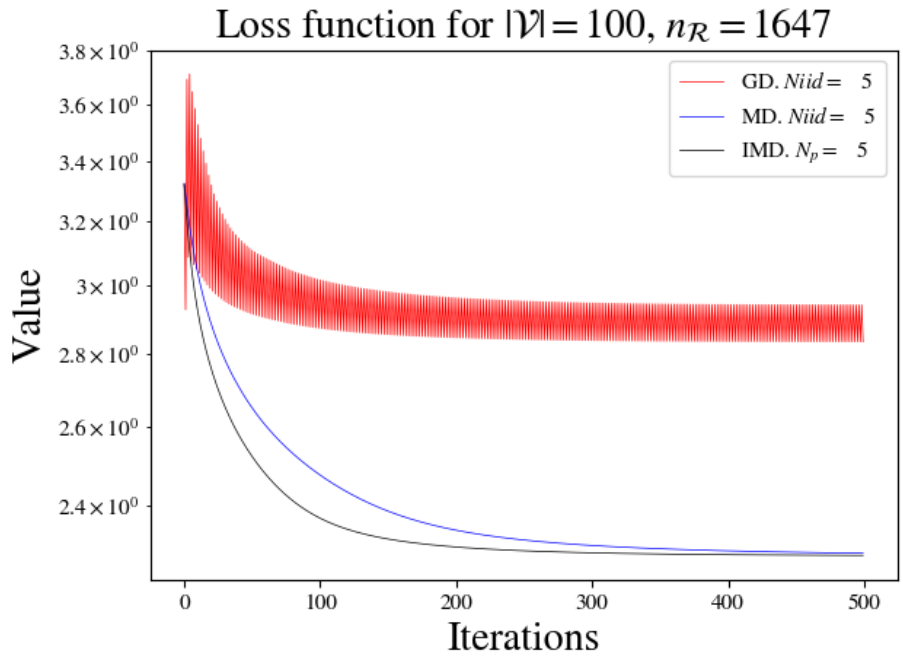

Figure 5.4: Traffic Assignment Problem: Convergence of deterministic optimisation with fixed learning rate: GD, MD and IMD in the case of higher dimensionaliy, $n_{\mathcal{R}} = 1647$.

In Figure 5.3 (b), we provide a histogram of the samples of then function value in the stochastic case. In the next sections we are going to be studying more about the variance of the samples for each method and we will study the effect of interaction in noise reduction.

5.2 Linear System

In this section we are concerned with a standard optimisation problem of a linear system, often referred to as (constrained) Linear Regression, with similar a setup as Section 5.1 of [10] and Example 9.19 of [4] (where they solve with PGD).

5.2.1 Formulation of the linear system problem with simplex constraints

Let $f(\boldsymbol{x}) = \frac{1}{2}||\boldsymbol{W}\,\boldsymbol{x} - \boldsymbol{b}||_2^2, \quad \boldsymbol{x} \in \mathbb{R}^n$, where $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $m \geq n$. We assume that $rank(\boldsymbol{W}) = n$. We are concerned with optimising f over the n-Simplex:

$$\min_{\boldsymbol{x}\in\mathcal{X}=\Delta^n} f(\boldsymbol{x}). \tag{5.3}$$

Note that f is a continuous and twice differentiable function:

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \boldsymbol{W}^T(\boldsymbol{W}\,\boldsymbol{x} - \boldsymbol{b}), \quad \boldsymbol{\nabla}^2 f(\boldsymbol{x}) = \boldsymbol{W}^T\boldsymbol{W}.$$

Since $rank(\boldsymbol{W}) = n$, the Hessian of f is positive definite $\forall\,\boldsymbol{x} \in \mathbb{R}^n$ and thus, f is strictly convex on $\mathcal{X} \subset \mathbb{R}^n$.

5.2.2 Numerical Results

Here we present convergence results of the three methods (PGD, MD, IMD) for $n = m = 1000$. For each case, we generate randomly $\boldsymbol{b} \sim \mathcal{N}(0, \boldsymbol{I}_n)$ and we also generate randomly one well and two ill-conditioned $\boldsymbol{W}$, with $\kappa(\boldsymbol{W}) = 10$ and $\kappa(\boldsymbol{W}) = 100, 200$ using the methods proposed by the authors in [6].

Single particle Optimisation: (S)GD vs (S)MD

First, we study the convergence of Stochastic Gradient Descent and Stochastic Mirror Descent (we set as the noise parameter $\sigma = 0.5$) executed in 10 identical and independent runs ($N_{iid} = 10$). We run both optimisation algorithms on two problems with condition of $\boldsymbol{W}$, $\kappa(\boldsymbol{W}) = 10$ and $\kappa(\boldsymbol{W}) = 100$. We then draw in parallel the loss function of each method for both cases and compare their convergence. Here, we run each algorithm for $T = 1000$ iterations and use decreasing learning rate $\eta_t = \frac{0.1}{\sqrt{t}}$.

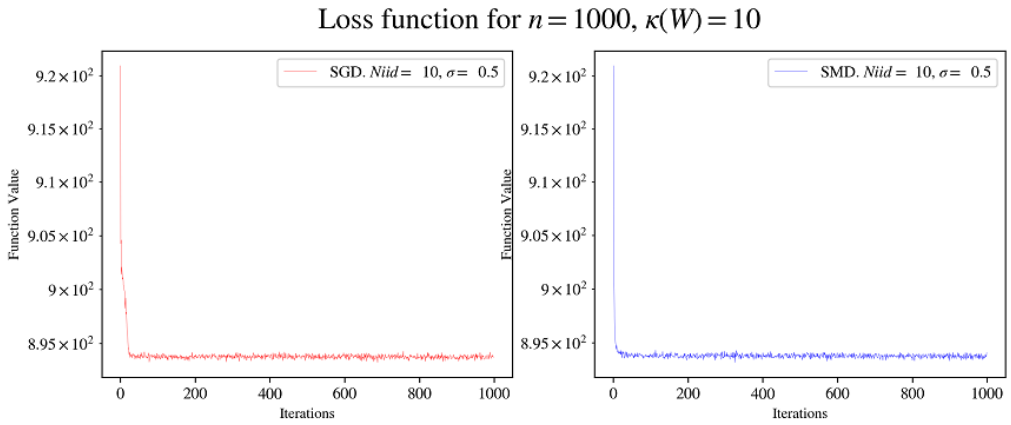

Figure 5.5: Linear System: SGD convergence (L) vs SMD convergence (R) for a well-conditioned W with $\kappa(\boldsymbol{W}) = 10$.

In Figures 5.5 and 5.6 we plot the convergence of the loss function for both cases. We observe that when $\kappa(\boldsymbol{W}) = 10$, the two methods attain similar performance as they both converge to the optimum in just a few iterations. When $\boldsymbol{W}$ is ill-conditioned, Mirror Descent seems to outperform Gradient Descent as shown in Figure 5.6, as the former converges in a few iterations, while the latter converges much slower and converges after approximately 800 iterations.

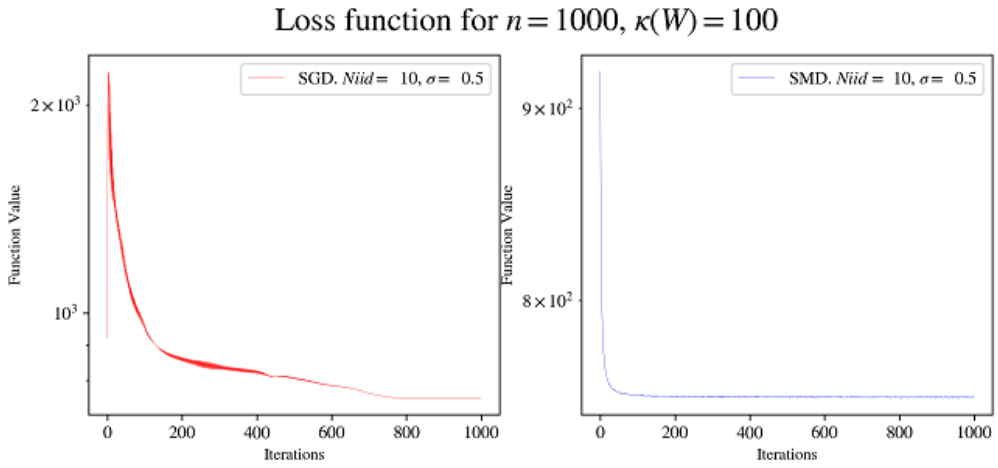

Figure 5.6: Linear System: SGD convergence (L) vs SMD convergence (R) for an ill-conditioned W with $\kappa(\boldsymbol{W}) = 100$.

Interacting particles Optimisation

Next, we conduct experiments to investigate the convergence properties of Interacting Mirror Descent. We run all algorithms in this subsection for $T = 1000$ iterations.

Comparing S(MD) vs (S)IMD. We compare the iterations of $N_{iid} = 10$ runs of MD and SMD with single particle against one run of IMD and SIMD with $N_p = 5$ interacting particles. Figure 5.7 shows that both methods converge to the optimum

in less than 100 iterations in the case of a well-conditioned W, $\kappa(\boldsymbol{W}) = 10$. Both algorithms are also able to converge even when adding some noise ($\sigma = 0.1$). In Figure 5.8, we illustrate a histogram of samples after convergence. Both methods perform similarly in both, stochastic and deterministic settings.

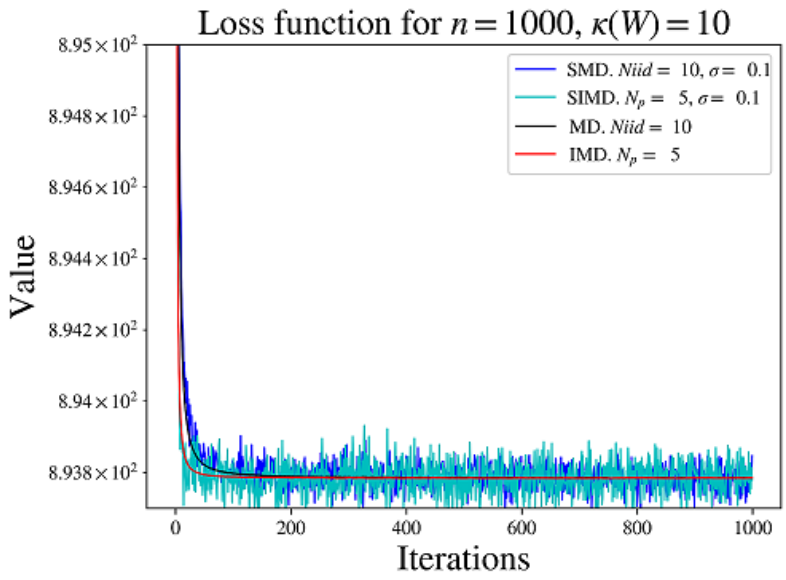

Figure 5.7: Linear System: (S)MD convergence vs (S)IMD convergence for a well-conditioned W with $\kappa(\boldsymbol{W}) = 10$.

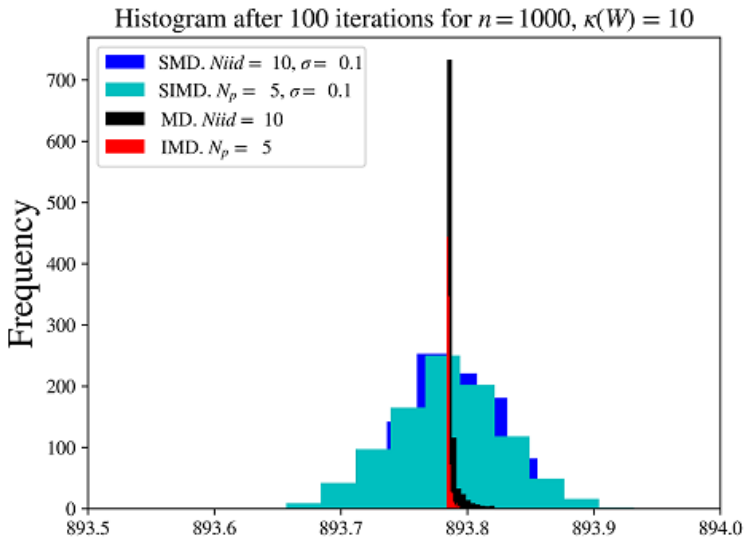

Figure 5.8: Linear System: Histogram of (S)MD and (S)IMD convergence for a well-conditioned W with $\kappa(\boldsymbol{W}) = 10$.

Interacting particles for Variance Reduction Theoretical results in Section 3 of [10] imply that better convergence is expected for larger number of interacting particles (N_p) used in Equation 4.21. In this paragraph we also investigate the effects of interaction in terms of the number of particles N_p against the variance of the objective function after some $t > T_0$ iterations of the algorithm.

Using decreasing learning rate $\eta_t = \frac{0.1}{\sqrt{t}}$ we run the stochastic case of IMD for $T = 1000$ iterations for various numbers of interacting particles $N_p \in \{1, 10, 50, 100\}$ in the cases when $\kappa(\boldsymbol{W}) = 10$ and when $\kappa(\boldsymbol{W}) = 200$ and plot the results in Figure 5.9.

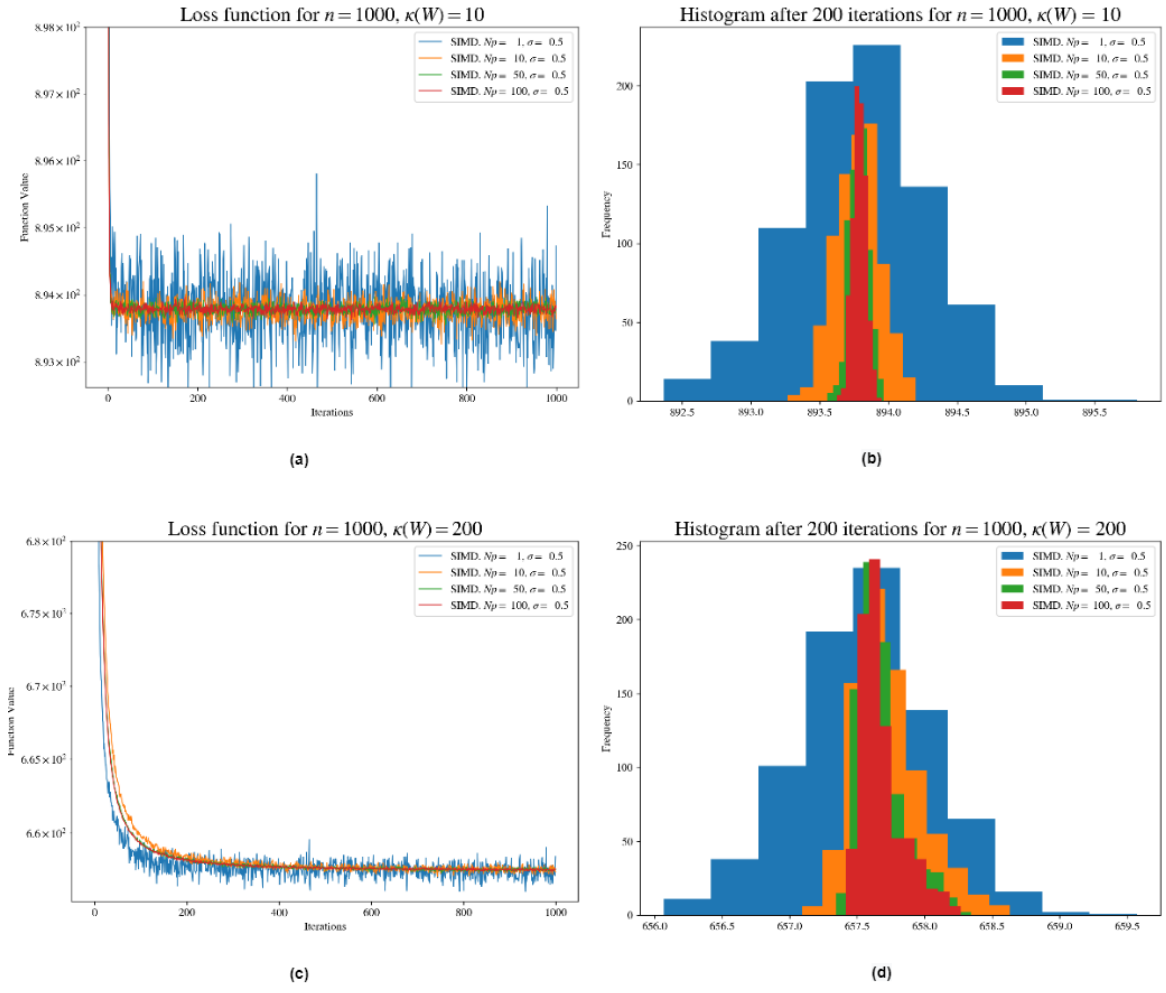

Figure 5.9: Linear System: Effect of number of interacting particles on variance of samples of the loss function after convergence for cases of ill and well-conditioned $\boldsymbol{W}$, $\kappa(\boldsymbol{W}) = 10, 200$.

In Table 5.1 we present the sample variance of the loss function for the two problems illustrated in Figure 5.9. Our calculations of the variance were performed on the samples obtained after $T_0 = 200$ iterations when all methods have converged to the optimum.

First, we observe that the variance of the samples is also dependent on the condition number of the matrix $\boldsymbol{W}$. Table 5.1 suggests that the variance is higher in the ill-conditioned $\boldsymbol{W}$ than the well-conditioned $\boldsymbol{W}$. Additionally, we remark (as preserved also from the illustration and as expected), the variance of the samples decreases as the number of interacting particles increases in both cases.

Variance of Interacting Mirror Descent		
$\boldsymbol{N_p}$	$\boldsymbol{\kappa(W) = 10}$	$\boldsymbol{\kappa(W) = 200}$
1	0.2324	0.2417
10	0.02553	0.0652
50	0.00480	0.0290
100	0.00242	0.0251

Table 5.1: Variance after convergence (after 200 iterations of SIMD) for different number of interacting particles and condition number $\kappa(\boldsymbol{W})$. Illustrated above in histograms.

5.3 Linear System: Mini-batch Optimisation

5.3.1 Problem formulation

In this section we investigate a problem with similar settings as the previous section with the difference that here we are interested in minimising the average over $M > 1$ data samples instead of a single one with Simplex constraints, where M is the sample size. Such setups are widely used in Machine and Deep Learning problems where it is standard to minimise an objective function of the form

$$f(\boldsymbol{x}) = \frac{1}{M} \sum_{m=1}^{M} f_m(\boldsymbol{x}).$$

Specifically, our optimisation framework here is:

$$\min_{\boldsymbol{x} \in \mathcal{X} = \Delta^n} f(\boldsymbol{x}) = \min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{M} \sum_{m=1}^{M} f_m(\boldsymbol{x}) \tag{5.4}$$

where

$$f_m(\boldsymbol{x}) = \frac{1}{2} ||\boldsymbol{W}_m \boldsymbol{x} - \boldsymbol{b}_m||_2^2$$

and $\boldsymbol{W}_m \in \mathbb{R}^{n \times n}$, $\boldsymbol{b}_m \in \mathbb{R}^n$ for $m = 1, ..., M$. It's important to note that since f_m is convex and continuously differentiable (see Section 5.2 for more details), f is also convex and continuously differentiable as the sum of convex and continuously differentiable functions.

Assuming that $\mathcal{O}$ is the set of all observations ($|\mathcal{O}| = M$), the gradient of the objective function in this setting is usually computed over a subset $\mathcal{B} \subseteq \mathcal{O}$ of size $M_{\mathcal{B}} \leq M$ i.e.:

$$\begin{aligned} \boldsymbol{\nabla} f(\boldsymbol{x}) \approx \boldsymbol{\nabla} f_{\mathcal{B}}(\boldsymbol{x}) &= \frac{1}{M_{\mathcal{B}}} \sum_{m\,:\,(\boldsymbol{W}_m, \boldsymbol{b}_m) \in \mathcal{B}} \boldsymbol{\nabla} f_m(\boldsymbol{x}) \\ &= \frac{1}{M_{\mathcal{B}}} \sum_{m\,:\,(\boldsymbol{W}_m, \boldsymbol{b}_m) \in \mathcal{B}} \boldsymbol{W}_m^T (\boldsymbol{W}_m \boldsymbol{x} - \boldsymbol{b}_m) \end{aligned} \tag{5.5}$$

where

$$f_{\mathcal{B}}(\boldsymbol{x}) = \frac{1}{M_{\mathcal{B}}} \sum_{m\,:\,(\boldsymbol{W}_m, \boldsymbol{b}_m) \in \mathcal{B}} f_m(\boldsymbol{x}).$$

This is usually called Mini-Batch Optimisation. The need for Mini-Batch Optimisation is due to the fact that for large M, processing of the entire observation set $\mathcal{O}$ before taking a step in gradient descent is expensive in terms of both time and computational power [11, 21].

5.3.2 Numerical Results

For the purposes of our experiments in this section all $\boldsymbol{W}_m$ and $\boldsymbol{b}_m$, $m = 1, ..., M$ are generated by adding some noise to a random non-singular full rank matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ with a specific condition number $\kappa(\boldsymbol{W})$ and a random bias vector $\boldsymbol{b} \in \mathbb{R}^n$, similarly to the previous section.

For our experiments we choose as the size of our observation set $|\mathcal{O}| = M = 100$. We plot and discuss results for $n = 500$ and $\kappa(\boldsymbol{W}) = 10, 100, 200$. For our calculations here we use a fixed learning rate $\eta = 0.05$.

First, we execute all three algorithms on a problem with an ill-conditioned $\boldsymbol{W}$, $\kappa(\boldsymbol{W}) = 100$ and $M_{\mathcal{B}} = 5$ with stochastic settings ($\sigma = 2$). We provide a plot in Figure 5.10 and compare their convergence. We observe that SIMD with 10 interacting particles governs the other two methods (averaged over 10 iid copies) as it manages to converge faster and with lower variance.

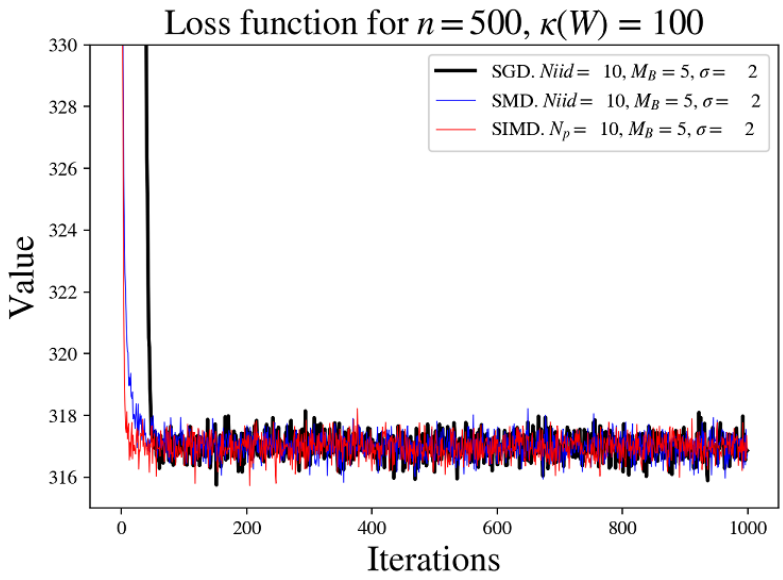

Figure 5.10: Mini-Batch Optimisation: Loss function for SGD and SMD ($N_{iid} = 10$) and SIMD ($N_p = 10$) for $M_{\mathcal{B}}$ in the case of $\kappa(\boldsymbol{W}) = 100$.

Effect of number of particles N_p in Mini-Batch Optimisation

Experiments for fixed mini-batch size $M_{\mathcal{B}}$ In this paragraph we present results for a well-conditioned $\boldsymbol{W}$ with condition number $\kappa(\boldsymbol{W}) = 10$. We study the effect of the number of interacting particles used in (S)IMD on the convergence of the loss function when the mini-batch size is rather small. We illustrate the results in Figures 5.11, and in Table 5.2 we calculate the variance of the samples for each method.

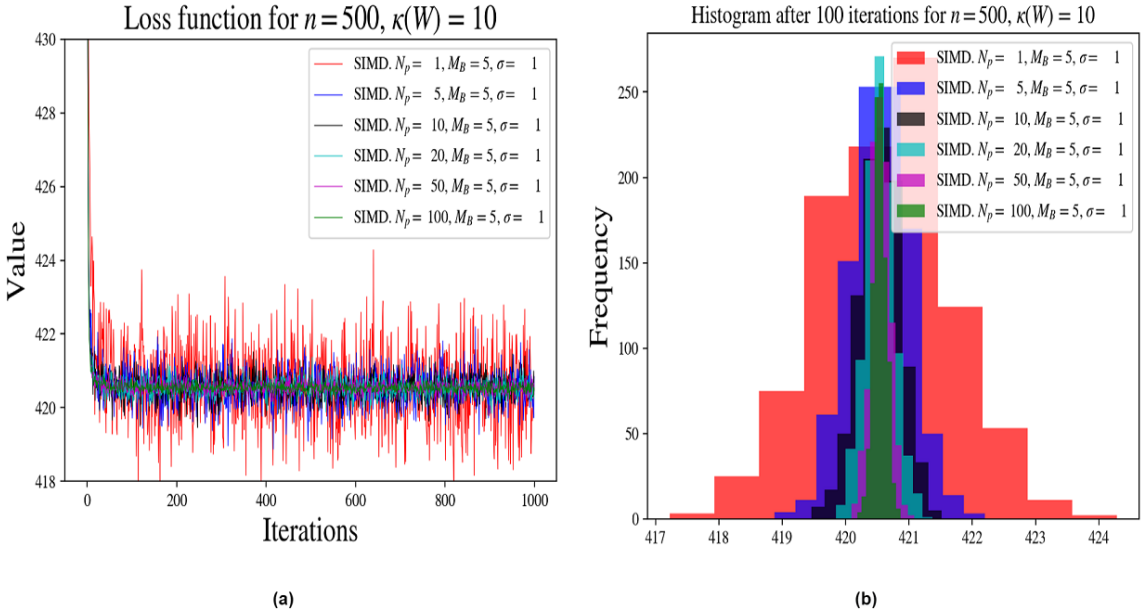

Figure 5.11: Mini-Batch Optimisation: **(a)** Loss function and **(b)** Histogram of loss samples after convergence for SIMD ($\sigma = 1.0$) for increasing number of interacting particles and fixed batch size $M_{\mathcal{B}} = 5$ in the case of $\kappa(\boldsymbol{W}) = 10$.

We execute the Stochastic Interacting Mirror Descent algorithm six times for different number of particles, one for each $N_p \in \{1, 5, 10, 20, 50, 100\}$ and fixed batch size $M_{\mathcal{B}} = 5$ and we plot the results. Figure 5.11 and Table 5.2 both show that a higher number of interacting particles (N_p) can decrease the noise of the calculations of the loss function even for a small sized batch.

N_p	$M_{\mathcal{B}}$	Variance of Interacting Mirror Descent for $\kappa(\boldsymbol{W}) = 10$
1	**5**	1.05135
5	**5**	0.21831
10	**5**	0.10661
20	**5**	0.05277
50	**5**	0.02210
100	**5**	0.01010

Table 5.2: Mini-Batch Optimisation: Variance of SIMD for fixed mini-batch size and increasing number of interacting particles after convergence.

Experiments for different combinations of N_p and $M_{\mathcal{B}}$ Ideally, the gradient of the loss function should be computed over the whole training batch. However, this can be expensive both, in terms of computation time and resources. Hence, we are interested in the performance of (S)IMD with respect to the size of the mini-batch since the gradient of the loss function is calculated over mini-batches and is expected to by noisy. We carry out experiments on two problems with $\kappa(\boldsymbol{W}) = 10$, 200 for

different combinations of the number of interacting particles and the mini-batch size and illustrate the results in Figures 5.12 and 5.13.

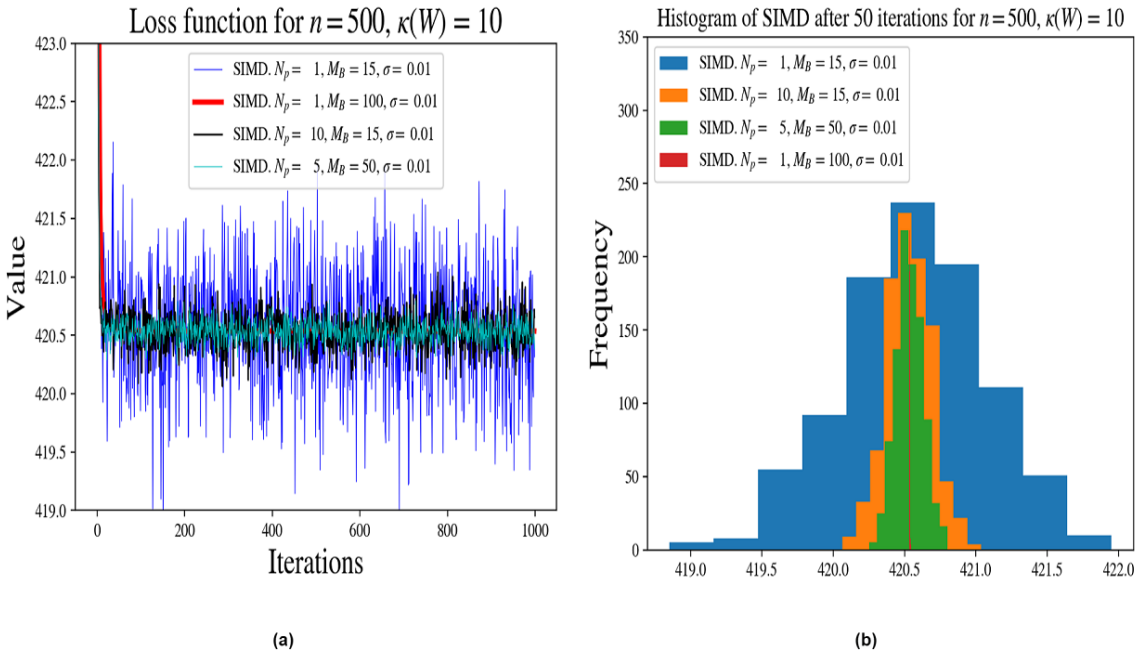

Figure 5.12: Mini-Batch Optimisation: **(a)** Loss function and **(b)** Histogram of loss samples after convergence for SIMD ($\sigma = 0.01$) for different combinations of batch sizes ($M_{\mathcal{B}}$) and number of interacting particles (N_p) in the case of $\kappa(\boldsymbol{W}) = 10$.

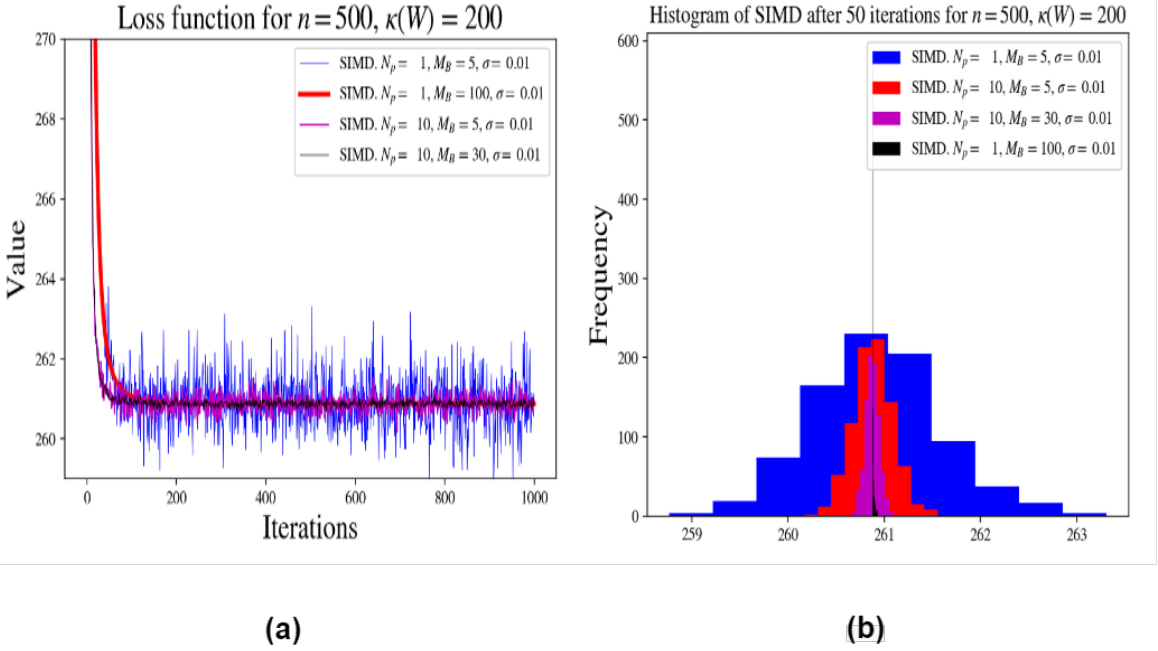

Figure 5.13: Mini-Batch Optimisation: **(a)** Loss function and **(b)** Histogram of loss samples after convergence for SIMD ($\sigma = 0.01$) for different combinations of batch sizes ($M_{\mathcal{B}}$) and number of interacting particles (N_p) in the case of $\kappa(\boldsymbol{W}) = 200$.

N_p	$M_{\mathcal{B}}$	$\kappa(W)$	Variance of Interacting Mirror Descent
1	**15**	**10**	0.25049
10	**15**	**10**	0.02466
5	**50**	**10**	0.00895
1	**100 (Full batch)**	**10**	0.00001
1	**5**	**200**	0.45695
10	**5**	**200**	0.04244
10	**30**	**200**	0.00515
1	**100 (Full batch)**	**200**	0.00066

Table 5.3: Mini-Batch Optimisation: Variance of SIMD for different number of interacting particles (N_p) and different mini-batch sizes ($M_{\mathcal{B}}$) after convergence.

As expected, computing the gradient over a full batch ($M_{\mathcal{B}} = 100$ here) exhibits less noise around the optimum in both cases. However, as illustrated in 5.12 and 5.13 when computing the gradient over mini-batches, using a higher number of interacting particles can result in reduction of the variance of the samples of the loss function. We also provide in Table 5.3 the variance of each method plotted in these figures.

Our calculations in the two previous paragraphs suggest that interaction is a way of combating noisy gradients and helps in reducing the variance. This result can prove helpful in cases in which it is more expensive to use large batch sizes rather than large number of interacting particles.

5.4 The Laplacian *K*-Modes for Clustering

5.4.1 Problem formulation

In this section we consider the Laplacian K-modes algorithm for clustering described in [51]. For N samples $\boldsymbol{x}_m \in \mathbb{R}^D$, $m = 1, ..., N$, we define $\boldsymbol{X} = [\boldsymbol{x}_1 \dots x_N]^T$. Let $\boldsymbol{W}$ be a symmetric matrix that denotes the affinity matrix of the undirected graph defined by $\boldsymbol{X}$, $\mathcal{G} = \mathcal{G}(\boldsymbol{X})$. Here, we assume that $\boldsymbol{W}$ is either binary (the Adjacency matrix of $\mathcal{G}$):

$$W_{mn} = \begin{cases} 1 & \text{if } \boldsymbol{x}_m \in kNN(\boldsymbol{x}_n) \\ 0 & otherwise, \end{cases}$$

or obtained by the Radial Basis Function (RBF):

$$W_{mn} = \exp\big(- \gamma \, ||\boldsymbol{x}_m - \boldsymbol{x}_n||_2^2 \big)$$

for $n, m = 1, ..., N$. Given the above, we want to optimise the objective function w.r.t. $\boldsymbol{Z} = [\boldsymbol{z}_1^T, .., \boldsymbol{z}_N^T]^T \in \mathbb{R}^{N \times K}$ and $\boldsymbol{C} = [\boldsymbol{c}_1^T, .., c_K^T]^T \in \mathbb{R}^{K \times D}$:

$$\begin{aligned} \min_{\boldsymbol{Z}, \boldsymbol{C}} \frac{\lambda}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} W_{mn} \, ||\boldsymbol{z}_m - \boldsymbol{z}_n||_2^2 - \sum_{m=1}^{N} \sum_{k=1}^{K} z_{mk} \, \mathcal{N}\bigg(\frac{||\boldsymbol{x}_m - \boldsymbol{c}_k||_2^2}{\sigma^2} \bigg) \\ s.t. \quad \sum_{k=1}^{K} z_{mk} = 1, \quad m = 1, ..., N, \\ z_{mk} \geq 0, \quad m = 1, ..., N \quad k = 1, ..., K, \end{aligned} \tag{5.6}$$

where $\lambda \geq 0$ represents a trade-off parameter, each $[z_{m1} .., z_{mK}]^T = \boldsymbol{z}_m \in \mathbb{R}^K$ represents soft assignments of sample $\boldsymbol{x}_m$ to K clusters and each $\boldsymbol{c}_k, \in \mathbb{R}^K$, $k = 1, ..., K$ represents modes (centers) of each cluster. Note that $\mathcal{N}(\cdot)$ denotes the Normal probability distribution function.

Let further $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ denote the Laplacian of $\mathcal{G}$, where $\boldsymbol{D} = \boldsymbol{diag}([\sum_{n=1}^{N} W_{mn}]_{m=1}^{N})$ is the degree matrix of $\mathcal{G}$. The matrix form of 5.6 is given by:

$$\begin{aligned} \min_{\boldsymbol{Z}, \boldsymbol{C}} f(\boldsymbol{Z}, \boldsymbol{C}) = \min_{\boldsymbol{Z}, \boldsymbol{C}} \lambda \, \mathrm{tr}(\boldsymbol{Z}^T \boldsymbol{L} \, \boldsymbol{Z}) - \mathrm{tr}(\boldsymbol{B}^T \boldsymbol{Z}) \\ s.t. \quad \boldsymbol{Z} \, \boldsymbol{1}_K = \boldsymbol{1}_N, \\ z_{mk} \geq 0, \quad m = 1, ..., N \quad k = 1, ..., K, \end{aligned} \tag{5.7}$$

where the elements of $\boldsymbol{B}$ are

$$B_{mk} = \mathcal{N}\bigg(\frac{||\boldsymbol{x}_m - \boldsymbol{c}_k||_2^2}{\sigma^2} \bigg), \quad m = 1, ..., N, \quad k = 1, ..., K.$$

Note that since $\boldsymbol{L}$ is positive semi-definite, it can be decomposeed as a product $\boldsymbol{L} = \boldsymbol{A} \, \boldsymbol{A}^T$ such that $\boldsymbol{A}$ has full row rank $r = rank(\boldsymbol{L})$ and thus, $\mathrm{tr}(\boldsymbol{Z}^T \boldsymbol{L} \, \boldsymbol{Z}) = ||\boldsymbol{Z}^T \boldsymbol{A}||_F^2$.

The optimisation problem 5.7 is convex w.r.t $\boldsymbol{Z}$:

Proof 9 *Assume a fixed* $\boldsymbol{B}$*. Let* $g : \mathcal{X} \longrightarrow \mathbb{R}$ *be defined by:*

$$g(\boldsymbol{Z}) = \lambda \operatorname{tr}(\boldsymbol{Z}^T \boldsymbol{L}\, \boldsymbol{Z}) - \operatorname{tr}(\boldsymbol{B}^T \boldsymbol{Z})$$

and let $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{N\times K}$*, such that* $\boldsymbol{x}_m, \boldsymbol{y}_m \in \mathcal{X} = \Delta^N$*,* $m = 1, ..., N$*,* $\alpha \in (0, 1)$*.*

$$\begin{aligned} g(\alpha\, \boldsymbol{X} + (1-\alpha)\, \boldsymbol{Y}) &= \frac{\lambda}{2}\big|\big|(\alpha\, \boldsymbol{X} + (1-\alpha)\, \boldsymbol{Y})^T \boldsymbol{A}\big|\big|_F^2 - \operatorname{tr}(\boldsymbol{B}^T(\alpha\, \boldsymbol{X} + (1-\alpha)\, \boldsymbol{Y})) \\ &\leq \frac{\lambda}{2}\big|\big|\alpha\, \boldsymbol{X}^T \boldsymbol{A}\big|\big|_F^2 + \frac{\lambda}{2}\big|\big|(1-\alpha)\, \boldsymbol{Y}^T \boldsymbol{A}\big|\big|_F^2 - \operatorname{tr}(\boldsymbol{B}^T(\alpha\, \boldsymbol{X} + (1-\alpha)\, \boldsymbol{Y})) \\ &= \alpha g(\boldsymbol{X}) + (1-\alpha) g(\boldsymbol{Y}). \end{aligned}$$

■

The gradient of f w.r.t $\boldsymbol{Z}$ is given by:

$$\boldsymbol{\nabla}_{\boldsymbol{Z}} f(\boldsymbol{Z}, \boldsymbol{C}) = 2 \cdot \lambda\, \boldsymbol{L}\, \boldsymbol{Z} - \boldsymbol{B} \tag{5.8}$$

and w.r.t. $\boldsymbol{c}_k$, $k = 1, .., K$:

$$\begin{aligned} \boldsymbol{\nabla}_{\boldsymbol{c}_k} f(\boldsymbol{Z}, \boldsymbol{C}) &= \sum_{m=1}^{N} \frac{2}{\sigma^2} \cdot z_{mk} \cdot \mathcal{N}'\left(\frac{||\boldsymbol{x}_m - \boldsymbol{c}_k||_2^2}{\sigma^2}\right)(\boldsymbol{x}_m - \boldsymbol{c}_k) \\ &= -\sum_{m=1}^{N} \frac{2}{\sigma^2} \cdot z_{mk} \cdot \frac{||\boldsymbol{x}_m - \boldsymbol{c}_k||_2^2}{\sigma^2} \cdot \mathcal{N}\left(\frac{||\boldsymbol{x}_m - \boldsymbol{c}_k||_2^2}{\sigma^2}\right)(\boldsymbol{x}_m - \boldsymbol{c}_k). \end{aligned} \tag{5.9}$$

The optimisation with respect to each row $\boldsymbol{z}_m$, $m = 1, ..., N$, of $\boldsymbol{Z}$ (for a fixed $\boldsymbol{C}$) is a convex constrained optimisation problem on the K-Simplex and therefore fits our framework. We utilise the matrix form of the problem and update $\boldsymbol{Z}$ "altogether" in the cases of MD and IMD. The optimisation with respect to $\boldsymbol{C}$ for a fixed $\boldsymbol{Z}$ is unconstrained and solved with the traditional method, GD.

5.4.2 Numerical Results

To solve (5.7) we alternate between optimising w.r.t. $\boldsymbol{Z}$ for a fixed $\boldsymbol{C}$ and optimising w.r.t. $\boldsymbol{C}$ for a fixed $\boldsymbol{Z}$. It should be mentioned that in [51] they suggest that there is a trade-off between the two hyperparameters λ and σ which we do not investigate here as it is out of the scope of this project. We specify λ and σ for each experiment.

We present results for two problems by generating synthetic data. For the first, we generate N random data-points in K isotropic Gaussian blobs using the method `make_blobs` from [42]. For the second problem, we generate N two-dimensional data-points split into two "half-moon" shaped clusters (i.e. $D = K = 2$) using the method `make_moons` from [42].

In Figures 5.14 and 5.15 are illustrated examples of generated datasets for each of

the problems. Note that for both problems we choose randomly the initial centroids $\boldsymbol{c}_k^0$, $k = 1, ..., K$ which we optimise.

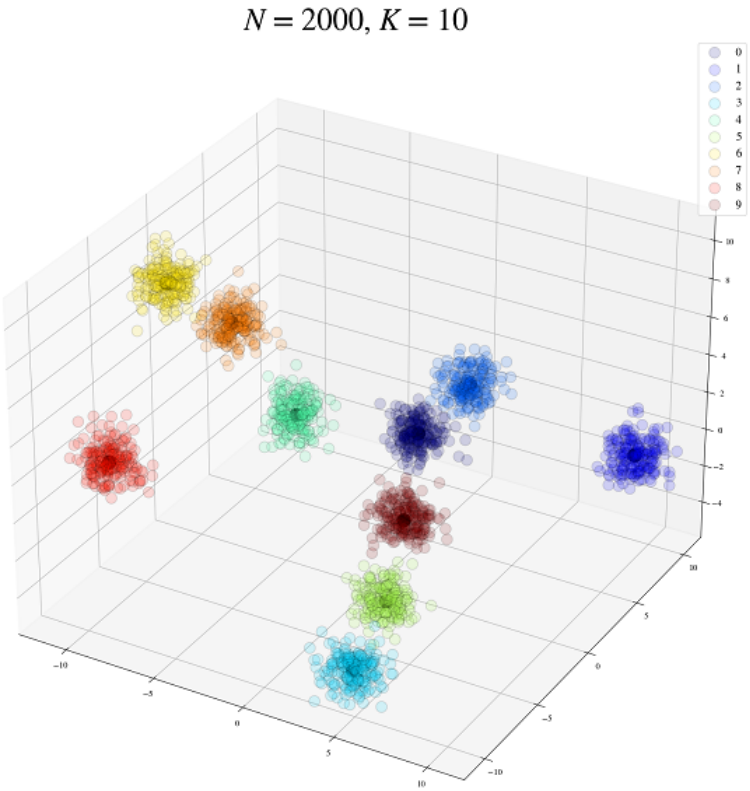

Figure 5.14: Laplacian K-Modes for Clustering: $N = 2000$ randomly generated data points in the 3D space divided into $K = 10$ clusters (isotropic Gaussian blobs). The black dots illustrate the (true) centroid $\boldsymbol{c}_k$ of each cluster.

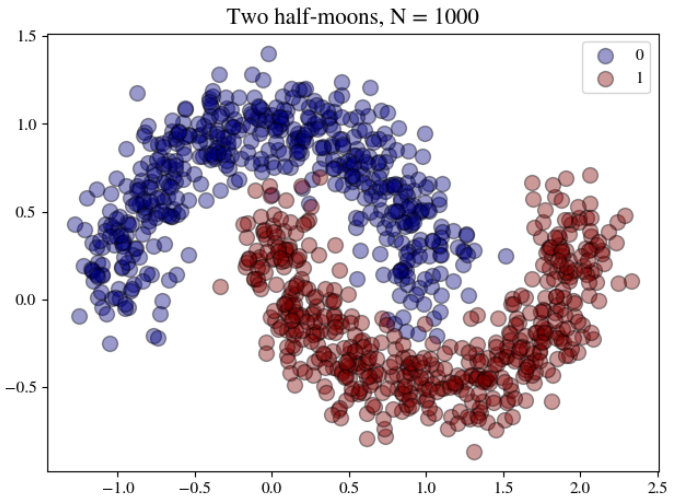

Figure 5.15: Laplacian K-Modes for Clustering: $N = 1000$ randomly generated data points in the 2D space divided into $K = 2$ half-moon shaped clusters.

Isotropic Gaussian blobs

In this paragraph, we report results for the optimisation of the problem proposed in Equation 5.7 for $K = 10$ 3D isotropic Gaussian blobs as shown in Figure 5.14. First, we study the performance of the deterministic Gradient Descent ran in two identical and independent runs and the deterministic Interacting Mirror Descent with two particles. For our experiments here as hyperparameters we set $\lambda = 100$ and $\sigma = 10$. Note that as an initialisation of $\boldsymbol{c}_k$ we randomly generate K vectors in the box bounded by all N samples.

We demonstrate the assignment results after running each algorithm for $T = 1000$ iterations in Figure 5.16 and Table 5.4. It should be mentioned that we assign each sample to a cluster by using a classification threshold $\theta = 0.1$, i.e. we assign a sample $\boldsymbol{x}_m, \quad m = 1, .., N$ in cluster $k, \quad k = 1, ..., K = 10$ if:

$$Z_{mk} \in \left[\frac{k-1}{K}, \frac{k}{K}\right).$$

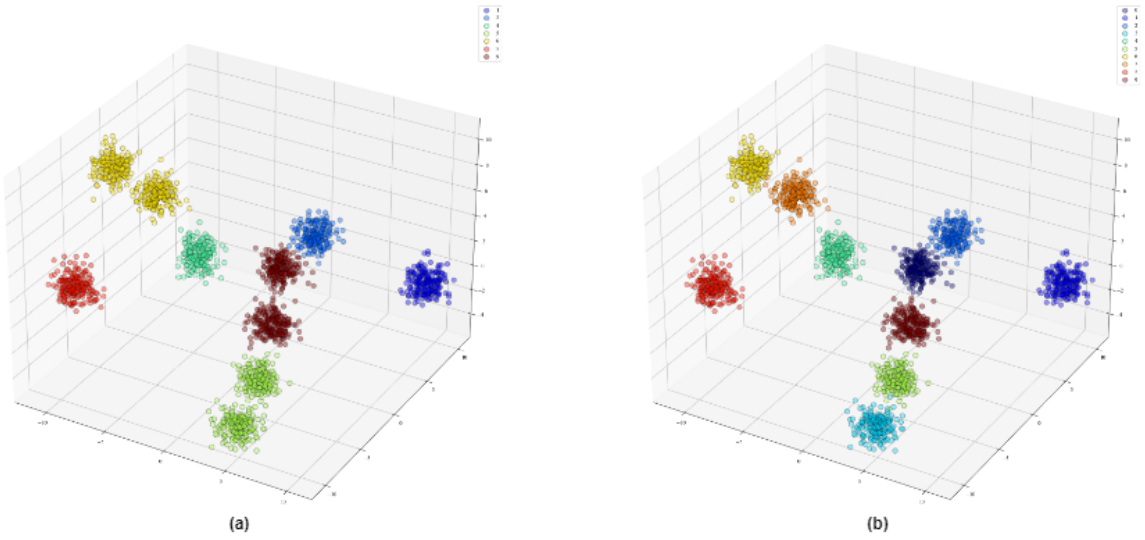

Figure 5.16: Laplacian K-Modes for Clustering: Cluster assignment results of (a) GD and (b) IMD using a classification threshold equal to 0.1. Their achieved numerical "accuracies" are reported in Table 5.4.

	AUC	**"Accuracy"**
GD	0.9833	0.8493
IMD	0.9996	0.9980

Table 5.4: Laplacian K-Modes for Clustering: Classification performance of GD and IMD in the case of $K = 10$ synthetically generated Isotropic Gaussian 3D blobs. Each blob contains approximately $\frac{N}{K} = \frac{2000}{10} = 200$ data samples.

We also plot in Figure 5.17 the performance of both methods on the optimisation problem for $N = 2000$. Once again, similarly to the previous problems, we remark that IMD reaches the global optimum faster than GD.

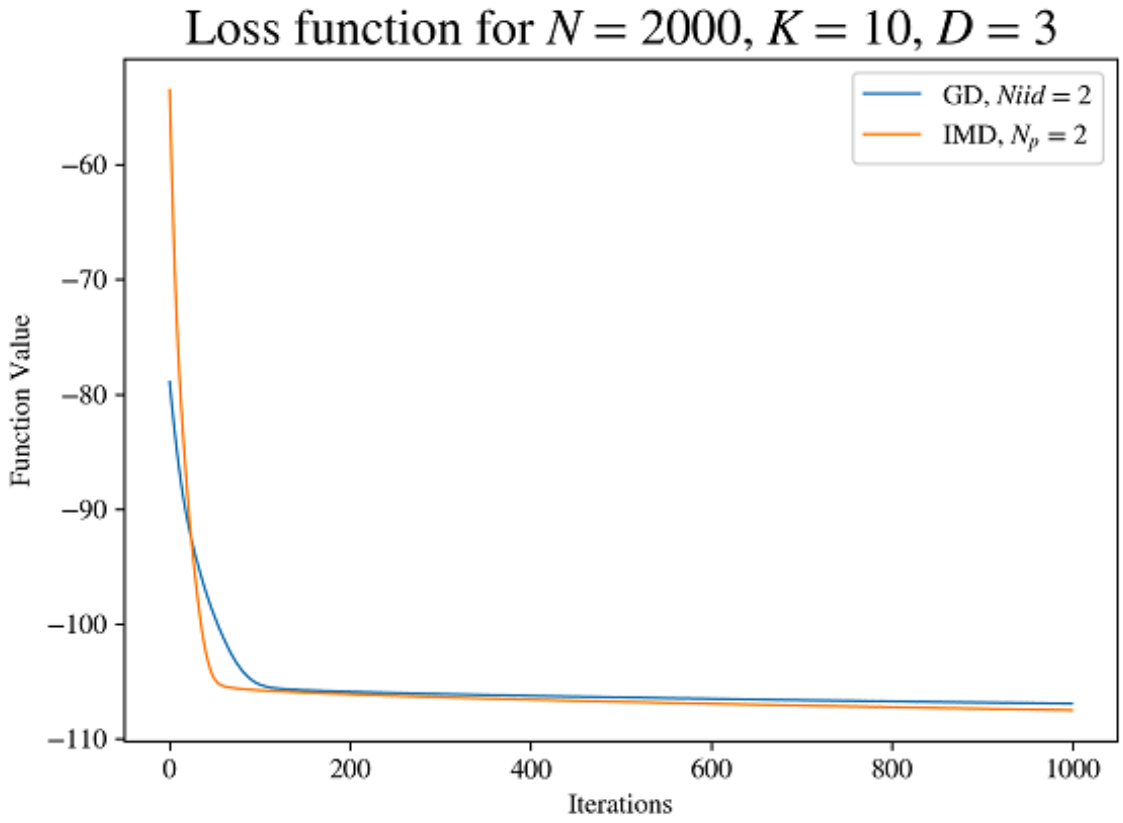

Figure 5.17: Laplacian K-Modes for Clustering: Loss convergence of GD and IMD for $N = 2000$.

Below, in Table 5.5, are indicated the average run-times for all three methods studied in this project. As anticipated, Gradient Descent is much more computationally expensive, as at each iteration we project $N = 2000$ vectors on the K-Simplex which is costly in terms of time and computational power. In contrast, Interacting Mirror Descent seems to be robust to large dimensionality problems, as expected.

	Average Runtime for a single iteration
GD	1.215 s for 1 iid run
IMD	0.840 s per particle

Table 5.5: Laplacian K-Modes for Clustering: Average run-time for one optimisation iteration for the three methods in the case of synthetic Isotropic Gaussian blobs.

Two "half-moons"

We study the performance of the deterministic versions of Gradient and Mirror Descent ran in two identical and independent runs and Interacting Mirror Descent with two particles. For our experiments here, as hyperparameters we set $\lambda = 2$ and $\sigma = 0.65$. We assign each sample to each cluster similarly to the previous problem with threshold $\theta = 0.5$, i.e. we assign $\boldsymbol{x}_n$ to cluster $k = 0$ if $z_{n0} < 0.5$ and to cluster $k = 1$ otherwise.

Below we include a plot in Figure 5.19 of the assignments of each method and we provide a table with their respective scores (Table 5.6). Also, in Figure 5.18, we plot the loss functions achieved by the three methods. We observe that in all three cases the performance is comparable. However, IMD achieves a higher "accuracy" and a better convergence to the minimum than the other methods.

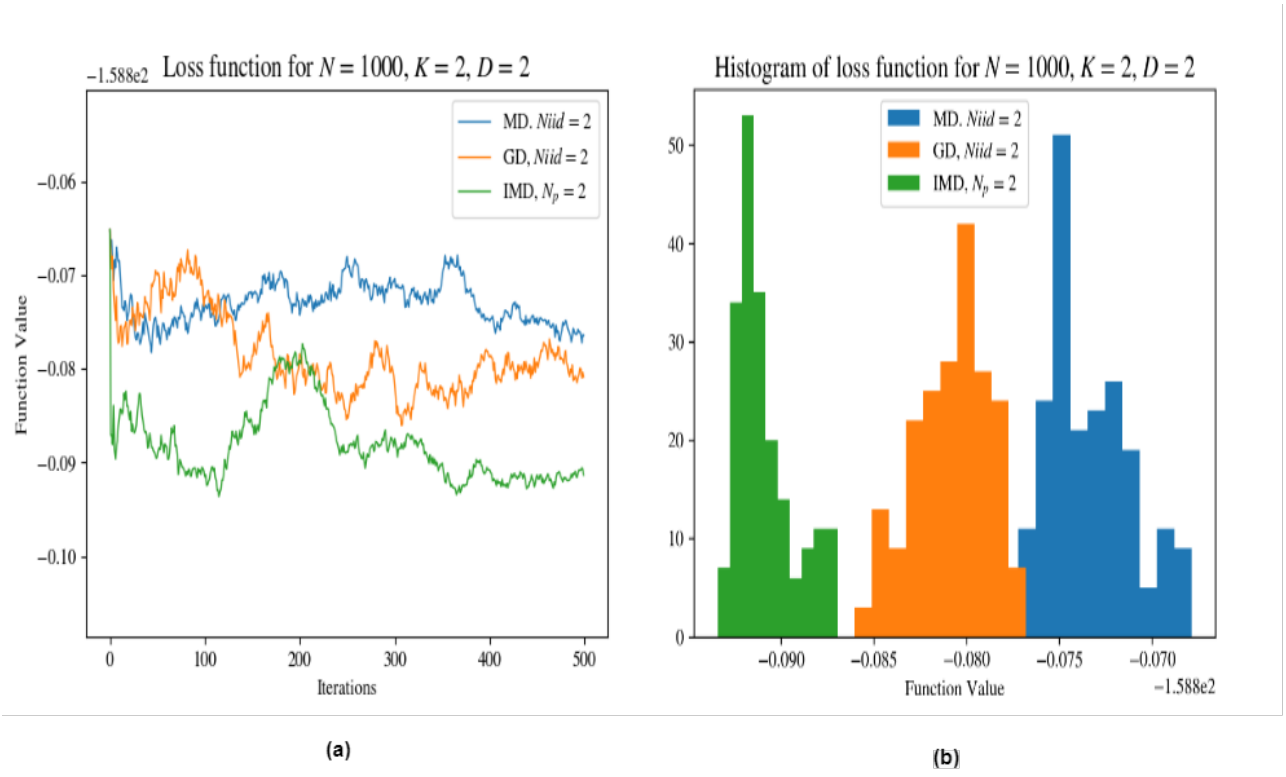

Figure 5.18: Laplacian K-Modes for Clustering: **(a)** Loss function and **(b)** Histogram of loss samples after convergence for the three methods for $N = 1000$.

	AUC	"Accuracy"
GD	0.815	0.8664
MD	0.814	0.8539
IMD	0.805	0.9086

Table 5.6: Laplacian K-Modes for Clustering: Classification performance of the three methods in the case of two synthetically generated half-moons clusters.

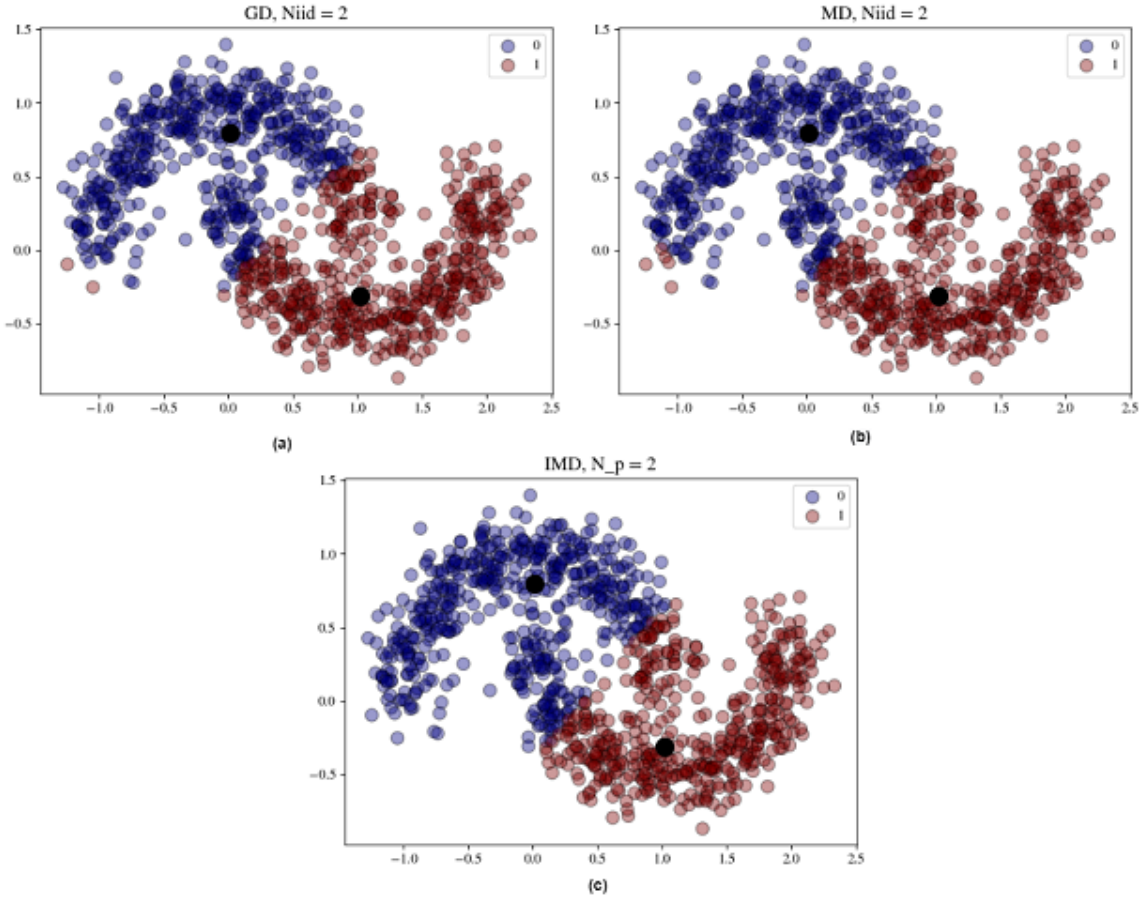

Figure 5.19: Laplacian K-Modes for Clustering: Cluster assignment results of **(a)** GD, **(b)** MD and **(c)** SIMD using a classification threshold equal to 0.5 in the case of two synthetically generated half-moons clusters. The black points illustrate the predicted centroids $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$.

5.5 Optimisation on SDPs

In this section we investigate a convex optimisation problem on SDPs for PSD matrices over the n-Spectrahedron. Specifically, we are concerned with connected subgraph anomaly detection in graph-structured data. Such problems can be applied in numerous instances, such as detection of anomalies in disease outbreaks, detection of anomalous incidents in social networks, identification of anomalous groups, clusters or paths in graphs, etc. We investigate a convex relaxation of a problem that aims in detecting whether there exist any connected sets of vertices in a graph that transmit abnormal signal values [1, 60].

5.5.1 Connected Subgraph Detection on SDPs: Problem formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a directed, weighted graph, where $\mathcal{V}$ denotes a set of n vertices and $\mathcal{E}$ the set of edges connecting the vertices. For root vertex $r \in S$, we consider an Elevated Mean Detection, i.e. we want to detect the existence of a connected in G subgraph $S \in \Lambda_r = \{S \subset V : r \in S, G_S \text{ is connected in G}\}$ with vertices transmitting an elevated mean compared to the other nodes. See [1] for a detailed formulation of the problem. In particular, we aim to solve a relaxed convex SDP optimisation problem as shown in Section 4 of [1]:

$$\begin{aligned} \max_{s \geq 0,\, \boldsymbol{M} \in \Delta^{n \times n}} \quad & \boldsymbol{C} \bullet \boldsymbol{M} - s \\ s.t. \quad & \boldsymbol{Q}_\gamma(\boldsymbol{M}) + s \cdot \delta \cdot \boldsymbol{D} \succeq \boldsymbol{0} \end{aligned} \tag{5.10}$$

where $\boldsymbol{C} = \boldsymbol{x}\,\boldsymbol{x}^T$ for $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \geq \boldsymbol{0}$, s a slack variable, $\delta \geq 0$ a fixed margin and $\boldsymbol{D}$ is the degree matrix of the given graph and $\boldsymbol{Q}_\gamma(\boldsymbol{M}) = \boldsymbol{L}_{G[\boldsymbol{M}]} - \frac{\gamma^2}{2}\boldsymbol{L}_{Star(r)[\boldsymbol{M}]}$. For more details on the operators $\boldsymbol{L}_{G[\cdot]}$, $\frac{\gamma^2}{2}\boldsymbol{L}_{Star(r)[\cdot]}, \boldsymbol{Q}_\gamma(\cdot)$ refer to [1]. Note that the value of γ is used to quantify the size and conductance of the anomalous graph. In our experiments in the next section we use different values of γ to get our results.

Now, let $\boldsymbol{Y} \succeq \boldsymbol{0}$ be the Lagrange multiplier in the Lagrangian of the optimisation problem associated to the constraint $\boldsymbol{Q}_\gamma(\boldsymbol{M}) + s \cdot \delta \cdot \boldsymbol{D}$. Then the dual of Equation 5.10 is given by the following problem :

$$\min_{\boldsymbol{Y} \in \Delta_D^{n \times n}} f(\boldsymbol{Y}) \tag{5.11}$$

where

$$f(\boldsymbol{Y}) = \max_{\boldsymbol{M} \in \Delta^{n \times n}} \left(\boldsymbol{C} + \frac{1}{\delta} \cdot \boldsymbol{P}_\gamma(\boldsymbol{Y})\right) \bullet \boldsymbol{Y}$$

and

$$\Delta_{\boldsymbol{D}}^{n \times n} = \left\{ \boldsymbol{X} \in \mathcal{S}_+^{n \times n} \mid \operatorname{tr}(\boldsymbol{D} \cdot \boldsymbol{X}) = 1 \right\}.$$

Note that $\Delta_{\boldsymbol{D}}^{n\times n}$ is a convex set in $\mathbb{R}^{n\times n}$ and therefore the optimisation over the Lagrange multiplier $\boldsymbol{Y}$ fits our framework. We utilise the MD and IMD algorithms for PSD matrices presented in Chapter 3 over the n-Spectrahedron. It should also be mentioned that for the optimisation of the primal variable M we utilise the update rule presented in Section 4 of [1] which is out of the scope of this project.

We present performance results of our methods on two problems: Disease Outbreak Anomaly Detection for a real case scenario and Elevated Mean anomaly classification on randomly generated geometric graphs. We will be referring to SNR $= \dfrac{\lambda_1}{\lambda_0}$ as signal-to-noise ratio where λ_0 denotes the base disease ratio and λ_1 the anomalous disease ratio.

5.5.2 Disease Outbreak Anomaly Detection: Numerical Results

Here we study a real-world graph $G = (\mathcal{V}, \mathcal{E})$, also studied in [1]. $\mathcal{V}$ constitutes of 129 vertices corresponding to 129 counties of Northeastern United Sates. For our null hypothesis (H_0) we consider a subgraph $S \subset G$ of size $|\mathcal{V}_S| = 16$ as the abnormal case, where $\mathcal{V}_S = \{69, 66, 105, 93, 106, 95, 49, 90, 68, 59, 104, 62, 88, 108, 84, 92\}$.

Letting p_j denote the population of each county, we assume the number of disease cases for each county as $y_j \sim Poisson(p_j\lambda_0)$ and $y_j \sim Poisson(p_j\lambda_1)$ for the non-abnormal and abnormal counties respectively. We then let $\boldsymbol{C} = \boldsymbol{x}\boldsymbol{x}^T$ where $x_j = \dfrac{y_j}{p_j}$ for $j = 1, ..., 129$ denotes the rate of infection in each county. See Figure 5.20 for an illustration of the graph and the ground truth subgraph.

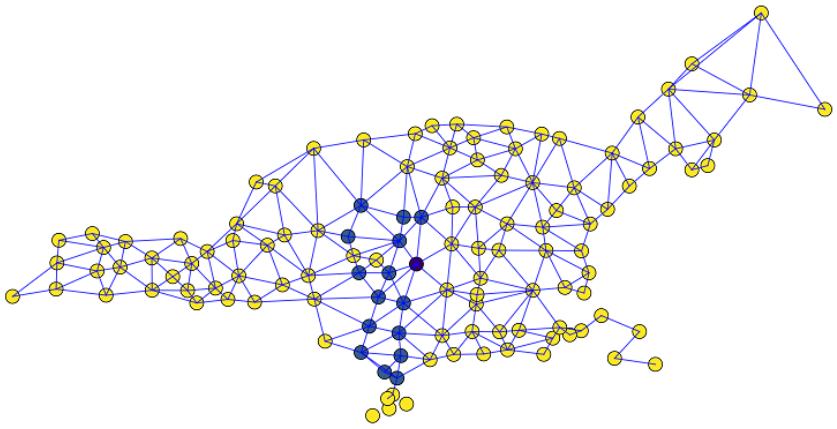

Figure 5.20: Disease Outbreak Anomaly Detection: a real case scenario, Graph of NE United States. Nodes colored blue illustrate H_0 for the anomalous subgraph (denoted $\mathcal{V}_S$ in the previous paragraph). The anchor node is colored purple.

Performance of IMD vs MD for different values of γ

In Figures 5.21 and 5.22 we illustrate the detection performance results of both, Mirror Descent and Interacting Mirror Descent respectively, for 10 different values of γ evenly spaced on log scale between 0.01 and 5.00. To measure the performance we use different values of threshold and calculate the AUC score by calculating for each threshold the true positive and false positive rates of the classification of anomalous clusters over a sample set generated randomly from the null hypothesis H_0 and alternative hypothesis H_1. We execute all optimisation algorithms over a random sample set of size $x_{samp} = 50$. All algorithms were ran with fixed learning rate $\eta = 5$ for $T = 50$ iterations.

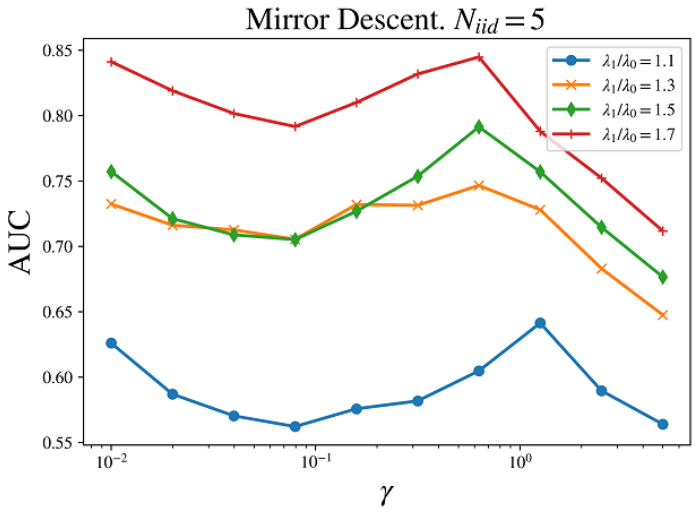

Figure 5.21: Disease Outbreak Anomaly Detection: AUC scores of MD (averaged over 5 iid runs) for disease ratios $\{1.1, 1.3, 1.5, 1.7\}$ and different values of γ.

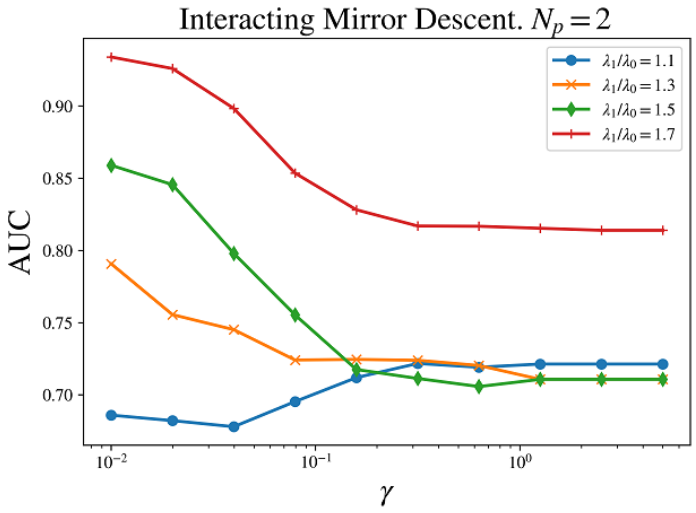

Figure 5.22: Disease Outbreak Anomaly Detection: AUC scores of IMD with 2 interacting particles for disease ratios $\{1.1, 1.3, 1.5, 1.7\}$ and different values of γ.

In Figures 5.21 and 5.22 we illustrate the AUC scores attained by Interacting Mirror Descent with 2 interacting particles and Mirror Descent, respectively, considering for each method four different SN-ratios $\frac{\lambda_1}{\lambda_0} = 1.1, 1.3, 1.5, 1.7$. We mention that for the base disease rate for all cases we set $\lambda_0 = 5 \times 10^{-5}$.

In Table 5.7 we report the best AUC scores for each disease ratio and plot them in Figure 5.23. Observing both, the Figures and the Table we can see that IMD with only two interacting particles outperforms an averaged over 5 runs MD for all values of ratios $\frac{\lambda_1}{\lambda_0}$. We also report the average run-time for the processing of a single particle for each method in Table 5.8. It is clear that IMD not only exhibits better performance than MD, it can also be computationally cheaper for a small number of interacting particles.

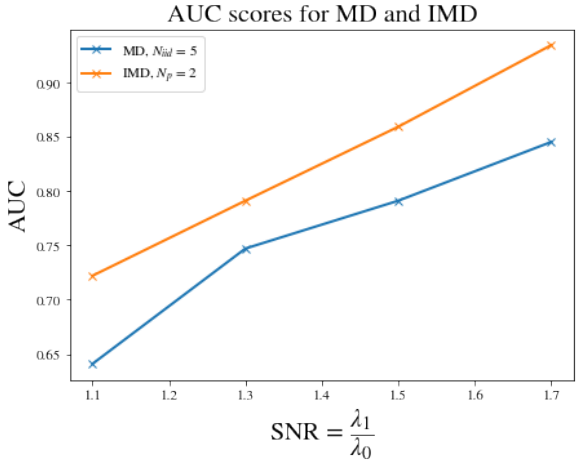

Figure 5.23: Disease Outbreak Anomaly Detection: Best AUC scores of MD and IMD for disease ratios $\{1.1, 1.3, 1.5, 1.7\}$.

AUC	SNR $= \lambda_1/\lambda_0$			
	1.1	**1.3**	**1.5**	**1.7**
MD	0.641	0.747	0.791	0.845
IMD	0.722	0.791	0.859	0.934

Table 5.7: Disease Outbreak Anomaly Detection: Comparison of MD and IMD for the optimal choice of γ as illustrated in Figures 5.21 and 5.22.

	N_{iid}	N_p	**Iters**	**Run-time for processing a single sample**
MD	5	-	50	2.02 secs
IMD	-	2	50	1.92 secs

Table 5.8: Disease Outbreak Anomaly Detection: Average time for the processing of one sample for MD and IMD.

5.5.3 Random Geometric Graphs Anomalous Clusters Classification: Numerical Results

In this section we present our results on the performance of Mirror Descent and Interacting Mirror Descent in the case of geometric graphs. We generate uniformly 3-dimensional geometric graphs with n nodes in the 3D cube $\mathcal{C}^3 = [-1, 1] \times [-1, 1] \times [-1, 1]$ with the methods presented in [43]. We then produce anomalous clusters by selecting groups of vertices that fall in origin-centered spheres or hyper-ellipsoids with specified radii. We denote with (r_x, r_y, r_z) the radius of each axis of the sphere or hyper-ellipsoid and with K the mean number of abnormal vertices in those clusters. In Figure 5.24 is illustrated a random generated graph consisting of $n = 200$ vertices placed uniformly in $\mathcal{C}^3$.

Similarly to the previous section, our aim here is to classify each cluster as abnormal or non-abrnormal with the elevated-mean detection method. For $i = 1, .., n$ we assign randomly to anomalous vertices $x_i \sim Poisson(\lambda_1)$ and normal vertices $x_i \sim Poisson(\lambda_0)$ (we set $\boldsymbol{C} = \boldsymbol{x}\boldsymbol{x}^T$). Below, we investigate the AUC performance of our methods with respect to different values of SNR ratios, increasing number of graph vertices and increasing K and different number of iterations. All graphs presented here, are generated with radii $r_x = r_y = r_z = r(K)$ and for all our computations we use a fixed $\gamma = \sqrt{\frac{1}{500}}$ and run our optimisation for $x_{samp} = 40$ samples. Any other parameter is specified in each problem.

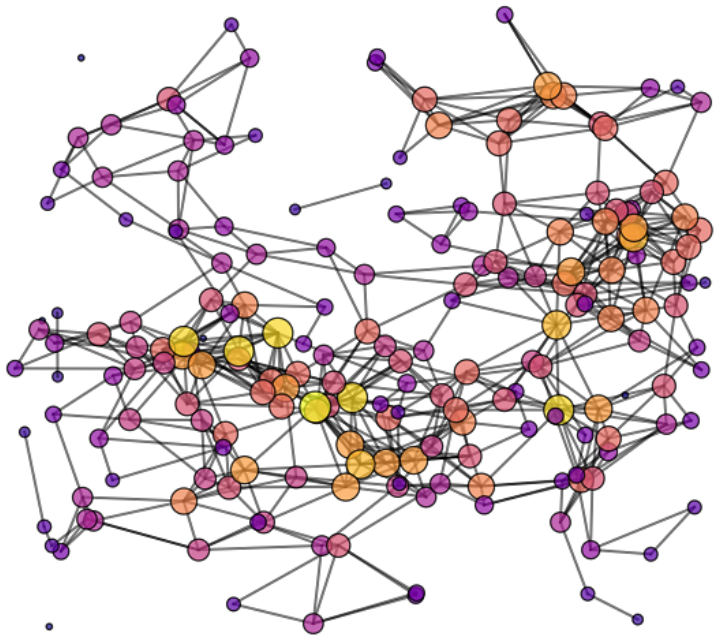

Figure 5.24: Random Geometric Graphs Anomalous Clusters Classification: 3D Random generated Geometric Graph consisting of 200 vertices. Nodes with higher degrees (more neighbours) are illustrated with a larger radius and lighter colours.

Performance of MD and IMD vs SNR values

We begin our experiments for the case of random geographic graphs by investigating the performance of Interacting Mirror Descent and Mirror Descent for various values of SNR. To show the scalability of Interacting Mirror Descent we generate a big-sized random geographical graph of $n = 1000$ vertices with abnormal cluster size $K = 40$. We run for $T = 50$ iterations each framework with base rate $\lambda_0 = 100$ and $\lambda_1 = 110, 130, 150$ and 170. In the case of IMD we use 2 interacting particles, while for MD we average the results over 2 iid copies.

We report the AUC performance results for both algorithms in Table 5.9. As observed in the previous section, we observe that both algorithms perform better for higher SNR values. Also, we can discern that for lower SNRs ($\lambda_1/\lambda_0 = 1.1$), IMD outperforms MD, while for higher SNRs ($\lambda_1/\lambda_0 = 1.3, 1.5, 1.7$) the performance of both methods is similar.

AUC	**SNR** $= \frac{\lambda_1}{\lambda_0}$			
	1.1	**1.3**	**1.5**	**1.7**
IMD	0.930	1.000	1.000	1.000
MD	0.831	0.999	1.000	1.000

Table 5.9: Random Geometric Graphs Anomalous Clusters Classification: Performance of IMD and MD for different SNR values.

Performance of MD and IMD vs Anomaly size K

In this paragraph we investigate the performance of Interacting Mirror Descent and Mirror Descent for various sizes of abnormal clusters K. We experiment with values of K between 5 and 40 and illustrate the results in Figure 5.25. We set $\lambda_0 = 100$ and $\lambda_1 = 110$ and we generate a random geographic graph consisting of $n = 1000$ nodes and run each optimiser for $T = 50$ iterations with fixed learning rate $\eta = 5$. In the case of IMD we use 2 interacting particles, while for MD we average the results over 2 iid copies.

We observe that the performance of both algorithms improves as the size of anomaly K increases. This result is consistent with the results presented in Section 5.2 of [1]. However, here we show that for all anomaly sizes IMD outperforms MD.

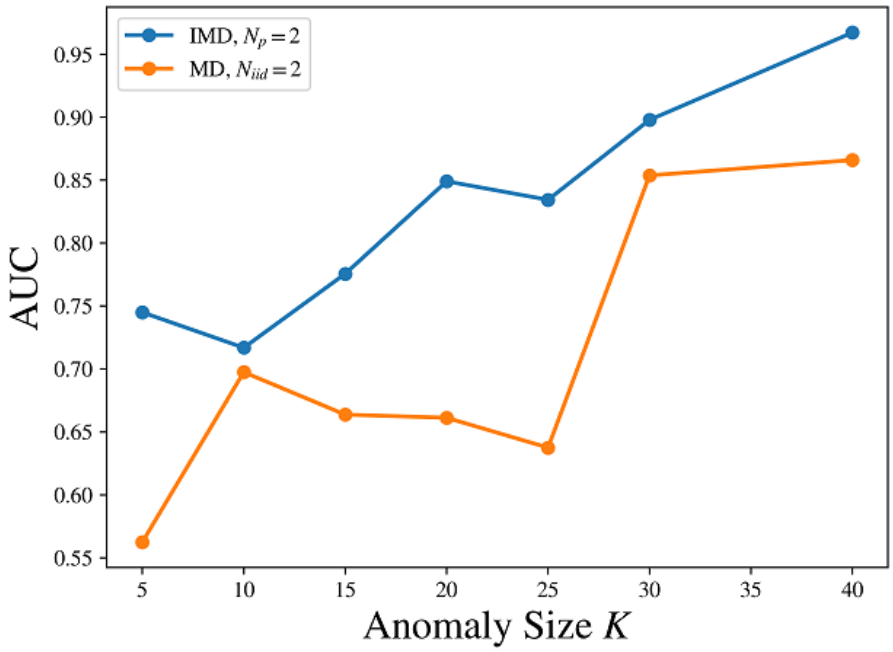

Figure 5.25: Random Geometric Graphs Anomalous Clusters Classification: AUC performance of MD and IMD for different anomaly sizes K.

Performance of MD and IMD vs graph size n

We further examine the time complexity of Mirror Descent and Interacting Mirror Descent algorithms with respect to the size n of the generated graph G. We use $K = 40$ and $T = 50$. We plot the average run-time per iteration for various graph sizes in semi-log axes in Figure 5.26. Note that in the case of the latter we use two interacting particles and we average the times with respect to the particles as well.

From the Figure we can observe that the average run-time per iteration for both methods grows exponentially with respect to the size of the graph. Additionally, we observe that for small-sized graphs, both methods perform similarly. However, for larger graph sizes, IMD takes more time to complete a single iteration than MD. Nevertheless, that was expected due to the fact that interaction requires the element-wise multiplication and exponentiation to the A_{ij}-power as per the update rule shown in Equation 4.30.

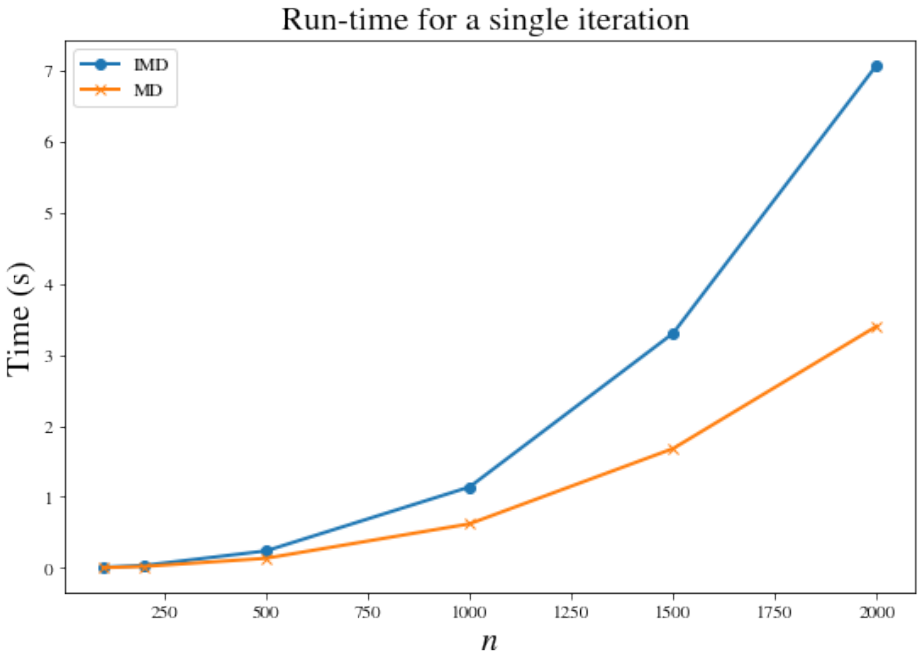

Figure 5.26: Random Geometric Graphs Anomalous Clusters Classification: Average run-time (in seconds) per iteration for the processing of one sample for various graph sizes n.

5.6 Langevin Diffusions

The last problem we are concerned with in this project is a popular optimisation problem arising in statistical and machine learning, called Markov Chain Monte Carlo (MCMC) which is used for sampling. Specifically, the sampling problem is as follows:

Given $V : \mathbb{R}^d \longrightarrow \mathbb{R}$ a convex, twice differentiable potential function, we want to sample from a negative log-convex target probability distribution π defined on $\mathbb{R}^d$ such that

$$d\pi = e^{-V(\boldsymbol{x})} d\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^d.$$

A widely used algorithm for MCMC sampling is Langevin Diffusions. LD is the solution $\{\boldsymbol{x}_t\}_{t=0}^T$ of the following SDE:

$$d\boldsymbol{x}_t = \boldsymbol{\nabla} V(\boldsymbol{x}_t)\, dt + \sqrt{2} d\boldsymbol{B}_t,$$

where $\{\boldsymbol{B}_t\}_{t=0}^T$ are standard Brownian motions in $\mathbb{R}^d$. For more information on Langevin Diffusions we refer the reader to [54].

5.6.1 Mirror Langevin Diffusion

Now we describe a variant of LD, called Mirror Langevin Diffusion (MLD), which was proposed initially in [59] and utilises Mirror Descent as the optimisation framework. The problem is as follows:

Let $\Psi : \mathbb{R}^d \longrightarrow \mathbb{R}$ be a mirror map as defined earlier in Chapter 3. The MLD then, is the SDE:

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{\nabla} \Psi^*(\boldsymbol{y}_t) \\ d\boldsymbol{y}_t &= -\boldsymbol{\nabla} V(\boldsymbol{x}_t)\, dt + \sqrt{2} \cdot \boldsymbol{\nabla}^2 \Psi(\boldsymbol{x}_t)^{\frac{1}{2}}\, d\boldsymbol{B}_t, \end{aligned} \tag{5.12}$$

where Ψ^* denotes the convex conjugate of Ψ and $\boldsymbol{\nabla}^2 \Psi(\boldsymbol{x}_t)^{\frac{1}{2}}$ is the solution $\boldsymbol{A}_t$ of

$$\boldsymbol{A}_t^2 = \boldsymbol{\nabla}^2 \Psi(\boldsymbol{x}_t).$$

In [17] the authors suggest picking as a mirror map $\Psi = V$, a potential function that meets all assumptions of a mirror map and propose a sampling method called Newton Langevin Diffusion. NLD has the form:

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{\nabla} V^*(\boldsymbol{y}_t) \\ d\boldsymbol{y}_t &= -\boldsymbol{\nabla} V(\boldsymbol{x}_t)\, dt + \sqrt{2} \cdot \boldsymbol{\nabla}^2 V(\boldsymbol{x}_t)^{\frac{1}{2}}\, d\boldsymbol{B}_t. \end{aligned} \tag{5.13}$$

We get the Newton Langevin Algorithm (NLA) by utilising the Euler discretization of Equation 5.13:

$$\begin{aligned} \boldsymbol{\nabla} V(\boldsymbol{x}_{t+1}) &= (1 - \epsilon) \boldsymbol{\nabla} V(\boldsymbol{x}_t) + \sqrt{2\epsilon} \boldsymbol{\nabla}^2 V(\boldsymbol{x}_t)^{\frac{1}{2}} \boldsymbol{\zeta}_t \\ \boldsymbol{x}_{t+1} &= \boldsymbol{\nabla} V^* \circ \boldsymbol{\nabla} V(\boldsymbol{x}_{t+1}), \quad t \geq 0, \end{aligned} \tag{5.14}$$

where ϵ denotes the descritisation parameter (here used as a learning rate) and $\boldsymbol{\zeta}_t \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

5.6.2 Interacting Mirror Langevin Diffusion

As an alternative to 5.12 we now propose an interacting particles SDE similar to the previous frameworks of this work, inspired by [9, 10], called **Interacting Mirror Langevin Diffusion** (IMLD) outlined by the SDE:

$$\begin{aligned} \boldsymbol{x}_t^i &= \boldsymbol{\nabla}\Psi^*(\boldsymbol{y}_t^i) \\ d\boldsymbol{y}_t^i &= -\boldsymbol{\nabla}V(\boldsymbol{x}_t^i)\,dt + \sum_{j=1}^{N_p} A_{ij}\big(\boldsymbol{y}_t^j - \boldsymbol{y}_t^i\big) + \sqrt{2}\cdot\boldsymbol{\nabla}^2\Psi(\boldsymbol{x}_t^i)^{\frac{1}{2}}\,d\boldsymbol{B}_t^i \\ i &= 1, 2, ..., N_p, \end{aligned} \tag{5.15}$$

where N_p denotes the number of interacting particles and $\boldsymbol{A}$ is a doubly stochastic matrix in $\mathbb{R}^{N_p\times N_p}$.

Similarly to [17], we use as a mirror map the potential function V and propose the Interacting Newton Langevin Diffusion (INLD), i.e.:

$$\begin{aligned} \boldsymbol{x}_t^i &= \boldsymbol{\nabla}V^*(\boldsymbol{y}_t^i) \\ d\boldsymbol{y}_t^i &= -\boldsymbol{\nabla}V(\boldsymbol{x}_t^i)\,dt + \sum_{j=1}^{N_p} A_{ij}\big(\boldsymbol{y}_t^j - \boldsymbol{y}_t^i\big) + \sqrt{2}\cdot\boldsymbol{\nabla}^2V(\boldsymbol{x}_t^i)^{\frac{1}{2}}\,d\boldsymbol{B}_t^i \\ i &= 1, 2, ..., N_p. \end{aligned} \tag{5.16}$$

The Euler ization of Equation 5.16 is gives our proposal algorithm, the Interacting Newton Langevin Algorithm (INLA):

$$\begin{aligned} \boldsymbol{\nabla}V(\boldsymbol{x}_{t+1}^i) &= (1-2\,\epsilon)\boldsymbol{\nabla}V(\boldsymbol{x}_t^j) + \epsilon\sum_{j=1}^{N_p} A_{ij}\boldsymbol{x}_t^j + \sqrt{2\epsilon}\boldsymbol{\nabla}^2V(\boldsymbol{x}_t^i)^{\frac{1}{2}}\boldsymbol{\zeta}_t^i \\ \boldsymbol{x}_{t+1}^i &= \boldsymbol{\nabla}V^*\circ\boldsymbol{\nabla}V(\boldsymbol{x}_{t+1}^i), \quad i = 1, ..., N_p, \quad t \geq 0, \end{aligned} \tag{5.17}$$

where again, ϵ denotes the discretization parameter and $\boldsymbol{\zeta}_t^i \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

Our results from the previous sections, suggest that interaction improves the performance of the optimisation problem. Hence, we expect that our numerical experiments in this section will show that our method INLD outperforms NLD. Note that in contrast with the previous problems, the optimisation problem here is unconstrained as we let $\boldsymbol{x}$ to lie in the whole space, i.e. $\boldsymbol{x} \in \mathbb{R}^d$.

5.6.3 Numerical Results

In this paragraph we provide results for our proposed method Interacting Newton Langevin Diffusion and we compare it with Newton Langevin Diffusion from [17]. Similarly to Section E.1 of [17], we use as a potential function

$$V(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}, \quad \boldsymbol{x} \in \mathbb{R}^d,$$

which is utilised to sample from a Gaussian Distribution. The matrix $\boldsymbol{\Sigma}$ denotes the scatter matrix of the desired distribution and it holds that $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$ and $\boldsymbol{\Sigma} \succ \mathbf{0}$.

Note that V is twice differentiable with gradient and Hessian:

$$\boldsymbol{\nabla} V(\boldsymbol{x}) = \boldsymbol{\Sigma}^{-1}\boldsymbol{x}, \quad \boldsymbol{\nabla}^2 V(\boldsymbol{x}) = \boldsymbol{\Sigma}^{-1}.$$

Since $\boldsymbol{\Sigma} \succ \mathbf{0}$, it holds that $\boldsymbol{\Sigma}^{-1} \succ \mathbf{0}$, and therefore V is a convex function defined on a convex domain ($\mathbb{R}^d$). It's also easy to proof that

$$V^*(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{z}, \quad \boldsymbol{\nabla} V^*(\boldsymbol{z}) = \boldsymbol{\Sigma}^{-1}\boldsymbol{z}.$$

We quantify the performance of each algorithm similarly to [17], by computing the $\log_{10}$ of running estimates at each iteration $t = 1, .., T$ of the mean $\bar{\boldsymbol{x}}_t = \frac{1}{t+1}\sum_{s=0}^{t}\boldsymbol{x}_s$ and by computing the $\log_{10}$ of the relative squared error of the covariance (scatter) matrix $\frac{||\,\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}\,||_F^2}{||\,\boldsymbol{\Sigma}\,||_F^2}$, where $\hat{\boldsymbol{\Sigma}} = \frac{1}{t+1}\sum_{s=0}^{t} cov(\bar{\boldsymbol{x}}_s)$.

We plot the results for each metric in Figures 5.27 and 5.28, respectively, in the case of $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d \end{bmatrix}$ and $d = 100$.

In the case of NLA we average the results over 10 independent and identical runs and in the case of INLA we use 10 interacting particles. For both algorithms we set a constant discretization parameter $\epsilon = 0.7$.

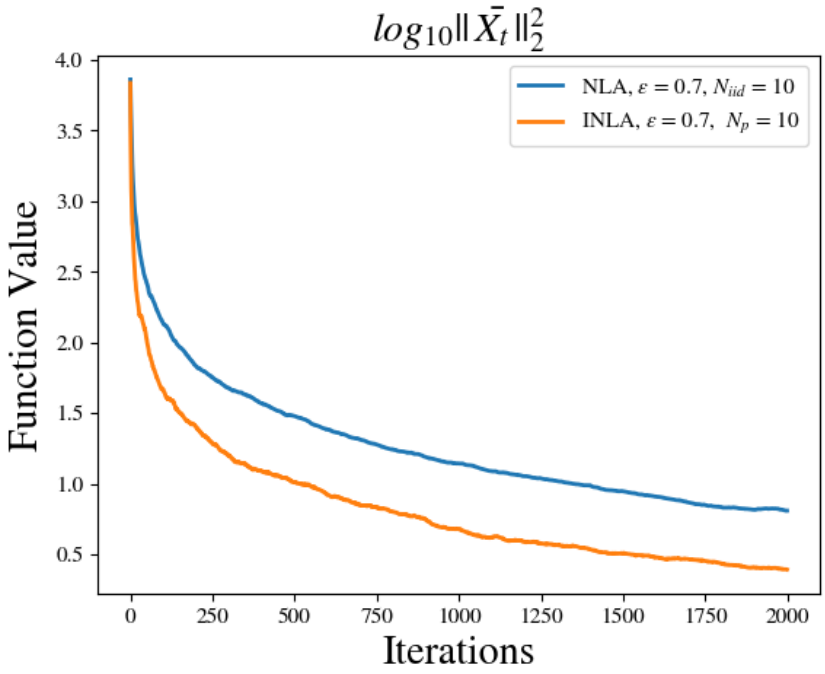

Figure 5.27: Mirror Langevin Diffusion: Convergence of the squared distance of the running mean $\bar{\boldsymbol{x}}_t$ from $\mathbf{0}$ for $d = 100$ in the case of NLA averaged over $N_{iid} = 10$ runs and INLA with $N_p = 10$.

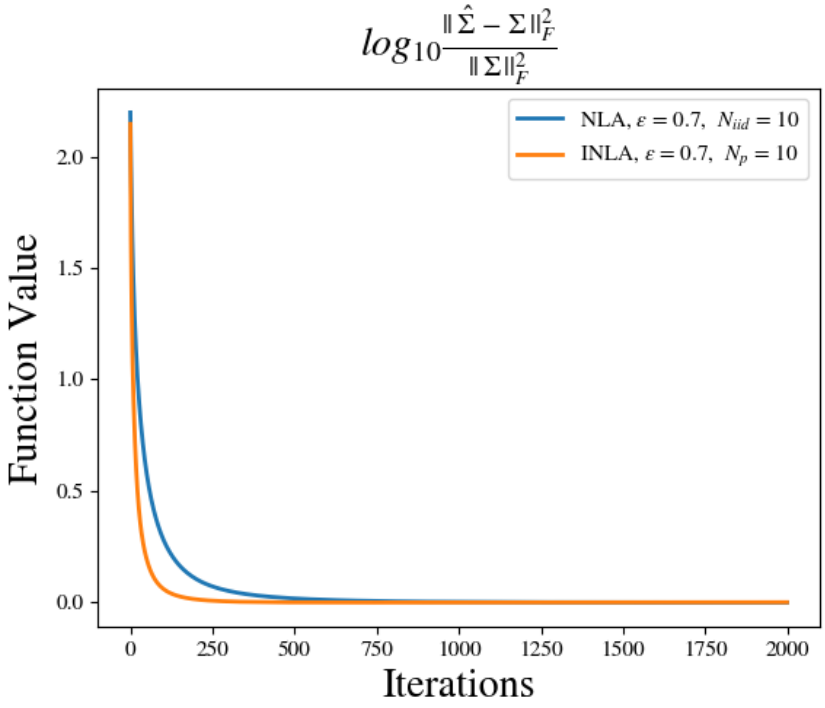

Figure 5.28: Mirror Langevin Diffusion: Convergence of the relative squared distance of the running estimate of $\hat{\boldsymbol{\Sigma}}$ from $\boldsymbol{\Sigma}$ for $d = 100$ in the case of NLA averaged over $N_{iid} = 10$ runs and INLA with $N_p = 10$.

Observing Figures 5.27 and 5.28 we can infer that INLA achieves a much greater performance than NLA, as the former manages to generate samples from the desired distribution significantly faster. Also, the speed of convergence of our proposed algorithm is greater than the speed of NLA.

Additionally, we plot in Figure 5.29 the histogram of the relative squared error and we present in Table 5.10 the variance for the two methods after 200 iterations. We then, compare the two methods. As expected from our previous numerical results, interaction aids in variance reduction. INLA attains a much lower variance than NLA as observed from Figure 5.29 and Table 5.10.

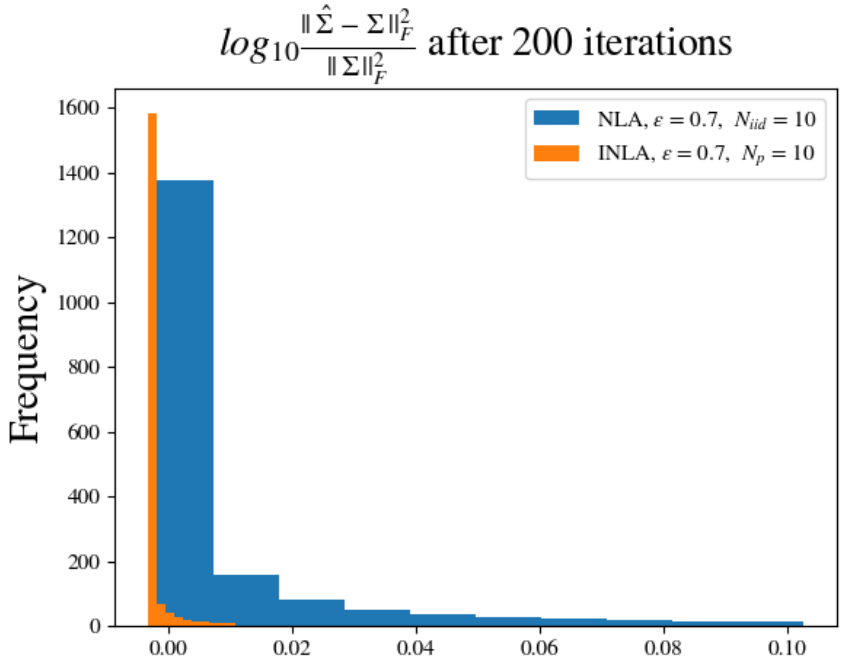

Figure 5.29: Mirror Langevin Diffusion: Histogram of the relative squared distance of the running estimate of $\hat{\mathbf{\Sigma}}$ from $\mathbf{\Sigma}$ for $d = 100$ in the case of NLA averaged over $N_{iid} = 10$ runs and INLA with $N_p = 10$ after convergence ($T_0 = 200$).

Variance after convergence	
NLA	0.000359875034
INLA	4.10370467e-06

Table 5.10: Mirror Langevin Diffusion: Variance of samples of $log_{10}\frac{||\hat{\mathbf{\Sigma}}-\mathbf{\Sigma}||_F^2}{||\mathbf{\Sigma}||_F^2}$ after convergence ($T_0 = 200$) for NLA and INLA.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this individual project, we have reviewed methods for optimising a convex function as an objective. We have applied Gradient Descent, Mirror Descent and Interacting Mirror Descent on six problems and we have compared their performance. Utilising the latter method, we have solved numerically for the first time two problems from the existing related literature (see subsections 5.4 and 5.5).

Additionally, in Section 5.6 we have proposed a novel Langevin diffusion-based method for sampling from a target distribution called Interacting Newton Langevin Algorithm. Our proposed method utilises interaction between particles as a variant of Newton Langevin Algorithm proposed in the related literature. Our numerical results suggest that our algorithm outperforms the original method as it establishes better convergence performance.

Moreover, throughout Chapter 5, we have analysed the benefits of using Mirror Descent and its variant Interacting Mirror Descent, over the Projected Gradient Descent. Our numerical experiments indicate that the first two surpass the latter algorithm.

We have also shown that interaction in IMD can speed up convergence of the optimiser to the optimum solution. Furthermore, we have seen that interaction can result in a reduction of noise even in cases such as Mini-batch optimisation. We have shown that using more interacting particles can decrease the variance of the samples of the loss function significantly, performing almost as good as full-batch optimisation. We also remark that we have ran experiments with fixed and decreasing learning rate. In both cases, MD and IMD exhibit better convergence properties than GD.

6.2 Future Work

We conclude this work by discussing several interesting directions for further future implementation and research. The results shown in this project spark the idea for the implementation of some possible future directions:

- For our experiments performed in Chapter 5, in the case of Interacting Mirror Descent we used as an interaction matrix a mean-field interaction, i.e. $A_{ij} = \frac{1}{N_p}$ where N_p denotes the number of interacting particles. We therefore allowed all particles to interact in the same way. An extension would be to study the convergence properties of the IMD method with respect to $\boldsymbol{A}$ by experimenting different interaction matrices by allowing some particles to interact more with some particles and less with others.

- A future direction could be the implementation of the Mirror Descent algorithm with interacting particles on a famous problem called Langevin Dynamics for Latent Dirichlet Allocation (LDA). This a similar setup to the last problem of this project (Langevin Diffusions) and it is usually applied for topic modelling problems. Such examples are [28, 41] where the authors utilise Stochastic projected Gradient Descent and Stochastic Mirror Descent on the probability simplex.

- In Section 5.6 of Chapter 5 we proposed a new algorithm called Interacting Mirror Langevin Diffusion. For our numerical results we used as a potential function $V(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$. Another future direction would be to apply our method for different choices of V that aim in sampling from ill-conditioned Gaussian distributions, such as $V(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x})^{\alpha}$, $\alpha \in (0, 1)$, similarly to [17] where they use $\alpha = 3/4$.

- Most of problems in ML and DL are non-convex and quite often optimisers get stack in local optima. Stochasticity can usually help escaping such situations. A possible direction would be the theoretical and experimental investigation of (Stochastic) Mirror Descent and (Stochastic) Interacting Mirror Descent for the optimisation of non-convex constrained problems.

Bibliography

[1] Cem Aksoylar, Lorenzo Orecchia, and Venkatesh Saligrama. Connected subgraph detection with mirror descent on SDPs. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 51–59, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. pages 6, 9, 56, 57, 61

[2] Jürgen Audretsch. *The von Neumann Entropy and Quantum Information*, chapter 6, pages 103–113. John Wiley & Sons, Ltd, 2008. pages 19

[3] S. Bahmani and B. Raj. A unifying analysis of projected gradient descent forp-constrained least squares. *Applied and Computational Harmonic Analysis*, 34(3):366–378, May 2013. pages 5

[4] Amir Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. pages 20, 39

[5] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 05 2003. pages 3, 6

[6] M. Bierlaire, Ph.L. Toint, and D. Tuyttens. On iterative algorithms for linear least squares problems with bound constraints, 1995. pages 39

[7] Jaya Bishwal. Estimation in interacting diffusions: Continuous and discrete sampling. *Applied Mathematics*, 02:1154–1158, 01 2011. pages 6

[8] Eliot Bolduc, George Knee, Erik Gauger, and Jonathan Leach. Projected gradient descent algorithms for quantum state tomography, 2016. pages 5

[9] Anastasia Borovykh, Nikolas Kantas, Panos Parpas, and Grigorios Pavliotis. To interact or not? the convergence properties of interacting stochastic mirror descent, 07 2020. pages 4, 65

[10] Anastasia Borovykh, Nikolas Kantas, Panos Parpas, and Grigorios A. Pavliotis. On stochastic mirror descent with interacting particles: convergence properties and variance reduction, 2020. pages 4, 7, 8, 20, 23, 26, 39, 42, 65

[11] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, August 2010. Springer. pages 44

[12] Milan Bradonjić, Aric Hagberg, and Allon G. Percus. Giant component and connectivity in geographical threshold graphs. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-Graph*, WAW'07, page 209–216, Berlin, Heidelberg, 2007. Springer-Verlag. pages 35

[13] Sébastien Bubeck. Convex optimization: Algorithms and complexity, 2014. pages 3, 19, 22, 23, 25

[14] Rene Carmona, Jean-Pierre Fouque, and Douglas Vestal. Interacting particle systems for the computation of rare credit portfolio losses. *Finance and Stochastics*, 13:613–633, 09 2009. pages 6

[15] Xi Chen, Qihang Lin, and Javier Pena. Optimal regularized dual averaging methods for stochastic optimization. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 395–403. Curran Associates, Inc., 2012. pages 6

[16] Xiaohui Chen. Maximum likelihood estimation of potential energy in interacting particle systems from single-trajectory data, 2020. pages 6

[17] Sinho Chewi, Thibaut Le Gouic, Chen Lu, Tyler Maunu, Philippe Rigollet, and Austin J. Stromme. Exponential ergodicity of mirror-langevin diffusions, 2020. pages 9, 64, 65, 66, 70

[18] John C. Duchi, Alekh Agarwal, Mikael Johansson, and Michael I. Jordan. Ergodic mirror descent, 2011. pages 6

[19] Dan Garber. On the convergence of stochastic gradient descent with low-rank projections for convex low-rank matrix problems, 2020. pages 5

[20] Josselin Garnier, George Papanicolaou, and Tzu-Wei Yang. Large deviations for a mean field model of systemic risk, 2012. pages 6

[21] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition. In Peter Grünwald, Elad Hazan, and Satyen Kale, editors, *Proceedings of The 28th Conference on Learning Theory*, volume 40 of *Proceedings of Machine Learning Research*, pages 797–842, Paris, France, 03–06 Jul 2015. PMLR. pages 44

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. pages 5

[23] Harshit Gupta, Kyong Hwan Jin, Ha Q. Nguyen, Michael T. McCann, and Michael Unser. Cnn-based projected gradient descent for consistent ct image reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1440–1453, Jun 2018. pages 5

[24] Nicholas Harvey. Machine learning theory: Mirror descent, 2018. pages 22

[25] Adrian Hauswirth, Saverio Bolognani, Gabriela Hug, and Florian Dörfler. Projected gradient descent on riemannian manifolds with applications to online power system optimization, 09 2016. pages 5

[26] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. Fundamentals of convex analysis / j.b. hiriart-urruty, c. lemaréchal., 01 2001. pages 13

[27] Tito Homem-de Mello. On rates of convergence for stochastic optimization problems under non–independent and identically distributed sampling. *SIAM Journal on Optimization*, 19(2):524–551, 2008. pages 26

[28] Ya-Ping Hsieh, Ali Kavis, Paul Rolland, and Volkan Cevher. Mirrored langevin dynamics, 2018. pages 6, 70

[29] Gautam Kunapuli and Jude Shavlik. Mirror descent for metric learning: A unified approach. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 859–874, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. pages 6

[30] Guanghui Lan, Arkadi Nemirovski, and Alexander Shapiro. Validation analysis of mirror descent stochastic approximation method. *Mathematical Programming*, 134:1–34, 09 2011. pages 6

[31] Måns Larsson, Anurag Arnab, Fredrik Kahl, Shuai Zheng, and Philip Torr. A projected gradient descent method for crf inference allowing end-to-end training of arbitrary pairwise potentials, 2017. pages 5

[32] Jie Liu and Martin Takac. Projected semi-stochastic gradient descent method with mini-batch scheme under weak strong convexity assumption, 2016. pages 5

[33] S. Liu, X. Li, P. Chen, J. Haupt, and L. Amini. Zeroth-order stochastic projected gradient descent for nonconvex optimization. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1179–1183, 2018. pages 5

[34] Mehrdad Mahdavi, Tianbao Yang, Rong Jin, Shenghuo Zhu, and Jinfeng Yi. Stochastic gradient descent with only one projection. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 494–502. Curran Associates, Inc., 2012. pages 5

[35] Naoki Masuda, Hiroyoshi Miwa, and Norio Konno. Geographical threshold graphs with small-world and scale-free properties. *Phys. Rev. E*, 71:036108, Mar 2005. pages 35

[36] P. Mertikopoulos, E. V. Belmega, R. Negrel, and L. Sanguinetti. Distributed stochastic optimization via matrix exponential learning. *IEEE Transactions on Signal Processing*, 65(9):2277–2290, 2017. pages 6

[37] Panayotis Mertikopoulos and Mathias Staudigl. On the convergence of gradient-like flows with noisy gradient input. *SIAM Journal on Optimization*, 28:163–197, 01 2018. pages 6

[38] Angelia Nedich and Soomin Lee. On stochastic subgradient mirror-descent algorithm with weighted averaging, 2013. pages 6

[39] Arkadi Nemirovski. Prox-method with rate of convergence o (1/ t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15:229–251, 01 2004. pages 6

[40] A.S. Nemirovskii and D.B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983. pages 3, 6

[41] Sam Patterson and Yee Whye Teh. Stochastic gradient riemannian langevin dynamics on the probability simplex. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3102–3110. Curran Associates, Inc., 2013. pages 70

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. pages 50

[43] D.M.S.M. Penrose, M. Penrose, and Oxford University Press. *Random Geometric Graphs*. Oxford studies in probability. Oxford University Press, 2003. pages 60

[44] M. Raginsky and J. Bouvrie. Continuous-time stochastic mirror descent on a network: Variance reduction, consensus, convergence. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 6793–6800, 2012. pages 22

[45] Clement Royer. Optimization for machine learning: Basics of constrained optimization. Dauphine University Paris -M2 IASD, dec 2019. pages 3, 5

[46] Shai Shalev-Shwartz. *Online Learning and Online Convex Optimization*. Now Publishers Inc, 01 2011. pages 6

[47] R. J. Smeed. Studies in the Economics of Transportation. *The Economic Journal*, 67(265):116–118, 03 1957. pages 33

[48] Tom Tirer and Raja Giryes. On the convergence rate of projected gradient descent for a back-projection based objective, 2020. pages 5

[49] Yann Traonmilin, Jean-François Aujol, and Arthur Leclaire. Projected gradient descent for non-convex sparse spike estimation, 2020. pages 5

[50] Weiran Wang and Miguel Á. Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application, 2013. pages 5, 8

[51] Weiran Wang and Miguel Á. Carreira-Perpiñán. The laplacian k-modes algorithm for clustering, 2014. pages 5, 8, 49, 50

[52] Fanyou Wu, Rado Gazo, Eva Haviarova, and Bedrich Benes. Efficient project gradient descent for ensemble adversarial attack, 2019. pages 5

[53] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(88):2543–2596, 2010. pages 6

[54] T. Xifara, C. Sherlock, S. Livingstone, S. Byrne, and M. Girolami. Langevin diffusions and the metropolis-adjusted langevin algorithm. *Statistics & Probability Letters*, 91:14–19, Aug 2014. pages 64

[55] Pan Xu, Tianhao Wang, and Quanquan Gu. Accelerated stochastic mirror descent: From continuous-time dynamics to discrete-time algorithms. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1087–1096, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR. pages 6

[56] Sheng Xu, Zhou Fan, and Sahand Negahban. Tree-projected gradient descent for estimating gradient-sparse parameters on graphs, 2020. pages 5

[57] Yao-Liang Yu. The strong convexity of von neumann's entropy, 2013. pages 17, 18, 19

[58] Jiawei Zhang. Gradient descent based optimization algorithms for deep learning models training, 2019. pages 5

[59] Kelvin Shuangjian Zhang, Gabriel Peyré, Jalal Fadili, and Marcelo Pereyra. Wasserstein control of mirror langevin monte carlo, 2020. pages 64

[60] B. Zhou and F. Chen. Graph-structured sparse optimization for connected subgraph detection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 709–718, 2016. pages 56

[61] Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Stephen Boyd, and Peter Glynn. On the convergence of mirror descent beyond stochastic convex programming, 2017. pages 6

[62] Martin Zinkevich, Markus Weimer, Alexander Smola, and Lihong Li. Parallelized stochastic gradient descent. *Advances in Neural Information Processing Systems*, 23:2595–2603, 01 2010. pages 26

Appendices

Appendix A

Code

The interested reader can find the Gitlab repository of this individual project in the link below:

https://gitlab.doc.ic.ac.uk/gy615/mirror-descent-optimisation .

In our Gitlab repository we provide all the code written for the implementation of all six optimisation problems solved in this project along with the results presented throughout Chapter 5. All code was written in Python3. A list of package-dependencies is also provided needed to run the code on a local machine.

The repository contains six project folders, one for each optimisation problem we solve in Chapter 5. The main functions that solve each optimisation problem with the methods afformentioned in the project are listed below:

- The Traffic Assignment Problem: `\run_opt.py`
- Linear System: `\run_opt.py`
- Linear System: Mini-batch Optimisation: `\run_opt.py`
- The Laplacian K-Modes for Clustering: `\run_opt.py`
- Optimisation on SDPs: `\startOpt\startOpt.py`
- Langevin Diffusions: `\run_opt.py`

The repository also includes all numerical results for the problems presented in Chapter 5 in the `\saved_items` subfolder of each problem, as well as all illustrations `\figures` subfolders.

Appendix B

Ethical Considerations

We declare that the work performed here poses no ethical, social or legal considerations, since we are concerned with the algorithmic performance and numerical results on mathematical problems. Additionally, all data used for this project were randomly generated and/or did not include any kind of personal data.

However, optimisation is a key component for every problem in artificial intelligence and machine learning and finding the best optimiser is always an open problem in every research field. Thus, the case of such algorithm being used and applied in scenarios that would cause harm to any extent to the environment or to any natural person, etc or for dual or medicinal use would not be improbable. Nevertheless, such cases are far from probable to happen and therefore we assume that they are no considerations of such.

In B.1 we review some ethical, social or legal considerations. I declare that everything stated on this table is true and correct and was completed solemnly with my own free will.

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		✓
Does your project involve the use of human embryos?		✓
Does your project involve the use of human foetal tissues / cells?		✓
Section 2: HUMANS		
Does your project involve human participants?		✓
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		✓
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		✓
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		✓
Does it involve processing of genetic information?		✓

Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		✓
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?		✓
Section 5: ANIMALS		
Does your project involve animals?		✓
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		✓
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		✓
Could the situation in the country put the individuals taking part in the project at risk?		✓
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		✓
Does your project deal with endangered fauna and/or flora /protected areas?		✓
Does your project involve the use of elements that may cause harm to humans, including project staff?		✓
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		✓
Section 8: DUAL USE		
Does your project have the potential for military applications?		✓
Does your project have an exclusive civilian application focus?		✓
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		✓
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		✓
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist abuse?		✓
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		✓
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?		✓

Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		✓
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		✓
Will your project use or produce goods or information for which there are data protection, or other legal implications?		✓
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into consideration?		✓

Table B.1: Imperial College London Ethics Checklist