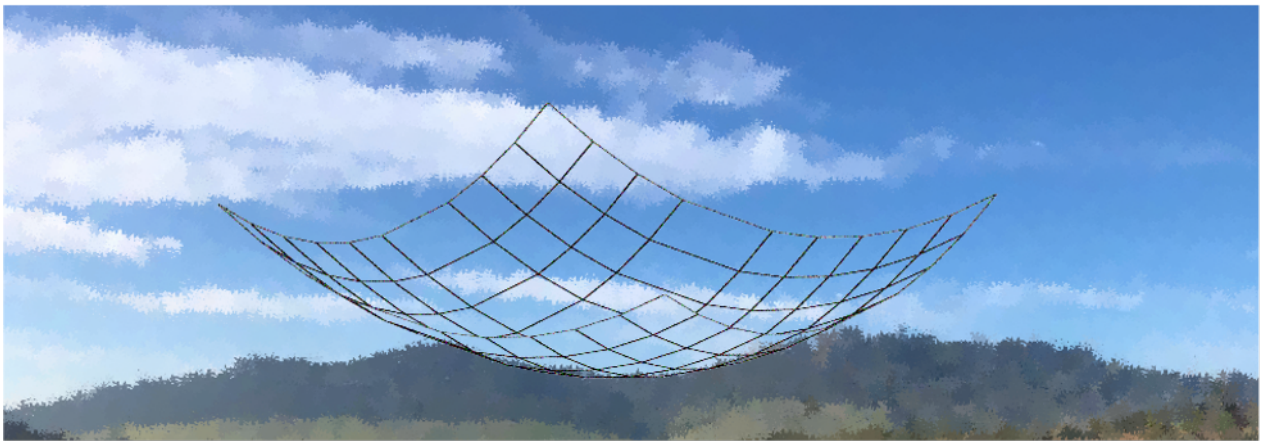
TÉCNICO LISBOA



COCO Denoiser

Using Co-Coercivity for Variance Reduction in Stochastic Convex Optimization

Manuel Monteiro Lança Madeira

Thesis to obtain the Master of Science Degree in

Biomedical Engineering

Supervisors: Prof. Pedro Manuel Quintas Aguiar
Prof. João Manuel de Freitas Xavier

Examination Committee

Chairperson: Prof. João Miguel Raposo Sanches
Supervisor: Prof. Pedro Manuel Quintas Aguiar
Member of the Committee: Prof. Mário Alexandre Teles de Figueiredo

January 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Preface

The work presented in this thesis was performed at the Institute for Systems and Robotics of Instituto Superior Técnico (Lisbon, Portugal), during the period February-December 2020, under the supervision of Prof. Pedro Aguiar (IST), Prof. João Xavier (IST) and Prof. Renato Negrinho (CMU).

Acknowledgments

This thesis marks the end of a 5-year journey at Instituto Superior Técnico; a journey ruled by a constant challenge but also, and more importantly, by huge learnings. When I entered IST, I could not ever have dreamed of the opportunities that I was provided with. I am deeply grateful and proud to have been a student at this institution.

I would like to thank my supervisors: to Prof. Pedro Aguiar for bridging all the elements together, for the continuous availability and effort, and all the wise advices; to Prof. João Xavier for the kind technical guidance to this complete layman in the field, and to Renato for his everlasting patience, invaluable insights and, essentially, friendship throughout the whole process. Without them, this thesis would never be possible.

Still regarding the work developed in this thesis, I would also like to thank José Monteiro, for his help with the design of some of the figures presented, and to Prof. Robert Gower, for kindly providing the code used as a starting point for the logistic regression problem considered.

To my friends, both from my hometown and IST, a huge thank you. Allow me an honorable mention to André Vieira and Tiago Rodrigues (KRC) and to the AEIST football team for being the fresh air in the darkest times and the light to follow in the happy ones.

Finally, I want to thank my family for providing all the conditions and support I could ever ask for. In particular, to my sister for the everyday company and happiness, and to my parents, who are the responsible for all this. They instilled me through example the values of hard work and kindness that I try to take with me at all times. This is for you.

Abstract

First-order methods for stochastic optimization gained undeniable relevance, in particular due to their pivotal role in machine learning. These methods blindly accept noisy gradients provided by an oracle, which may be inconsistent with structural properties of the underlying objective function. We exploit gradient co-coercivity of L-smooth convex objective functions to obtain more accurate gradient estimates. The method introduced in this thesis is then coined as the co-coercivity (COCO) denoiser.

Our denoiser is a joint Maximum Likelihood (ML) estimator for the gradients, constrained by pairwise co-coercivity conditions. Although ML leads to a Quadratically Constrained Quadratic Problem, we introduce an efficient first-order algorithm for COCO, which is based on the Fast Dual Proximal Gradient method. For the denoiser that deals with a single pair of gradients, we derive the closed-form solution for the ML problem, which is relevant in practice, since our experiments have shown that even this simple scenario leads to variance reduction gains in stochastic optimization.

We carry out a theoretical analysis that provides insight into parameter tuning and estimator results, showing in particular why COCO necessarily improves with respect to the noisy oracle. The experimental analysis corroborates these results and shows that the COCO estimator, although not unbiased, leads to a reduction of the mean squared error of the function gradients. To illustrate the impact in stochastic optimization, we use both synthetic data and a real online learning task. Our experiments show that COCO leads to improvements in variance reduction with respect to baseline algorithms.

Keywords

Stochastic Optimization; First-Order Algorithms; Convex Optimization; Co-coercivity; COCO Denoiser; Variance Reduction; Quadratically Constrained Quadratic Problem; Machine Learning.

Resumo

Os algoritmos de primeira-ordem para optimização estocástica têm ganho inegável relevância, em particular pelo papel preponderante que desempenham em Aprendizagem Automática. Estes algoritmos aceitam cegamente gradientes ruidosos facultados por um oráculo, que podem ser inconsistentes com propriedades estruturais da função de custo em questão. Nós exploramos a co-coercividade do gradiente de funções convexas L-suaves para obter estimativas mais precisas desses gradientes. O método apresentado nesta tese é então designado por filtro de Co-coercividade (COCO).

O nosso filtro é um estimador conjunto de máxima verosimilhança (MV) para os gradientes, restringidos dois a dois por condições de co-coercividade. Embora a MV nos leve a um Problema Quadrático Quadraticamente Restringido, propomos um algoritmo de primeira-ordem eficiente para o COCO, baseado no método *Fast Dual Proximal Gradient*. Para o Filtro que apenas considera um único par de gradientes, derivamos a solução para o problema de MV em forma fechada , resultado relevante na prática, dado que as nossas experiências demonstram haver ganhos em termos de redução de variância em optimização estocástica mesmo para este cenário simples.

Realizamos uma análise teórica que fornece intuição acerca do ajuste de parâmetros e dos resultados dados pelo estimador, demonstrando, em particular, porque é que o COCO necessariamente leva a melhorias em relação ao oráculo com ruído. A nossa análise experimental corrobora estes resultados e mostra que o COCO, apesar de ser um estimador enviesado, leva à diminuição do erro quadrático médio dos gradientes da função de custo. Para ilustrar o seu impacto em optimização estocástica, testamo-lo usando quer dados sintéticos, quer uma tarefa real de aprendizagem *online*. Estes testes demonstram que o COCO leva a melhorias na redução de variância em relação a algoritmos correntes.

Palavras Chave

Optimização Estocástica; Algoritmos de Primeira Ordem; Optimização Convexa; Co-coercividade; Filtro COCO; Redução de Variância; Problema Quadrático Quadraticamente Restringido; Aprendizagem Automática.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACC	Stochastic Accelerated schemes
APPA	Approximate Proximal Point Algorithm
COCO	Co-coercivity
GD	Gradient Descent
ISTA	Iterative Shrinkage-Thresholding Algorithms
FDPG	Fast Dual Proximal Gradient
FISTA	Fast Iterative Shrinkage-Thresholding Algorithms
KKT	Karush-Kuhn-Tucker
MISO	Minimization by Incremental Surrogate Optimization
ML	Maximum Likelihood
MSE	Mean Squared Error
NAG	Nesterov Accelerated Gradient
QCQP	Quadratically Constrained Quadratic Problem
S2GD	Semi-Stochastic Gradient Descent
SA	Stochastic Approximation
SAG	Stochastic Average Gradient
SAGA	Stochastic Average Gradient “amelioré”
SDCA	Stochastic Dual Coordinate Ascent
SGD	Stochastic Gradient Descent
SVRG	Stochastic Variance-Reduced Gradient
VR	Variance Reduction

1

Introduction

Contents

We start by motivating the problem addressed in the thesis and summarizing existing approaches. Then, we describe our methods, emphasizing their novelty. Finally, we outline the organization of the thesis.

1.1 Problem and Motivation

Optimization is an intrinsic part of Nature. In Physics, systems tend to states of minimum energy and photons follow paths that take them from one point to another in the least time possible. In Biology, evolution theory shows how the fittest individuals to the environment survive, leading to the continuous improvement of population genetics along time. Besides the usefulness of optimization to describe the universe around us, it has also more practical applications: *mathematical optimization* is a pivotal tool when rationalizing the process of decision making. In Economics and Industry, its relevance comes from, for example, the aim of maximizing profits, utilities, or efficiencies, under constraints of avoiding excessive risk, having access to finite resources or limited ranges for some parameters in engineering designs. Moreover, as a consequence of the extraordinarily fast increase in computational power in the last few decades, its importance has only been rising.

Formally, a *mathematical optimization problem* can be written as [3]

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \ i = 1, \ldots, m \\ & h_i(x) = 0, \ \ i = 1, \ldots, p\,, \end{aligned} \tag{1.1}$$

where the *optimization variable* is the d-dimensional vector x[1], the *objective function* is $f : \mathbb{R}^d \to \mathbb{R}$, and the constraints are coded via *equality constraint functions* $h_1, \ldots, h_p : \mathbb{R}^d \to \mathbb{R}$ and *inequality constraint functions* $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$. The vectors that satisfy the constraints constitute the *feasible set*, interchangeably referred to by *domain*[2]; an *optimal vector*, usually denoted by x^*, leads to the smallest possible value for the objective function in the feasible set.

The flexibility of the framework in Equation (1.1) enables expressing many problems of our daily life; this formalization is usually referred to as *modelling*. However, finding an algorithm that computes an optimal x^*, within a given accuracy, for a generic instance of Equation (1.1) is very hard (in fact, there is not an universal solution method for the general mathematical optimization problem in a reasonable time [4]). In spite of this, particular classes of the general problem are tractable and the appropriate modelling of a real life problem as an efficiently solvable mathematical optimization problem is generally accepted as a key step. This thesis focuses on one of such classes, the *convex optimization* problems, which nowadays underlie methods that tackle several practical applications, such as in machine learning,

[1] We consider d-dimensional variables to live in an Euclidean space and denote by $\|\cdot\|$ the Euclidean norm.
[2] Rigorously, the *domain* of the problem is the intersection between the feasible set and the domain of the objective function.

signal processing, portfolio optimization, or device sizing [3, 5].

We start by recalling the definitions of convex set and convex function [6].

Definition 1.1.1 (Convex Set)**.** A set $\mathcal{X} \subset \mathbb{R}^d$ is said to be convex if it contains all segments whose extremes are in $\mathcal{X}$, *i.e.*, if

$$\forall x, y \in \mathcal{X}, \gamma \in [0,1] : \qquad (1-\gamma)x + \gamma y \in \mathcal{X} .$$

Definition 1.1.2 (Convex Function)**.** A function $f : \mathcal{X} \to \mathbb{R}$ is said to be convex if it lies below all its chords, *i.e.*, if

$$\forall x, y \in \mathcal{X}, \gamma \in [0,1] : \qquad f\left(\gamma x + (1-\gamma)y\right) \leq \gamma f(x) + (1-\gamma) f(y) .$$

The general optimization problem of Equation (1.1) corresponds to a convex optimization problem when:

1. The objective function f is convex;
2. The inequality constraint functions $f_1, \ldots, f_m$ are convex;
3. The equality constraint functions are affine, *i.e.*, $h_i(x) = b_i^T x + c_i$, where b_i and c_i are constants.

The domain of this class of problems is convex (it results from the intersection of convex sets), thus, in convex optimization, one basically seeks to minimize a convex objective function over a convex set [3].

A key aspect of this class of optimization problems is that convex functions exhibit a local to global phenomenon: their gradients, $\nabla f(x)$, which in general only contain local information about the function, provide global information in the convex case, through a global linear lower bound [6]. Moreover, their local minima are, in fact, global minima. In these points, the gradient vanishes, *i.e.*, $\nabla f(x^*) = 0$. These properties lay the foundation of strong theoretical guarantees for convex optimization algorithms, which along with their wide range of applications [3], motivate the focus of this thesis.

Since the beginning of the development of mathematical optimization, a variety of algorithms has been developed for many different instances of the general optimization problem in Equation (1.1). Therefore, it is with no surprise that the same applies in the scope of convex optimization. The vast majority of these algorithms are iterative and usually classified into two main families [4]:

1. **First-order algorithms** - use only first-order derivatives (in the multivariate case, gradients), requiring then that the objective function must be differentiable.
2. **Second-order algorithms** - use second-order derivatives (in the multivariate case, Hessian matrices), requiring then that the objective function must be twice-differentiable.

First-order algorithms are usually characterized by exhibiting slower convergence rate to the optimal solution but much cheaper cost per iteration, when compared to second-order ones. In fact, the tragic

way the computational cost of second-order iterations scales with the dimension of the problem often makes them prohibitive for problems of high dimensionality, which are precisely the ones relevant for many applications, such as machine learning and signal/image processing. Besides, in many of these applications, the objective function is just a proxy to the real-life problem, so that cheap approximate solutions are vastly more important than expensive accurate ones [4]. For example, in machine learning, not only the dataset may contain errors but also overfitting to the minimum of the empirical risk is not desirable, since it often leads to a worse performance outside the used training set corresponding to a loss of generalization power [7]. For these reasons, first-order algorithms are nowadays prominent.

The most popular first-order algorithm is the Gradient Descent (GD), which performs greedy local steps: starting from an initial guess x_0, GD computes each subsequent iterate by taking a step in the opposite direction of the one pointed by the gradient of the objective function f. Naturally, if the step is of appropriate length, the iterate x_{i+1} will be closer to the optimal solution than x_i. It can be shown that if the steps are simply proportional to the magnitude of the gradient, GD converges asymptotically to the optimal solution[3]. The simplicity of GD comes from just requiring access to the gradient of the objective function at each iterate, $\nabla f(x_i)$. An interesting question emerges: *what if, instead of being able to compute the exact gradient $\nabla f(x_i)$, we can only have access to an oracle that provides a noisy version of $\nabla f(x_i)$?* This is the specific problem we address in the thesis.

In fact, there are cases in which the exact gradient can not be easily computed, for example due to the computational cost involved. This is a real scenario often found in high-dimensional optimization problems. Maybe the more glaring examples come from machine learning applications, where algorithms like the Stochastic Gradient Descent (SGD), which can be seen as greedy methods that use noisy versions of the true gradient, are a requirement in order to achieve timely solutions [7].

1.2 Overview of Current Approaches

The problem we motivated in the previous section lives in the field of stochastic optimization. Although the machine learning frenzy of the last few years has strongly contributed to the development and enhancement of algorithms like SGD, the first approaches to stochastic optimization date back to the fifties of the last century, as detailed in the following chapter. Here, we single out results that inspire the methods proposed in the thesis. The scenario is simply described: the objective function f is unknown but can be accessed through queries to a first-order *oracle*, *i.e.*, an unit that takes as input a vector x and outputs the gradient $\nabla f(x)$ [6]. While GD uses an exact oracle, SGD has to deal with an inexact one.

Unlike with GD, a fixed step size for SGD does not make it converge to the optimal solution, even

[3]While we described GD disregarding the constraints in Equation (1.1), *i.e.*, considering domain $\mathbb{R}^d$, the equivalent algorithm for other convex domains, usually referred to as *Projected* GD, just includes the extra step of projecting onto the domain at each iteration. For the sake of simplicity, without loss of generality, we will consider domain $\mathbb{R}^n$ except otherwise stated.

in the convex setting. In fact, the convergence of SGD can be analyzed considering two terms: (i) the bias, which represents the dependence of the convergence on the initial distance to the optimum (*e.g.*, $\|f(x_0) - f(x^*)\|$); and (ii) the variance, which represents the dependence of the convergence on the noise of the oracle itself. Although the bias vanishes under a convenient selection of a fixed step size, that does not happen to the variance. For this reason, the algorithm gets stuck when the bias term has vanished, originating random iterates within a certain "ball of uncertainty" (see Chapter 2 for details).

This problem in the stochastic setting can be solved by using a diminishing step size, *i.e,*, by selecting at iteration i a step size $\gamma_i = C/i$, where C is constant. Basically, this strategy progressively reduces the referred ball of uncertainty, enabling convergence. It can be shown that the convergence rates obtained using this approach are asymptotically optimal when the expected value of the oracle equals the true gradient (*e.g.*, whenever the noise affecting the gradients is additive and zero mean) [8,9]. In spite of this, further significant improvements were achieved through online averaging (the so-called Polyak-Ruppert averaging [10]) and adaptive step size techniques, *e.g.*, Adam [11]. Naturally, these improvements in constants on the convergence rate are of utmost importance in practice, where one necessarily deals with a finite horizon in terms of number of iterations.

In the last decade, a new direction has driven the research community to a paradigm that enhances the aforementioned asymptotic bounds. For objectives that can be decomposed as a finite sum of functions, which is precisely the case of many machine learning problems, the so-called Variance Reduction (VR) techniques significantly improve the performance of the optimization algorithms. In fact, the rise of algorithms such as the Stochastic Average Gradient (SAG) [12] made it possible to close the gap that existed between the asymptotic convergence rate gap of the deterministic GD and the ones of all the stochastic oracle counterparts.

Nevertheless, in the deterministic scenario, it is also known that GD is sub-optimal among the methods that perform each step still using only gradient information but obtained at more than one point [13]. Considering the class of algorithms that generate each iterate using a linear combination of the gradients of all previously visited locations, *i.e.*, such that $x_i \in x_0 + \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_{i-1})\}$, the work developed by Polyak, with its heavy-ball method [14], then refined by Nesterov [15], lead to the so-called Nesterov Accelerated Gradient (NAG) descent algorithm, which achieves the optimal rate. In the stochastic scenario, the optimal rates were naturally shown to be worse [16, 17] and are attained by a family of stochastic accelerated algorithms, first introduced by the reference [18].

Our goal in this thesis is to evaluate the possibility of further improving the performance of the state-of-the-art stochastic optimization algorithms referred above by providing them cleaner gradient estimates. These gradient estimates will be obtained by exploiting properties of convex functions that, although widely used for convex optimization algorithm analysis, have been left out of algorithm design.

1.3 Proposed Methods

Two structural properties of many convex functions are the following:

Definition 1.3.1 (L-Smooth Function)**.** A continuously differentiable function, $f : \mathbb{R}^d \to \mathbb{R}$, is said to be L-smooth (or to have Lipschitz-continuous gradients with constant $L \in \mathbb{R}^+$) if

$$\forall x, y \in \mathbb{R}^d : \qquad \|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\| .$$

Definition 1.3.2 (μ-Strongly Convex Function)**.** A continuously differentiable function, $f : \mathbb{R}^d \to \mathbb{R}$, is said to be μ-strongly convex (or to be strongly convex with modulus $\mu \in \mathbb{R}^+$) if

$$\forall x, y \in \mathbb{R}^d : \qquad f(x) \geq f(y) + \nabla f(y)^T(x - y) + \frac{\mu}{2}\|x - y\|^2 . \tag{1.2}$$

Strong convexity immediately provides a quadratic lower bound for the function, as stated by Equation (1.2). In turn, it can be shown that L-smoothness imposes a quadratic upper bound [19], expressed by

$$f(x) \leq f(y) + \nabla f(y)^T(x - y) + \frac{L}{2}\|x - y\|^2 .$$

Furthermore, it can also be shown that, if a function f is twice differentiable, μ strong-convexity imposes $\mu I \preceq \nabla^2 f$ and L-smoothness imposes $\nabla^2 f \preceq LI$, where ∇^2 represents the Hessian operator and $\preceq$ is the standard notation for matrix inequalities based on positive semi-definiteness [6]. Thus, we necessarily have $\mu \leq L$. In fact, constants L and μ characterize, respectively, the maximum and minimum curvature of function f and the ratio between them is known by the condition number of the convex function f, $k = L/\mu \geq 1$.

The relevance of strong convexity and L-smoothness is clear: while strong convexity imposes that at every point there is at least some curvature of the function, L-smoothness grants that such curvature can not be arbitrarily high. These properties have shown to be very useful for the analysis of convex optimization algorithms without excessively restricting the convex settings in which they can be verified, as they are still very general [13]. However, to the best of our knowledge, existing algorithms for stochastic convex optimization do not make explicit use of this kind of restrictions to the function curvature, naively accepting the noisy gradients provided by the oracle. This motivated us to exploit L-smoothness (usually considered an even weaker assumption than strong convexity [20]) and mere convexity (which can be seen as a relaxation of strong-convexity to $\mu = 0$) to constrain function gradients. It has been shown that L-smoothness and convexity can be merged into one single condition, usually referred to as gradient *co-coercivity* [19], which suggested coining our method as the *Co-coercivity (COCO) denoiser*. We formulate the denoising problem as the joint Maximum Likelihood (ML) estimation of a set of function gradients, constrained by the co-coercivity conditions, from their noisy observations provided by the or-

acle, where, as often done, the noise is assumed to be zero mean white Gaussian. Our workflow for stochastic optimization, schematically represented in Figure 1.1, consists in using COCO as a "plug-in" denoiser, *i.e.*, in feeding the denoised gradients to a baseline algorithm (such as the ones outlined in the previous section) rather that letting it consult directly the oracle.

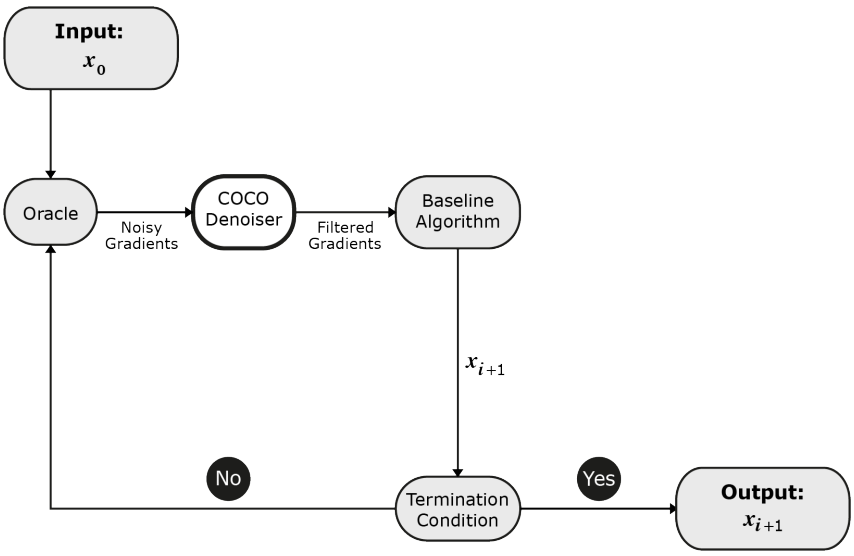


Figure 1.1: Our workflow for stochastic optimization with first-order algorithms. The approach can be interpreted as using a "new oracle", composed by the original one coupled with the proposed COCO denoiser.

The ML estimation in the COCO denoiser leads to a particular convex optimization problem known as Quadratically Constrained Quadratic Problem (QCQP). This problem can be solved by using available convex optimization packages, such as the popular CVX [21], but its complexity may turn out prohibitive. For this reason, we exploit the particular structure of the ML estimation problem to derive an original first-order algorithm, based on the so-called Fast Dual Proximal Gradient (FDPG) method [22], which succeeds in providing an approximate solution in reasonable time.

In spite of the efficiency of the proposed algorithm, the number of co-coercivity constraints grows quadratically with the number of points simultaneously processed by COCO. Thus, although cleaner gradient estimates are obtained by processing simultaneously the ever growing number of points visited, *i.e.*, points $x_0, \ldots, x_i$, in Figure 1.1, we also considered performing the denoising using just a fixed number K of last visited points, *i.e.*, $x_{i-K+1}, \ldots, x_i$. For shortness of reference, we call this denoiser COCO_K (naturally, the globally optimal denoiser is then COCO_{i+1}). This point deserves particular attention because our experiments have shown that convergence gains in stochastic optimization are obtained with values of K as small as 2 and we were able to find the *closed-form solution* for COCO_2.

By theoretically analysing COCO, we were able to provide insight regarding the estimator results, showing in particular why COCO necessarily improves with respect to the noisy oracle. We also evaluate the probability of the co-coercive constraints being "active", which enables interpreting the impact of the

chosen Lipschitz constant L in our approach. Our experimental analysis corroborates these results and show that the COCO estimator, although not unbiased, leads to a reduction of the Mean Squared Error (MSE) of the function gradients, as desired.

In the context of stochastic optimization, to evaluate the impact of using the proposed COCO denoiser, we consider two scenarios. Firstly, using synthetic data that follow the assumptions underlying the design of the denoiser. Secondly, using an online learning task (logistic regression [20]), in which the noise affecting the gradients falls out of those assumptions. Our experiments have shown that COCO leads to improvements in variance reduction with respect to baseline algorithms such as SGD and Adam.

1.4 Summary of Contributions

We emphasize the following aspects as original contributions of the work in the thesis:

- Exploration of L-smoothness and convexity (*i.e.*, co-coercivity) in the context of the Maximum Likelihood estimation of function gradients from noisy observations, leading to the COCO denoiser;
- Efficient first-order solution method for COCO (FDPG);
- Closed-form solution for COCO_2;
- Analytic study of COCO, providing insights into parameter tuning and expected error reduction;
- Experimental analysis of COCO, regarding estimator bias and mean squared error;
- Framework for stochastic optimization using first-order algorithms with the COCO denoiser;
- Experiments illustrating variance reduction in convex stochastic optimization using COCO.

The bibliographic review presented in the following chapter also deserves mention, in particular due to the summary of asymptotically optimal convergence rates, which is not conveyed in such condensed form even in recent surveys. The main original results of this work will also appear in [23, 24].

1.5 Thesis Organization

The remaining of this document is organized as follows. Chapter 2 contains an extensive review of first-order methods in stochastic convex optimization. Chapter 3 details the proposed approach for the exploitation of co-coercivity to filter noisy gradients. We formalize the problem, derive methods to address it, and study properties of the proposed solutions. In Chapter 4 we describe experiments that demonstrate the effectiveness of the proposed denoiser and its usefulness for stochastic convex optimization, using both synthetic and real datasets. Chapter 5 concludes the thesis by summarizing the work and pointing out directions for future research.

2

State of the Art

Contents

In this chapter, we provide an extensive review on the main developments of first-order stochastic convex optimization algorithms. We start from the deterministic setting, by further detailing the original Gradient Descent (GD) algorithm, in order to naturally introduce Stochastic Gradient Descent (SGD). We describe its vanilla version, followed by the Polyak-Ruppert averaging and adaptive step size techniques. Then, we proceed to the recent stochastic Variance Reduction (VR) methods. Finally, we briefly describe the Nesterov Accelerated Gradient (NAG) algorithm in order to motivate the Stochastic Accelerated schemes (ACC).

Table 2.1 summarizes the different asymptotically optimal convergence rates for each family of algorithms mentioned in the following sections.

Table 2.1: Asymptotically optimal convergence rates ($\mathbb{E}[f(x_i) - f(x^*)]$) for algorithms GD, NAG, SGD (which includes averaging and adaptive methods), VR (only for finite sums) and ACC (only for finite sums), under different assumptions on the objective function, f. See text for the meaning of constants k, $k_{\max}$, and n.

	Deterministic		**Stochastic**		
Assumption(s)	GD	NAG	SGD	VR	ACC
Convexity	$O\left(\frac{1}{\sqrt{i}}\right)$	$O\left(\frac{1}{\sqrt{i}}\right)$	$O\left(\frac{1}{\sqrt{i}}\right)$	$O\left(\frac{1}{\sqrt{i}}\right)$	$O\left(\frac{1}{\sqrt{i}}\right)$
+ L-Smoothness	$O\left(\frac{1}{i}\right)$	$O\left(\frac{1}{i^2}\right)$	$O\left(\frac{1}{\sqrt{i}}\right)$	$O\left(\frac{1}{i}\right)$	$O\left(\frac{1}{i^2}\right)$
+ Strong Convexity	$O\left(e^{-\frac{i}{k}}\right)$	$O\left(e^{-\frac{i}{\sqrt{k}}}\right)$	$O\left(\frac{1}{i}\right)$	$O\left(e^{-\frac{i}{k_{\max}+n}}\right)$	$O\left(e^{-\frac{i}{\sqrt{nk_{\max}}+n}}\right)$

2.1 Gradient Descent

The Gradient Descent (GD) algorithm (Algorithm 2.1) belongs to the family of first-order algorithms. Historically, its discovery is assigned to Cauchy, who suggested it in 1847 [25], even though its convergence properties for non-linear optimization problems were only later analysed by Haskel Curry, in 1944 [26].

Algorithm 2.1: Gradient Descent (GD)

Input: Initial point: x_0; Number of steps: T; Step size: γ.
for $i = 1, \dots, T$ **do**
  $x_i = x_{i-1} - \gamma \, \nabla f(x_{i-1})$
Output: Final point, x_T.

Intuitively, the reasoning behind this approach is quite straightforward: since the direction of maximum growth of the objective function at a given point x_i is given by its gradient, $\nabla f(x_i)$, and we want to find the minimum of the function, we must move in the opposite direction of the gradient since it will be the one of maximum decrease. Therefore, the idea of the algorithm is to apply these steps iteratively.

Regarding the step size, it can be chosen in many ways - it even might change from one iterate to another - and, actually, it plays a key role in the algorithm convergence to the minimum of the objective function. Moreover, for a well defined range of step sizes, this algorithm is guaranteed to converge to the optimal value x^* [13]. Note, nevertheless, that this range of appropriate step sizes depends on characteristics of the objective function.

2.2 Stochastic Gradient Descent

2.2.1 Vanilla SGD

The original GD algorithm has been around for almost two centuries but we are particularly interested in an alternative but similar method: the Stochastic Gradient Descent (SGD) algorithm, where the gradient $\nabla f(x_{i-1})$ in the update rule of GD is replaced by a noisy version of that gradient at the same point, $g(x_{i-1})$, which for the sake of simplicity we denote by g_{i-1}, as represented in Algorithm 2.2. It is clear that both these algorithms resort to a first-order oracle, *i.e.*, require access to the gradient of the objective function. However, while in the GD case the oracle is exact, in the SGD case the oracle is inexact. Consequently, while GD is classified as a solution method considered in convex optimization problems, the SGD algorithm belongs to the stochastic optimization framework. This framework is generically described as the class of optimization problems which involve random variables whether they appear in the formulation of the problem itself (more specifically, in the objective function or constraint functions) or even in the form of random iterates of the solution method [27].

Algorithm 2.2: Stochastic Gradient Descent (SGD)

Input: Initial point: x_0; Number of steps: T; Step size: γ_i.
for $i = 1, \ldots, T$ **do**
  $x_i = x_{i-1} - \gamma_i \, g_{i-1}$
Output: Final point: x_T.

But why should we care about using an inexact oracle? In fact, it is reasonable to argue that a first-order algorithm resorting to a noisy version of the gradient will always be outperformed by one resorting to an exact one. There are cases, however, in which it might be preferable to consult the stochastic version because the computational cost of consulting the exact gradient is much higher than consulting its noisy version. This reduction of the computational burden traded for a lower precision of the gradient version allows the algorithm to iterate faster. As a consequence, despite the slower convergence rate, overall it might be advantageous in terms of the time required to achieve a sufficiently good approximate solution of the problem.

This is the case for many machine learning problems where the objective function can be defined as a finite sum of terms, each of these concerning a different example from the dataset. In this case, it

turns out that an inexact oracle is obtainable from the evaluation of a single data point, while its exact version requires a full pass through the whole dataset. Thus, in datasets where the number of points is very large, an inexact oracle becomes a requirement [7]. This case is further explored in section 2.3.

Historically, the main ideas behind SGD can be traced back to an older but closely related method of the stochastic optimization field, which is known as Stochastic Approximation (SA). That method, first proposed by Robbins and Monro in 1951 [28], solves the following problem: given a function M and a constant α, such that $M(x) = \alpha$ has a unique root , find that root x^*, without directly accessing M, rather only sampling a random variable $N(x;\xi)$, where ξ is a random vector and $\mathbb{E}_\xi[N(x;\xi)] = M(x)$.

The Robbins-Monro algorithm approaches this problem by generating iterates of the following form:

$$x_{i+1} = x_i - \gamma_i(N(x_i;\xi) - \alpha)\,.$$

This sequence of iterates is guaranteed to converge to the optimal value x^* with probability one under the following assumptions [29]:

Assumption 2.2.1. *$N(x;\xi)$ has finite variance: $\mathbb{E}\left[(N(x;\xi) - M(x))^2\right] < \infty$ for all x.*

Assumption 2.2.2. *M is a nondecreasing Lebesgue-measurable function.*

Assumption 2.2.3. *$M'(x^*)$ exists and is positive.*

Assumption 2.2.4. *$\gamma_i > 0$, $\sum_{i=0}^{\infty}\gamma_i = \infty$, and $\sum_{i=0}^{\infty}\gamma_i^2 < \infty$ (e.g., $\gamma_i = a/i,\ a > 0$, as suggested in reference [28]).*

Extrapolating this framework for the gradient descent scheme in a convex problem paradigm, we observe the following correspondences: $M(x_i) = \nabla f(x_i)$, $N(x_i;\xi) = g_i$, $\alpha = 0$ and γ_i corresponds to the step size. Note that Assumption 2.2.3 is clearly related to the positive definite Hessian matrix of a strictly convex function at least in a neighbourhood of the optimal point, x^*. In fact, a broader definition of convex function only implies a positive semi-definite Hessian matrix, but, in that case, we might fall into a situation of multiple minima. Nevertheless, in that case, the method is still guaranteed to converge to one of those points.

To have a better insight of the SGD behavior, it is quite illustrative to carefully examine the results provided in its convergence analysis, which are presented under the following assumptions:

Assumption 2.2.5. *The objective function is L-smooth and μ-strongly convex.*

Assumption 2.2.6. *$\mathbb{E}[\, g(x_i;\xi_i)|\, \xi_1,\ \ldots,\ \xi_{i-1}\,] = \nabla f(x_i)$, i.e., $g(x_i;\xi_i)$ is an unbiased estimate of $\nabla f(x_i)$ given $\{\xi_1, ..., \xi_{i-1}\}$.*

Assumption 2.2.7. *The (uncentered) variance of $g(x;\xi)$ is uniformly bounded, i.e., $\mathbb{E}\left[\|g(x;\xi)\|^2\right] \leq \sigma^2$.*

We start by proving Lemma 2.2.1, which will be further required:

Lemma 2.2.1. *Under Assumption 2.2.5, Assumption 2.2.6, and Assumption 2.2.7, we have:*

$$\mathbb{E}\left[(x_i - x^*)^T g(x_i;\xi_i)\right] \geq \mu\, \mathbb{E}\left[\|x_i - x^*\|^2\right]$$

Proof. Taking into account Assumption 2.2.6:

$$\begin{aligned}
\mathbb{E}\left[(x_i - x^*)^T g(x_i;\xi_i)\right] &= \mathbb{E}\left[\, \mathbb{E}\left[(x_i - x^*)^T\, g(x_i;\xi_i) \mid \xi_1, ..., \xi_{i-1}\,\right]\,\right] && (2.1)\\
&= \mathbb{E}\left[\, (x_i - x^*)^T\, \mathbb{E}\left[\, g(x_i;\xi_i) \mid \xi_1, ..., \xi_{i-1}\,\right]\,\right] && (2.2)\\
&= \mathbb{E}\left[\, (x_i - x^*)^T\, \nabla f(x_i)\,\right], && (2.3)
\end{aligned}$$

where in Equation (2.1) the law of total expectation is applied to the initial expression; in Equation (2.2), the term $(x_i - x^*)^T$ is constant and Equation (2.3) comes directly from Assumption 2.2.6.

On the other hand, note that:

$$\begin{aligned}
(x_i - x^*)^T\, \nabla f(x_i) &= (x_i - x^*)^T\, (\nabla f(x_i) - \nabla f(x^*)) && (2.4)\\
&\geq \mu\, \|x_i - x^*\|^2, && (2.5)
\end{aligned}$$

where Equation (2.4) is a result of the fact that $\nabla f(x^*) = 0$ and Equation (2.5) comes from the strong convexity of $f(x)$ (Assumption 2.2.5).

Consequently, analysing Equation (2.5) in expectation, we obtain the intended result □

Now we can prove the convergence of SGD; first, for a fixed step size (Theorem 2.2.2); then, for a variable (diminishing) step size (Theorem 2.2.3).

Theorem 2.2.2. *Under Assumption 2.2.5, Assumption 2.2.6, and Assumption 2.2.7, if* $0 < \gamma < 1/\mu$*, the iterates* x_i *generated by SGD satisfy:*

$$\mathbb{E}[\|x_i - x^*\|^2] \leq (1 - 2\mu\gamma)^{i+1}\, \|x_0 - x^*\|^2 + \left[\gamma \frac{1 - (1 - 2\mu\gamma)^{i+1}}{2\mu}\right] \sigma^2\,.$$

Proof. We begin by expanding the expression for $\|x_{i+1} - x^*\|^2$ by sequentially considering the simple Euclidean norm property $\|v\|^2 = v^T v$:

$$\begin{aligned}
\|x_{i+1} - x^*\|^2 &= \|x_i - \gamma_i\, g(x_i;\xi_i) - x^*\|^2\\
&= \|x_i - x^*\|^2 - 2\gamma_i\, (x_i - x^*)^T g(x_i;\xi_i) + \gamma_n^2\, \|g(x_i;\xi_i)\|^2\,.
\end{aligned}$$

Analyzing the same expression in expectation, we obtain:

$$\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] = \mathbb{E}\left[\|x_i-x^*\|^2\right] - 2\gamma_i\,\mathbb{E}\left[(x_i-x^*)^T g(x_i;\xi_i)\right] + \gamma_i^2\,\mathbb{E}\left[\|g(x_i;\xi_i)\|^2\right] \tag{2.6}$$

$$\leq (1-2\mu\gamma_i)\,\mathbb{E}\left[\|x_i-x^*\|^2\right] + \gamma_i^2\sigma^2 \tag{2.7}$$

$$\leq (1-2\mu\gamma_i)(1-2\mu\gamma_{i-1})\,\mathbb{E}\left[\|x_{i-1}-x^*\|^2\right] + \left(\gamma_i^2+\gamma_{i-1}^2(1-2\mu\gamma_i)\right)\sigma^2 \tag{2.8}$$

$$\leq \dots \tag{2.9}$$

$$\leq \mathbb{E}\left[\|x_0-x^*\|^2\right]\,\prod_{k=0}^{i}(1-2\mu\gamma_k)\;+\sigma^2\sum_{k=0}^{i}\left[\gamma_k^2\prod_{j=k+1}^{i}(1-2\mu\gamma_j)\right] \tag{2.10}$$

$$\leq \|x_0-x^*\|^2\prod_{k=0}^{i}(1-2\mu\gamma_k)\;+\sigma^2\sum_{k=0}^{i}\left[\gamma_k^2\prod_{j=k+1}^{i}(1-2\mu\gamma_j)\right], \tag{2.11}$$

where Equation (2.6) comes from applying the linearity of the expectation operator; Equation (2.7) comes from applying Lemma 2.2.1; from Equation (2.8) to Equation (2.10) we apply the previous inequality ($\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] \leq (1-2\mu\gamma_i)\mathbb{E}\left[\|x_i-x^*\|^2\right]+\gamma_i^2\sigma^2$) iteratively. Finally, Equation (2.11) results from neither x_0 nor x^* being random variables (they are constants).

At this point, it should be noted that on the right-hand side of the inequality in Equation (2.11) we have two independent terms: one depends on $\|x_0-x^*\|^2$ and is called the *bias term*; the other depends on σ^2 and is known as the *variance term*. Now, assuming a constant step size, $\gamma_i=\gamma$:

$$\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] \leq (1-2\mu\gamma)^{i+1}\|x_0-x^*\|^2 + \left[\gamma^2\sum_{k=1}^{i}(1-2\mu\gamma)^k\right]\sigma^2$$

$$= (1-2\mu\gamma)^{i+1}\|x_0-x^*\|^2 + \left[\gamma^2\frac{1-(1-2\mu\gamma)^{i+1}}{1-(1-2\mu\gamma)}\right]\sigma^2, \quad \text{if} \quad 0<\gamma<\frac{1}{\mu} \tag{2.12}$$

$$= (1-2\mu\gamma)^{i+1}\|x_0-x^*\|^2 + \left[\gamma\frac{1-(1-2\mu\gamma)^{i+1}}{2\mu}\right]\sigma^2, \tag{2.13}$$

where Equation (2.12) is directly obtained from the sum of a geometric series. Its simplification yields Equation (2.13). □

In particular, we are interested in what happens when the number of iterations increases ($i\to\infty$). Therefore, it is clear that when $0<\gamma<1/\mu$:

$$\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] \;\leq\; (1-2\mu\gamma)^{i+1}\left(\|x_0-x^*\|^2-\frac{\gamma}{2\mu}\sigma^2\right)+\frac{\gamma}{2\mu}\sigma^2 \;\overset{i\to\infty}{\to}\; \frac{\gamma}{2\mu}\sigma^2\,. \tag{2.14}$$

This result is extremely elucidative of the SGD behavior. It shows how, at first, it converges as both the bias term and part of the variance term present linear convergence (*i.e.*, $\mathbb{E}[\|x_{i+1}-x^*\|^2]$ decreases); then, when those components become negligible, a non-convergent part from the variance term stands out and the algorithm generates random iterates within a ball centered at x^* with radius $\gamma\sigma^2/(2\mu)$. Nat-

urally, when $\sigma^2 = 0$, we recover the linear convergence of gradient descent under strong convexity [13], as shown in Table 2.1.

As a consequence, in practical applications, it is common to implement the SGD algorithm with a slight variation from its original version: whenever the convergence apparently stops, it halves the step size [30], which halves the radius of the ball containing the random iterates. This idea of iteratively reducing the radius of the ball in which the SGD may generate iterates naturally leads to the idea of using a diminishing step size in a pre-defined schedule in order to ensure convergence. Theorem 2.2.3 explores exactly this point:

Theorem 2.2.3. *Under Assumption 2.2.5, Assumption 2.2.6 and Assumption 2.2.7, if* $\gamma_i = \theta/(i+1)$ *with* $\theta > 1/(2\mu)$*, the iterates generated by SGD satisfy:*

$$\mathbb{E}\left[\|x_i - x^*\|^2\right] \leq \frac{c_\theta}{i+1},$$

where $c_\theta = \max\left\{\frac{2\theta^2}{2\mu\theta - 1}\sigma^2,\ \|x_0 - x^*\|^2\right\}$.

Proof. We use mathematical induction. The first step is to verify that the inequality holds for $i = 0$, *i.e.*,

$$\begin{aligned}
\mathbb{E}\left[\|x_0 - x^*\|^2\right] &= \|x_0 - x^*\|^2 && (2.15)\\
&\leq \max\left\{\frac{2\theta^2\sigma^2}{2\mu\theta - 1},\ \|x_0 - x^*\|\right\} && (2.16)\\
&= \frac{c_\theta}{1}, && (2.17)
\end{aligned}$$

where Equation (2.15) results from x_0 and x^* being constants; Equation (2.16) is trivially verified from the $\max$ operator properties and Equation (2.17) recovers the intended result for $i = 0$.

The next and final step is to show that if the inequality holds for iteration i, then it also holds for the

next iteration, $i+1$:

$$\mathbb{E}\left[\|x_{i+1}-x^*\|^2\right] \leq (1-2\mu\gamma_i)\,\mathbb{E}\left[\|x_i-x^*\|^2\right]+\gamma_i^2\,\sigma^2 \tag{2.18}$$

$$= \left(1-2\mu\frac{\theta}{i+1}\right)\mathbb{E}\left[\|x_i-x^*\|^2\right]+\left(\frac{\theta}{i+1}\right)^2\sigma^2 \tag{2.19}$$

$$\leq \left(1-2\mu\frac{\theta}{i+1}\right)\frac{c_\theta}{i+1}+\left(\frac{\theta}{i+1}\right)^2\sigma^2 \tag{2.20}$$

$$= \left(\frac{1}{i+1}-\frac{2\mu\theta}{(i+1)^2}\right)c_\theta+\frac{2\mu\theta-1}{2(i+1)^2}\frac{2\theta^2}{2\mu\theta-1}\sigma^2, \quad \text{if} \quad \theta>\frac{1}{2\mu} \tag{2.21}$$

$$\leq \left(\frac{1}{i+1}-\frac{2\mu\theta}{(i+1)^2}+\frac{\mu\theta-\frac{1}{2}}{(i+1)^2}\right)c_\theta \tag{2.22}$$

$$= \left(\frac{1}{i+1}-\frac{\mu\theta+\frac{1}{2}}{(i+1)^2}\right)c_\theta \tag{2.23}$$

$$\leq \left(\frac{1}{i+1}-\frac{1}{(i+1)^2}\right)c_\theta \tag{2.24}$$

$$\leq \left(\frac{1}{i+2}\right)c_\theta, \tag{2.25}$$

where Equation (2.18) recovers the result obtained in Equation (2.7); Equation (2.19) results from imposing $\gamma_i = \theta/(i+1)$; Equation (2.20) is obtained from assuming the induction hypothesis for i: $\mathbb{E}\left[\|x_i-x^*\|^2\right] \leq c_\theta/(i+1)$; in Equation (2.21) the second term is divided and multiplied by $(2\mu\theta-1)/2$; Equation (2.22) is a direct result from the definition of c_θ; Equation (2.23) is a mere simplification from Equation (2.22); Equation (2.24) is obtained considering that $\theta > 1/(2\mu)$ (from Equation (2.21)) and Equation (2.25) is trivially verified from Equation (2.24). □

Therefore, in the case of a diminishing step size, it is readily verified that when the number of iterations increases ($i \to \infty$), the distance from the iterate to the optimum vanishes, *i.e.*, when $\theta > 1/(2\mu)$:

$$\mathbb{E}\left[\|x_i-x^*\|^2\right] \leq \frac{c_\theta}{i+1} \overset{i\to\infty}{\to} 0\,.$$

Moreover, note that the result provided by Theorem 2.2.3 recovers the optimal rate observed for SGD in a convex, L-smooth and strongly convex function - $O\left(1/i\right)$.

Both in Theorem 2.2.2 and Theorem 2.2.3, we assumed strong convexity of the objective function for the sake of simplicity. Nevertheless, the analysis can also be performed by relaxing this assumption, although that deteriorates the asymptotic optimal convergence rate (see Table 2.1).

Furthermore, the analysis here performed was made in the vector space ($\mathbb{E}\left[\|x_i-x^*\|^2\right]$), even though it can also be commonly found on the function value space ($\mathbb{E}\left[\|f(x_i)-f(x^*)\|^2\right]$). Examples of equivalent SGD convergence proofs to the ones presented here but in the function value space are readily available at reference [30], both for the fixed and diminishing step sizes. Note that, in this reference, the assumption of finite uncentered variance (Assumption 2.2.7) is relaxed to $\mathbb{E}\left[\|g(x;\xi)\|^2\right] \leq$

$\sigma^2 + c_g\|\nabla f(x)\|^2$ (the uncentered variance of the oracle output is no longer required to be uniformly bounded, as it may grow quadratically with the groundtruth gradient). Despite this slight difference, their SGD analysis can also be decomposed in bias and variance terms. Taking this into consideration, we emphasize the importance of that decomposition.

2.2.2 Averaging Algorithms

Even though the idea of $O(1/i)$ step sizes allowed for the mitigation of noise effect and, thus, the convergence of SGD for convex functions, from a deterministic (noiseless) perspective, a step size that follows that decay rate is excessively conservative. In fact, it is in the search of algorithms that allow for larger step sizes without compromising their convergence that the averaging algorithms emerge: instead of simply taking into account the convergence of the iterates x_i, the following sequence is considered:

$$\bar{x}_i = \frac{1}{i+1}\sum_{j=0}^{i} x_j \,.$$

This online averaging technique is usually known as Polyak-Ruppert averaging [10] and by damping the oscillations of the generated iterates, it ends up reducing their variance. This is the intuition which, in this scheme, supports the possibility of using step sizes with slower decays (*i.e.*, $O\left(1/i^{\alpha}\right),\ 0<\alpha<1$) or, under some additional conditions, even fixed step sizes ($\alpha = 0$).

In fact, as shown in Table 2.1, for the generic convex optimization, there is no cost for having a stochastic oracle instead of an exact one as far as asymptotic rates are concerned (the rate of $O\left(1/\sqrt{i}\right)$ can not be beaten). However, when we impose smoothness, it can be shown that while it does not bring any benefit for the general stochastic case [31] (it is still $O\left(1/\sqrt{i}\right)$), in the deterministic setting it achieves $O(1/i)$. Nevertheless, while being true for the generic stochastic setting, it can also be shown that for specific functions smoothness provides in fact some improvement. For example, it was shown that for a least-squares regression problem, the rate of $O(1/i)$ is attainable resorting to Polyak-Ruppert averaging method with fixed step size [32].

Another important indication regarding the Polyak-Ruppert averaging was given by the reference [33], where by plugging it to a typical accelerated algorithm for the deterministic setting, again in a least-squares regression problem, it was possible to find the first algorithm achieving $O\left(1/i^2\right)$ on the bias term (comparable to NAG - see Table 2.1), without compromising the optimal $O(1/i)$ of the variance term [31], which is what happens when one blindly applies deterministic acceleration into a stochastic setting [34].

Finally, different averaging schemes have been proposed recently, which also lead to similar results in accelerating schemes, in spite of their more tortuous analysis. One clear example is provided by the

reference [35], which considers a "tail-averaged" sequence, *i.e.*,:

$$\bar{x}_{t,i} = \frac{1}{i-t} \sum_{j=t+1}^{i} x_j \, .$$

In this paradigm, there is an unaveraged/burn-in phase and only the $i - t$ iterates are averaged. Note that t can be a function of i (more precisely, in reference [35], results are provided for $t = \lfloor \frac{i}{2} \rfloor$[1]).

2.2.3 Adaptive Algorithms

Generally speaking, algorithms that are robust to initialization are usually preferable as they adjust to the optimization problem instance by themselves and do not depend on the tuning of hyperparameters (in the case of SGD, the step size). In fact, to tune a $O\left(1/i\right)$ sequence of step sizes can be extremely difficult for a given dataset, as it has to ensure that the algorithm does not stop too early, before reaching the solution, or too late, wasting resources.

It is in this demand that the adaptive (step size) algorithms (*e.g.*, AdaGrad [36], Adadelta [37], RMSprop [38], Adam [11], AdaMax [11] and Nadam [39]) appear. These algorithms receive as input an initial step size and then adjust it successively in each dimension according to the magnitude of the progress in that same dimension, *i.e.*, for dimensions of low curvature (thus, small gradients), the step size is increased; for dimensions of high curvature (thus, large gradients), the step size is reduced. Therefore, this approach drastically improves the performance over SGD, namely in problems with high condition number, being particularly popular in training deep neural networks [40].

Among these algorithms, the one most often used is Adam (Algorithm 2.3), whose name is inspired on "adaptive moment estimation". We briefly describe it here, in order to illustrate this family of algorithms. The Adam algorithm computes different adaptive learning rates for each dimension by storing in memory an exponentially decaying average of past gradients m_i and an exponentially decaying average of past squared gradients, v_i, similar to momentum:

$$\begin{aligned} m_i &= \beta_1 m_{i-1} + (1-\beta_1) g_i \\ v_i &= \beta_2 v_{i-1} + (1-\beta_2) g_i^2, \end{aligned}$$

where g_i^2 denotes the element-wise square of that vector.

Therefore, m_i and v_i are estimates of the first and second moment - the mean and the uncentered variance, respectively - of the gradient, respectively. Furthermore, the authors of the algorithm still

[1] $\lfloor \cdot \rfloor$ denotes the floor function, *i.e.*, $\lfloor x \rfloor$ is the greatest integer not greater than x.

propose bias-corrected estimates for those variables based on the expressions above:

$$\hat{m}_i = \frac{m_i}{1-\beta_1^i}$$
$$\hat{v}_i = \frac{v_i}{1-\beta_2^i}\,.$$

Finally, these values are used to compute the new iterate according to the Adam update rule:

$$x_{i+1} = x_i - \gamma \frac{\hat{m}_i}{\sqrt{\hat{v}_i}+\epsilon},$$

where γ is the step size and ϵ a smoothing term that avoids division by zero. Note that here the fraction bar denotes the element-wise division between $\hat{m}_i$ and $\sqrt{\hat{v}_i}+\epsilon$.

Algorithm 2.3: Adam

Input: Initial point: x_0; Number of steps: T; Step size: γ; Moment vectors (1^{st} and 2^{nd}, respectively): $m_0, v_0 = 0$; Exponential decay rates for the moment estimates: $\beta_1, \beta_2 \in [0, 1)$; Smoothing term: ϵ.

for $i = 1, \ldots, T$ **do**

$m_i = \beta_1 m_{i-1} + (1-\beta_1) g_i$
$v_i = \beta_2 v_{i-1} + (1-\beta_2) g_i^2$
$\hat{m}_i = \frac{m_i}{1-\beta_1^i}$
$\hat{v}_i = \frac{v_i}{1-\beta_2^i}$
$x_i = x_{i-1} - \gamma \frac{\hat{m}_i}{\sqrt{\hat{v}_i}+\epsilon}$

Output: Final point: x_T.

From Algorithm 2.3, it is possible to observe that in this method, there are 3 hyperparameters: β_1, β_2 and ϵ. The original paper defines default values for them which were verified to usually lead to a good performance of the algorithm: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ [11].

Regarding the convergence of the adaptive algorithms, there are many cases to consider. For example, AdaGrad, the first adaptive step size to emerge (in 2011), was proven to converge to the optimal solution in its original paper [36]. In the case of Adam, its original paper provided a proof of convergence [11], in which an error was later detected [41]. Moreover, reference [41] also shows how Adam fails to surely find an optimum through a counter-example in a convex optimization setting. This failure occurs due to Adam's exponential moving average procedure, putting at stake all the other known adaptive algorithm which also employ that technique (*e.g.*, Adadelta, RMSprop or Nadam). Since the detection of the error in Adam's analysis, newer proofs of convergence have been emerging (*e.g.*, in reference [42]) but based on more restrictive assumptions. Furthermore, other Adam-like algorithms have also been proposed, whose modifications allow for a proof of convergence and present enhanced empirical performance in most cases, such as AMSGrad (provided as a solution for Adam in the mentioned reference [41]) and Adam$^+$ [43].

2.3 Stochastic Variance Reduction Algorithms

Even though the results obtained for the general stochastic setting [8, 9, 31] have been well defined for a while (see column SGD in Table 2.1), in the last few years the research community has been specially focused on a specific stochastic setup: problems in which the objective function can be decomposed as a finite sum of functions:

$$f(x) = \frac{1}{n} \sum_{j=1}^{n} f_j(x), \tag{2.26}$$

with $f_j : \mathbb{R}^d \to \mathbb{R}$.

This decomposition of the objective function is ubiquitous in machine learning problems, in which the loss function is usually the finite sum of the losses in each data point of the dataset (in this case, there are n data points).

A classic technique for decreasing the variance is the mini-batching technique, which is specially useful when it is possible to obtain multiple gradients in parallel. The update rule in this procedure is:

$$x_{i+1} = x_i - \gamma \frac{1}{|B_i|} \sum_{j \in B_i} \nabla f_j(x_i), \tag{2.27}$$

where $B_i \subset \{1, \ldots, n\}$ is a set of random indices, whose size is $|B_i|$. It can be shown that when B_i is sampled uniformly with replacement, the variance of this gradient estimator, $1/|B_i| \sum_{j \in B_i} \nabla f_j(x_i)$, is inversely proportional to $|B_i|$, *i.e.*, $O\left(1/|B_i|\right)$. However, the cost of this iteration is proportional to $|B_i|$, therefore, this variance reduction comes at an extra computational cost.

An alternative and better approach can be used in the finite sum setting [20]: instead of using one (or more) $\nabla f_j(x_i)$ directly as an approximation of $\nabla f(x_i)$, use them to update an estimate $G_i \in \mathbb{R}^n$ of the gradient such that $G_i \approx \nabla f(x_i)$, but ensuring that $\mathbb{E}\left[\|G_i - \nabla f(x_i)\|^2\right] \overset{i \to \infty}{\to} 0$. These ideas can be traced back to 2007 [44] and allow for the application of techniques that lead to a drastic variance reduction of the estimator of the gradient of the objective in a given point. These are called the VR methods and, surprisingly, it turns out that by using them it is possible to dodge the asymptotic optimal rates of the general stochastic case. These algorithms are able to recover the asymptotic results of the deterministic GD (with worse constants, as expected), closing the gap which existed between that setting and the stochastic general case (see column VR, Table 2.1). Furthermore, regarding the total runtime (given by the product of the iteration complexity and the iteration cost) the VR methods are still advantageous in the finite sum setting for large datasets (large n), as the cost of a GD iteration is $O(n)$ (it has to scan the whole dataset for each step), whereas the VR algorithms cost is $O(1)$ (it only evaluates one instance of the dataset to update G_i). Nevertheless, in turn, the VR methods usually require a significant memory overhead.

In order to better illustrate the perspective often found in these approaches, we present Stochastic

Average Gradient (SAG) [12], the method which actually introduced the VR scenario in the first-order stochastic convex optimization. As synthesized in Algorithm 2.4, SAG maintains an estimate $v_i^j \approx \nabla f_j(x_i)$ for each dataset example j, whose average constitutes the estimate of the full gradient. At each iteration, SAG samples $k_i \in \{1, \ldots, n\}$ and the mentioned estimates are iteratively updated:

$$v_{i+1}^j = \begin{cases} v_i^j & \text{if } j \neq k_i \\ \nabla f_{k_i}(x_i) & \text{if } j = k_i \,. \end{cases}$$

Based on this, G_i can be efficiently updated:

$$\begin{aligned} G_i &= \frac{1}{n} \sum_{j=1}^{n} v_i^j \\ &= \frac{1}{n} \sum_{j=1, j \neq k_i}^{n} v_i^j + \frac{1}{n} v_i^{k_i} \\ &= G_{i-1} - \frac{1}{n} v_{i-1}^{k_i} + \frac{1}{n} \nabla f_{k_i}(x_i) \,. \end{aligned}$$

Algorithm 2.4: Stochastic Average Gradient (SAG)

Input: Initial point: x_0; Number of steps: T; Step size: γ; Gradient estimates: $v_0^j = 0 \in \mathbb{R}^n,\ j = 1, \ldots, n$.

for $i = 1, \ldots, T$ **do**

- Sample $k_i \in \{1, \ldots, n\}$
- $G_i = G_{i-1} - \frac{1}{n} v^{k_i}$
- $v^{k_i} = \nabla f_{k_i}(x_i)$
- $G_i = G_i + \frac{1}{n} v^{k_i}$
- $x_i = x_{i-1} - \gamma\, G_i$

Output: Final Point: x_T.

Note that SAG needs to store the gradient estimates v_i^j. Recalling that each each gradient estimate is a d-dimensional vector, SAG requires an $O(nd)$ memory overhead, which easily becomes infeasible when n or d are large. A method which addresses this limitation is, for example, the Stochastic Variance-Reduced Gradient (SVRG) [45], as it only requires a $O(d)$ storage. Moreover, the proof of convergence of SAG is extremely elaborated, even including some computer-aided steps. In turn, Stochastic Average Gradient "amelioré" (SAGA) [46] curbs this by, contrarily to SAG, resorting to an unbiased estimate of the gradient (G_i) based on control variates. In fact, many other alternatives and improvements on this topic have been developed afterwards, *e.g.*, Stochastic Dual Coordinate Ascent (SDCA) [47], Semi-Stochastic Gradient Descent (S2GD) [48], Minimization by Incremental Surrogate Optimization (MISO) [49], and Finito [50].

A final note is left on the rate obtained for VR methods in Table 2.1. For the strongly-convex case,

$$k_{\max} := \frac{\max_j L_j}{\mu} \geq k,$$

where L_j denotes the Lipschitz constant of the gradient of f_j(x). This inequality implies that the iteration complexity of GD is necessarily better than the one of VR methods (regarding the constants in Table 2.1, n is always the total number of points in the dataset).

2.4 Accelerated Algorithms

The sub-optimality of GD in the deterministic setting motivates the concept of acceleration as a possible improvement to the asymptotic rates obtained for VR methods. Taking this into consideration, first, we introduce the Nesterov Acceleration - a well-known technique which allowed the closing of the sub-optimality gap for the first-order family of algorithms and one of the most astonishing results in optimization; then, we describe the appropriate extrapolation of this result to the stochastic setting.

2.4.1 Nesterov Acceleration

Recalling the deterministic setting of convex optimization, it is well known that GD has some trouble in minimizing objective functions that present "ravines", *i.e.*, regions where the difference between curvatures of different dimensions is significant, which in optimization are common near local optima of the objective functions[2]. In those cases, while the convergence for the dimensions of higher curvature is quickly achieved, on the ones of lower curvature, it progresses very slowly [51]. In the attempt of fastening the convergence in lower curvature, it is natural to increase the step size. However, this leads to oscillations on the higher curvature dimensions [52], as illustrated in the left image of Figure 2.1.

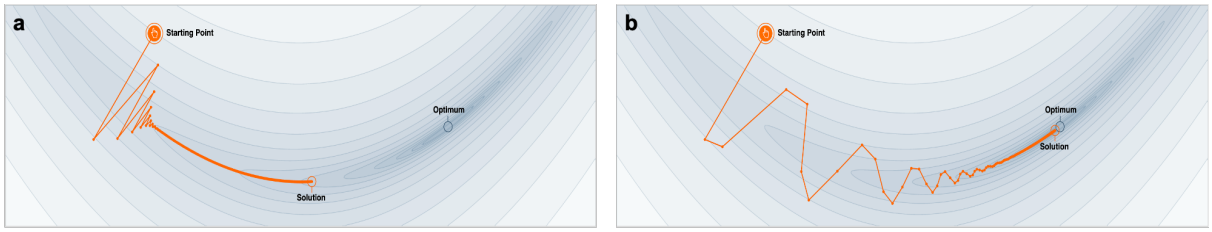


Figure 2.1: Differences between classical GD (a) and the Heavy-Ball method (b). Although the starting point and the number of iterations is the same for both, the Heavy-Ball ends up much closer to the optimal point. [3]

To prevent these oscillations, a reasonable strategy is to try to consult the gradients "history" in order to predict that the just described behavior will arise. In fact, that is what Polyak's Momentum algorithm

[2]Note that for the specific case of convex optimization, there is only one local optimum, which is also the global minimum.
[3]These images were produced using the website in reference [51].

(also known as Heavy-Ball method) achieves. Its name is assigned to the clear inspiration of the physical phenomenon of a ball rolling down a hill: the ball accumulates momentum along the descent as it moves faster and faster. Then, the update rule for the Polyak's momentum algorithm is [10]:

$$x_{i+1} = x_i - \gamma \nabla f(x_i) + \eta(x_i - x_{i-1}), \qquad \eta \in [0, 1],\ \gamma > 0\,.$$

In the inertial term added, η can be intuitively seen as the mass of the rolling ball. In fact, η is not limited to the interval referred above; that interval is only required to achieve a damping effect (*e.g* in the example of a ball, the dissipative effects caused by air resistance, friction, etc.). This updating rule can be decomposed into two different rules, one for the momentum itself and the other for the next position, as in Algorithm 2.5, where the momentum term is explicitly computed. This momentum vector, m_i, is the term that aggregates the information related to the "history" of the gradient, since $m_i \in \text{span}\{\nabla f(x_0), \ldots, \nabla f(x_i)\}$.

Algorithm 2.5: Polyak's Momentum/Heavy-Ball Algorithm

Input: Initial point: x_0; Number of steps: n; Step size: γ; Momentum constant: η.
$m_0 = 0$
for $i = 1, \ldots, n$ **do**
  $m_i = \gamma\ \nabla f(x_{i-1}) + \eta\ m_{i-1}$
  $x_i = x_{i-1} - m_i$
Output: Final point: x_n.

The momentum term increases for the dimensions whose gradients always point in the same directions, showing a cumulative effect, while for dimensions whose gradients often change directions, they end up cancelling each other, reducing its momentum component. Consequently, this inertial term leads to a decrease in oscillations and, thus, a faster convergence, as the same rate of convergence is achieved for all curvatures [53]. These effects can be observed in the right image of Figure 2.1.

Note that the Heavy-Ball method is not guaranteed to converge for all convex cases. In fact, it fails for relatively simple examples of convex functions, under a given choice of hyperparameters [54]. In response to this limitation, Nesterov took advantage of the ideas conveyed by the Heavy-Ball method in order to create a slightly different algorithm: instead of computing the gradient at the point at which the iterate is before applying the momentum, it is computed at the point that would result from adding the momentum component. This method is usually known by Nesterov Accelerated Gradient (NAG) and is illustrated in Figure 2.2 and detailed in Algorithm 2.6. This anticipatory update increases the "responsiveness" of the iterates as we have a smarter version of the rolling ball: intuitively, it now has notion of where it is going to and, so, manages to slow down before progressing in undesirable directions [55].

This algorithm was originally analysed by Nesterov under a proper initialization regarding the step size and momentum constant. In particular, the step size is defined as a function of the Lipschitz

Figure 2.2: Updates performed by the Heavy-Ball method (on the left) and NAG (on the right). Naturally, the gradient (orange arrow) is always orthogonal to the level set. The main distinction is that Heavy-Ball computes the gradient at x_1, whereas NAG computes it at the point that results from applying momentum (red arrow) to x_1, *i.e.*, at the red arrowhead. The blue arrow is the resulting step. (Illustration from [1]).

Algorithm 2.6: Nesterov Accelerated Gradient Descent (NAG)

Input: Initial point: x_0; Number of steps: n; Step size: $\gamma = \frac{1}{L}$.

$\lambda_0 = 0$

$m_0 = 0$

for $i = 1, \ldots, n$ **do**

$\lambda_i = \frac{1+\sqrt{1+\lambda_{i-1}^2}}{2}$

$\eta_i = \frac{\lambda_{i-1}-1}{\lambda_i}$

$m_i = \gamma \nabla f(x_{i-1} - \eta_i \, m_{i-1}) + \eta_i \, m_{i-1}$

$x_i = x_{i-1} - m_i$

Output: Final point: x_n.

constant of an L-smooth function, making that definition of the hyperparamenters only applicable in settings where that assumption is considered. Nevertheless, for the general convex case, NAG can still be shown to converge for an appropriate fixed step size, thus overcoming the main limitation of the Heavy-Ball method. The convergence of NAG for general convex case is $O\left(1/\sqrt{i}\right)$ [13], matching the asymptotically the optimal convergence rate for that setting [13]. It is worth mentioning that GD also matches that rate for the general convex case.

Therefore, the great improvement of NAG comes from its analysis in L-smooth and strongly convex functions. In fact, the reason behind the initialization of Algorithm 2.6 is its role in the analysis proposed by Nesterov, in 1983, in which that method is shown to achieve a convergence rate of $O\left(1/i^2\right)$ for L-smooth functions [15], matching the optimal rate achievable for convex optimization solution methods of that class and in which the iterates verify $x_i \in x_0 + \text{span}\left\{\nabla f(x_0), \ldots, \nabla f(x_{i-1})\right\}$ [13]. Thus, noting that GD only achieves a convergence rate $O\left(1/i\right)$, NAG closes the gap generated by the sub-optimality of the former. Moreover, when the strongly convex assumption is also considered, this method matches the optimal rate of $O\left(\exp(-i/\sqrt{k})\right)$, reinforcing the sub-optimality of GD - $O\left(\exp(-i/k)\right)$ - also in this

setting.

As a final note, it is also worth mentioning that the Heavy-Ball method matches the optimal rates with guaranteed convergence but only for specific instances of the convex setting. For example, if the objective function is twice continuously differentiable, strongly convex and L-smooth, the Heavy-Ball method has even better convergence factors than NAG. However, that convergence becomes compromised when the objective function is not twice differentiable [56], which is the setting that the example from reference [54] considers. Therefore, the Heavy-Ball method and NAG, along with the conjugate gradient descent method (not mentioned here, but which reaches the optimal rate at the same contexts as the Heavy-Ball method) are usually called the fast gradient methods as a consequence of the improvements provided over GD.

2.4.2 Stochastic Accelerated Algorithms

Given the improvements provided by NAG to the deterministic setting it is reasonable to think of those approaches in order to leverage the algorithms in the stochastic setting. Even though some attempts were performed by plugging it in the raw SGD, those revealed to lead to error accumulation and not to yield accelerated methods [34].

Meanwhile, bounds for the convergence rate for finite sums in the stochastic setting were proven [16, 17], as indicated in the column ACC of Table 2.1. On the one hand, under the assumption of the L-smoothness of the objective function, the NAG convergence rate is recovered. On the other hand, for the strongly convex setting, the optimal rate found is worse than the one that classical acceleration provided in the deterministic setting, but still not caught by the family of stochastic VR algorithms.

Nevertheless, this sub-optimality gap was readily closed by a new wave of algorithms that actually had started appearing before the optimal bounds were proven. Essentially, this new family of algorithms takes advantage of an inner-outer loop first proposed in an accelerated version of SDCA [18]. This approach had developments in, for example, Approximate Proximal Point Algorithm (APPA) [57], Catalyst [58] or Katyusha [34], achieving the desired stochastic accelerated convergence rates. Still, for ill-conditioned settings, these accelerated variants have an evident relevance, as shown by the rates in Table 2.1 for $k_{\max} \gg n$.

3

COCO Denoiser

Contents

This chapter describes our approach. First, we formulate COCO as a Maximum Likelihood estimator constrained by co-coercivity conditions; then, we propose efficient methods to compute its solution. Finally, we study theoretical properties of the estimator.

3.1 Maximum Likelihood Estimation

In convex optimization, a global linear lower bound on the objective function can be easily built, by definition, gathering only local information (the gradient of the function at one point), through the following inequality:

$$f(x) \geq f(y) + \nabla f(y)^T (x - y) \tag{3.1}$$

Recalling the definition of strong-convexity (Definition 1.3.2) and L-smoothness (Definition 1.3.1), by adding a quadratic term to the right-hand side of Equation (3.1), both a stricter lower bound and an upper bound can also be attained, respectively. These results are represented graphically in Figure 3.1.

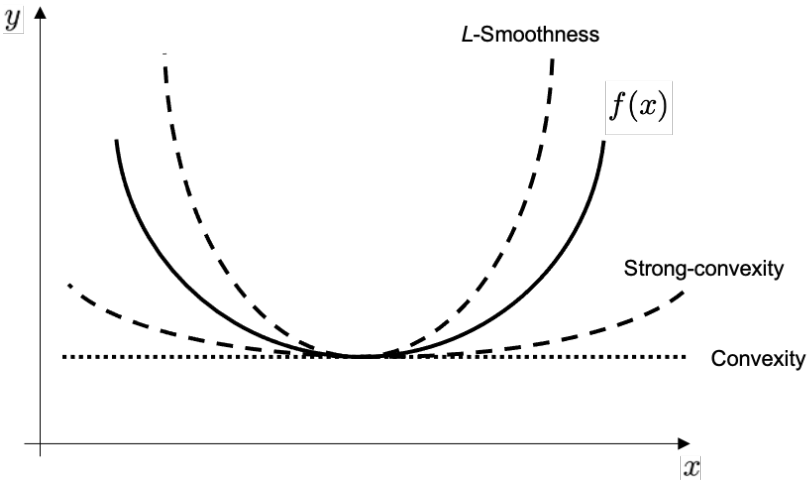


Figure 3.1: Bounds for a convex function $f(x)$: an upper bound via L-smoothness, a linear lower bound via convexity and a stricter lower bound via strong-convexity.

In particular, a standard result in convex analysis is that the gradient of a L-smooth convex function (note that Definition 1.3.1 does not assume convexity) is co-coercive [3], *i.e.*,:

$$\forall x, y \in \mathbb{R}^n,\ L \in \mathbb{R}^+ : \qquad \frac{1}{L} \|\nabla f(y) - \nabla f(x)\|^2 \leq \langle \nabla f(y) - \nabla f(x),\ y - x \rangle . \tag{3.2}$$

Note that Equation (3.2) is stronger than the inequality in L-smoothness definition (Definition 1.3.1), since the inequality in Definition 1.3.1 follows from Equation (3.2). This fact can be easily observed by applying the Cauchy-Schwartz inequality on the right-hand side of Equation (3.2).

In our approach, despite not knowing the objective function f, we assume the following:

Assumption 3.1.1. *We know a Lipschitz constant of its gradient, L.*

This assumption is commonly adopted in stochastic optimization methods as a result of its usefulness in the analysis of those algorithms, without narrowing excessively the universe of possible applications. This imposition prevents the gradients from changing arbitrarily fast from one point to another. Moreover, L-smoothness is usually considered a weaker assumption than strong convexity [20]. In fact, as a consequence of their high-dimensionality, typical machine learning problems have correlated variables, yielding non-strongly convex objective functions (*i.e.*, $\mu \approx 0$) [32]. By additionally considering that estimating L is often easier than estimating μ, we emphasize the pertinence of this assumption.

Assumption 3.1.2. *We have access to an oracle which, given an input $x \in \mathbb{R}^d$, outputs a noisy version of the gradient of f at x; specifically, we assume that the oracle outputs $g(x; w) = \nabla f(x) + w$, where $w \in \mathbb{R}^d$ is a sample of a Gaussian distribution with zero mean, i.e., $w \sim \mathcal{N}(0, \Sigma)$, where the covariance matrix Σ is assumed to be known. Moreover, we assume the noise samples are independent across the oracle consultations.*

The motivation for the noise model comes, naturally, from the simplicity that it provides to our method. Moreover, the Central Limit Theorem states that the sum (or mean) of a given number of independent and identically distributed random variables with finite variances will tend to a normal distribution as the number of variables grows. Note that in machine learning, the mini-batch scheme is a common procedure to obtain gradient estimates at a point and its gradient estimator is defined as a mean of independent random variables (recall Equation (2.27)). This reasoning reinforces the relevance of this assumption.

We assume that the oracle was consulted at the input points $x_1, \ldots, x_K$, and returned the outputs $g_1, \ldots, g_K$. This is our available data, which we arrange in the vector

$$g = \begin{bmatrix} g_1 \\ \vdots \\ g_K \end{bmatrix} \in \mathbb{R}^{Kd}.$$

We address the problem of estimating the gradients $\nabla f(x_1), \ldots, \nabla f(x_K)$ from the available data. Thus, the parameter we're interested in estimating is

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_K \end{bmatrix} \in \mathbb{R}^{Kd}, \tag{3.3}$$

where $\theta_k = \nabla f(x_k) \in \mathbb{R}^d$.

From Assumption 3.1.2, the available data g is related to the parameter of interest θ by the observation model

$$g = \theta + w, \tag{3.4}$$

where

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix} \in \mathbb{R}^{Kd}$$

is distributed as $w \sim \mathcal{N}(0,\, \Sigma_w)$. Note that Σ_w is a block-diagonal matrix, each block being Σ.

We also have the following information about the parameter θ, which comes from the co-coercivity condition (Equation (3.2)):

$$\theta \in \Theta = \{(\theta_1, \ldots, \theta_K) : \frac{1}{L}\|\theta_m - \theta_l\|^2 \leq \langle \theta_m - \theta_l,\ x_m - x_l \rangle, 1 \leq m < l \leq K\}\,.$$

The Maximum Likelihood (ML) estimate of θ is

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmax}} \quad p(g|\theta)\,,$$

where, in accordance to our observation model in Equation (3.4),

$$p(g|\theta) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(g-\theta)^T \Sigma_w^{-1}(g-\theta)}\,.$$

Consequently, it is immediate that computing our solution $\hat{\theta}$ corresponds to solving the following optimization problem:

$$\begin{aligned} \underset{\theta_1,\ldots,\theta_K}{\text{minimize}} \quad & \sum_{k=1}^{K} (g_k - \theta_k)^T \Sigma^{-1} (g_k - \theta_k) \\ \text{subject to} \quad & \frac{1}{L}\|\theta_m - \theta_l\|^2 \leq \langle \theta_m - \theta_l, x_m - x_l \rangle, \quad 1 \leq m < l \leq K. \end{aligned} \tag{3.5}$$

In fact, this instantiation of the convex optimization problem can be classified as a Quadratically Constrained Quadratic Problem (QCQP) [3], since both the objective and the constraints are quadratic functions. In this type of problem, the objective function (quadratic) is minimized over a feasible region that results from the intersection of ellipsoids. Despite their ubiquity in many engineering and scientific applications, solving a generic nonconvex QCQP problem is NP-hard [59]. Nevertheless, for convex instances of those problems, it is possible to explore the structure of the problem and attain a tractable solution method. This is the case of the problem considered and the methods proposed are discussed in Section 3.2.

The main idea supporting our approach is the observation that all the methods mentioned in Chap-

ter 2, even though often assuming the L-smoothness of the objective function for their analyses, do not take advantage of it in their methodology. In fact, after taking this assumption into consideration, it is unsatisfactory to blindly accept the gradients that the oracle is outputting. It is on the demand of making those observations coherent with this assumption that our approach is based on. Since the co-coercivity constraints play the pivotal role of merging two important conditions (convexity and L-smoothness of the objective function) into only one expression, we call our method the Co-coercivity (COCO) denoiser.

The number of constraints in this approach scales quadratically with K, the number of consulted points. More precisely, the number of constraints is given by $K(K-1)/2$, as each constraint is imposed between every two points from the ones considered. This drawback motivates a simplification of the original COCO denoiser: instead of considering all the consulted points across all the i iterations, we fix a given number of points, K ($1 \leq K \leq i$), and only consider the information belonging to the last K points to denoise the consulted gradients. For example, if we fix $K = 2$, the denoise is performed only considering x_i, x_{i-1}, g_i and g_{i-1}. To this denoiser using a fixed window of length K, we call COCO$_K$.

3.2 Efficient Solutions for COCO$_K$

In this section, a solution method is proposed for the QCQP raised by COCO$_K$. We start by providing closed-form solutions for $K = 1$ and $K = 2$ and then propose an efficient iterative algorithm which yields an approximate solution for arbitrary K.

3.2.1 Closed-form Solution for COCO$_2$

Given the initial convex minimization problem expressed in Equation (1.1), the Lagrangian for that problem is defined as:

$$L(x, \mu, \upsilon) = f(x) + \sum_{i=1}^{m} \mu_i f_i(x) + \sum_{i=1}^{p} \upsilon_i h_i(x), \tag{3.6}$$

where $\mu = (\mu_1, \ldots, \mu_m)$ and $\upsilon = (\upsilon_1, \ldots, \upsilon_p)$. Consequently, the Lagrange dual function is defined as:

$$g(\mu, \upsilon) = \inf_x L(x, u, v)$$

and the Lagrange dual problem as:

$$\begin{aligned} \underset{\mu,\upsilon}{\text{maximize}} \quad & g(\mu, \upsilon) \\ \text{subject to} \quad & \mu_i \geq 0, \quad i = 1, \ldots, m\,. \end{aligned} \tag{3.7}$$

Whilst x is known as the primal variable, as it is the variable over which we minimize in the primal problem (Equation (1.1)), μ and υ are known as the dual variables as, in the Lagrange dual problem,

these are the variables over which we maximize the objective function $g(\mu, \upsilon)$.

The Karush-Kuhn-Tucker (KKT) conditions are a set of conditions on x, μ and υ which are known to be sufficient to ensure the optimality of those variables, *i.e.*, if we find x^*, μ^* and υ^* which verify the KKT conditions, then those are primal and dual solutions, respectively [60]. The KKT conditions are presented below:

1. **Stationarity**: $0 \in \partial(f(x^*) + \sum_{i=1}^{m} \mu_i f_i(x^*) + \sum_{i=1}^{p} \upsilon_i h_i(x^*))$;
2. **Complementary Slackness**: $\mu_i \cdot f_i(x^*) = 0,$ for $i = 1, \ldots, m$;
3. **Primal Feasibility**: $f_i(x^*) \leq 0,\ h_j(x^*) = 0,$ for $i = 1, \ldots, m$ and $j = 1, \ldots, p$;
4. **Dual Feasibility**: $\mu_i \geq 0,$ for $i = 1, \ldots, m$.

In Condition 1, $\partial(\cdot)$ denotes the subdifferential operator[1].

Using the KKT conditions it is possible to find closed-form expressions for the denoised gradients when $K = 1$ and $K = 2$. Those expressions can be found in Theorem 3.2.1 and Theorem 3.2.2.

Theorem 3.2.1. *The solution to Equation* (3.5) *for* $K = 1$ *is given by:*

$$\hat{\theta}_1 = g_1 \,. \tag{3.8}$$

Proof. It is immediate, since, in this case, the optimization problem in Equation (3.5) becomes unconstrained. In fact, the KKT conditions may not even be considered and the problem is simply

$$\underset{\theta_1}{\text{minimize}} \quad (g_1 - \theta_1)^T \Sigma^{-1} (g_1 - \theta_1) \,,$$

a quadratic function with a positive-definite matrix, whose minimum is trivially obtained: $\hat{\theta}_1 = g_1$. □

Note that the result for Theorem 3.2.1 holds for a generic Σ, while for Theorem 3.2.2 the result is specified under Assumption 3.2.1:

Assumption 3.2.1. *The covariance matrix of the multivariate Gaussian distribution of noise is multiple of the identity matrix, i.e.,* $\Sigma = \sigma^2 I$.

Theorem 3.2.2. *Under Assumption 3.2.1, the solution to Equation* (3.5) *for* $K = 2$ *is given by:*

If $\|g_1 - g_2\|^2 \leq L \langle g_1 - g_2,\ x_1 - x_2 \rangle$*:*

$$\begin{cases} \hat{\theta}_1 = g_1 \\ \hat{\theta}_2 = g_2 \,. \end{cases}$$

[1] For a continuous function $f : \mathbb{R}^d \to \mathbb{R}$, $c \in \mathbb{R}^d$ is a subgradient of f at $x \in \mathbb{R}^d$ if and only if $f(y) - f(x) \geq c^T(y - x)$, with $y \in \mathbb{R}^d$. The set of all the subgradients of f at x is called the subdifferential of f at x, $\partial(f(x))$.

If $\|g_1 - g_2\|^2 > L\,\langle g_1 - g_2,\, x_1 - x_2\rangle$*:*

$$\begin{cases} \hat{\theta}_1 = \frac{g_1+g_2+\frac{L}{2}(x_1-x_2)}{2} + \|\frac{L}{4}(x_1-x_2)\|\frac{g_1-g_2-\frac{L}{2}(x_1-x_2)}{\|g_1-g_2-\frac{L}{2}(x_1-x_2)\|} \\ \hat{\theta}_2 = \frac{g_1+g_2-\frac{L}{2}(x_1-x_2)}{2} - \|\frac{L}{4}(x_1-x_2)\|\frac{g_1-g_2-\frac{L}{2}(x_1-x_2)}{\|g_1-g_2-\frac{L}{2}(x_1-x_2)\|} \end{cases}.$$

Proof. For $K = 2$, the problem can be formalized in the following way:

$$\begin{aligned} \underset{\theta_1,\theta_2}{\text{minimize}} \quad & (g_1-\theta_1)^T\Sigma^{-1}(g_1-\theta_1) + (g_2-\theta_2)^T\Sigma^{-1}(g_2-\theta_2) \\ \text{subject to} \quad & \|\theta_1-\theta_2\|^2 - L\langle\theta_1-\theta_2,\, x_1-x_2\rangle \leq 0\,. \end{aligned}$$

In order to solve this problem, the KKT conditions will now be used. It can be observed that there are no equality constraints, then no $h_i(x)$ or dual variables v_i need to be considered. Moreover, we have:

$$\begin{aligned} f(\theta_1,\theta_2) &= (g_1-\theta_1)^T\Sigma^{-1}(g_1-\theta_1) + (g_2-\theta_2)^T\Sigma^{-1}(g_2-\theta_2) \\ f_1(\theta_1,\theta_2) &= \|\theta_1-\theta_2\|^2 - L\langle\theta_1-\theta_2,\, x_1-x_2\rangle\,. \end{aligned}$$

Since both functions are differentiable and convex, we can use $\partial(f(x)) = \{\nabla f(x)\}$. This can be applied for simplification of the stationarity condition, through the linearity of the gradient operator. Therefore, the KKT conditions yield the following system of equations:

$$\begin{cases} 2\Sigma^{-1}(\hat{\theta}_1-g_1) + \mu_1\left[2(\hat{\theta}_1-\hat{\theta}_2) - L(x_1-x_2)\right] = 0 & [\ \textit{i.}\ \text{Stationarity in order to } \hat{\theta}_1\ ] \\ 2\Sigma^{-1}(\hat{\theta}_2-g_2) - \mu_1\left[2(\hat{\theta}_1-\hat{\theta}_2) - L(x_1-x_2)\right] = 0 & [\ \textit{ii.}\ \text{Stationarity in order to } \hat{\theta}_2\ ] \\ \mu_1\ \left(\|\hat{\theta}_1-\hat{\theta}_2\|^2 - L\langle\hat{\theta}_1-\hat{\theta}_2,\, x_1-x_2\rangle\right) = 0 & [\ \textit{iii.}\ \text{Complementary Slackness}\ ] \\ \|\hat{\theta}_1-\hat{\theta}_2\|^2 - L\langle\hat{\theta}_1-\hat{\theta}_2,\, x_1-x_2\rangle \leq 0 & [\ \textit{iv.}\ \text{Primal Feasibility}\ ] \\ \mu_1 \geq 0 & [\ \textit{v.}\ \text{Dual Feasibility}\ ]\,. \end{cases}$$

From *iii.*, two cases must be considered:

- $\boldsymbol{\mu_1 = 0}$: In this case, from complementary slackness (*iii.*), $\|\hat{\theta}_1-\hat{\theta}_2\|^2 < L\langle\hat{\theta}_1-\hat{\theta}_2,\, x_1-x_2\rangle$

In that case, from *i.* and *ii.*, it is easy to conclude that $\hat{\theta}_1 = g_1$ and $\hat{\theta}_2 = g_2$. Therefore, we note that this happen when $\|g_1-g_2\|^2 < L\langle g_1-g_2,\, x_1-x_2\rangle$

- $\boldsymbol{\mu_1 > 0}$: In that case, from complementary slackness (*iii.*), $\|\hat{\theta}_1-\hat{\theta}_2\|^2 = L\langle\hat{\theta}_1-\hat{\theta}_2,\, x_1-x_2\rangle$.

By summing *i.* and *ii.*:

$$\hat{\theta}_1 + \hat{\theta}_2 = g_1 + g_2\,.$$

This equality is particularly interesting and further discussed in Section 3.3. By replacing it in *i.*

and *ii.*, we obtain:

$$\begin{aligned}\hat{\theta}_1 &= (\Sigma^{-1} + 2\mu_1 I)^{-1}[(\Sigma^{-1} + \mu_1 I)g_1 + \mu_1 g_2 + \mu_1 \frac{L}{2}(x_1 - x_2)] \\ \hat{\theta}_2 &= (\Sigma^{-1} + 2\mu_1 I)^{-1}[\mu_1 g_1 + (\Sigma^{-1} + \mu_1 I)g_2 - \mu_1 \frac{L}{2}(x_1 - x_2)]\,.\end{aligned} \tag{3.9}$$

Then, by replacing those results in $\|\hat{\theta}_1 - \hat{\theta}_2\|^2 = L\langle\hat{\theta}_1 - \hat{\theta}_2,\ x_1 - x_2\rangle$, it yields the following expression:

$$I\mu_1^2 + \Sigma^{-1}\mu_1 - (\Sigma^{-1})^2 C = 0, \tag{3.10}$$

with $C = (\|g_1 - g_2\|^2 - L\langle g_1 - g_2,\ x_1 - x_2\rangle)/(L^2\|x_1 - x_2\|^2)$. Note that $C \geq 0$, since, otherwise, we would have $\|g_1 - g_2\|^2 < L\langle g_1 - g_2,\ x_1 - x_2\rangle$ and we would be in the case of $\mu_1 = 0$. Imposing Assumption 3.2.1 in the equation above, it yields for each diagonal entry:

$$\mu_1^2 + \frac{1}{\sigma^2}\mu_1 - (\frac{1}{\sigma^2})^2 C = 0\,. \tag{3.11}$$

Note that the non-diagonal entries are not informative, as they are all zero. The only root from Equation (3.11) that respects dual feasibility (*v.*) is:

$$\mu_1 = \frac{1}{\sigma^2}\left(\frac{-1 + \sqrt{1 + 4C}}{2}\right).$$

Replacing this value of μ_1 in Equation (3.9), we obtain the intended result. □

The two cases in which we decompose the closed-form solution for COCO$_2$ have an intuitive explanation supporting them: when the two observed gradients are co-coercive ($\|g_1 - g_2\|^2 \leq L\langle g_1 - g_2,\ x_1 - x_2\rangle$), they are on the feasible set of the problem, so, they are also the estimated gradients; when the two observed gradients are not co-coercive ($\|g_1 - g_2\|^2 > L\langle g_1 - g_2,\ x_1 - x_2\rangle$), their difference is orthogonally projected onto its feasible set (which is a ball, as shown in the following section). This projection is achieved through the expression obtained for each of the estimated gradients in Theorem 3.2.2. This interpretation is illustrated in Figure 3.2.

Our closed-form solution is of the utmost relevance, since the experiments described in the following chapter show that COCO leads to significant improvements in stochastic optimization, even for the simple case that considers only two gradients.

3.2.2 Fast Dual Proximal Gradient Method for Arbitrary K

Since the closed-form solution for COCO$_K$ for $K \geq 3$ could not be found (even with the help of symbolic manipulation packages of *Matlab* and *Mathematica*), a straightforward procedure to solve the QCQP in Equation (3.5) is its implementation onto *CVX* [21]. This tool is designed in such a way that it can be

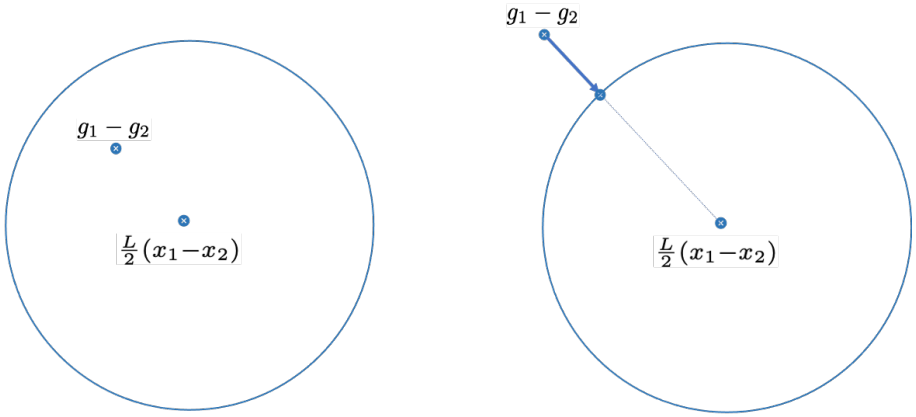


Figure 3.2: Closed-form solution for COCO$_2$. Left: the observed gradients are co-coercive (their difference is on the feasible set), so the estimate coincides with the observation. Right: the observed gradients are not co-coercive (their difference is outside the feasible set), and the estimated gradients are obtained by an orthogonal projection.

used as black-box, in the sense that the user does not need to understand how the problem can be solved, as far as that problem is presented in the format required by the software. As a consequence of its generality, this tool resorts to higher order methods (*e.g.*, second-order cone programming methods) which ensure high precision but necessarily end up being slower than methods which are specifically tailored for a given problem. This fact combined with the quadratic growth of the number of constraints with the K motivates an alternative approach. Therefore, in this section, a first-order algorithm which explores the particular structure of the problem in Equation (3.5) is presented.

Reformulation of the Problem

For $\Sigma = \sigma^2 I$, the QCQP in Equation (3.5) can be rewritten in the following way:

$$\begin{aligned} \underset{\theta_1,\ldots,\theta_K}{\text{minimize}} \quad & \frac{1}{2}\sum_{k=1}^{K} \|g_k - \theta_k\|^2 \\ \text{subject to} \quad & \|\theta_m - \theta_l - \frac{L}{2}(x_m - x_l)\| \leq \frac{L}{2}\|x_m - x_l\|, \quad 1 \leq m < l \leq K, \end{aligned}$$

where the factor $1/2$ replaces the original $1/\sigma^2$ for the sake of simplicity in the next steps and the new form of the constraints is given by completing the square in the expression from the original formulation

(Equation (3.5)):

$$
\begin{aligned}
&\frac{1}{L}\|\theta_m - \theta_l\|^2 \leq (\theta_m - \theta_l)^T (x_m - x_l) \\
\Leftrightarrow\ &\|\theta_m - \theta_l\|^2 - L(\theta_m - \theta_l)^T (x_m - x_l) + \frac{L}{4}\|x_m - x_l\|^2 - \frac{L}{4}\|x_m - x_l\|^2 \leq 0 \\
\Leftrightarrow\ &\|\theta_m - \theta_l - \frac{L}{2}(x_m - x_l)\|^2 \leq \|\frac{L}{2}(x_m - x_l)\|^2 \\
\Leftrightarrow\ &\|\theta_m - \theta_l - \frac{L}{2}(x_m - x_l)\| \leq \|\frac{L}{2}(x_m - x_l)\|,
\end{aligned}
\tag{3.12}
$$

where in Equation (3.12) we add and subtract $L\|x_m - x_l\|^2/4$ and all the other steps are simple manipulations. Note that, in this case, $\theta_m - \theta_l \in \mathcal{B}\left(L(x_m - x_l)/2,\ L\,\|x_m - x_l\|/2\right)$[2].

Now, performing the change of variables $\alpha_k = \theta_k - g_k$, the problem becomes:

$$
\begin{aligned}
\underset{\alpha_1,\ldots,\alpha_K}{\text{minimize}} \quad & \frac{1}{2}\sum_{k=1}^{K}\|\alpha_k\|^2 \\
\text{subject to} \quad & \|\alpha_m - \alpha_l + c_{ml}\| \leq r_{ml}, \quad 1 \leq m < l \leq K,
\end{aligned}
$$

where $c_{ml} = (g_m - (L/2)\,x_m) - (g_l - (L/2)\,x_l)$ and $r_{ml} = L\|x_m - x_l\|/2$.

The indicator function can be defined as

$$
\mathbf{1}_E(x) = \begin{cases} 0 & \text{if } x \in E \\ \infty & \text{if } x \notin E\,. \end{cases}
$$

Using this definition, the primal problem can be finally formulated as

$$
\underset{\alpha}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|\alpha\|^2}_{p(\alpha)} + \underbrace{\mathbf{1}_{\mathcal{B}}(A\alpha + c)}_{q(A\alpha)},
$$

where $\alpha = [\alpha_1,\ \alpha_2,\ \ldots,\ \alpha_K]^T$, $A\alpha = [\alpha_1 - \alpha_2,\ \alpha_1 - \alpha_3,\ \ldots,\ \alpha_1 - \alpha_K,\ \alpha_2 - \alpha_3,\ \ldots,\ \alpha_{K-1} - \alpha_K]^T$, $c = [c_{12},\ c_{13},\ \ldots,\ c_{1K},\ c_{23},\ \ldots,\ c_{K-1K}]^T$ and $\mathcal{B} = \mathcal{B}(0,\ r_{12}) \times \mathcal{B}(0,\ r_{13}) \times \ldots \times \mathcal{B}(0,\ r_{1K}) \times \mathcal{B}(0,\ r_{23}) \times \ldots \times \mathcal{B}(0,\ r_{K-1K})$.

In this formulation, we want to minimize the sum of two convex functions, where the first is differentiable and the second is non-differenttiable, but still closed[3]. This is the setup to which the Iterative Shrinkage-Thresholding Algorithms (ISTA) [61] are designed for. In particular, when the non-differentiable function is a simple indicator function, that method can be interpreted as Projected Gradient Descent. However, in this formulation, that function is composed with a linear map A, case in which there is no closed-form for the proximity operator.

[2]The notation $\mathcal{B}(c, r)$ denotes the set of points within a ball centered at c and of radius r, *i.e.*, $\mathcal{B}(c, r) = \{x \in \mathbb{R}^n : \|x - c\| \leq r\}$.
[3]A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be closed if for each $\alpha \in \mathbb{R}$, the sublevel set $\{x \in \text{dom} f | f(x) \leq a\}$ is a closed set.

Given this, a reformulation using Lagrange duality is used. First, the problem can be rewritten as:

$$\begin{aligned} \underset{\alpha,\beta}{\text{minimize}} \quad & p(\alpha) + q(\beta) \\ \text{subject to} \quad & A\alpha = \beta\,. \end{aligned}$$

It is possible to write the Lagrangian (recall its definition in Equation (3.6)) for the reformulated problem:

$$\begin{aligned} L(\alpha, \beta, s) \quad &= p(\alpha) + q(\beta) + s^T(A\alpha - \beta) \\ &= p(\alpha) + s^T A\alpha + q(\beta) - s^T\beta\,. \end{aligned}$$

The Lagrange dual function can be computed:

$$\begin{aligned} L(s) \quad &= \inf_{\alpha,\beta} L(\alpha,\ \beta,\ s) \\ &= \inf_{\alpha} \ \big(p(\alpha) + s^T A\alpha\big) + \inf_{\beta} \ \big(q(\beta) - s^T\beta\big)\ . \end{aligned}$$

Thus,

$$-\,L(s) \quad = \sup_{\alpha} \ \big((-A^T s)^T\alpha - p(\alpha)\big) + \sup_{\beta} \ \big(s^T\beta - q(\beta)\big)\ .$$

By definition, for a generic function, its (Fenchel) conjugate is defined as $f^*(s) = \sup_x (s^T x - f(x))$. Therefore, it is possible to conclude that:

$$-L(s) = p^*(-A^T s) + q^*(s)\,.$$

It remains to obtain the specific form of $p^*(s)$ and $q^*(s)$. Regarding the former:

$$\begin{aligned} p^*(s) \quad &= \sup_{\alpha} \ (s^T\alpha - \frac{1}{2}\|\alpha\|^2) \\ &= \frac{1}{2}\|s\|^2, \end{aligned}$$

where the second equality easily comes from differentiating $s^T\alpha - 1/2\ \|\alpha\|^2$ with respect to α and equating to zero. Therefore, the value obtained for α is then replaced on the original expression.

Regarding q^*:

$$\begin{aligned} q^*(s) \quad &= \sup_{\beta} \ \big(s^T\beta - \mathbf{1}_{\mathcal{B}}(\beta + c)\big) \\ &= \sup_{\beta} \ \big\{s^T\beta : \beta + c \in \mathcal{B}\big\} \\ &= \sup_{\beta} \ \left\{ \sum_{1\le m<l\le K} s_{ml}^T\beta_{ml} : \beta_{ml} + c_{ml} \in \mathcal{B}(0, r_{ml}) \right\} \\ &= \sum_{1\le m<l\le K} \ \sup_{\beta} \ \big\{s_{ml}^T\beta_{ml} : \|\beta_{ml} + c_{ml}\| \le r_{ml}\big\} \\ &= \sum_{1\le m<l\le K} r_{ml}\|s_{ml}\| - s_{ml}^T c_{ml}\,. \end{aligned}$$

Therefore, the minimization problem can be rewritten in the following form:

$$\underset{s}{\text{minimize}} \quad \underbrace{\frac{1}{2}\| - A^T s\|^2}_{p^*(-A^T s)} + \underbrace{\sum_{1 \leq m < l \leq K} r_{ml}\|s_{ml}\| - s_{ml}^T c_{ml}}_{q^*(s)} \,.$$

Note that, at this point, the linear mapping A has been transferred to the differentiable term. This change allows us now to find a closed-form expression for the proximity operator of $q^*(s)$, as the gradient of the first term can be computed even considering the composition with A^T.

Proximity Operator Computation

By definition, the proximity operator of a generic closed, convex function f is:

$$\text{prox}_f(x) = \underset{u}{\text{argmin}} \, \frac{1}{2}\|u - x\|^2 + f(u) \,.$$

We are interested in obtaining $\text{prox}_{\mu q^*}(s)$, for any given $\mu > 0$. Thus:

$$\begin{aligned} \text{prox}_{\mu q^*}(s) &= s - \text{prox}_{(\mu q^*)^*}(s) && (3.13)\\ &= s - \text{prox}_{(q^*)^* \cdot \mu}(s) && (3.14)\\ &= s - \text{prox}_{q \cdot \mu}(s), && (3.15) \end{aligned}$$

where in Equation (3.13) it is applied the well-known Moreau identity $\text{prox}_f(x) = x - \text{prox}_{f^*}(x)$; in Equation (3.14), we used $(\mu f)^*(x) = f^* \cdot \mu \; (x) = \mu \, f^*(x/\mu)$ and, in Equation (3.15), the property $(f^*)^* = f$, which holds for any closed, convex function. Now, note that:

$$\begin{aligned} q \cdot \mu \; (s) &= \mu \; q \left(\frac{s}{\mu} \right) \\ &= \mu \; \mathbf{1}_{\mathcal{B}} \left(\frac{s}{\mu} + c \right) \\ &= \mathbf{1}_{\mathcal{B}} \left(\frac{s}{\mu} + c \right) \,, && (3.16) \end{aligned}$$

since, in Equation (3.16), μ can be dropped as $\mathbf{1}_{\mathcal{B}}$ returns either 0 or ∞.

Therefore,

$$\begin{aligned}
\text{prox}_{\mu q^*}(s) &= s - \underset{u}{\text{argmin}} \left(\frac{1}{2}\|u - s\|^2 + \mathbf{1}_{\mathcal{B}}\left(\frac{u}{\mu} + c\right)\right) \\
&= s - \mu \left(\underset{v}{\text{argmin}} \left(\frac{1}{2}\|\mu(v - c) - s\|^2 + \mathbf{1}_{\mathcal{B}}(v) \right) - c \right) \\
&= s - \mu \left(\underset{v \in \mathcal{B}}{\text{argmin}} \left(\frac{1}{2}\|v - (c + \frac{s}{\mu})\|^2 \right) - c \right) \\
&= s - \mu \left(v_{\text{proj}} - c \right),
\end{aligned} \tag{3.17}$$

where the change of variable $v = \frac{u}{\mu} + c$ was used in Equation (3.17) and the orthogonal projection of $c_{ml} + s_{ml}/\mu$ onto the ball $\mathcal{B}(0,\ r_{ml})$, with $1 \leq m < l \leq K$, is denoted by v_{ml}, whose stacking results in $v_{\text{proj}} = \underset{v \in \mathcal{B}}{\text{argmin}} \|v - (c + s/\mu)\|^2$.

Fast Dual Proximal Gradient Method

Recalling Section 3.2.2, we now have a first term, $p^*(-A^T s)$, differentiable, for which the gradient has a closed-form and a second term, $q^*(s)$, non-differentiable but for which we can compute also a closed-form and inexpensive proximity operator. These are the conditions for which the ISTA can be applied, where the iterates are generated by alternating between following a gradient step of the differentiable function and a proximal step.

The gradient for $p^*(-A^T s)$ can be easily computed: $\nabla_s\ p^*(-A^T s) = AA^T s$. Furthermore, from this expression is straightforward to observe that the first term, $p^*(-A^T s)$, is necessarily L-smooth, with $L = \lambda_{\max}(A)^2$.[4] Given this, not only the optimal step size for a gradient update is known ($\gamma = 1/L$), but also, just as happened with first-order algorithms (for differentiable functions) in the deterministic convex setting, it is possible to accelerate the ISTA resorting to a Nesterov Accelerated Gradient (NAG) similar scheme, *i.e.*, enabling momentum to contribute in the generated iterates. The accelerated version of the ISTA are known as Fast Iterative Shrinkage-Thresholding Algorithms (FISTA) [61]. Moreover, this perspective of applying the FISTA to the dual problem is a well studied technique, known as the Fast Dual Proximal Gradient (FDPG) method [22]. This approach is presented for the specific case of the COCO denoiser in Algorithm 3.1.

Through FDPG, it is possible to find an approximation of the optimal solution of the dual problem, s^*. However, what we are interested in is in recovering the solution of the primal problem, α^*, which, nevertheless, can be easily obtained through: $\alpha^* = -A^T s^*$. Consequently, the recovery of the gradient estimates is attained through: $\hat{\theta}_k = \alpha_k^* + g_k$.

We should state that strong duality, *i.e.*, $p(\alpha^*) + q(A\alpha^*) = -\big(p^*(-A^T s^*) + q(s^*)\big)$, holds for this convex optimization problem. For example, a Slater point can be easily obtained by considering $\theta_k =$

[4] The notation $\lambda_{\max}(A)$ denotes the maximum singular value of matrix A.

Algorithm 3.1: FDPG (applied to COCO Denoiser)

Input: Initial Point: s_0; Number of steps: T; L-Smoothness Constant: L; Momentum Auxiliary Iterate: $y_0 = s_0$; Initial Momentum Constant: $t_0 = 1$

for $i = 1, \ldots, T$ **do**

$s_i = \text{prox}_{\frac{1}{L}q^*}(\, y_{i-1} - \frac{1}{L}\nabla p^*(-A^T y_{i-1}) \,)$

$t_i = \frac{1+\sqrt{1+4t_{i-1}^2}}{2}$

$y_i = s_i + \frac{t_{i-1}-1}{t_i}(s_i - s_{i-1})$

Output: Final Point: s_T

$L/2 \; x_k$, assuming the iterates to be different from each other ($x_i \neq x_j$ if $i \neq j$). This is expectable if we presume that these iterates are generated through a stochastic first-order method.

An important note has to be made regarding the convergence rate of this method: even though the rate of convergence of the dual objective function sequence towards its optimal value is $O\left(1/i^2\right)$ (it is the convergence rate of FISTA [61]), it can be proven that the primal rate of convergence is only $O\left(1/i\right)$ [22]. From this result, it is immediate to verify that the iteration complexity[5] from our method is $O(1/\epsilon)$, while the iteration runtime is $O(K^2 d)$, as this is the cost of computing the proximity operator. These results yield a total algorithmic runtime[6] of $O(K^2 d/\epsilon)$.

Taking this into account, in order to provide a COCO gradient estimation given noisy oracle consultations, two possible solutions are then available for the case in which $\Sigma = \sigma^2 I$: (i) a closed-form solution for $K \leq 2$ and (ii) an iterative first-order algorithm for arbitrary K. Noting that the closed-form COCO gradient estimation has an iteration runtime of $O(d)$ and achieves the solution in 1 iteration, this results in a total runtime of $O(d)$. This puts in perspective the obvious preference for the closed-form procedure when it is available ($K \leq 2$). This statement is reinforced by the memory requirements: while the closed-form solution requires K points and respective gradients (both d-dimensional vectors) to be kept in memory, therefore implying an $O(Kd)$ memory overhead, the memory requirements for the iterative method largely surpasses this value. For example, just to store the auxiliary structured matrix A, there is a $O(K^2 d^2)$ memory overhead.

3.3 Properties of the Estimator

3.3.1 Relation between the Oracle and COCO Estimates

As anticipated in the proof of Theorem 3.2.2, there is an easily interpretable property that relates the sum of the noisy gradients (COCO input) to the sum of the denoised ones (COCO output). It is expressed by the following theorem.

[5] We define iteration complexity as the smallest i verifying the inequality $E\left[f(x_i) - f(x^*)\right] < \epsilon$.

[6] The total runtime of an algorithm is obtained by the product of the iteration complexity and the iteration runtime.

Theorem 3.3.1. *The gradients estimated by the COCO$_K$ denoiser, $\hat{\theta}_1, \ldots, \hat{\theta}_K$, verify the following relation with its raw inputs, $g_1, \ldots, g_K$:*

$$\sum_{i=1}^{K} \hat{\theta}_i = \sum_{i=1}^{K} g_i \,.$$

Proof. From the COCO denoiser formalization (Equation (3.5)), the KKT conditions yield K stationarity equations. Its i-th equation is of the form:

$$2\Sigma^{-1}(\hat{\theta}_i - g_i) + \sum_{j=1, j\neq i}^{K} \mu_l[\, 2(\hat{\theta}_i - \hat{\theta}_j) - L(x_i - x_j)\,] = 0,$$

where the index l is arbitrary, *i.e.*, it refers to the only constraint involving $\hat{\theta}_i$ and $\hat{\theta}_j$.

Summing the K equations, all the constraint terms cancel out pairwisely, yielding:

$$\sum_{i=1}^{K} 2\Sigma^{-1}(\hat{\theta}_i - g_i) = 0 \Leftrightarrow 2\Sigma^{-1} \sum_{i=1}^{K} (\hat{\theta}_i - g_i) = 0 \Leftrightarrow \sum_{i=1}^{K} \hat{\theta}_i = \sum_{i=1}^{K} g_i \,.$$

□

Note that this property holds for generic Σ and not only for $\Sigma = \sigma^2 I$, contrarily to the proof of Theorem 3.2.2. Moreover, by multiplying both sides of the equality from Theorem 3.3.1 by $1/K$, we obtain:

$$\frac{1}{K} \sum_{i=1}^{K} \hat{\theta}_i = \frac{1}{K} \sum_{i=1}^{K} g_i \,,$$

showing that the centroid of the estimated and consulted gradients is the same, reinforcing the interpretability of this relation.

3.3.2 Mean Squared Error

In this section, the notion of Mean Squared Error (MSE) is explored, as it can be used as a performance metric for estimators (the smaller the better). For a given estimator $\hat{Y}$ of an actual random vector Y, the MSE is defined as:

$$\text{MSE}(\hat{Y}) = \mathbb{E}[\|\hat{Y} - Y\|^2] \,.$$

In particular, we are interested in comparing the MSE from the COCO estimator with the one of the oracle itself. It is in this perspective that Theorem 3.3.2 arises:

Theorem 3.3.2. *From Equation* (3.5) *and assuming $\Sigma = \sigma^2 I$, the following inequality holds:*

$$\textit{MSE}(\hat{\theta}) \leq \textit{MSE}(g), \tag{3.18}$$

where $\hat{\theta}$ denotes the stacked vector of the different $\hat{\theta}_k$ outputted from the COCO$_K$ denoiser and g denotes the stacked vector of the different g_k outputted from the oracle.

Proof. The Orthogonal Projection operator on a set S is defined as

$$\begin{aligned} P_S(x): \quad & \mathbb{R}^d \quad \to \mathbb{R}^d \\ & x \quad \mapsto \underset{y \in S}{\operatorname{argmin}} \|x - y\| . \end{aligned}$$

In the case in which $S \subset \mathbb{R}^d$ is closed and convex, the following property holds:

$$\|P_S(a) - P_S(b)\| \leq \|a - b\| .$$

Let us now define ∇f as the vector which results from stacking the different $\nabla f(x_k)$. Note that MSE$(\hat{\theta}) = \mathbb{E}[\|\hat{\theta} - \nabla f\|^2]$ and MSE$(g) = \mathbb{E}[\|g - \nabla f\|^2]$

Let also S be the feasible set of the problem in Equation (3.5). Note that, in that case, S is a convex and closed set as it results from the intersection of ellipsoids, which are convex and closed sets themselves. Moreover, from the definition of the problem, $\hat{\theta} = \underset{\theta \in S}{\operatorname{argmin}} \, 1/\sigma^2 \, \|\theta - g\|^2 = \underset{\theta \in S}{\operatorname{argmin}} \, \|\theta - g\|$, thus $\hat{\theta} = P_S(g)$. Noting that $\nabla f = P_S(\nabla f)$ since $\nabla f \in S$, *i.e.*, the true gradients of an L-smooth and convex function are necessarily co-coercive[7], it follows:

$$\|\hat{\theta} - \nabla f\| = \|P_S(g) - P_S(\nabla f)\| \leq \|g - \nabla f\| . \tag{3.19}$$

Squaring both sides of the inequality in Equation (3.19) and applying the Expectation operator, the result intended is obtained. □

3.3.3 Constraints Tightness

Every constraint included in the COCO$_K$ denoiser problem involves a pair of estimated gradients. Every time that two gradient observations, g_i and g_j are not co-coercive between them (note that to conclude that, it is required to also know x_i and x_j), the solution will have to output two estimated gradients, $\hat{\theta}_i$ and $\hat{\theta}_j$ which respect that condition and, thus, necessarily different from g_i and g_j. Noting that g_i and g_j are random variables, an important question to answer is: how often are the observed gradients incoherent with the co-coercivity constraint?

To answer this question, we focus on the one-dimensional situation ($d = 1$), for simplicity. We have access to two different points, x_1 and x_2. Without loss of generality, let us assume $x_1 > x_2$. The true gradients on those points are $\nabla f(x_1)$ and $\nabla f(x_2)$, whose noisy versions are g_1 and g_2. Therefore,

[7] Note that this statement is only true for $L \geq L_{\text{real}}$, where L_{real} denotes the minimal Lipschitz constant of ∇f.

$g_1 \perp\!\!\!\perp g_2$[8] and $\Sigma = \sigma^2$, which is as general as possible for the one-dimensional case. Note that $g_i \sim \mathcal{N}(\nabla f(x_i),\, \sigma^2)$. Moreover, the co-coercivity constraint between g_1 and g_2 is inactive when:

$$\begin{aligned}\|g_1 - g_2\|^2 < L\,\langle g_1 - g_2,\, x_1 - x_2\rangle &\Leftrightarrow (g_1 - g_2)^2 - L(g_1 - g_2)(x_1 - x_2) < 0 \\ &\Leftrightarrow (g_1 - g_2)(g_1 - g_2 - L(x_1 - x_2)) < 0 \\ &\Leftrightarrow 0 < g_1 - g_2 < L(x_1 - x_2)\,.\end{aligned}$$

Therefore, noticing that $g_1 - g_2 \sim \mathcal{N}(\nabla f(x_1) - \nabla f(x_2),\, 2\sigma^2)$ and defining $\Delta_x = x_1 - x_2$ and $\Delta_{\nabla f} = \nabla f(x_1) - \nabla f(x_2)$:

$$\begin{aligned}P(\|g_1 - g_2\|^2 < L\,\langle g_1 - g_2,\, x_1 - x_2\rangle) &= P(0 < g_1 - g_2 < L\Delta x) \\ &= \Phi\left(\frac{L\Delta_x - \Delta_{\nabla f}}{\sqrt{2}\sigma}\right) - \Phi\left(\frac{-\Delta_{\nabla f}}{\sqrt{2}\sigma}\right) \\ &= p_{inactive}\,.\end{aligned}$$

Consequently, $p_{active} = P(\|g_1 - g_2\|^2 \geq L\,\langle g_1 - g_2,\, x_1 - x_2\rangle) = 1 - p_{inactive}$ by event complementarity. Moreover, note that $\Delta_x > 0$ (since $x_1 > x_2$) and, thus, by the definition of convexity (Definition 1.1.2), $\Delta_{\nabla f} \geq 0$.

In order to have a better insight on the expressions above, a simple analysis for a quadratic function - $f(x) = (L_{real}/2)\, x^2$ - is performed. The motivation of using a quadratic comes not only, as we are in the one-dimensional case, from the second derivative of this function being, actually, L_{real} in all the domain, but also from the fact that every twice-differentiable convex function can be approximated, at least locally, by a quadratic function. Moreover, this analysis is made considering that the Lipschitz constant used as an input for the denoiser, L, may not correspond to L_{real}. For the given quadratic objective function and defining $\Delta_L = L - L_{real}$, it is possible to obtain the following result:

$$\begin{aligned}\Delta_{\nabla f} - L\Delta_x &= L_{real}\Delta_x - L\Delta_x \\ &= -\Delta_L\Delta_x\,.\end{aligned}$$

Consequently:

$$p_{inactive} = \Phi\left(\frac{\Delta_L\Delta_x}{\sqrt{2}\sigma}\right) - \Phi\left(\frac{-L_{real}\Delta_x}{\sqrt{2}\sigma}\right)\,. \tag{3.20}$$

Taking into account that $\Phi(-\infty) = 0$, $\Phi(0) = 0.5$ and $\Phi(+\infty) = 1$, some high level observations

[8] The notation $\perp\!\!\!\perp$ denotes independence between random variables.

about the behaviour of $p_{inactive}$ as a function of Δ_x can be readily determined:

$$\begin{cases} \text{if } L \text{ is overestimated:} & \Delta_L > 0 : p_{inactive} \overset{\Delta_x \to 0}{\to} 0;\ p_{inactive} \overset{\Delta_x \to +\infty}{\to} 1 \\ \text{if } L \text{ is perfectly estimated:} & \Delta_L = 0 : p_{inactive} \overset{\Delta_x \to 0}{\to} 0;\ p_{inactive} \overset{\Delta_x \to +\infty}{\to} 0.5 \\ \text{if } L \text{ is underestimated:} & \Delta_L < 0 : p_{inactive} \overset{\Delta_x \to 0}{\to} 0;\ p_{inactive} \overset{\Delta_x \to +\infty}{\to} 0\,. \end{cases}$$

In order to further illustrate the analysis performed, in Figure 3.3, p_{active} is represented as a function of Δ_x:

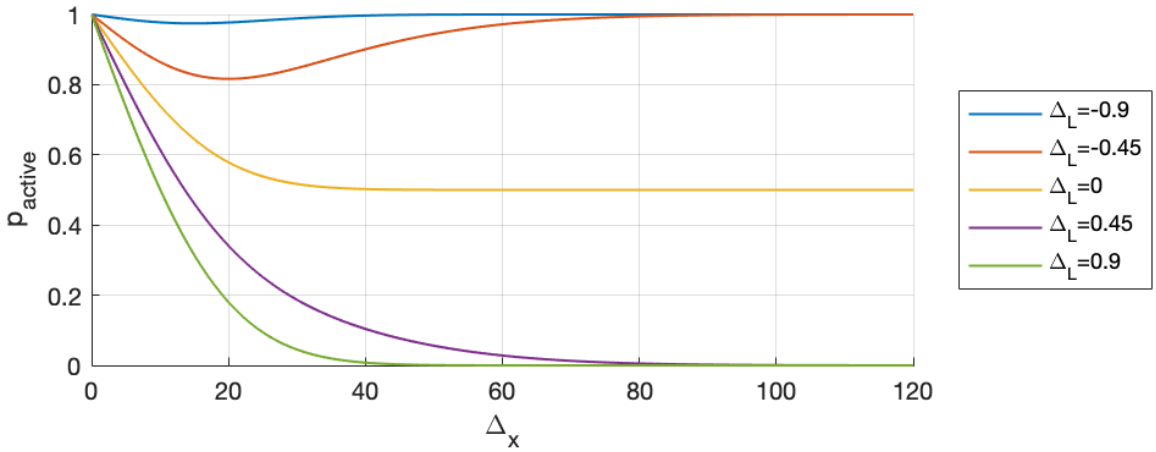


Figure 3.3: Representation of the probability of two sampled noisy gradients, g_1 at x_1 and g_2 at x_2, not being co-coercive as a function of the distance between x_1 and x_2 ($p_{active}(\Delta_x)$) for different Δ_L. We considered $L_{real} = 1$ and $\sigma = 10$.

These results can be intuitively explained. Those explanations are only given for $p_{active}(\Delta x)$, as they are the ones represented in Figure 3.3 and can be easily transferred to $p_{inactive}(\Delta_x)$ due to their complementarity:

- Independently of Δ_L, $p_{active}(0) = 1$ as from the co-coercivity inequality (Equation (3.2)), the right-hand side is zero, and therefore the constraint will always be violated with exception of the case in which $g_1 = g_2$, a set of points which, nevertheless, has zero Lebesgue measure. Moreover, as Δ_x increases, the right-hand side of (Equation (3.2)) increases, allowing the increase of the Lebesgue measure of the set of points which do not violate co-coercivity. Given this, obviously p_{active} decreases for all Δ_L. After this point, different Δ_L lead to different behaviours.
- When the curvature is overestimated ($\Delta_L > 0$), the true gradients, $\nabla f(x_1)$ and $\nabla f(x_2)$ are always co-coercive, and if only those were considered, there wouldn't be active constraints. Therefore, the decrease in the tightness of the constraint (increase of the right-hand side of Equation (3.2)) is directly related to the increasing relevance of the true gradients with the distance, explaining why p_{active} tends to zero. Note that, naturally, the higher the Δ_L, the faster p_{active} tends to zero.

- When the curvature is precisely the one estimated, $\Delta_L = 0$, the true gradients always lead to a case of equality in Equation (3.2)). Due to noise, when the perturbation leads the noisy gradients to be further away than supposed, the constraint activates. By the same token, the constraint is inactive when the perturbation decreases the difference between gradients. As the noise follows a Gaussian distribution, it increases or decreases the difference between gradients with the same probability, explaining why p_{active} tends for 0.5 for larger distances.

- When the curvature is underestimated, $\Delta_L < 0$, the variation of the true gradients is always incoherent with the $L_{estimated}$ (true gradients change faster than allowed). Consequently, the constraint tends to be active (p_{active} tends to 1) as the distance increases. Note that the lower the Δ_L, the more incoherent the true gradients are with the co-coercivity constraint and, therefore, the faster the p_{active} tends to 1.

A careful observation of Equation (3.20) also exposes the dependence of p_{active} (or $p_{inactive}$) on the level of noise in the problem, expressed through σ. Therefore, it is expected that higher noise leads to slower convergence of those probabilities to the aforementioned values as it attenuates the influence from the true gradients. This intuition is confirmed in Figure 3.4.

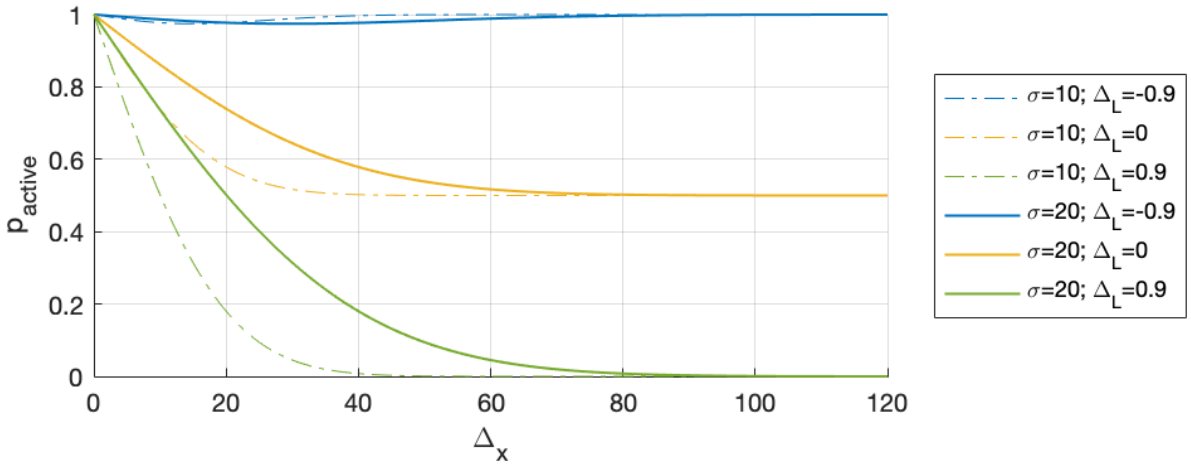


Figure 3.4: Comparison of p_{active} for different levels of noise. The dashed lines are retrieved from the bottom plot of Figure 3.3 ($\sigma = 10$), while the solid lines are obtained by doubling noise magnitude ($\sigma = 20$). The real value of the parameter L_{real} is the same: $L_{real} = 1$.

4

Computational Experiments

Contents

In this chapter, we present results of computational experiments with the COCO denoiser. First, we illustrate the relevance of using co-coercivity as a filter. Then, we describe experiments targeted to characterizing the estimator. Finally, we use the COCO_K denoiser as a plug-in for stochastic optimization, illustrating the gains both with synthetic data and with a learning application (logistic regression) that uses real data.

4.1 Pertinence of Co-coercivity

In this section, we describe an illustrative experiment that shows the impositions obtained through co-coercivity. As it was previously mentioned, this condition allows the fusion of two distinct properties of the objective function - convexity and L-smoothness - into one expression.

In order to get some insight on the solutions provided by the Quadratically Constrained Quadratic Problem (QCQP) of COCO denoiser (see Equation (3.5)), an experiment on a one-dimensional ($d = 1$) convex function is carried out, with the aim of comparing the benefit brought by COCO not only in relation to the raw oracle, but also to a similar QCQP but in which, instead of co-coercivity constraints, L-smoothness conditions (recall the L-smoothness definition - Definition 1.3.1) were imposed, *i.e.*,:

$$\begin{aligned} \underset{\theta_1,\ldots,\theta_K}{\text{minimize}} \quad & \sum_{k=1}^{K} (g_k - \theta_k)^T \Sigma^{-1} (g_k - \theta_k) \\ \text{subject to} \quad & \|\theta_m - \theta_l\| \leq L\|x_m - x_l\|, \quad 1 \leq m < l \leq K\,. \end{aligned} \tag{4.1}$$

In the case represented in Figure 4.1, the function chosen was $f(x) = x^2$. 64 oracle consultations from evenly spaced points between $x = 1$ and $x = -1$ were sampled. This oracle followed the noise model assumed (Gaussian additive noise). All the gradients were considered ($K = 64$), with the constant L used in the denoisers being equal to the real one: $L = 2$. Moreover, $\Sigma = \sigma^2 = 1$.

From the results in Figure 4.1, it can be seen that both the COCO denoiser and the denoiser with L-smooth constraints provide better gradient estimates than the oracle, as it can be observed, for example, by the reconstruction of the original function. Nevertheless, the difference between the two denoising schemes is subtle. From the bottom plot, where the reconstructions are represented, it is not clear which denoiser better approximates the original function (for $x \in [-1, 0)$ COCO seems to perform better, while for $x \in (0, 1]$ it is the other way around). However, from the top plot, it is possible to verify that the estimates provided by COCO are in fact closer to true gradients, namely at the ends of the x-axis. Since co-coercivity assumes convexity of the objective function, the gradients estimated are necessarily monotonically non-decreasing (this is easily seen in the top plot of Figure 4.1), a property that is not verified by the gradient estimates coming from the L-smoothness denoiser. More precisely, while co-coercivity imposes $0 \leq f''(x) \leq L$, L-smoothness only imposes $-L \leq f''(x) \leq L$, justifying the

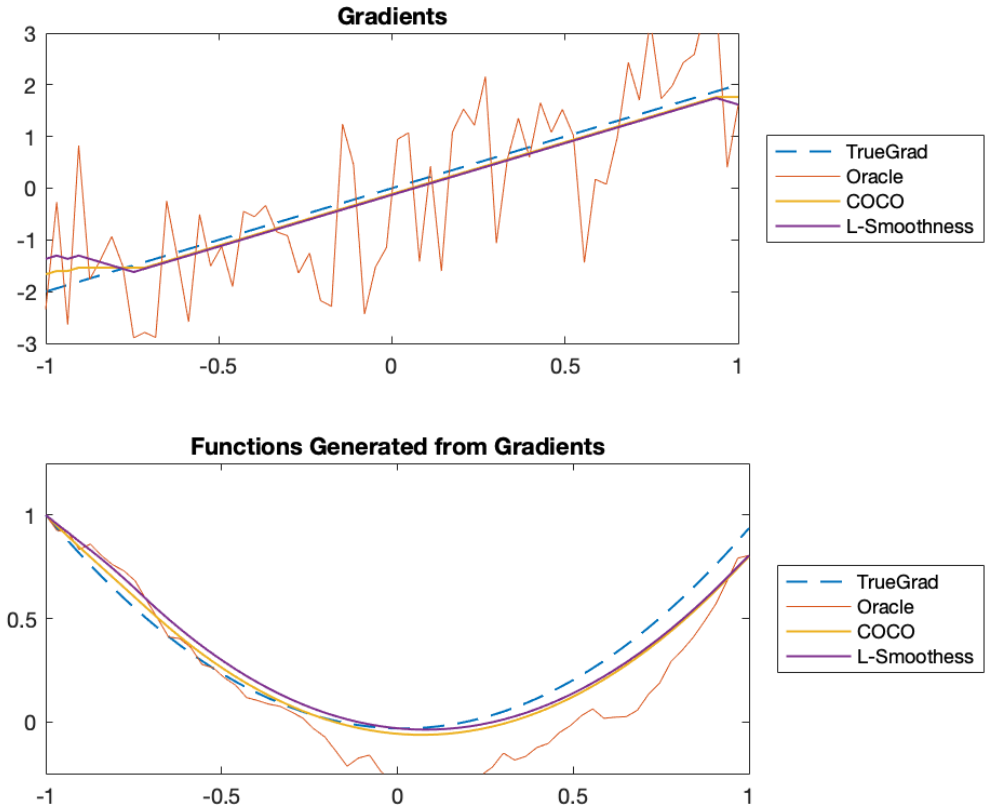


Figure 4.1: Top: Representation of the true gradients, oracle consultations, and gradient estimates by COCO and by imposing only L-smoothness, for the convex function $f(x) = x^2$.
Bottom: Functions generated by first-order approximations, $f(x_i) = f(x_{i-1}) + f'(x_{i-1})(x_i - x_{i-1})$, with $f(-1) = 1$, based on the gradients above.

differences observed.

To better clarify this point, we run the same experiment with a non-convex function, $f(x) = \sin(x)$. For this function, $L = 1$, which was the value considered for both denoisers. The magnitude of noise was the same ($\Sigma = 1$), as well as K, now with $x \in [0, 2\pi]$. The results are represented in Figure 4.2. As expected, the COCO denoiser outputs gradients which reconstruct a convex function (COCO tries to approximate the sinusoidal function by a convex one), which is not true for the L-smoothness denoiser (which is therefore capable of following the concave part of the original function).

Even though the present analysis was performed for the one-dimensional case, its results provide insights about the behaviours of the two denoisers for higher dimensions. In fact, while the COCO denoiser assumes $0 \preceq \nabla^2 f(x) \preceq LI$, L-smoothness alone just imposes $-LI \preceq \nabla^2 f(x) \preceq LI$ (for twice-differentiable functions). This justifies using the COCO denoiser over the L-smoothness one, in the stochastic convex setting. Also, when not having access to the value of L, it might be overestimated ($L >$

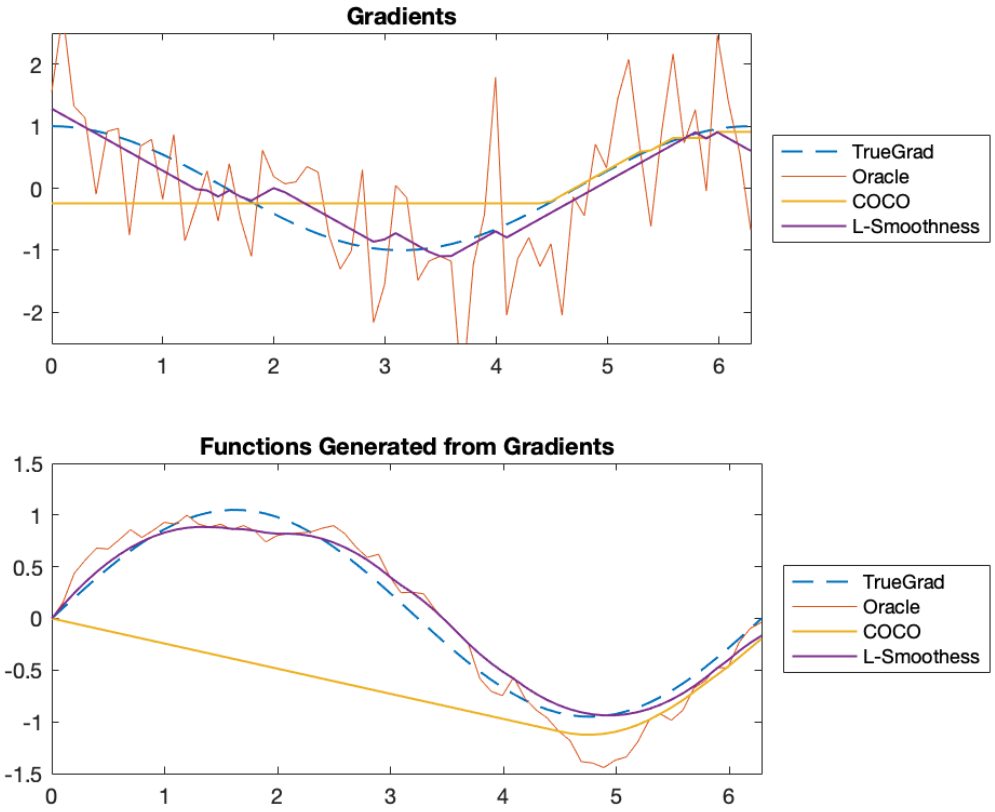


Figure 4.2: Top: representation of the true gradients, oracle consultations, and gradient estimates by COCO and by imposing only L-smoothness, for the non-convex function $f(x) = \sin(x)$.
Bottom: functions generated by first-order approximations, $f(x_i) = f(x_{i-1}) + f'(x_{i-1})(x_i - x_{i-1})$, with with $f(0) = 0$, based on the gradients above.

L_{real}), situation in which both denoisers allow faster variations of the gradient, thus the L-smoothness denoiser ends up allowing more non-monotonic behaviours.

4.2 Properties of the Estimator

In this section, we describe computational experiments to study the properties of the COCO estimator.

4.2.1 Mean Squared Error

Theorem 3.3.2, in Chapter 3, stated that:

$$\text{MSE}(\hat{\theta}) = \sum_{k=1}^{K} E[\|\hat{\theta}_k - \nabla f(x_k)\|^2] \leq \text{MSE}(g) = \sum_{k=1}^{K} E[\|g_k - \nabla f(x_k)\|^2]\,.$$

Note, nevertheless, that this result only provides an upper bound ensuring that the gradient estimation using COCO is preferable than raw oracle. Given this, an interesting problem is to study to what extent the former outperforms the latter. In addition, from Section 3.3.3 it was possible to obtain, for the one-dimensional case, an expression which relates the constraint tightness probability as a function of the distance between the two sampled points. Therefore, noting that in the case in which the constraint is not active, the COCO denoiser outputs the result provided by the oracle, and in the case in which the constraint is active, the denoiser filters the output of the oracle, MSE($\hat{\theta}$) is expected to be a function of the constraint tightness and, as a consequence, implicitly a function of the distance between points.

Taking this reasoning into consideration, the experiment represented in Figure 4.3 recovers the theoretical results obtained in Section 3.3.3. Therefore, a one-dimensional quadratic function, $f(x) = 1/2\, x^2$ was used, with $L_{real} = 1$, where two points were considered: one fixed at $x_1 = 0$ and a variable point at $x_2 = \Delta_x$. The oracle consultations provided gradient estimates with additive Gaussian noise with $\Sigma = \sigma^2 = 100$. In these conditions, the probability of the constraint being active, p_{active} for each Δ_x was estimated through Monte Carlo simulations:

$$\widehat{p_{\text{active}}} = \frac{N_{\text{active}}}{N},$$

where N_{active} is the number of repetitions in which the oracle outputted non co-coercive gradient estimates and N is the total number of repetitions.

The MSE($\hat{\theta}$) was also estimated for each Δ_x as

$$\widehat{\text{MSE}}(\hat{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{2}\|\hat{\theta}_{ki} - \nabla f(x_k)\|^2\right),$$

where N is, again, the total number of repetitions and $\hat{\theta}_{ki}$ is the gradient estimated by the COCO denoiser for the point x_k at repetition i.

Moreover, it is possible to obtain a closed-form result for the MSE(g) for a general number of points considered, K, a general dimension d and $\Sigma = \sigma^2 I$:

$$\text{MSE}(g) = E[\sum_{k=1}^{K}\|g_k - \nabla f(x_k)\|^2] = \sum_{k=1}^{K}E[\|w_k\|^2] = \sum_{k=1}^{K} d\sigma^2 = Kd\sigma^2,$$

where w_k is a realization from $W \sim \mathcal{N}(0,\ \sigma^2 I)$. Replacing it with the corresponding values from this case, it yields MSE(g) $= 200$, which is also depicted for reference in Figure 4.3.

As expected, the results for p_{active} recover those obtained in Section 3.3.3.

Regarding the MSE($\hat{\theta}$), it can be observed that:

- for $\Delta_x = 0$, all the curves have $p_{\text{active}} = 1$ and MSE($\hat{\theta}$) $= 100 = \sigma^2/2$. This recovers a well known

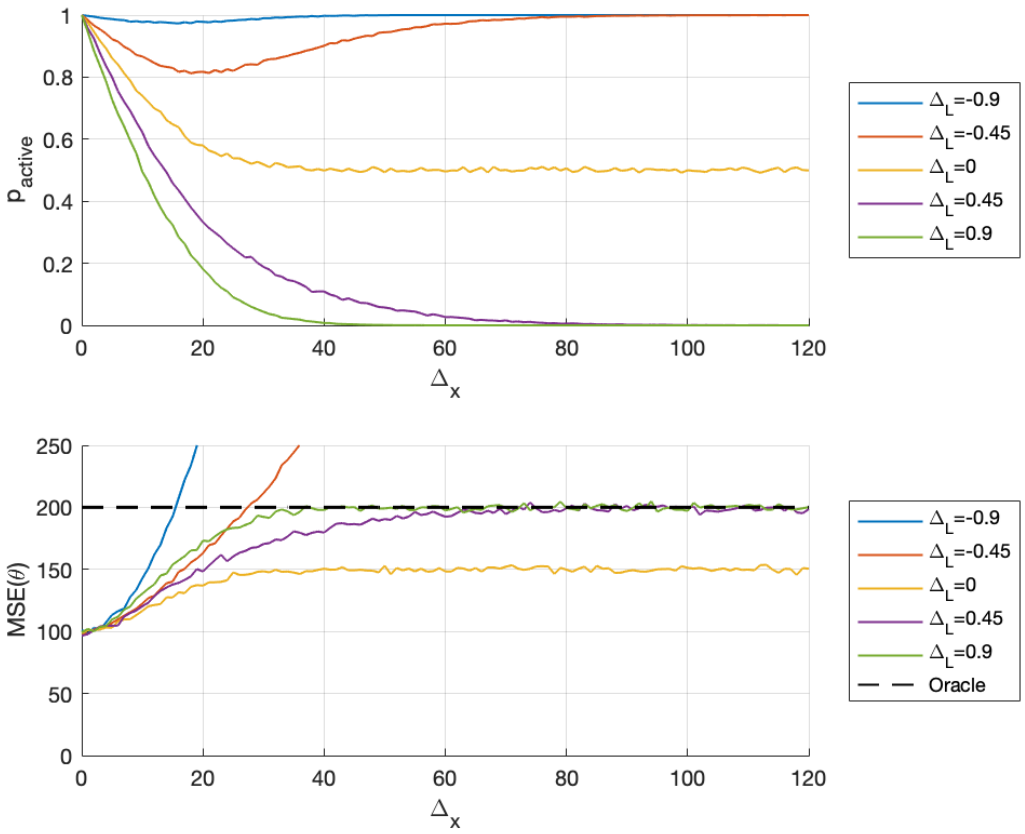


Figure 4.3: Top: Experimental plot for p_{active} as a function of Δ_x for different values of Δ_L.
Bottom: Computed MSE$(\hat{\theta})$; the dashed line denotes the (theoretical) value for the oracle.
Number of Monte-Carlo simulations (for both plots): $N = 10000$.

result for the average of K random variables with Gaussian distributions: their MSE[1] is σ^2/K. In fact, when $\Delta_x = 0$, COCO$_K$ denoiser outputs the average of the observed gradients (recall closed-form solution for COCO$_2$ - Theorem 3.2.2). Furthermore, COCO denoiser can therefore be considered an extension for the variance reduction through averaging method, but which tolerates samples from different points. This can be viewed as one of the main advantages of COCO;

- for the cases in which the L is underestimated ($\Delta_L < 0$), the MSE$(\hat{\theta})$ is not guaranteed to be lower than MSE(g). Nevertheless, note that there still is a range of Δ_x where MSE$(\hat{\theta}) \leq$ MSE(g). The more underestimated L is, the smaller this region becomes. This observation not only recalls that the result from Theorem 3.3.2 only holds for $\Delta_L \geq 0$, but also reinforces the importance of ensuring that the L considered for COCO denoiser is an upper bound for L_{real};

- for the case in which the L is perfectly estimated ($\Delta_L = 0$), just as the p_{active} tends to an intermediate

[1] The MSE corresponds to the variance of an unbiased estimator, which is the case of the average of random variables following normal distributions.

value, so it happens with MSE$(\hat{\theta})$. This is the ideal situation, as MSE$(\hat{\theta})$ is minimal for every Δ_L. Moreover, note that when the p_{active} curve stabilizes, the MSE$(\hat{\theta})$ also stabilizes, reinforcing the expected relation between those curves;

- for the cases in which the L is overestimated ($\Delta_L > 0$), just as p_{active} tends to 0, the MSE$(\hat{\theta})$ also tends to the MSE(g) reference curve. Moreover, it is possible to see that when p_{active} stabilizes around 0, so it happens to MSE$(\hat{\theta})$ around the oracle's curve. This is easily explained, again, by the fact that when the constraints are loose, the COCO denoiser outputs the oracle results without any "filtering";

- Regarding the noise variance, σ^2, it should be stated that, as previously seen in Section 3.3.3, its increase would shift the stabilization of the curves from the $\Delta_L > 0$ cases towards higher Δ_x.

The analysis above was performed for a metric that combines the information from every point considered. To have a better insight about what happens at each point, it is preferable to look at the elementwise MSE, *i.e.*, for x_k, MSE$(\hat{\theta}_k) = E[\|\hat{\theta}_k - \nabla f(x_k)\|^2]$ and MSE$(g_k) = E[\|g_k - \nabla f(x_k)\|^2] = \sigma^2 d = 100$. These results are represented in Figure 4.4 and seem to suggest that MSE$(\hat{\theta})$ divides itself equally by the two points. It also presents empirical evidence for the following result: if $\Delta_L > 0$, then $E[\|\hat{\theta}_k - \nabla f(x_k)\|^2] \leq E[\|g_k - \nabla f(x_k)\|^2]$. Note that this inequality is stronger than the one from Theorem 3.3.2, as the former imposes each term on the left-hand side from the latter to be smaller or equal than the respective term on its right-hand side.

In order to reinforce the empirical evidence of this result, we investigate the possibility of generalizing the statement $E[\|\hat{\theta}_k - \nabla f(x_k)\|^2] \leq E[\|g_k - \nabla f(x_k)\|^2]$ for arbitrary dimension d, and number of points K. It must be stated that whereas for the case in which there are only two points the geometry between them is always a line and, thus, their distance is the only parameter to be considered, for a higher number of points considered, this is not true. In fact, the different geometries that the considered points assume play a role on how the constraints interact among them (which is, in fact, the reason why we could not find a closed-form solution for COCO$_K$ with $K > 2$) and, consequently, a role on the estimator properties.

Since exploring all those possibilities is not possible in practice, we consider illustrative random configurations for the visited points. For each 8-point configuration in $\mathbb{R}^3$, each point is sampled from an uniform distribution in a cube centered at the origin with edge length $2l$, *i.e.*, $x_k \in [-l,\ l] \times [-l,\ l] \times [-l,\ l]$. This setting is repeated for different values of l. The eigenvalues of the Hessian were chosen to be linearly spaced between 1 and $1/3$ and $\sigma = 10$.

The estimator for MSE$(\hat{\theta}_k)$ is

$$\widehat{\text{MSE}}(\hat{\theta}_k) = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\theta}_{ki} - \nabla f(x_k)\|^2,$$

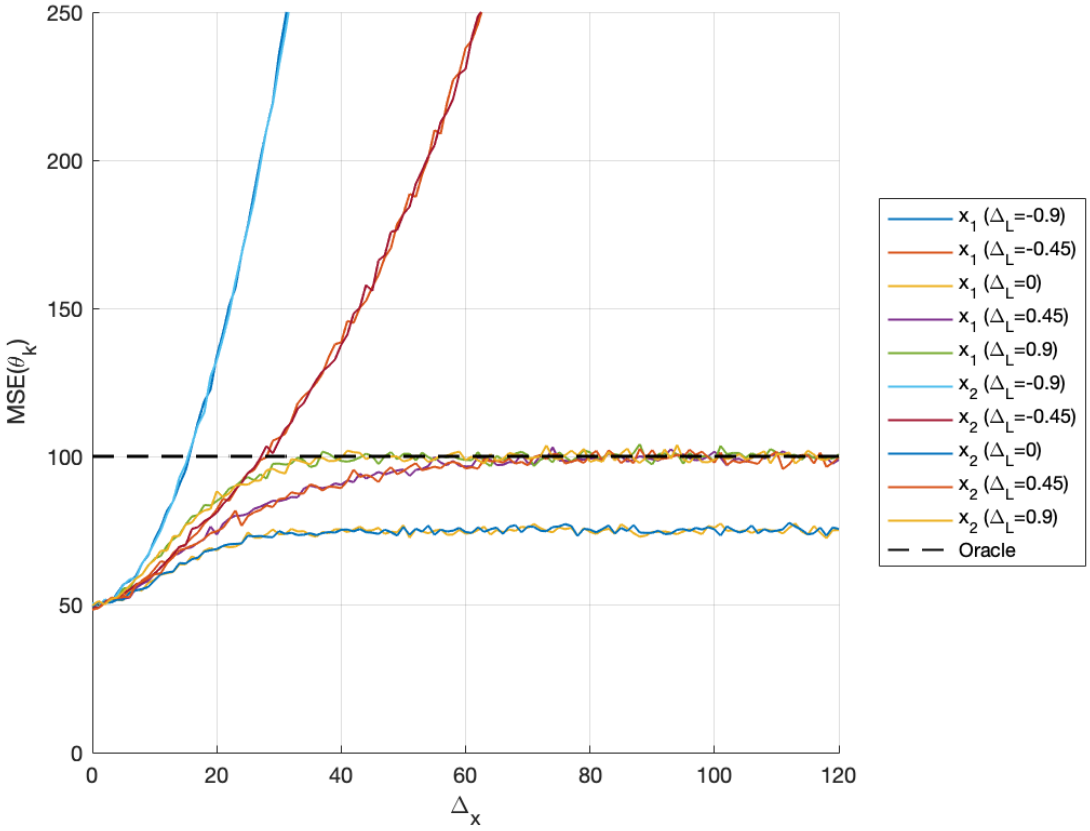


Figure 4.4: Experimental plot for MSE($\hat{\theta}_k$) as a function of Δ_x, for different Δ_L; the dashed line denotes the (theoretical) value for the oracle. Number of Monte-Carlo simulations: $N = 10000$.

where N is the number of Monte-Carlo simulations and $\hat{\theta}_{ki}$ is the gradient estimated by the COCO denoiser for the point x_k at repetition i. Similarly, the estimator for MSE(g_k) is

$$\widehat{\text{MSE}}(g_k) = \frac{1}{N}\sum_{i=1}^{N} \|g_{ki} - \nabla f(x_k)\|^2,$$

where g_{ki} is the gradient outputted by the oracle for the point x_k at repetition i. The results of our experiments are synthesized in Figure 4.5, Figure 4.6, Figure 4.7, for 6 different configurations and 3 values of l. Note that now MSE$(g_k) = \sigma^2 d = 300$.

From this figure, three main conclusions can be drawn:

- As conjectured with only two points, the idea that closer iterates allow lower MSE is reinforced for higher number of points and dimension. Moreover, when distance is sufficiently high, the COCO denoiser approximates its behaviour from the oracle, as the COCO gradient estimates progressively approximate themselves from the ones provided by the oracle with the increase of l;

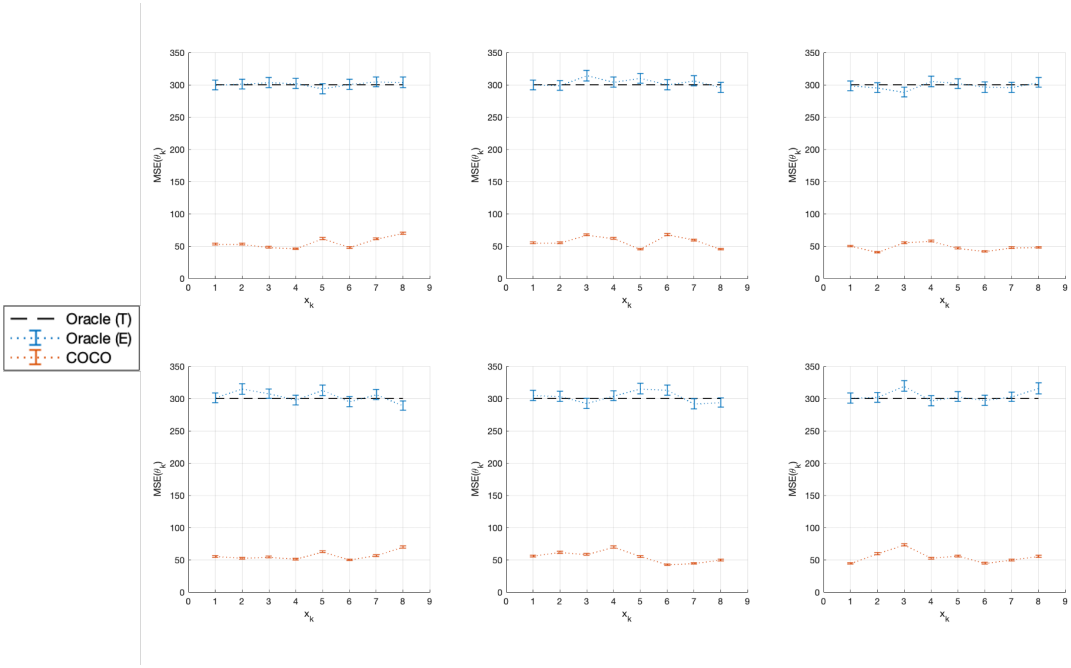


Figure 4.5: $\widehat{\text{MSE}}(\hat{\theta}_k)$ (COCO), $\widehat{\text{MSE}}(g_k)$ (Oracle (E)), and $\text{MSE}(g_k)$ (Oracle (T)), for points inside cubic boxes with edge lengths $2l$, centered at the origin, for each of 6 random configurations. Number of Monte-Carlo simulations: $N = 1000$. In this case, $l = 10$.

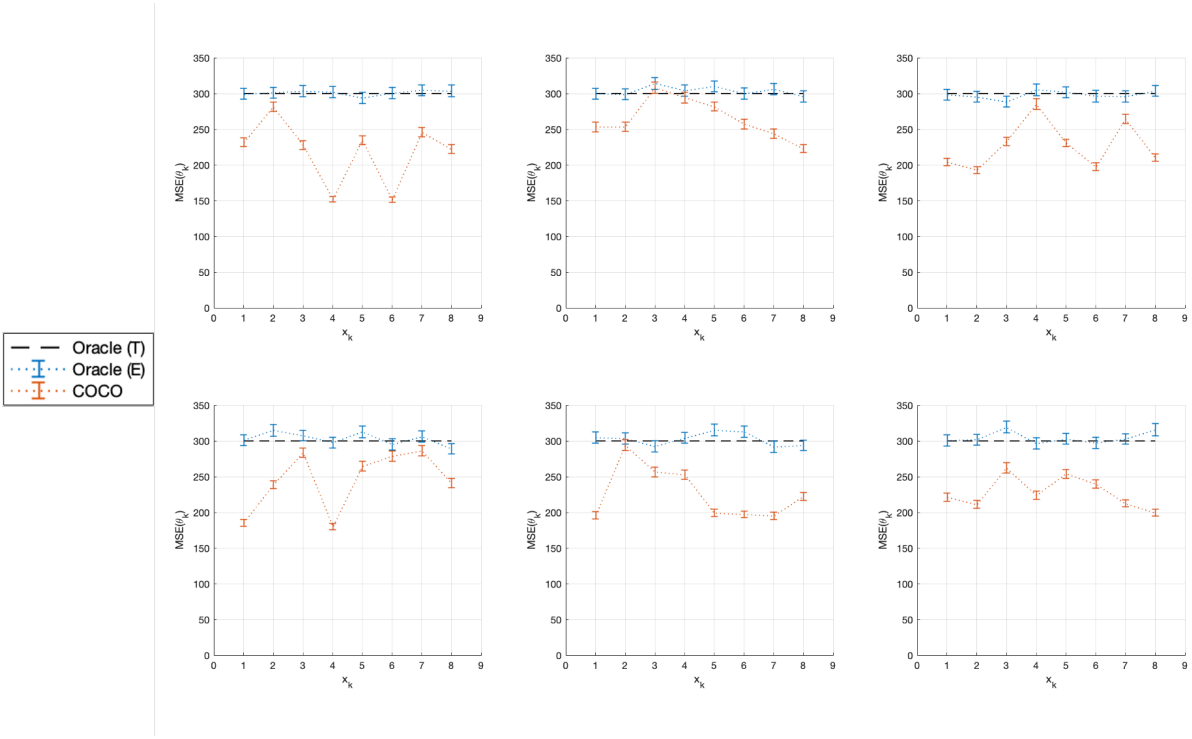


Figure 4.6: Same as Figure 4.5, now with $l = 100$.

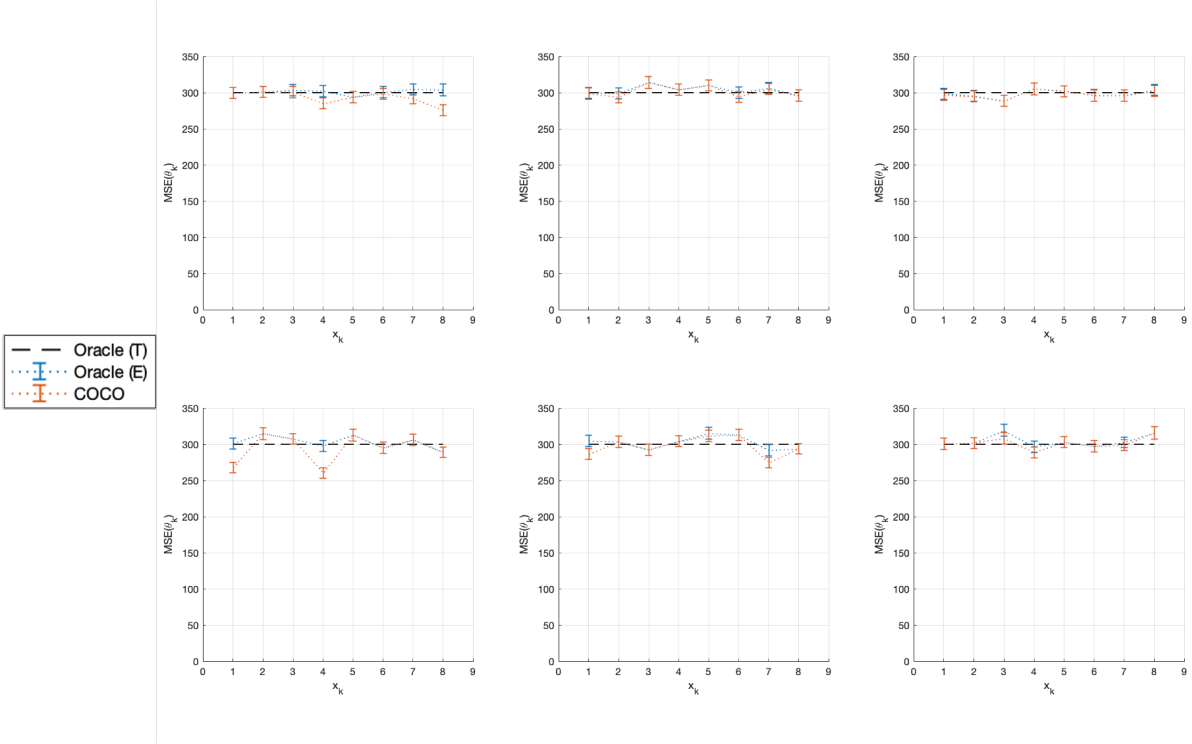


Figure 4.7: Same as Figure 4.5, now with $l = 1000$.

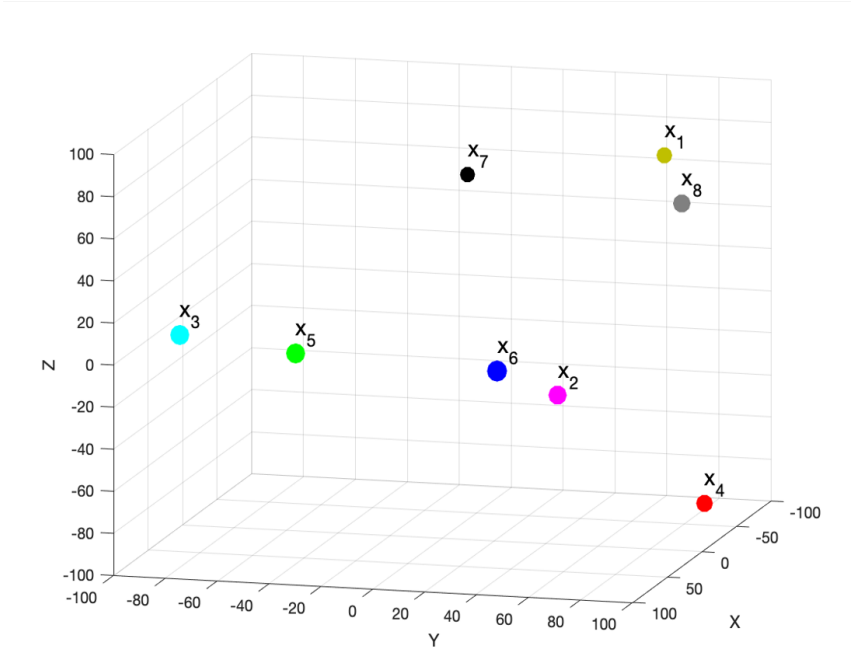


Figure 4.8: Spatial configuration that yields the results in the plot of the 1^{st} row, 3^{rd} column of Figure 4.6 (to provide some depth insight, marker size is proportional to the point x-coordinate).

- It can be also observed that $E[\|\hat{\theta}_k - \nabla f(x_k)\|^2] \leq E[\|g_k - \nabla f(x_k)\|^2]$ for every point in every setting, therefore reinforcing the empirical evidence on that sense.

- Contrarily to what was supposed for the case in which $K = 2$, it is possible to verify that the MSE$(\hat{\theta})$ does not distribute evenly among the different points, as the MSE$(\hat{\theta}_k)$ varies from point to point when $K > 2$. In fact, that is a consequence of the symmetry from the COCO$_2$ closed-form solution, property which does not hold for higher K. In particular, points which have other points closer present lower MSE$(\hat{\theta}_k)$, whereas more isolated points show higher MSE$(\hat{\theta}_k)$. This can be easily assessed by comparing the relative positions of the points represented in Figure 4.8 with MSE$(\hat{\theta}_k)$ obtained for each of them.

A rigorous observation of Figure 4.5 suggests that, in this case, MSE$(\hat{\theta}_k) <$ MSE$(g_k)/2$, while for $K = 2$ it was verified that it was the best that we could achieve through COCO. Therefore, in order to analyse how those MSE$(\hat{\theta}_k)$ vary with K a new experiment was carried out.

We now study the MSE$(\hat{\theta})$ for different numbers of points considered, K. In particular, in this case, $1 \leq K \leq 10$ and the different iterates are generated randomly following an uniform distribution inside a cube centered at the origin with edge length of 10 ($x_k \in [-5,\ 5] \times [-5,\ 5] \times [-5,\ 5]$). Moreover, we remain with $d = 3$, an anisotropic Hessian with eigenvalues linearly spaced between 1 and $1/3$ and $N = 1000$. Note that this choice for the size of the cube is not inadequate having in mind that the first-order algorithms in which this scheme will be applied, as, for example, for GD, its optimal step size is $\gamma = 1/L = 1$.

The results obtained are shown in Figure 4.9. We recovered the known result for the oracle: MSE$(g) = Kd\sigma^2$, which in this plot is represented by a line of intercept 0 and slope Kd. Surprisingly, the results for COCO$_K$ suggested that those results could as well be approximated by a line, but still with 0 intercept (when there is no noise, the noisy gradients correspond to the real ones, so no error is expected for both estimators). In order to measure the adequacy of this linear approximation for COCO$_K$, the coefficients of determination, R^2, obtained for the oracle and through COCO$_K$ were compared. These are represented in Table 4.1. It can be observed that those coefficients are comparable between the two cases (in fact, the ones regarding the oracle are slightly higher, but the difference can be easily assigned to precision errors in the solutions provided by COCO$_K$). Taking into consideration that we know that for the oracle the linear approximation is completely adequate, we assume that for $COCO_K$ it holds as well.

From Figure 4.5, it is possible to observe that for points inside a tight cube, *i.e.*, points which are sufficiently close to each other, the MSE$(\hat{\theta}_k)$ distributes almost uniformly across the iterates x_k. Since in this case we are in an even tighter cube and, in order to have an approximation of that MSE$(\hat{\theta}_k)$, each MSE$(\hat{\theta})$ curve from Figure 4.9 is divided by the corresponding K. The result observed is in Figure 4.10. Furthermore, using the information from this figure, the slope of each of the curves was depicted as a function of K in Figure 4.11.

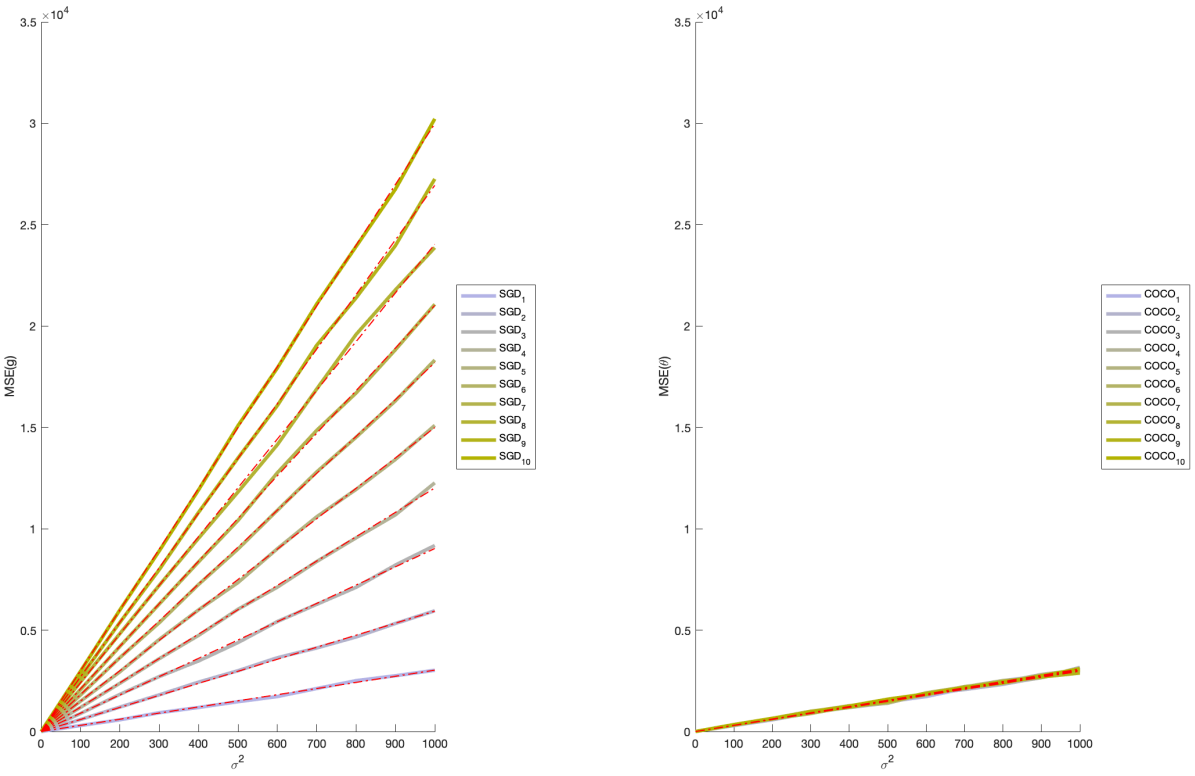


Figure 4.9: $\widehat{\text{MSE}}(g)$ (left) and $\widehat{\text{MSE}}(\hat{\theta})$ (right) as functions of the noise variance σ^2, for several numbers K of points considered. The dashed-dotted red lines result from linear regressions with intercept fixed at 0. Number of Monte-Carlo simulations: $N = 1000$. For each simulation, a different set of points is randomly generated from an uniform distribution in a cube centered at the origin with edge length 10.

Figure 4.11 suggests that the MSE$(\hat{\theta}_k)$ is $O(1/K)$ for COCO$_K$, while the MSE(g_k) remains constant independently of K. This result was, in fact, predictable from Figure 4.9. On the one hand, for the raw oracle consultations, we have MSE$(g) = Kd\sigma^2$ and therefore MSE(g_k) = MSE$(g)/K = d\sigma^2$, whose slope is $d = 3$. On the other hand, it is also observable that for the case of COCO$_K$ the MSE$(\hat{\theta})$ remained approximately constant, independently of the K considered. Given that COCO$_1$ is equivalent to the raw oracle, it is thus possible to conclude that MSE$(\hat{\theta}) \approx d\sigma^2$, which therefore implies that MSE$(\hat{\theta}_k)$ = MSE$(\hat{\theta})/K = d\sigma^2/K$, whose slope is therefore $3/K$, resulting in a decreasing MSE$(\hat{\theta}_k)$ with K.

This result is remarkable, as this is the usual result for the common averaging of normally distributed variables. In this case, that averaging would require the sampling of K gradient estimates of the oracle at each iterate x_k. With COCO$_K$, it is possible to achieve the same MSE$(\hat{\theta}_K)$ without having to be stuck on the same position. Therefore, this result reinforces the interpretation of COCO$_K$ as an extension to that procedure in the sense that it allows to integrate more information for more precise gradient estimates without having to stop the iterate progression.

Table 4.1: R^2 obtained for the linear regressions from Figure 4.9 for SGD_K and COCO_K for different K

	SGD	$COCO_K$
$K = 1$	0.9980	0.9980
$K = 2$	0.9994	0.9989
$K = 3$	0.9991	0.9955
$K = 4$	0.9994	0.9991
$K = 5$	0.9998	9.9957
$K = 6$	0.9999	0.9990
$K = 7$	0.9998	0.9981
$K = 8$	0.9995	0.9979
$K = 9$	0.9997	0.9970
$K = 10$	0.9998	0.9988

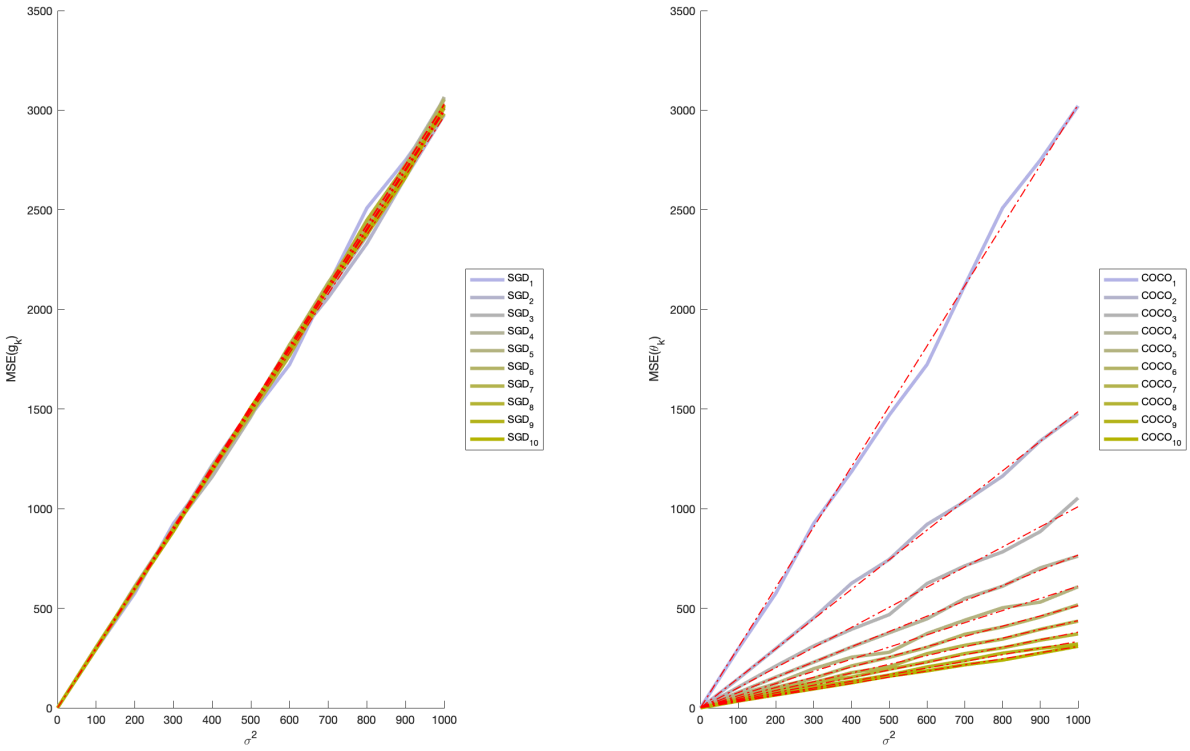


Figure 4.10: Same as Figure 4.9, now for $\widehat{\text{MSE}}(g_k)$ (left) and $\widehat{\text{MSE}}(\hat{\theta}_k)$ (right).

4.2.2 Bias

The bias of the gradient estimators here analysed can be formulated as: for each point x_k, $Bias(\hat{\theta}_k) = E[\hat{\theta}_k - \nabla f(x_k)] = E[\hat{\theta}_k] - \nabla f(x_k)$ and $Bias(g_k) = E[g_k - \nabla f(x_k)] = E[g_k] - \nabla f(x_k)$. In fact, by definition, $E[g_k] = \nabla f(x_k)$, which therefore implies $Bias(g_k) = 0$. Therefore, the oracle whose noise follows the additive and normally distributed model is an unbiased estimator of the gradient. We are interested in also having some characterization of the behavior of $\hat{\theta}_k$ in this respect. In order to test the bias of the COCO denoiser estimator, we estimated $\|Bias(\hat{\theta}_k)\|$ (note that if it is an unbiased estimator, then

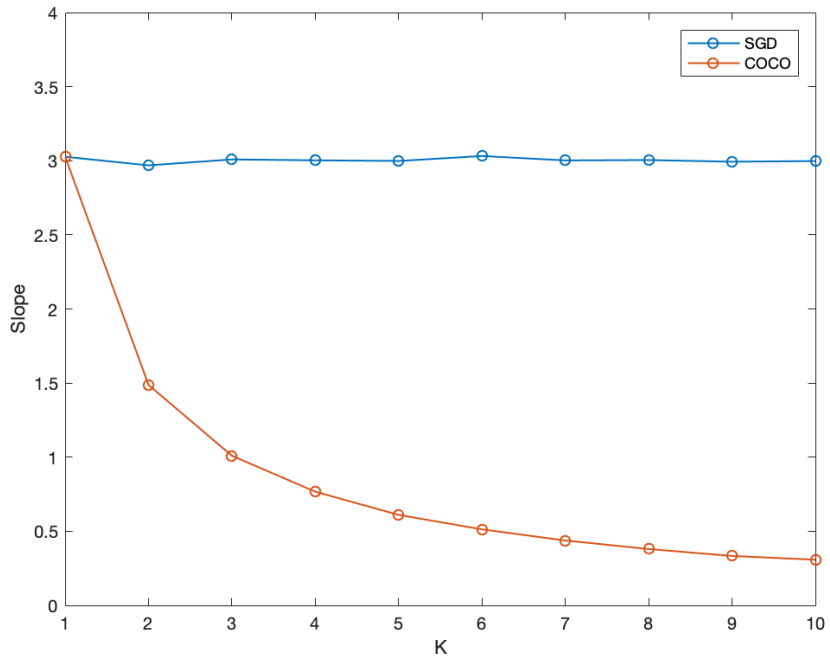


Figure 4.11: Slopes of the linear regressions in the Figure 4.10 as functions of the number of points considred, K.

$\|Bias(\hat{\theta}_k)\| = 0$) through the Monte-Carlo method,

$$\widehat{\|Bias(\hat{\theta}_K)\|} = \left\| \frac{1}{N} \left(\sum_{i=1}^{N} \hat{\theta}_{ki} \right) - \nabla f(x_k) \right\|,$$

where N is the total number of repetitions and $\hat{\theta}_{ki}$ is the gradient estimated by the COCO denoiser for the point x_k at repetition i. Therefore, using this estimator in the same setup used to obtain Figure 4.3 and Figure 4.4, the results in Figure 4.12 were achieved.

The conclusions to take from Figure 4.12 are similar to the ones in the previous section:

- In all cases, for $\Delta_x = 0$, the COCO estimator is unbiased since it consists of the averaging estimator;
- For $\Delta_L < 0$, the bias of this estimator seems to grow linearly with Δ_x. The smaller the Δ_L, the higher the slope of that linear relation;
- For $\Delta_L = 0$, the estimator is biased as well. That bias grows until a stabilization, which happens at the Δ_x that it happened with p_{active};
- For $\Delta_L > 0$, the estimator is also biased. As the COCO estimator outputs become more similar to the ones of the oracle, its bias decreases. Moreover, the higher the Δ_L, the lower the bias (as the constraints are less restrictive).

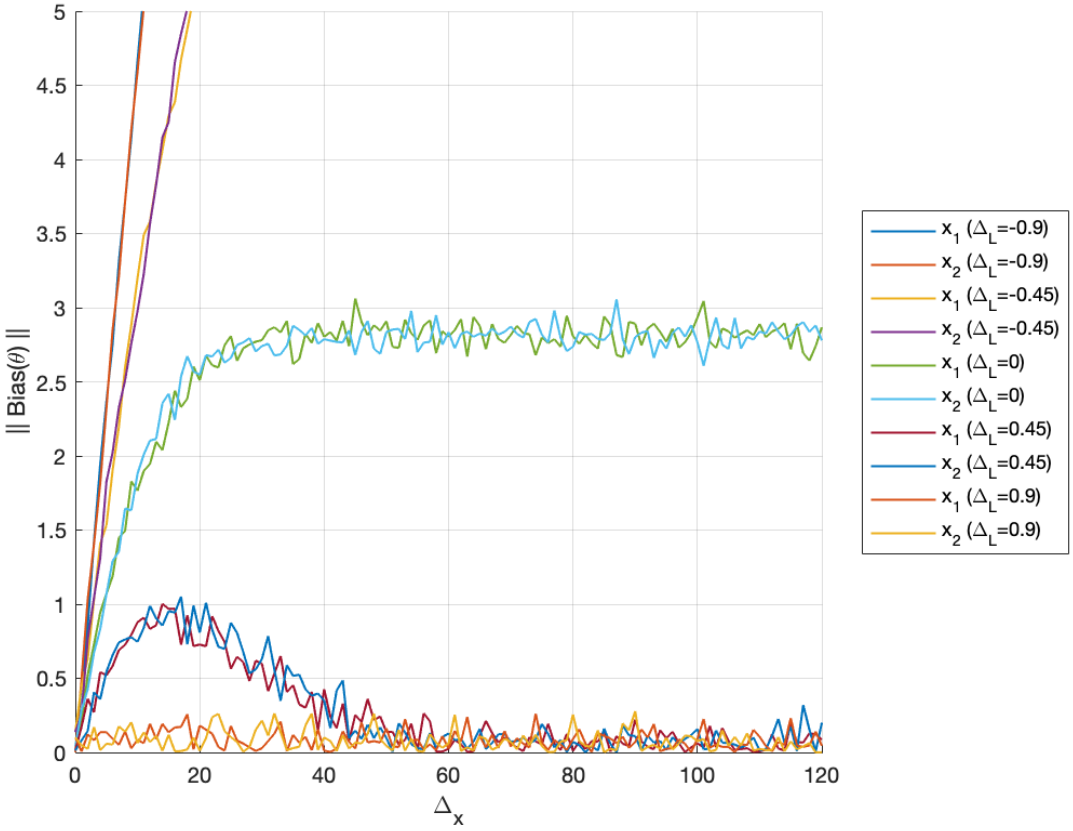


Figure 4.12: Bias as a function of the distance between the points considered,for the setup of Figure 4.3.

These observations suggest that if the constraint between two gradient estimates is active, then it imposes bias on the COCO estimator. On the other hand, when the constraint is inactive, the COCO estimator outputs the oracle consultations, which are known to be unbiased.

4.3 Stochastic Optimization with COCO

In this section, the COCO$_K$ denoiser is coupled to typical first-order algorithms and their performance is evaluated in synthetic and real contexts. As introduced in Chapter 1, the denoisers are essentially viewed as plug-ins to be coupled to the oracles to which the already existent algorithms in stochastic optimization resort, *i.e.*, instead of having direct access to the raw observation from the oracle, g_i, they are fed with the convex and L-smooth coherent observations of that same oracle, $\hat{\theta}_i$. This scheme is represented in Figure 4.13.

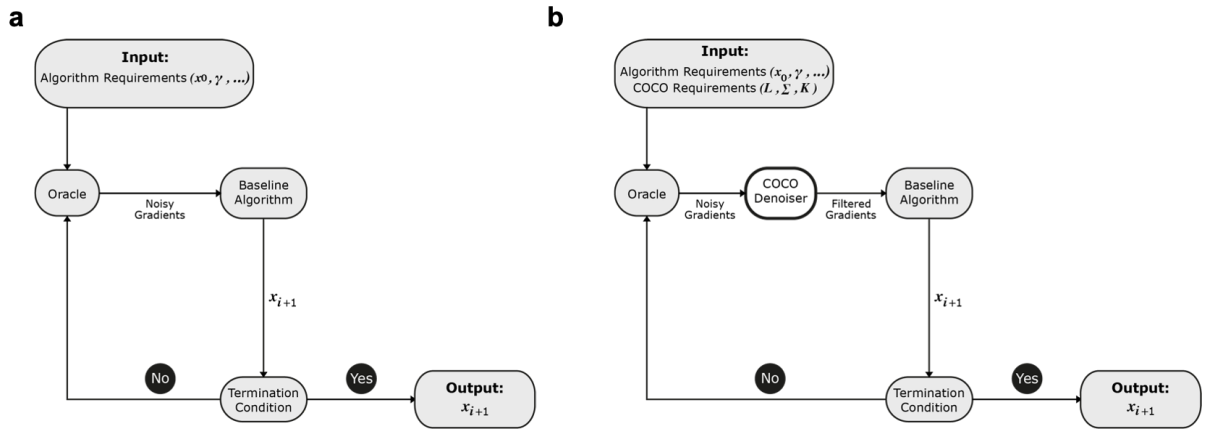


Figure 4.13: Typical approach to first-order stochastic optimization (a) and the proposed workflow (b). The COCO denoiser is simply plugged-in to provide more accurate gradients to any baseline algorithm.

4.3.1 Trajectories of the Algorithms

In order to get some insight on the type of output provided by the COCO denoiser, some experiments were carried out in a 2-dimensional ($d = 2$) quadratic function, as in that case it is possible to observe the different trajectories that the algorithm performs. These are represented in Figure 4.14.

In the experiments from Figure 4.14, the parameters for the COCO denoiser are both perfectly estimated (L and Σ match the ground truth). Therefore, even though in the case represented the increase in K leads to a deterioration of the performance (the algorithms ends up further from the optimum), it is not statistically representative. In fact, it is easily explainable by the fact that we are considering a model of additive noise, whose variance is fixed. Thus, when we are further from the minimum, the magnitude of the true gradients considered is significantly larger than the magnitude of the noise and,

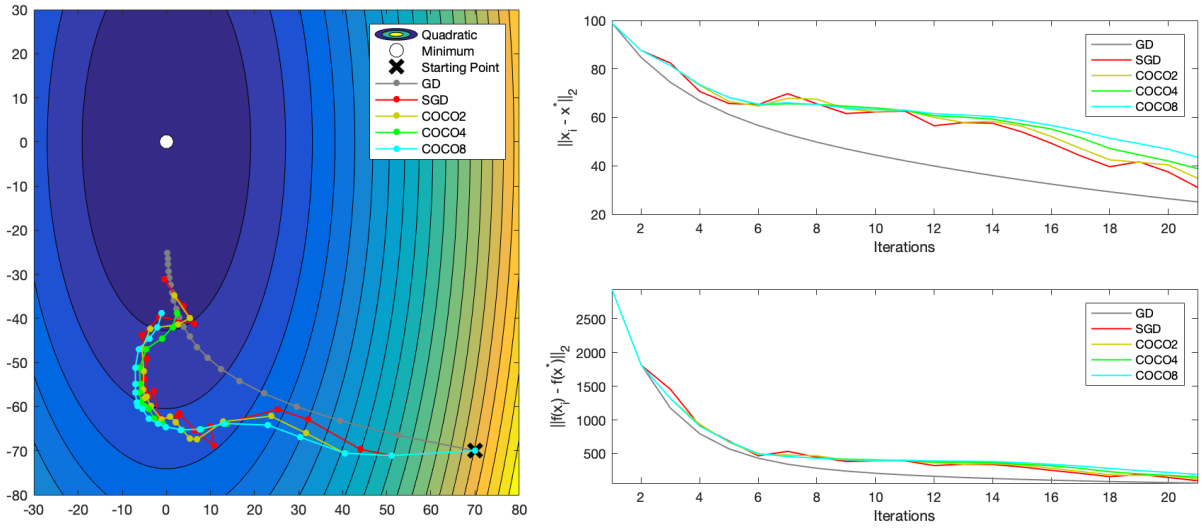


Figure 4.14: Results (20 steps) of 5 algorithms, for a quadratic function $(1/2)\ x^T Ax$, with $A = diag(1, 0.2)$. On the left, the trajectories of the algorithms; on the right, the distance to the optimum in the vector space (top) and in the function space (bottom). Oracle parameters: $\Sigma = 100I$. SGD hyperparameters: $\gamma = 0.25$. COCO hyperparameters: $L = 1$; $\Sigma = 100I$. When $K \geq i$, COCO$_K$ uses the total number i of visited points. The noise at each iteration is the same for all algorithms (the random seed is the same).

consequently, their sum corresponds essentially to the true gradient, leading to a situation where apparently no denoise would be preferable. In situations where the magnitude of the gradients generated becomes closer to the magnitude of the noise generated, the results are different. The major insight to take from this experiment is that it is possible to observe that, with the increase in K, the COCO$_K$ trajectory of the algorithm is smoothed, in contrast with the erratic trajectory of SGD.

4.3.2 Warm-Starting

By coupling a baseline algorithm with COCO$_K$, note that, at iteration i, only the oldest gradient (g_{i-K}) is forgotten and a new one (g_i) is kept in memory. Thus, it is reasonable to think of taking advantage from the COCO$_K$ solution obtained for the previous iterate to obtain a new solution faster.

We introduce a warm-starting procedure for the COCO$_K$ solution method (FDPG) which enables this utilization of past information. In particular, we achieve it by a careful initialization of the dual variable, s. In fact, s is the vector that results from stacking the different s_{ml}, where each s_{ml} addresses the co-coercivity constraint between the COCO estimates for gradient m, $\hat{\theta}_m$, and for gradient l, $\hat{\theta}_l$. Since we expect the estimates for old gradients to only have small variations among them on the new iterate as they have been "filtered" at least once, we initialize these s_{ml} to the values obtained for the correspondent dual variables in the previous COCO$_K$ solution. For the different s_{ml} concerning the new gradient, we do not have any information yet, thereby being initialized to the default value (zero).

In Figure 4.15, we expose the improvements provided by this approach. The warm-starting proce-

dure allows the iterative method to start with a much better guess of s^*, thereby achieving satisfactory approximate solutions faster.

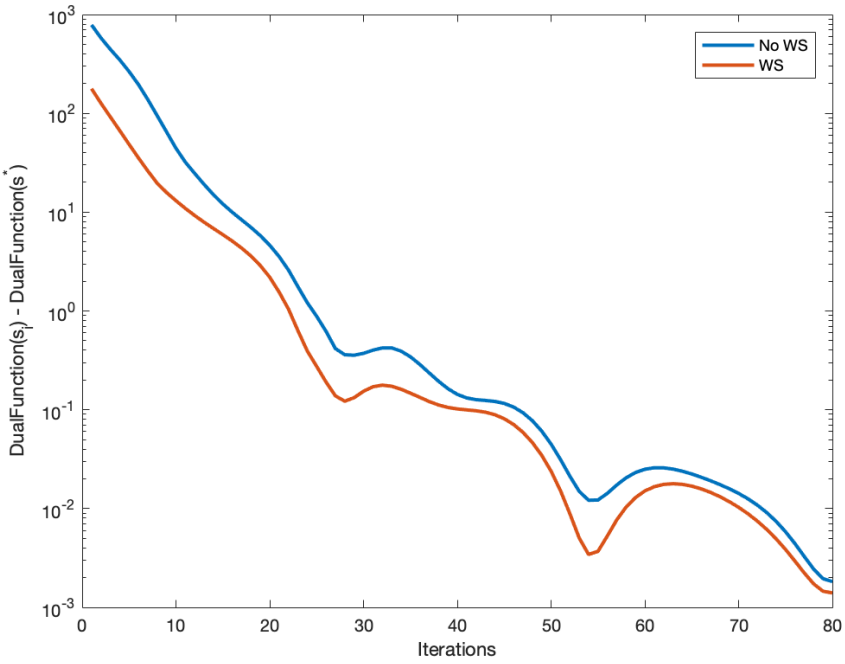


Figure 4.15: Dual objective function obtained for the different iterates of the COCO$_K$ solution method, using the warm-starting procedure (WS) and without using it (No WS). DualFunction(s) is the dual objective function $p^*(-A^T s) + q^*(s)$, s_i is the vector that results from stacking the different s_{ml} at iteration i, and s^* is the corresponding optimal vector.

4.3.3 Synthetic Data

In this section, the utility of COCO$_K$ is assessed in a scenario that perfectly matches the assumptions under which the denoiser was proposed. In this case, the objective function is a 10-dimensional ($d = 10$) quadratic function, $f(x) = 1/2\ x^T Ax$, where matrix A is the Hessian of the objective function. In this case we consider an anisotropic Hessian, with eigenvalues linearly separated between 1 and $1/10$, with the minimum at $x^* = (0, 0, \ldots, 0)^T$. Moreover, the first-order oracle provides a gradient estimate whose noise is additive and normally distributed, with $\Sigma = 100\ I$. The initial iterate was kept the same through all the simulations, $x_0 = (100, 100, \ldots, 100)^T$.

Performance of COCO coupled to SGD and Adam

Considering the setup described above, the COCO$_K$ denoiser is coupled both to SGD and Adam, where the latter is picked as a representative of its class of algorithms (adaptive step size algorithms). Note

that algorithms that address finite sum objectives are not tested here, as the setup considered falls out of their scope. In both cases, the COCO$_K$ hyperparameters are correctly set: $L = 1$, $\Sigma = 100\ I$. Given that we are in a stochastic setting, the quantity that we are interested in following across iterations is $E[\|x_i - x^*\|]$, which can be estimated via Monte-Carlo simulation through:

$$\widehat{E[\|x_i - x^*\|]} = \frac{1}{N}\sum_{j=1}^{N} \|x_{ij} - x^*\|,$$

where N is the total number of repetitions and x_{ij} is the point obtained from the algorithm in iteration i at repetition j.

In Figure 4.16, the results obtained are shown with $N = 100$.

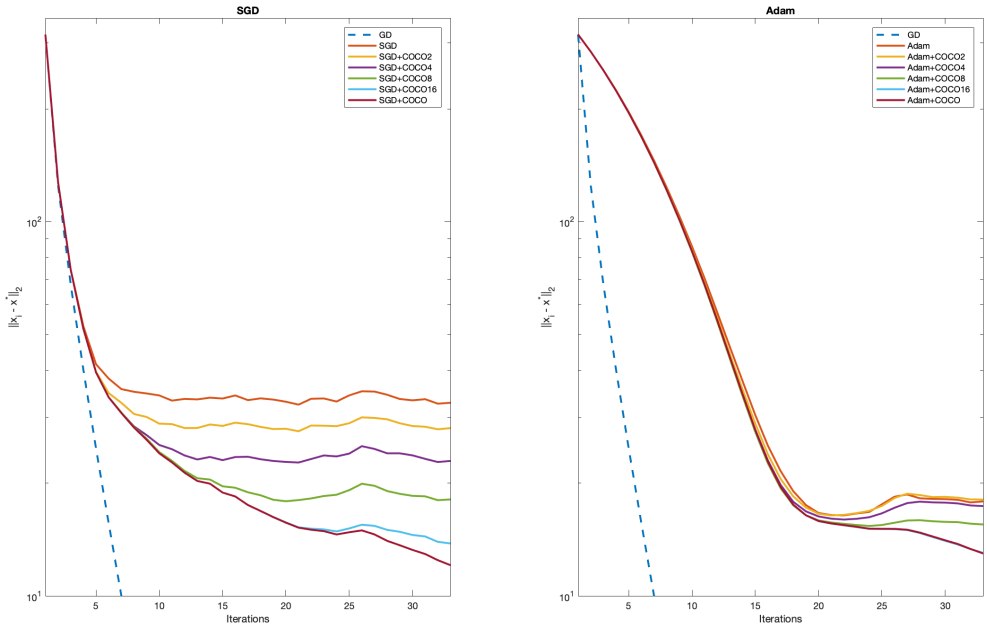


Figure 4.16: Results of stochastic optimization with the proposed COCO denoiser, when used with baselines SGD (left) and Adam (right). The performance is measured in terms of $\widehat{E[\|x_i - x^*\|]}$ (the performance of GD is also depicted for reference; the lines for "Adam + COCO$_{16}$" and "Adam + COCO" are superimposed). Number of Monte-Carlo simulations: $N = 100$.

From this figure, it is possible to recall the result shown in Equation (2.14): there is an initial *bias regime*, where all the algorithms seem to converge linearly (see Table 2.1; we are in an L-smooth and strongly convex setting, where GD is known to converge linearly and the stochastic algorithms are able to keep up with it initially); across iterations that convergence is successively slowed down and eventually leads to a stagnation to which we call *variance regime*. In fact, from Figure 4.16 we can observe how a higher K in COCO$_K$ leads to improved performance at least in terms of the variance regime (without

compromising the bias one). Moreover, still from Equation (2.14), it is possible to conclude that the "level" at which the stochastic algorithm stops converging is directly dependent on the (uncentered) variance of the oracle. This reinforces the variance reduction achieved by coupling COCO_K to a baseline algorithm.

Gains in Variance

In this section, we further characterize to what extent the COCO_K coupling is able to reduce the variance of the provided gradient estimate. In fact, from Section 4.2.1, it was possible to empirically characterize the COCO_K estimator ($\text{MSE}(\hat{\theta}_k) \approx d/K\sigma^2$ for sufficiently close points K points). A well-known relation between the MSE and the variance of an estimator $\hat{Y}$ is:

$$\text{MSE}(\hat{Y}) = \text{Var}(\hat{Y}) + \|Bias(\hat{Y})\|^2, \tag{4.2}$$

where $\text{Var}(\hat{Y}) = E\left[\|\hat{Y} - E[\hat{Y}]\|^2\right]$, the centered variance of the estimator $\hat{Y}$.

Considering this, it is clear that $\text{MSE}(g_k) = \text{Var}(g_k)$. On the other hand, it is also possible to conclude that $\text{Var}(\hat{\theta}_k) \leq \text{MSE}(\hat{\theta}_k)$. In Section 4.2, we found empirical evidence supporting $\text{MSE}(\hat{\theta}_k) \leq \text{MSE}(g_k)$, from where we conclude $\text{Var}(\hat{\theta}_k) \leq \text{Var}(g_k)$, which ensures the variance reduction obtained via COCO coupling.

However, again from Equation (2.14), it is possible to observe that the "level" in which SGD convergence curve stabilizes is directly related to the uncentered variance of the gradient estimator. In turn, the uncentered variance can be related to the MSE via

$$E\left[\|\hat{\theta}_k\|^2\right] = \text{MSE}(\hat{\theta}_k) + \|E[\hat{\theta}_k]\|^2 - \|Bias(\hat{\theta}_k)\|^2,$$

which can be easily derived from Equation (4.2).

This expression does not yield an obvious result. But note that, when we couple COCO_K to Stochastic Gradient Descent (SGD), the update rule becomes $x_i = x_{i-1} + \gamma\ \hat{\theta}_{i-1}$, allowing us to get some insight on the uncentered variance of the COCO estimator by the "level" at which both (SGD and its COCO coupling) stagnate. In order to assess that, Figure 4.17 compares the performance of SGD coupled with COCO_2 with the original SGD in two different situations: one resorts to an oracle of the same variance (σ^2) and the other to an oracle whose variance is halved ($\sigma^2/2$), as in that case the MSE is halved as well.

From this figure, it is possible to observe that, by using the denoiser, unfortunately the oracle uncentered variance does not decrease as much as $\text{MSE}(\hat{\theta}_k)$, fact which is assigned to the bias of the COCO estimator.

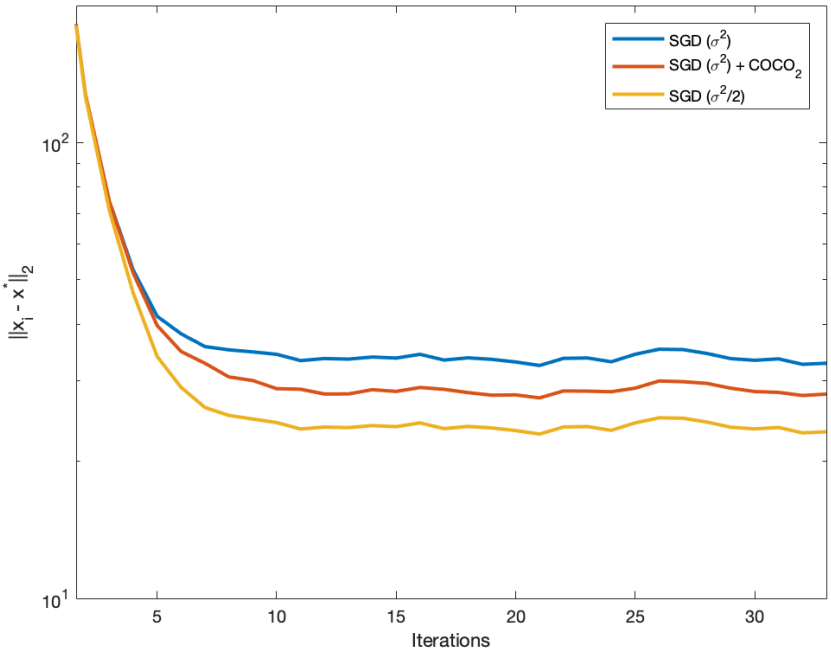


Figure 4.17: Performance gain with the proposed computationally simplest denoiser, COCO$_2$.

The Effect of Using a Wrong Lipschitz Constant

When applying the COCO$_K$ denoiser to a baseline algorithm, three additional hyperparameters have to be considered: Σ, K and L.

Regarding the choice of Σ, it has to follow the model to which we provide a solution method: $\Sigma = \sigma^2\ I$. The different constants σ^2 that the user might choose yield the same QCQP. Thus, the choice of σ^2 does not put at stake the performance of the denoiser. In the case of K, it is a choice of the user in the trade-off performance *versus* time of computation. However, the selection of L might imperil the algorithm's performance. Therefore, it is of the utmost importance to correctly pick this value, as well as to understand the different outcomes that different choices might lead to.

In order to assess the effect of a wrong choice of L, the results from Figure 4.16 for SGD + COCO$_8$ are recovered. Note that, in this case, we have $L = L_{real} = 1$ and so we are interested in "recording" the algorithm's performance considering different values of L. These results are represented in fig. 4.18.

As expected, from this figure it is possible to verify that the better results are for the correctly estimated L. Moreover, when we have $L > L_{real}$, we can see that the algorithm seems to always achieve better performance than SGD, even if only marginally. This is a direct consequence from the fact that, when we overestimate excessively the L ($\Delta_L \to \infty$), the COCO$_K$ estimators tend to the ones provided by the oracle. On the other hand, for underestimated L ($L < L_{real}$), surprisingly some good results are

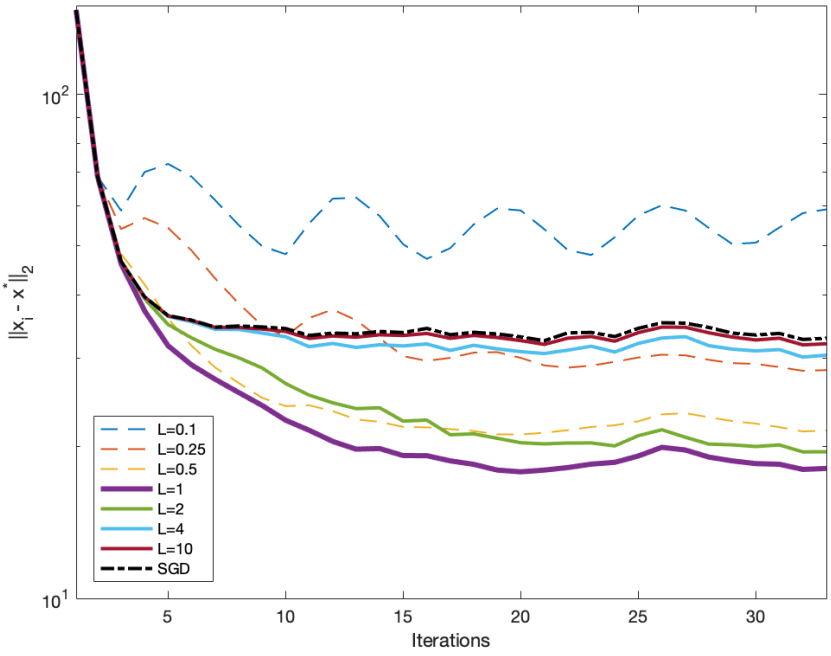


Figure 4.18: Performance of COCO$_8$ for several choices for the denoising parameter L, when $L_{real} = 1$.

observed ($L = 0.5$ and $L = 0.25$) in terms of the variance regime. These results are assigned to the fact that, in Figure 4.4, there are some values of Δ_x for which the MSE$(\hat{\theta}_k) <$ MSE(g_k). For $L = 0.1$, the steps given by the algorithm might fall in a region where we already have MSE$(\hat{\theta}_k) >$ MSE(g_k), leading to a worse performance than SGD.

An interesting observation comes from the ripples generated in the case of underestimation of L, which is a consequence of an excessive smoothening of the objective function. Moreover, it was also verified that the smaller the L considered, the higher the computation time of the iterative solution method to COCO, as the co-coercivity constraints are likely to be more severely violated by the gradient estimates provided by the oracle, requiring more iterations from the FDPG method to find a solution. These two considerations might play a pivotal role in settings where the user is not able to compute the true L and, therefore, require the tuning of this hyperparameter.

Stabilization through COCO

In Section 4.2, we suggested that COCO$_K$ could be interpreted as an extension of an averaging strategy of the gradients estimates provided by the oracle for variance reduction, but in which the oracle consultations come from different points. In fact, even though this gradient averaging technique is different from the Polyak's averaging scheme (see Chapter 2), where the averaging is performed on the iterates, they both share an interesting property: the stabilization of the algorithm for larger step sizes.

Our experimental exploration on COCO$_K$ proceeds to the analysis of this capacity. In order to illustrate it, the step size of the baseline algorithm (in this case, SGD) was increased to a magnitude to which not even GD converges: $\gamma = 2/L_{real} = 2$. The results obtained are depicted in Figure 4.19.

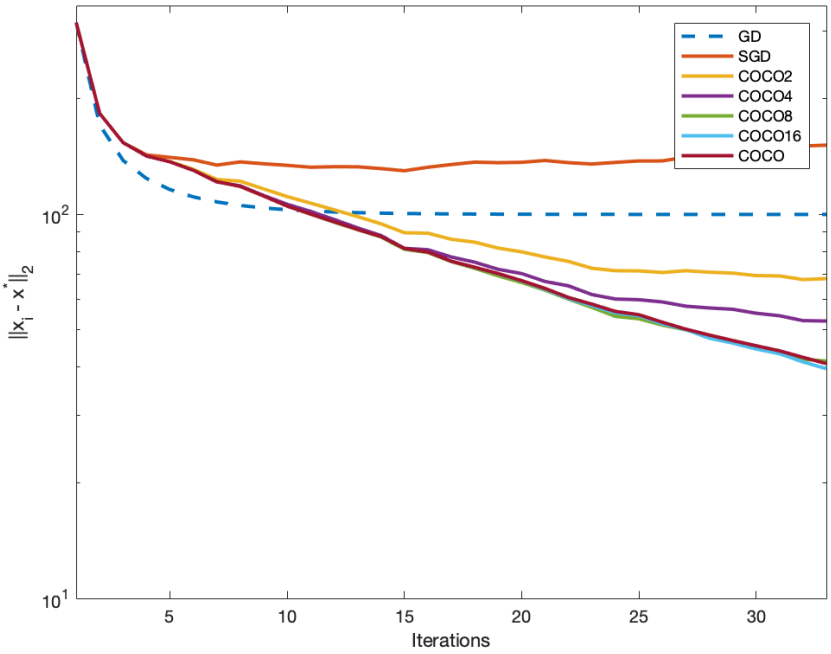


Figure 4.19: Performance of COCO$_K$ for a larger step size. The line for GD is also depicted for reference.

From this figure, it is observable that for this choice of step size, both GD and SGD convergences stagnate way further from the optimum when compared with SGD coupled with COCO$_K$. In fact, the coupling of the baseline algorithm with the denoiser, allowed it to bear larger step sizes, inclusively beating its deterministic version (GD). This result clearly reinforces the benefits of resorting to COCO$_K$, as besides improving the baseline algorithm performance, it also may play an important role in its stabilization.

4.3.4 Real Online Learning Application

In this section, the coupling of COCO$_K$ denoiser to first-order algorithms is now tested in a real context, *i.e.*, are applied to readily available datasets. In particular, we analyse the performance of coupling COCO$_K$ to SGD, Adam, and Stochastic Average Gradient (SAG) in regularized logistic regression problems based on the "fourclass" and "mushrooms" datasets [2]. Therefore, we first introduce the logistic regression problem as a convex and finite sum objective function and then present the results for the mentioned datasets.

Note that, by applying COCO$_K$ to finite sum objectives, we fall out of the assumptions of its formulation, as (i) the different gradients sampled are not completely independent samples (at least the ones coming from the same example in the dataset) and (ii) the oracle's noise does not follow an additive and normally distributed model (even though, by resorting to mini-batches to compute the gradient estimate, its distribution approximates itself from a Gaussian). Nevertheless, considering these limitations, we assume these simulations as robustness tests to the proposed framework.

Logistic Regression

The datasets considered in this section are organized in the following form: there is one feature matrix, A, where each example, a_i, is displayed in each row, and a vector of labels, b, *i.e.*,:

$$A = [a_1, \dots, a_n]^T \quad \text{and} \quad b = [b_1, \dots, b_n]^T,$$

where A is a matrix with n rows and d columns, while b is a n-dimensional vector. Considering this structure, the type of problems that we approach - logistic regression with *Tikhonov* regularization - are a specific instance of convex finite sum objectives, which can be formulated as:

$$\min_x \quad \frac{1}{n}\sum_{j=1}^{n} \mathcal{L}(a_j^T x, b_j) \; + \; \frac{\lambda}{2}\|x\|^2,$$

where $\mathcal{L}(z, b) = \log(1 \; + \; \exp(-bz))$.

Note that this formulation is equivalent to the one in Equation (2.26), with $f_j(x) = \mathcal{L}(a_j^T x, b_j) \; + \; \lambda/2 \; \|x\|^2$. Therefore, it is trivial to verify that

$$\nabla f_j(x) = -\frac{b_j}{1 \; + \; \exp(-b_j \, a_j^T x)} a_j + \lambda x \, .$$

Moreover, denoting by L the Lipschitz constant of ∇f and by L_j the Lipschitz constant of ∇f_j, it can be shown that

$$L = \frac{\|A^T A\|_2}{4n} + \lambda \quad \text{and} \quad L_j = \frac{1}{4}\|a_j\|^2 + \lambda,$$

where $\|\cdot\|_2$ denotes the matrix spectral norm, which corresponds to the matrix maximal singular value.

"Fourclass" Dataset

From the information provided in the previous section, we are in conditions of applying stochastic first-order algorithms to the presented convex problem. The feature matrix A and the vector of labels b used belong to the "fourclass" dataset [2], with $n = 862$ and $d = 2$ (a small dataset). The stochastic first-order algorithms to which the COCO$_K$ coupling is analysed are: (i) SGD as the basic algorithm; (ii) Adam as

a representative of the adaptive step size methods; (iii) SAG as a representative of the recent Variance Reduction (VR) techniques. To begin with, the three methods are plotted for reference in Figure 4.20.

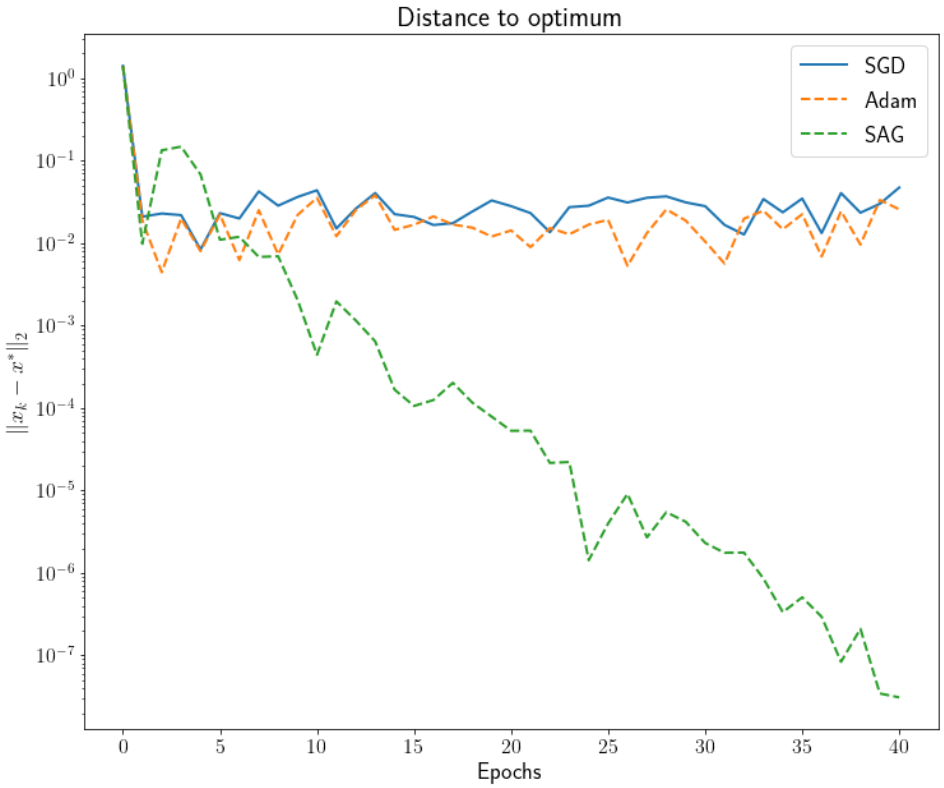


Figure 4.20: Results obtained for SGD, Adam, and SAG, with the dataset "fourclass" [2]. Note the linear convergence of SAG in comparison to the stagnation of SGD and Adam. We used a mini-batch size of 1 and the following hyperparameters: for SGD, step size $\gamma = 1/L$; for Adam, step size $\gamma = 0.008$ and default values for the other parameters ($\beta_1 = 0.9,\ \beta_2 = 0.999,\ = 10^{-8}$); for SAG, step size $\gamma = 1/(\max_j\ L_j)$.

Taking these results into consideration, we now observe the benefits brought by coupling COCO_K to SGD and Adam, considering two different mini-batch sizes: 1 (which was also the one considered in Figure 4.20) and 32. Those results are represented in Figure 4.21 and Figure 4.22. Note that, in these plots, the results are obtained for a single run of the algorithm (with random seeds fixed), contrarily to the plots obtained throughout Section 4.3.3.

From Figure 4.21, it is possible to conclude that the COCO_K is effective when coupled to SGD regardlessly of the mini-batch size. In fact, despite slowing down the convergence on the bias term, it is possible to observe that the gains in the variance regime are recovered. This delay in the bias regime can be assigned to the interpretation of COCO_K as an extension of averaging techniques which considers different iterates, given that this is a typical behavior on those methods. Furthermore, this result is remarkable, given that for a mini-batch size of 1 the argument considering the tendency of the noisy gradient's distribution towards a Gaussian distribution (Central Limit Theorem) is completely implausible. This fact strongly supports the robustness provided by COCO_K.

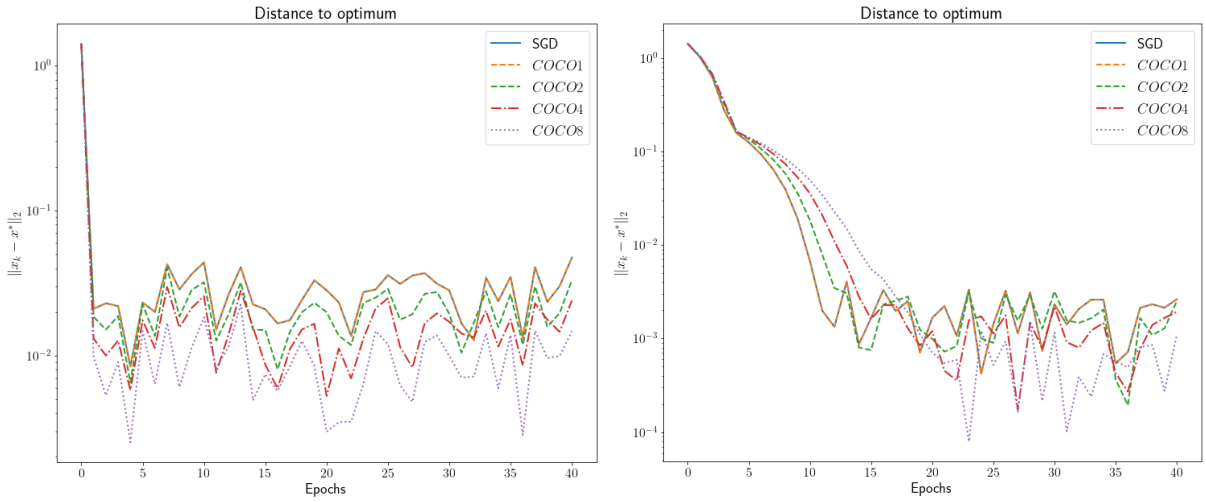


Figure 4.21: Results of SGD with the proposed denoiser COCO$_K$, for the "fourclass" dataset [2], using a mini-batch size of 1 (left) and 32 (right). Note the performance improvement with the increase of K.

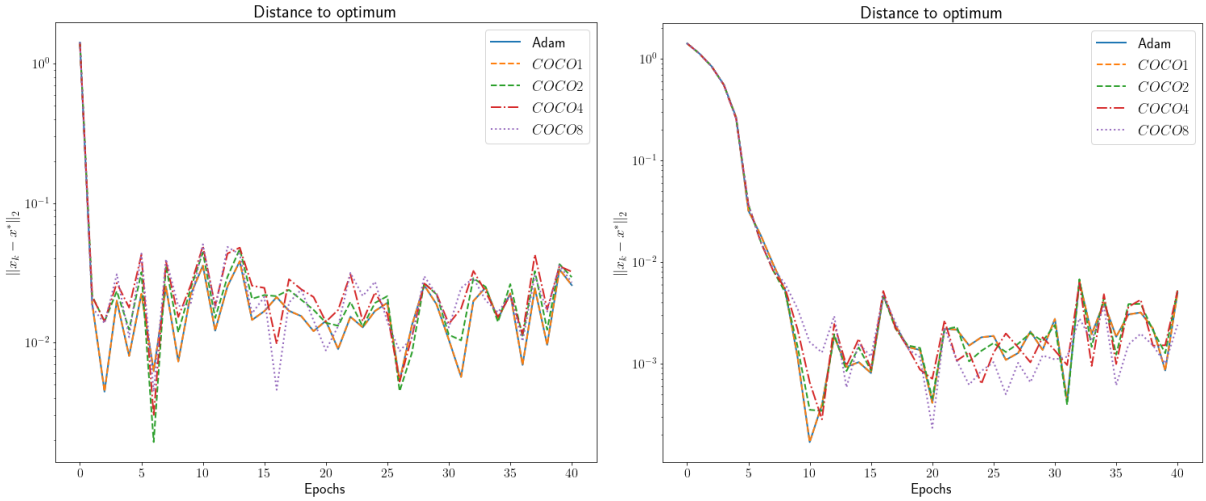


Figure 4.22: Same as Figure 4.21, now for baseline algorithm Adam.

In its turn, the results regarding the coupling of COCO$_K$ to Adam are less conclusive. While for a mini-batch size of 1 there seems not to exist any benefit in using COCO$_K$, for a mini-batch size of 32, some advantage can be easily noticed stemming from it (for the COCO$_8$ it is specially clear). Again, this conclusion can be assigned to the noisy gradients' distribution being different from the one considered in the assumptions of COCO$_K$ for a mini-batch size of 1. It should be noted that, in this case, there is no delay in the bias regime for the coupled COCO$_K$ cases, since Adam adapts the step size in each dimension accordingly to the magnitude of the gradient provided.

Finally, regarding SAG, its coupling with COCO$_K$ is analysed in Figure 4.23. From this plot, it is possible to conclude that the naive coupling between the former and the latter does not bring any benefit. In particular, it halts the linear convergence which characterizes the SAG method. Therefore, this example clearly defines the barrier beyond which COCO$_K$ should not be used. In fact, given that SAG is specifically designed for finite sum objectives, setup which, as previously explained, falls out of the assumptions of COCO$_K$, this failure is understandable.

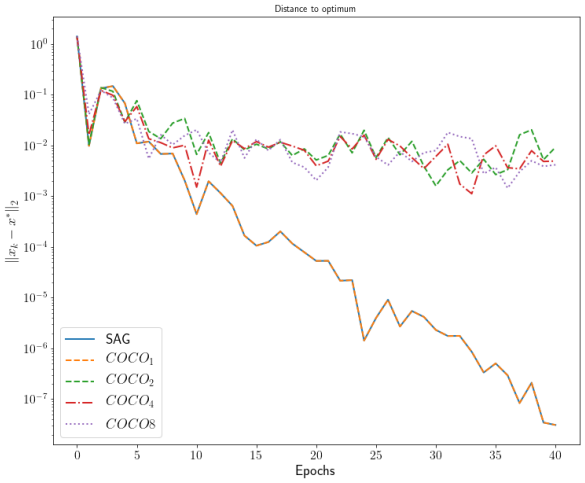


Figure 4.23: Same as Figure 4.21, now for baseline algorithm SAG. In this case, no improvements are observed with the increase in K and the linear convergence of SAG is even compromised by COCO.

"Mushrooms" Dataset

Taking into account the tests performed for the small dataset in the previous section, we now extend those results for a larger one. In this section, we consider the "Mushrooms" dataset [2], with dimensions $n = 8124$ and $d = 112$ on which, again, the Tikhonov regularized logistic regression is solved. The results for SGD and Adam are presented for a mini-batch size of 64 in Figure 4.24.

On the one hand, the results regarding SGD are according to what was expected: a higher K in the COCO$_K$ used leads to a delay in the bias term, compensated by an improved performance in terms of variance regime. On the other hand, the results obtained for Adam, show that the COCO$_K$ coupling

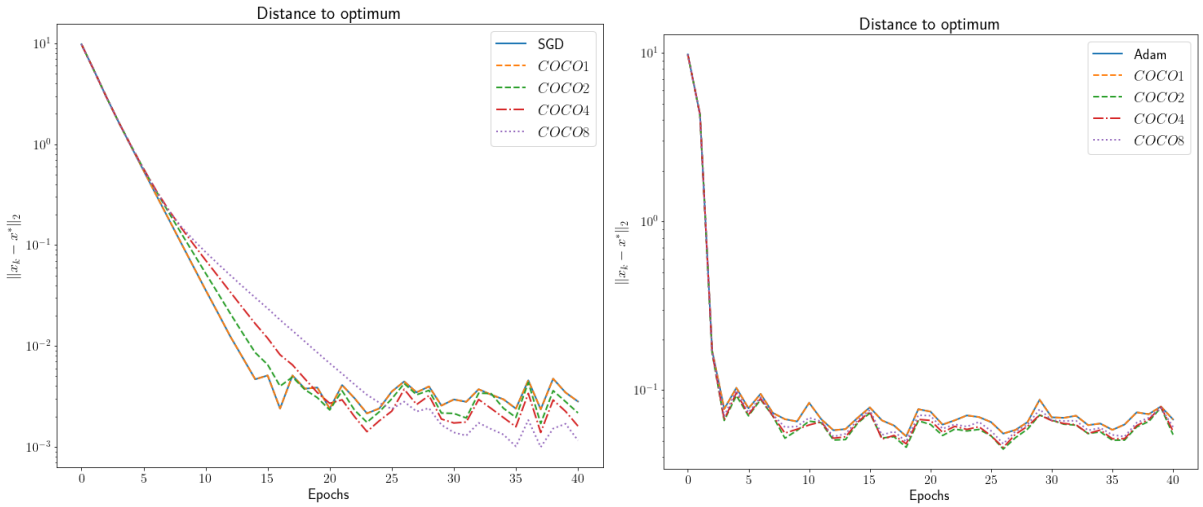


Figure 4.24: Performance of SGD (left) and Adam (right) with COCO_K, for the "Mushrooms" dataset [2]. We used the mini-batch size of 64 (and, in this experiment, a step size of 0.004 for Adam).

brings advantages in the variance regime, even though this improvement is not monotonic in relation to K. In fact, it is possible to observe that in this case, the runs with COCO_4 and COCO_8 are beaten by the one with COCO_2. Nevertheless, these results reinforce the benefit of using the COCO_K denoiser.

5

Conclusion

Contents

This chapter summarizes our work and suggests possible extensions and future research directions.

5.1 Summary

This thesis introduced the COCO denoiser, which exploits co-coercivity of convex and L-smooth objective functions to denoise gradient estimates provided by a stochastic oracle. Our denoiser is based on the joint ML estimation of the gradients, constrained by the co-coercivity conditions. Our theoretical analysis enables finding an interpretable relation between the observations and COCO estimates.

By assuming a noise model with covariance proportional to the identity, we proved that the estimates provided by COCO are necessarily more accurate than the oracle, in what respects to MSE. For this case, we introduced an efficient first-order solution to COCO, based on the FDPG method. By considering the simpler scenario of optimizing a function of a single variable, we conclude that the MSE deteriorates with the distance between the points where the gradients are observed and with the model mismatch in what regards to the Lipschitz constant L.

Our computational experiments corroborate the theoretical results above and have also shown that the elementwise MSE decreases with the rate of $O(1/K)$, where K is the number of gradients simultaneously estimated from sufficiently close points. This is the same rate obtained for a gradient averaging estimator that had access to K observations of the gradient at the same point, which supports interpreting the COCO denoiser as an extension that allows incorporating information from different points.

In stochastic optimization, our experiments with synthetic data have shown that current first-order methods coupled with COCO_K lead to variance reduction, an increase in performance that is noticed even for the case in which only two points are considered. This is particularly relevant because we derived the closed-form solution for COCO_2.

To illustrate the usefulness of COCO in a real online learning task, we solve a logistic regression problem. Although algorithms such as SAG, which exploits the finite sum decomposition of the objective function, do not gain by using COCO estimates, our experiments show that more general baseline algorithms, such as SGD or Adam, clearly exhibit variance reduction.

5.2 Future Work

A simple task that deserves attention in the immediate future is the experimental exploration of the limits of COCO in what respects to dealing with situations that do not fully match the design assumptions. For example, even for problems requiring the (local) minimization of a non-convex function (*e.g.*, deep learning), there is hope for improvement of baseline first-orders methods when coupled with COCO, since the objective function is often locally convex.

First-order algorithms for stochastic optimization can be considered to also estimate (in a non-explicit way) the function gradient (SGD estimates it as the noisy observation itself, while others, *e.g.*, Adam or SAG, have their own operations on the noisy gradient). This observation motivates the possibility of using COCO for stochastic optimization in a slightly different way that the one explored in the thesis: instead of feeding baseline algorithms with the output of COCO, why not feed COCO with the gradient estimates provided by the baseline algorithms? The standard gradient descent steps would then use directly the output of COCO.

Naturally, our theoretical analysis of COCO can be extended. It would be interesting to demonstrate the universality of the evidence provided by our experiments, namely in what respects to the estimator bias and variance (at least for COCO_2, for which there is closed-form solution) and the decrease with K of the elementwise MSE of COCO_K. Regarding the usage of COCO as a plug-in for stochastic optimization, it would be important to study convergence guarantees (which, naturally, also depend on the baseline algorithm) and to quantify the gains in variance reduction.

Aspects of computational efficiency can also motivate future work. For example, the extension of the proposed FDPG efficient method for COCO to deal with more general noise covariance matrices. This would certainly bring robustness to the denoiser, which, despite the predictable higher computational cost, could widen the range of application scenarios. Another interesting line of thought concerns dealing with the quadratic scaling of the number of constraints with the number of points simultaneously considered. In fact, our analysis showed that the larger gains in denoising come from close-by points, which could motivate strategies to reduce (maybe to a linear dependence) the number of constraints that could effectively be considered without compromising the results.

More exploratory lines of research would address the possibility of denoising gradients using different assumptions on the underlying objective function. For example, strong convexity, which has lead to better convergence rates for stochastic optimization algorithms, or the finite sum decomposition that is omnipresent in machine learning applications. Even in the non-convex setting, it could be interesting to consider the single assumption of L-smoothness, since, just as in the convex case, it would prevent arbitrarily fast changes of the gradient, thereby promising denoising capabilities.

Finally, our insights relative to the influence of the location of the query points may motivate strategies for active learning, *i.e.*, for actively selecting those points, rather than passively using only the past iterates.

Bibliography

[1] I. Mitliagkas, "Nesterov's Momentum, Stochastic Gradient Descent," 2020. [Online]. Available: http://mitliagkas.github.io/ift6085-2020/ift-6085-lecture-6-notes.pdf

[2] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: https://doi.org/10.1145/1961189.1961199

[3] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004.

[4] A. Taylor, "Convex interpolation and performance estimation of first-order methods for convex optimization," Ph.D. dissertation, UCL - Université Catholique de Louvain, 2017. [Online]. Available: https://dial.uclouvain.be/pr/boreal/object/boreal:182881

[5] S. Sra, S. Nowozin, and S. J. Wright, Eds., *Optimization for machine learning*, ser. Neural information processing series. Cambridge, Mass: MIT Press, 2012, oCLC: ocn701493361.

[6] S. Bubeck, "Convex Optimization: Algorithms and Complexity," *arXiv:1405.4980 [cs, math, stat]*, Nov. 2015, arXiv: 1405.4980. [Online]. Available: http://arxiv.org/abs/1405.4980

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016.

[8] A. S. Nemirovsky and D. B. Yudin, *Problem complexity and method efficiency in optimization*, ser. Wiley-Interscience series in discrete mathematics. Chichester ; New York: Wiley, 1983.

[9] A. Agarwal, P. L. Bartlett, P. Ravikumar, and M. J. Wainwright, "Information-Theoretic Lower Bounds on the Oracle Complexity of Stochastic Convex Optimization," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3235–3249, May 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6142067/

[10] B. T. Polyak and A. B. Juditsky, “Acceleration of Stochastic Approximation by Averaging,” *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, Jul. 1992. [Online]. Available: http://epubs.siam.org/doi/10.1137/0330046

[11] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[12] M. Schmidt, N. L. Roux, and F. Bach, “Minimizing Finite Sums with the Stochastic Average Gradient,” *arXiv:1309.2388 [cs, math, stat]*, May 2016, arXiv: 1309.2388 version: 2. [Online]. Available: http://arxiv.org/abs/1309.2388

[13] Y. E. Nesterov, *Introductory lectures on convex optimization: a basic course*, ser. Applied optimization. Boston: Kluwer Academic Publishers, 2004, no. v. 87.

[14] B. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, Jan. 1964. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0041555364901375

[15] Y. E. Nesterov, “A method for solving the convex programming problem with convergence rate O(1/kˆ2),” *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983. [Online]. Available: https://ci.nii.ac.jp/naid/10029946121/

[16] G. Lan and Y. Zhou, “An optimal randomized incremental gradient method,” *arXiv:1507.02000 [cs, math, stat]*, Oct. 2015, arXiv: 1507.02000. [Online]. Available: http://arxiv.org/abs/1507.02000

[17] B. Woodworth and N. Srebro, “Tight Complexity Bounds for Optimizing Composite Objectives,” *arXiv:1605.08003 [cs, math, stat]*, Apr. 2019, arXiv: 1605.08003. [Online]. Available: http://arxiv.org/abs/1605.08003

[18] S. Shalev-Shwartz and T. Zhang, “Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization,” in *International Conference on Machine Learning*. PMLR, Jan. 2014, pp. 64–72, iSSN: 1938-7228. [Online]. Available: http://proceedings.mlr.press/v32/shalev-shwartz14.html

[19] X. Zhou, “On the Fenchel Duality between Strong Convexity and Lipschitz Continuous Gradient,” *arXiv:1803.06573 [math]*, Mar. 2018, arXiv: 1803.06573. [Online]. Available: http://arxiv.org/abs/1803.06573

[20] R. M. Gower, M. Schmidt, F. Bach, and P. Richtarik, “Variance-Reduced Methods for Machine Learning,” *arXiv:2010.00892 [cs, math, stat]*, Oct. 2020, arXiv: 2010.00892. [Online]. Available: http://arxiv.org/abs/2010.00892

[21] M. Grant and S. Boyd, “CVX: Matlab Software for Disciplined Convex Programming, version 2.1,” Mar. 2014. [Online]. Available: http://cvxr.com/cvx/

[22] A. Beck and M. Teboulle, “A fast dual proximal gradient algorithm for convex minimization and applications,” *Operations Research Letters*, vol. 42, pp. 1–6, Jan. 2014.

[23] M. Madeira, R. Negrinho, J. Xavier, and P. Aguiar, “COCO - Exploring co-coercivity to filter noisy gradients,” *To be submitted*, 2020.

[24] ——, “Variance Reduction in Stochastic Convex Optimization using Using Co-Coercivity,” *To be submitted*, 2020.

[25] A. Cauchy, “Methode generale pour la resolution des systemes d’equations simultanees,” *C.R. Acad. Sci. Paris*, vol. 25, pp. 536–538, 1847. [Online]. Available: https://ci.nii.ac.jp/naid/10026863174/en/

[26] H. B. Curry, “The method of steepest descent for non-linear minimization problems,” *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261, Oct. 1944. [Online]. Available: http://www.ams.org/qam/1944-02-03/S0033-569X-1944-10667-3/

[27] J. C. Spall, *Introduction to stochastic search and optimization: estimation, simulation, and control*, ser. Wiley-Interscience series in discrete mathematics and optimization. Hoboken, N.J: Wiley-Interscience, 2003.

[28] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951. [Online]. Available: http://projecteuclid.org/euclid.aoms/1177729586

[29] J. R. Blum, “Approximation Methods which Converge with Probability one,” *The Annals of Mathematical Statistics*, vol. 25, no. 2, pp. 382–386, Jun. 1954. [Online]. Available: http://projecteuclid.org/euclid.aoms/1177728794

[30] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization Methods for Large-Scale Machine Learning,” *arXiv:1606.04838 [cs, math, stat]*, Feb. 2018, arXiv: 1606.04838. [Online]. Available: http://arxiv.org/abs/1606.04838

[31] A. Tsybakov, “Optimal Rates of Aggregation,” vol. 2777, Jan. 2003, pp. 303–313.

[32] F. Bach and E. Moulines, “Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n),” *arXiv:1306.2119 [cs, math, stat]*, Jun. 2013, arXiv: 1306.2119. [Online]. Available: http://arxiv.org/abs/1306.2119

[33] A. Dieuleveut, N. Flammarion, and F. Bach, “Harder, Better, Faster, Stronger Convergence Rates for Least-Squares Regression,” *arXiv:1602.05419 [cs, math, stat]*, Feb. 2016, arXiv: 1602.05419. [Online]. Available: http://arxiv.org/abs/1602.05419

[34] Z. Allen-Zhu, “Katyusha: The First Direct Acceleration of Stochastic Gradient Methods,” *arXiv:1603.05953 [cs, math, stat]*, Sep. 2018, arXiv: 1603.05953. [Online]. Available: http://arxiv.org/abs/1603.05953

[35] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, “Accelerating Stochastic Gradient Descent For Least Squares Regression,” *arXiv:1704.08227 [cs, math, stat]*, Jul. 2018, arXiv: 1704.08227. [Online]. Available: http://arxiv.org/abs/1704.08227

[36] J. Duchi, E. Hazan, and Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[37] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *arXiv:1212.5701 [cs]*, Dec. 2012, arXiv: 1212.5701. [Online]. Available: http://arxiv.org/abs/1212.5701

[38] G. Hinton and T. Tieleman, “Lecture 6.5 - RMSprop: Divide the Gradient by a Running Average of Its Recent Magnitude,” *COURSERA: Neural Networks for Machine Learning*, vol. 4, pp. 26–31, 2012. [Online]. Available: //www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[39] T. Dozat, “Incorporating Nesterov Momentum into Adam,” Feb. 2016. [Online]. Available: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ

[40] R. Tibshirani, “Modern Stochastic Methods,” 2019. [Online]. Available: http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/modern-sgd.pdf

[41] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” *arXiv:1904.09237 [cs, math, stat]*, Apr. 2019, arXiv: 1904.09237. [Online]. Available: http://arxiv.org/abs/1904.09237

[42] A. Défossez, L. Bottou, F. Bach, and N. Usunier, “On the Convergence of Adam and Adagrad,” *arXiv:2003.02395 [cs, stat]*, Mar. 2020, arXiv: 2003.02395. [Online]. Available: http://arxiv.org/abs/2003.02395

[43] M. Liu, W. Zhang, F. Orabona, and T. Yang, “Adam$ˆ+$: A Stochastic Method with Adaptive Variance Reduction,” *arXiv:2011.11985 [cs, math]*, Nov. 2020, arXiv: 2011.11985. [Online]. Available: http://arxiv.org/abs/2011.11985

[44] D. Blatt, A. O. Hero, and H. Gauchman, “A Convergent Incremental Gradient Method with a Constant Step Size,” *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51,

Jan. 2007, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/040615961

[45] R. Johnson and T. Zhang, "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 315–323. [Online]. Available: http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf

[46] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives," *arXiv:1407.0202 [cs, math, stat]*, Dec. 2014, arXiv: 1407.0202. [Online]. Available: http://arxiv.org/abs/1407.0202

[47] S. Shalev-Shwartz and T. Zhang, "Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization," *arXiv:1209.1873 [cs, math, stat]*, Jan. 2013, arXiv: 1209.1873. [Online]. Available: http://arxiv.org/abs/1209.1873

[48] J. Konečný and P. Richtárik, "Semi-Stochastic Gradient Descent Methods," *arXiv:1312.1666 [cs, math, stat]*, Jun. 2015, arXiv: 1312.1666. [Online]. Available: http://arxiv.org/abs/1312.1666

[49] J. Mairal, "Optimization with First-Order Surrogate Functions," *arXiv:1305.3120 [cs, math, stat]*, May 2013, arXiv: 1305.3120. [Online]. Available: http://arxiv.org/abs/1305.3120

[50] A. J. Defazio, T. S. Caetano, and J. Domke, "Finito: A Faster, Permutable Incremental Gradient Method for Big Data Problems," *arXiv:1407.2710 [cs, stat]*, Jul. 2014, arXiv: 1407.2710. [Online]. Available: http://arxiv.org/abs/1407.2710

[51] G. Goh, "Why Momentum Really Works," *Distill*, vol. 2, no. 4, p. e6, Apr. 2017. [Online]. Available: http://distill.pub/2017/momentum

[52] R. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, pp. 823–832, 1986. [Online]. Available: https://ci.nii.ac.jp/naid/10012746305/

[53] I. Mitliagkas, "Accelerated Methods - Polyak's Momentum (Heavy Ball Method)," 2020. [Online]. Available: http://mitliagkas.github.io/ift6085-2020/ift-6085-lecture-5-notes.pdf

[54] L. Lessard, B. Recht, and A. Packard, "Analysis and Design of Optimization Algorithms via Integral Quadratic Constraints," *arXiv:1408.3595 [cs, math]*, Oct. 2015, arXiv: 1408.3595. [Online]. Available: http://arxiv.org/abs/1408.3595

[55] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv:1609.04747 [cs]*, Jun. 2017, arXiv: 1609.04747. [Online]. Available: http://arxiv.org/abs/1609.04747

[56] E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson, “Global convergence of the Heavy-ball method for convex optimization,” in *2015 European Control Conference (ECC)*, Jul. 2015, pp. 310–315.

[57] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, “Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization,” *arXiv:1506.07512 [cs, stat]*, Jun. 2015, arXiv: 1506.07512. [Online]. Available: http://arxiv.org/abs/1506.07512

[58] H. Lin, J. Mairal, and Z. Harchaoui, “Catalyst Acceleration for First-order Convex Optimization: from Theory to Practice,” *Journal of Machine Learning Research*, vol. 18, no. 212, pp. 1–54, 2018. [Online]. Available: http://jmlr.org/papers/v18/17-748.html

[59] K. Basu, A. Saha, and S. Chatterjee, “Large-Scale Quadratically Constrained Quadratic Program via Low-Discrepancy Sequences,” Oct. 2017. [Online]. Available: https://arxiv.org/abs/1710.01163v1

[60] R. Tibshirani, “Karush-Kuhn-Tucker Conditions,” 2019. [Online]. Available: https://www.stat.cmu.edu/~ryantibs/convexopt/lectures/kkt.pdf

[61] A. Beck and M. Teboulle, “A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems,” *SIAM J. Imaging Sciences*, vol. 2, pp. 183–202, Jan. 2009.