SAPIENZA
UNIVERSITÀ DI ROMA

# Optimization techniques for large scale finite sum problems


Scuola di Scienza e Tecnologia dell'informazione e delle comunicazioni

Dottorato di Ricerca in Automatica Bioingegneria e Ricerca Operativa – XXXII Ciclo

Candidate

Tommaso Colombo
ID number 1308415

Thesis Advisor

Prof. Stefano Lucidi

Co-Advisor

Prof. Alberto De Santis


A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Automatica Bioingegneria e Ricerca Operativa

October 31, 2019

Thesis defended on February 21, 2020
in front of a Board of Examiners composed by:

Prof. Gianluca Antonelli, Prof. Luca Faes, Prof. Marco Sciandrone (chairman)

---

**Optimization techniques for large scale finite sum problems**
Ph.D. thesis. Sapienza – University of Rome



This thesis has been typeset by LaTeX and the Sapthesis class.

Version: March 17, 2020

Author's email: tommaso.colombo@uniroma1.it

# Abstract

With the explosion of machine learning and artificial intelligence applications, the need for optimization methods specialized in the training of such models has been steadily growing for the last 10-20 years. Indeed, given the big data regime and the special structure of the optimization problems to be solved in these settings, a number of new, efficient optimization methods have been developed.

A large amount of these new methods strongly rely on the finite sum structure of the objective function to be minimized, namely on the fact that the function $f$ can be written as $f(w) = \frac{1}{N}\sum_{i=1}^{N} f_i(w)$, where the indices $i = 1,\dots,N$ often refer to the availability of $N$ input-output pairs on which the model should be trained, i.e. the training set. Nevertheless, this is not the only application where a finite sum structure of the objective function appears. Indeed, beyond the training of Neural Networks (NN) and Support Vector Machines (SVM), which depend by definition on a dataset of input-output pairs, a finite sum structure can also be recognized in Reinforcement Learning (RL) applications, due to the need of estimating expected values by sample approximation. In all these cases, $N$ is usually huge, in the order of millions, or even billions, therefore making the exact computation of the function and gradient infeasible for many real life applications.

This is one of the reasons why the field has seen a flourishing of publications from the most diverse communities, beyond the operations research one, for example the dynamical control, computer science, stochastic optimization ones. Many new methods have been developed by these communities, both deterministic and stochastic algorithms, although their comparison is made difficult by the different approaches coming from the different communities the new algorithms belong to.

Due to the above considerations, the focus of this dissertation is on how to solve optimization problems where the function is structured as a finite sum of component functions. In this finite sum setting, a function $f_i$ can be referred to as a component function, and its gradient $\nabla f_i$ as a component gradient.

In particular, a deep investigation of the algorithms developed so far to solve such problems is carried on, with a specific interest in showing the similarities and differences of the convergence analysis when it is developed in the deterministic vs stochastic cases. The target of the investigation is the case when the component gradients are continuously differentiable, and easily computable, like in many machine learning settings (e.g., neural networks training).
In this framework, dynamic minibatching schemes are addressed. These are employed to determine the size of the sample to be used during the optimization process, especially in gradient-based methods, when the gradient is estimated by subsampling the component gradients, namely, when it is estimated based on a subset of the indices $\{1,\dots,N\}$. The aim of dynamic minibatching schemes is to dynamically test the quality of the gradient approximation, and consequently suggest if the sample size should grow or not. A new technique is proposed, based on statistical analysis of the gradient estimates. The new technique is based on the well-known Analysis of Variance (ANOVA) test, and the convergence of a subsampled gradient-based method is proved when such technique is employed. Numerical experiments are reported on standard machine learning tasks, like (nonlinear) regression and binary classification.

Then, the derivative free setting is explored, i.e. the setting where the component functions come from a black-box-like process and the component gradients are not directly available. An example of such setting is policy optimization for reinforcement learning, where only sample approximations of the stochastic reward function are available. Therefore, in literature, Derivative Free Optimization (DFO) methods have been applied to solve this problem, in particular by trying to estimate the gradient by computing only sample approximations of the function. An analysis of the convergence guaran-

tees of stochastic optimization methods in this setting is performed, showing that approximating the gradient by only computing sample-based estimates of the function brings a further approximation error, leading to poorer theoretical results. The special case of policy optimization for reinforcement learning is analysed, showing that such application is even harder, since the sample approximation of the function, in general, does not have continuity guarantees.

Finally, a new class of distributed algorithms is introduced to solve linearly constrained, convex problems, with potential application to the dual formulation of the support vector machines training problem. This employs augmented Lagrangian and primal-dual theory to develop a simple, distributable and parallelizable class of algorithms to solve convex problems with simple bound and hard (i.e. coupling all the variables), linear constraints. Such class of algorithms is of particular interest for training support vector machines, since it allows to fully distribute the data, i.e. the input-output pairs, to the available parallel processes, simplifying the (often infeasible) storage of such large amount of data.

## Acknowledgments

*The author wants to thank Stefano Lucidi and Alberto De Santis, for the daily support and fundamental advise during these 3 years.*

*Special thanks to Laura Palagi and Simone Sagratella for the fruitful research carried on together.*

# Contents

# Chapter 1

# The finite sum problem

The subject of this dissertation is the solution of the so-called finite sum problem, i.e.

$$\min_{w\in\mathbb{R}^d}\left\{f(w)=\frac{1}{N}\sum_{i=1}^{N}f_i(w)\right\}, \tag{1.1}$$

where $f_i:\mathbb{R}^d\to\mathbb{R}$, $i=1,\ldots,N$, are named the *component functions* and $N$ is huge, namely in the order of millions, or even billions.

In Machine Learning (ML), the training phase of three very well-known models need to solve an optimization problem like problem 1.1: Neural Networks (NN), Reinforcement Learning (RL) and Support Vector Machines (SVM). In particular,

(i) for Neural Networks training, the gradients of the component functions $\nabla f_i$ are assumed to be $L_w$-Lipschitz continuous and can be computed efficiently, thanks to the well-known back-propagation algorithm [51];

(ii) for policy optimization in Reinforcement Learning [107], the components functions $f_i$ are output of a black-box, i.e. the gradients of the component functions can not be computed exactly;

(iii) for Support Vector Machines training [81], the component functions $f_i$ are of the type $f_i = max\{0,g_i\}$, where $g_i:\mathbb{R}^d\to\mathbb{R}$, $i=1,\ldots,N$, are convex and continuously differentiable functions.

## 1.1 Neural Networks training problem

In deterministic optimization, the vast majority of algorithms rely on the computation of the objective function and its derivative. When applied to the training of Neural Networks, this is not always possible. In fact, the objective of such training process is the minimization of the *expected risk*

$$F(w)=\int f(w;x,y)dP(x,y), \tag{1.2}$$

which is impossible to compute since the joint probability distribution $P(x,y)$ is not known. Therefore, the *empirical risk* is minimized

$$\min_{w\in\mathbb{R}^d} f(w)=\left\{\tfrac{1}{N}\textstyle\sum_{i=1}^{N}f_i(w)\triangleq\tfrac{1}{N}\sum_{i=1}^{N}\phi(w;x^i,y^i)\right\}, \tag{1.3}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ are assumed to be continuously differentiable for all $i = 1, \ldots, N$ with Lipschitz-continuous gradients and $\{x_i, y_i\}^i, i = 1, \ldots, N$, are the input-output samples available.

The reason why problem 1.3 has been deeply investigated for the last decade is that training a (Deep) NN requires the solution of an unconstrained finite sum problem. Figure 1.1 represents a NN with 4-dimensional inputs (i.e. $n = 4$), 3-dimensional outputs and two hidden layers of 5 and 7 neurons respectively. $W_1 \in \mathbb{R}^{5\times4}$, $W_2 \in \mathbb{R}^{7\times5}$ and $W_3 \in \mathbb{R}^{3\times7}$ are the weight matrices between the input and first hidden, first hidden and second hidden, second hidden and output layers, respectively. The biases could easily be included by adding a neuron with fixed value of 1 to each hidden layer [48], but are not fundamental to the aim of this example. $g_1(\cdot)$ and $g_2(\cdot)$ are the so-called activation functions, that can be linear, piece-wise linear or nonlinear, depending on the degree of nonlinearity needed in the model. In particular, the output of a hidden layer $h$ relative to a generic input $x$ can be written as

$$o_h(x) = g_h(W_h o_{h-1}(x)) + b_h,$$

where $g_h$ is the (non-)linear activation function just introduced. Altogether, the example in figure 1.1 would imply solving an optimization problem like problem 1.3 with respect to a vector of variables $w \in \mathbb{R}^d$, with $d = 20 + 35 + 21 = 76$, representing the three above mentioned weight matrices.

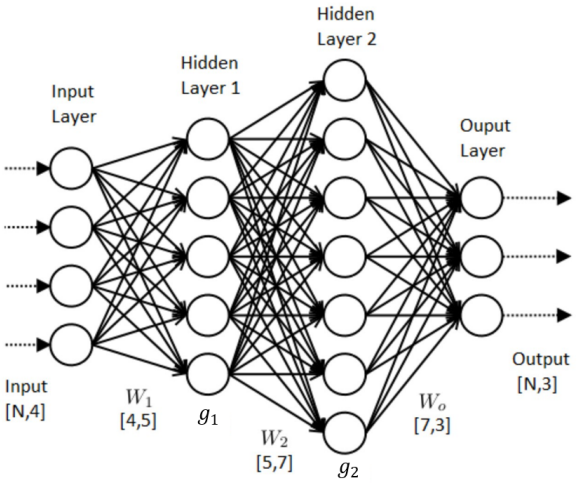


**Figure 1.1.** A two-hidden layer Neural Network

More formally, a generic, $H$-hidden-layered NN produces a predicted output $\widehat{y}^i$ by computing

$$\widehat{y}^i(w) = g_O(W_O * g_H(W_H * g_{H-1}(W_{H-1} * g_{H-2}(\ldots(g_2(W_1 x^i) + b_1)\ldots) + b_{H-1}) + b_H) + b_O), \qquad (1.4)$$

where $g(\cdot)$ represents the component-wise application of the activation function $g$, $W_1, \ldots, W_H$ and $b_1, \ldots, b_H$ represent the weight matrices and biases of the $H$ hidden layers, and $W_O, b_O$ represent the weight matrix and bias of the output layer. The objective of the training problem is to solve

$$\min_{w\in\mathbb{R}^d} \left\{ f(w) = \tfrac{1}{N}\textstyle\sum_{i=1}^N f_i(w) = \tfrac{1}{N}\sum_{i=1}^N \tfrac{1}{2}\|\widehat{y}^i(w) - y^i\|^2 \right\}, \qquad (1.5)$$

where $d$ can be very large if, for example, the number of hidden layers $H$ and the number of neurons of each layer $n_h$, $h = 1, \ldots, H$, are very large, or if the size of the input samples $n$ is huge. If the activation functions $g(\cdot)$ are linear, then problem 1.5 is a convex problem and the optimum value is unique. In general, $g(\cdot)$ is nonlinear since this is necessary to make NN a universal function approximator [54].

Notice that the function $f$ defined in (1.3) is an approximation of function $F$ defined in (1.2), since the unknown, continuous probability distribution $P$ is substituted by a discretized sample average approximation. This notwithstanding, the number of samples $N$ available in current real applications can be in the order of millions, or billions. For this reason, even computing the function $f$ as the sum of all the functions $f_i, i = 1,\dots,N$, may be prohibitive from a computational point of view. The same applies to the gradient calculation. This is the reason behind the renewed interest in the gradient-based method, first introduced by Robbins & Monro [84] in 1951, where the gradient is estimated over a subsample of the available population $S \subseteq \{1,\dots,N\}$.

In literature, this led to an increasing number of algorithms that exploit the structure of $F$ and $f$ to produce a gradient estimate that is as accurate as possible and which computational cost is as low as possible. These gradient approximation methods will be further detailed in chapter 3, but in general have the goal of estimating the gradient by computing only a few, or only one, $\nabla f_i$ at each iteration. These methods usually comprise two components: one in high frequency, aiming to inject "novelty" in the gradient approximation by using the information gained at the current iteration; the other in low frequency, trying to leverage all the past computations to gather more information on the true gradient $\nabla f$. This framework is often denoted as the Stochastic Gradient (SG) framework, due to the fact that the seminal paper [84] comes from the stochastic optimization community. Nevertheless, the analysis of such methods, i.e. methods approximating gradients via subsampling, has been also carried out in the deterministic setting , i.e. the Ingremental Gradient (IG) framework, as will be further explained later.

## 1.2 Policy optimization problem for Reinforcement Learning

In stochastic optimization, often the aim is to minimize the expected value of a certain quantity, namely to solve the problem

$$\min_{w\in\mathbb{R}^d} F(w), \tag{1.6}$$

where $F(w) = E_\zeta[f^0(w,\zeta)]$, and $f^0$ is a function of $w$ and the random variable $\zeta$. The solution of problem 1.6 arises, e.g., in simulation-optimization [5] and policy optimization for reinforcement learning [106]. Note that problem 1.6, after discretization of the values the random vector $\zeta$ can take, can be seen as the minimization of an infinite sum, i.e.

$$\min_{w\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} f_i^0(w), \tag{1.7}$$

where $f_i^0(w) := \phi(w,\zeta_i)$ and $\zeta_i$ is a realization of the random variable $\zeta$, and $f^0 \to F$ as $N \to \infty$. In practice, such infinite sum is usually approximated by a finite sum of fixed size, e.g., when in simulation-optimization the number of replicates over which the average is computed is fixed a priori.

In the following, reinforcement learning is introduced in more details, in order to show how policy optimization for reinforcement learning can be seen as a finite sum problem too.

In reinforcement learning, an agent acts in a stochastic environment by sequentially choosing actions over a sequence of time steps, in order to maximize a cumulative reward, as sketched in figure 1.2. The reinforcement learning paradigm was introduced almost 30 years ago, but has regained popularity in the last years thanks to the introduction of deep networks to parameterize the policy, aka deep reinforcement learning (DeepRL). These recent advances led to unprecedented results on a number of tasks, from sophisticated robotic manipulations [2, 64, 82, 108], to the control of complex dynamical systems like games, e.g. AlphaGo [97] and Atari games [73].

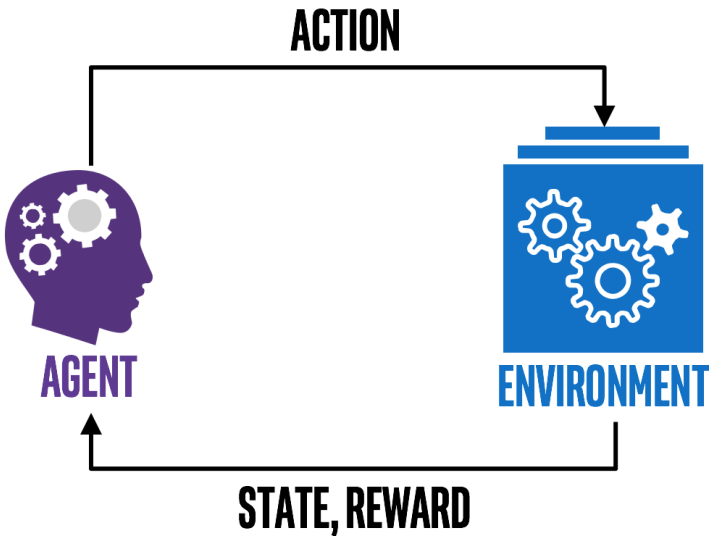


**Figure 1.2.** Generic reinforcement learning framework

Reinforcement learning aims at finding an optimal control (i.e. the policy) for a dynamical system that can be described as

$$s_{t+1} = \phi_t(s_t, a_t, \zeta_t), \tag{1.8}$$

where $\phi_t$ is a function mapping a state $s_t$ to the next state $s_{t+1}$, under the influence of an action $a_t$ and a disturbance $\zeta_t$. The main difference between reinforcement learning and classical optimal control problems is that, in the former, the function $\phi_t$ is neither known nor identified during the process. Indeed, the environment is considered a black box, providing the agent, at each time step, with some quantities representing the current state $s_t$ (e.g., position, velocity, ...) and some notion of reward. The goal of the policy optimization is therefore to design a policy that, applied at every time step, maximizes the cumulative reward received by the agent from the environment.

In the following, the name *episode* will represent a simulation of the actor movement in the environment from time step $t = 1$ to time step $t = T$. In RL literature, an episode can be referred to as a *trajectory*, underlining that a simulation yields a sequence of states and actions $\{s_1, a_1, s_2, a_2, \ldots, s_{T-1}, a_{T-1}, s_T\}$.

A reinforcement learning system can be modeled as a Markov Decision Process (MDP), defined by a tuple $(\mathscr{S}, \mathscr{A}, r, \mathscr{P})$. In the rest of the section, the focus will be on continuous control tasks, where the following can be defined:

- the *state space* $\mathscr{S} \subseteq \mathbb{R}^s$, with the states observed at each time step $t$ identified by $s_t \in \mathscr{S}$, for all $t = 1, \ldots, T$;
- the *action space* $\mathscr{A} \subseteq \mathbb{R}^a$, with the actions taken at each time step $t$ identified by $a_t \in \mathscr{A}$, for all $t = 1, \ldots, T$. The actions are usually modeled as the realization of a normally distributed random variable $A_t \sim N(\mu_A, \sigma_A)$, where $\mu_A$ and $\sigma_A$ are, respectively, its mean and standard deviation;
- the *total reward function* $R : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$

$$R = \sum_{t=1}^{T} r_t,$$

where $r_t \in \mathbb{R}$ are the rewards received from the environment at each time step $t$, modeling the amount of reward gained by the actor over the entire time horizon of an episode;

- the *state-transition probabilities*, modeling the probability of transitioning to a state $s'$ from the current state $s$ taking action $a$, i.e. $\mathscr{P} : \mathscr{S} \times \mathscr{A} \to \mathscr{S}$, assumed to satisfy the Markov property

$$\mathscr{P}(s_{t+1}|s_1, a_1, \ldots, s_t, a_t) = \mathscr{P}(s_{t+1}|s_t, a_t).$$

A deep neural network, aka *policy network*, is usually employed to encode the function $\pi : \mathbb{R}^s \to \mathbb{R}^a$ determining a policy, i.e. the function that given a state $s$ returns the action $a$ to be taken by the agent. A common way of modeling it is to consider the output of the *policy network* as the mean of the normally distributed random variable $A_t$ representing the stochastic actions, namely

$$\mu_A = \pi(w; s_t), \quad \text{for all } t = 1, \ldots, T,$$

where $w \in \mathbb{R}^d$ is the vector of parameters (i.e. the weights of the policy network) to be learned. Notice that the standard deviation is often taken as fixed, i.e. $\sigma_A = 1$, or diminishing during the optimization, i.e. $\sigma_A^k \sim \frac{1}{k}$, where $k$ is the index of the optimization iterations.

The aim of policy optimization for reinforcement learning is to find an optimal policy $\pi^\star$ and, since the policy is parameterized by $w$, the objective is to find $w^\star$ such that the expected value of the total reward is maximized.

Given the function $R$ defined above, one could easily apply a derivative free optimization (DFO) algorithm to maximize a sample approximation of its expected value with respect to the parameters $w$. Indeed, $R$ inherently depends on a black box computation, i.e. the simulation of the episode(s), and therefore its derivatives are not directly computable.

If, with some abuse of notation, the transition probabilities $\mathscr{P}$ include the effect of the policy $\pi$, the objective of a DFO algorithm for policy optimization in RL is therefore to solve the problem

$$\min_{w \in \mathbb{R}^d} F(w), \tag{1.9}$$

where $F(w) = -\mathscr{R}(w) := -E_{\mathscr{P}}[R]$. Observe that there exists stochasticity due to three factors:

1. the sample approximation due to the impossibility of computing an expectation at each iteration, i.e. the number of episodes (replicates) run at each iteration are finite. It is common to run a single episode, or a small minibatch of episodes, at each iteration. In the case of multiple episodes run at each iteration, the episodes can be indexed by $j$ and the total reward would be

$$R = \sum_{j \in B} \sum_{t=1}^{T} r_t^j,$$

   where $B$ is a minibatch of episodes;

2. the simulation of the environment, which is stochastic;

3. the normally distributed actions, the means of which come from the outputs of the policy network, but whose standard deviations may not be 0.

Solving problem 5.1 is incredibly hard. Indeed, the function is nonconvex and potentially nonsmooth and noncontinuous, as will be shown in chapter 5, where some gradient approximation methods used in the RL literature will be introduced and analyzed.

## 1.3 Support Vector Machines training problem

Support Vector Machines had been first developed to solve classification problems, namely to find separating surfaces between (labeled) classes of inputs $x^i$. Assume to have a data set $\{x^i, y^i\}_{i=1}^N$, where $x^i \in \mathbb{R}^n, i = 1,\dots,N$ are the input vectors to be classified and $y^i \in \{-1,+1\}, i = 1,\dots,N$, are the classes. Note that this definition of the $y^i$ implies the problem is a binary classification one. Then, the most straightforward manner to define a classifier is to find a linear operator allowing to "separate" the two classes of points, i.e. the $y^i$ with value $-1$ and the $y^i$ with value $+1$. Furthermore, observe that a linear classifier of this type only exists if the $x^i$ are linearly separable. The linear classifier can be defined by a weights vector $\widetilde{w} \in \mathbb{R}^n$ and a bias $\widetilde{b} \in \mathbb{R}$, and the most straightforward way of classifying the points $x^i$ would be to find $\widetilde{w}, \widetilde{b}$ such that

$$sign(\widetilde{w}^T x^i + \widetilde{b}) = y^i \quad \forall\, i = 1,\dots,N,$$

where the operator $sign(t)$ denotes the sign of $t \in \mathbb{R}$, namely it returns $+1$ if $t \geq 0$, $-1$ otherwise.

One way to do this is to solve the system of inequalities

$$y^i(\widetilde{w}^T x^i + \widetilde{b}) \geq 0, \quad i = 1,\dots,N$$

which, if solvable, guarantees that

$$\begin{cases} y^i \geq 0, & \widetilde{w}^T x^i + \widetilde{b} \geq 0, \\ y^i < 0, & \widetilde{w}^T x^i + \widetilde{b} < 0. \end{cases}$$

Including the bias $\widetilde{b}$ in a unique vector $w = \left[\widetilde{w}, \widetilde{b}\right] \in \mathbb{R}^d$ with $d = n+1$ (this can be done by adding a $-1$ to each input sample $x^i$), the system of inequalities to be solved is

$$y^i w^T x^i \geq 0, \quad i = 1,\dots,N.$$

It is well-known that a solution to a system of inequalities, if a solution exists, can be found by solving an optimization problem. Indeed, in the above case it suffices to solve

$$\text{minimize}_{w \in \mathbb{R}^d} \quad \sum_{i=1}^N \max\{0, -y^i w^T x^i\}, \tag{1.10}$$

where $N$ can be in the order of millions, or even billions. Notice that problem 1.10 is a finite sum problem like problem 1.1. Nevertheless, a solution to problem 1.10 finds *one of the hyperplanes*, which are in general infinite, able to separate the two classes of input samples. Support Vector Machines, instead, aim at finding the *maximum margin hyperplane*, as shown in figure 1.3. It can be shown [81] that this is equivalent to solve problem 1.11, which is similar to problem 1.10 and, again, is a finite sum problem like problem 1.1.

$$\text{minimize}_{w \in \mathbb{R}^n} \quad \sum_{i=1}^N \max\{0, 1 - y_i w^T x_i\} + \tfrac{1}{2}\|w\|^2 \tag{1.11}$$

Problem 1.11 has two issues: (i) it assumes that the the inputs are linearly separable and (ii) it is a nonsmooth problem. To solve (i), the concept of soft margin SVM was developed [81], together with the so-called *kernel trick* that allows to find nonlinear separating curves by implicitly projecting the input samples to higher (infinite) dimensional spaces thanks to the application of a kernel operator $K(\cdot)$ to the inputs $x^i$ [81].

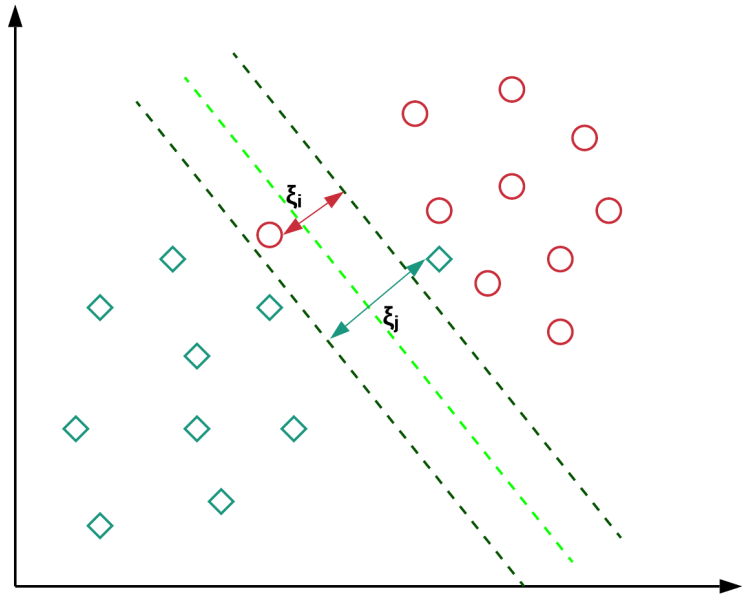


**Figure 1.3.** Linear, soft margin SVM classifier

The linear kernel SVM Problem 1.11 can be rewritten, by including the concept of soft margin, as a constrained problem

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\|w\|^2 + C\textstyle\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i w^T x_i \geq 1-\xi_i, \qquad i=1,\ldots,N, \\ & \xi_i \geq 0, \qquad i=1,\ldots,N, \end{aligned} \tag{1.12}$$

also known as the *SVM primal formulation*, where $\{x_i, y_i\}_{i=1}^{N}$ are the data samples, with $x_i \in \mathbb{R}^n$, $y_i \in \{-1,1\}$. This is a convex, linearly constrained optimization problem with many inequality constraints, i.e. one for each data sample. By duality theory (see, e.g., [81]), the corresponding *SVM dual formulation* can be written as

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\alpha^T Q\alpha - \mathbb{1}^T\alpha \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \qquad i=1,\ldots,N, \\ & y^T\alpha = 0, \end{aligned} \tag{1.13}$$

where the matrix $Q$ is the so-called *kernel matrix*.

Problem 1.13 is a quadratic convex problem with one single (difficult) linear constraint over all the variables $\alpha_i$, $i=1,\ldots,N$, and many simple box constraints. The constraint

$$y^T\alpha = 0 \tag{1.14}$$

is known in literature as *coupling constraint*, since it couples all the variables of the problem in one equality constraint, thus making problem 1.13 hard to parallelize/ distribute. This makes the solution of problem 1.13 hard in practice, since the huge dimension of the dataset, namely the magnitude of $N$,

poses severe challenges in the computation and storage of the above introduced matrix $Q$. This is the reason why many decomposition algorithms have been proposed in literature to solve problem 1.13 in a computationally efficient way. These will be further discussed in chapter 6.

# Chapter 2

# General assumptions, definitions and preliminaries

The assumptions listed below are presented altogether to avoid repetitions throughout the dissertation, but will be recalled in the analysis when necessary.

**Assumption 1** (Lipschitz gradients)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable function. There exists a constant $L_w > 0$ such that*

$$\|\nabla f(w_1) - \nabla f(w_2)\| \leq L_w \|w_1 - w_2\| \quad \forall\, w_1, w_2 \in \mathbb{R}^d. \tag{2.1}$$

*Furthermore, if the function has a finite sum structure, i.e. $f(w) = \sum_{i=1}^N f_i(w)$, there exist $N$ constants $L_w^i, i = 1, \ldots, N$, such that*

$$\|\nabla f_i(w_1) - \nabla f_i(w_2)\| \leq L_w^i \|w_1 - w_2\| \quad \forall\, w_1, w_2 \in \mathbb{R}^d, i = 1, \ldots, N.$$

Observe that, in the finite sum setting where $f(w) = \sum_{i=1}^N f_i(w)$, one can choose $L_w = \sum_{i=1}^N L_w^i$.

**Remark 1.** *Assumption 1 implies that $d^T \nabla^2 f(w) d \leq L_w \|d\|^2$, for all $w$.*

**Assumption 2** (Bound on gradient norm)**.** *For all $w \in \mathbb{R}^d$, there exists $B > 0$ such that*

$$\|\nabla f(w)\| \leq B.$$

**Assumption 3** (Strong convexity)**.** *There exists a constant $\mu > 0$ such that for any $d \in \mathbb{R}^d$*

$$d^T \nabla^2 f(w) d \geq \mu \|d\|^2 \qquad \text{for all } w.$$

**Definition 1** (Condition number)**.** *Suppose a function $f$ satisfies both assumptions 1 and 3. Then the condition number can be defined as*

$$\kappa = \frac{L_w}{\mu} \geq 1.$$

**Assumption 4** (Strict convexity)**.** *For all $w, d \in \mathbb{R}^d$ it holds that*

$$d^T \nabla^2 f(w) d > 0.$$

**Assumption 5** (General convexity)**.** *For all $w, d \in \mathbb{R}^d$ it holds that*

$$d^T \nabla^2 f(w) d \geq 0.$$

**Remark 2.** *Assumption 3 implies that*

$$2\mu(f(w)-f^\star) \leq \|\nabla f(w)\|^2 \quad \forall\, w \in \mathbb{R}^d,$$

*where $f^\star$ is the optimal value of $f$.*

**Remark 3.** *Assumption 3 implies that there exists a unique minimizer $w^\star$ and, when assumption 1 also holds, the following relations apply*

$$\|\nabla f(w)\| \leq L_w \|w-w^\star\| \quad \forall\, w \in \mathbb{R}^d,$$

$$f(w)-f(w^\star) \leq \frac{L_w}{2}\|w-w^\star\|^2 \quad \forall\, w \in \mathbb{R}^d.$$

*Furthermore, it holds [75] that*

$$\nabla f(w)^T(w-w^\star) \geq \frac{\mu L_w}{\mu+L_w}\|w-w^\star\|^2 + \frac{1}{\mu+L_w}\|\nabla f(w)\|^2 \quad \forall\, w \in \mathbb{R}^d.$$

All the above properties will be useful in the subsequent analysis. In particular, in chapter 3 some of the most important methods to solve the unconstrained finite sum problem 1.3 will be introduced and analyzed, with the aim of characterizing their main convergence properties, for example:

- the necessary assumptions on the problem, e.g., Lipschitz-continuity, convexity;
- the necessary assumptions on the algorithm, e.g., assumptions on the stepsizes sequence, on the direction employed;
- whether the whole sequence $\{w_k\}$ generated by the iterative method converges to a finite value;
- whether the sequence $\{w_k\}$ generated by the iterative method converges to a solution (stationary point) of problem 1.3, or to a neighborhood;
- the rate of convergence of the sequence $\{w_k\}$.

# Chapter 3

# Optimization for large-scale unconstrained finite sum problems

In this chapter, a review of the most well-known optimization methods to solve large-scale unconstrained finite sum problems is carried on. But first, some results for standard gradient-based optimization methods are reported.

In algorithm 1, a generic gradient-based method is sketched, where the iterate $w_k \in \mathbb{R}^d$ is updated based on a search direction $d_k \in \mathbb{R}^d$ and a stepsize $\alpha_k > 0$.

**Algorithm 1:** Generic gradient-based scheme

```
for k = 0, 1, ... do
    Compute a search direction d_k such that condition (3.1) is satisfied;
    Choose a stepsize α_k > 0;
    Update w_{k+1} = w_k + α_k d_k.
end
```

The typical condition that gradient-based methods assume $d_k$ to satisfy is the angle condition (3.1)

$$d_k^T \nabla f(w_k) \geq -c_1 \|\nabla f(w_k)\|^2, \tag{3.1}$$

for some $c_1 > 0$. This condition guarantees that $d_k$ is a descent direction, which implies that a stepsize $\alpha_k > 0$ exists such that, if the iterate update is $w_{k+1} = w_k + \alpha_k d_k$, then

$$f(w_{k+1}) \leq f(w_k),$$

i.e. there is a descent in the objective function value. In such methods, where the angle condition is guaranteed at each iteration, a wide range of stepsize rules has been developed, including but not limited to fixed stepsizes, diminishing stepsizes, linesearch techniques. In particular, linesearch methods guarantee global convergence to (at least) a stationary point if condition (3.2) holds

$$\lim_{k \to \infty} \frac{\nabla f(w_k)^T d_k}{\|d_k\|} = 0. \tag{3.2}$$

For the fixed stepsize gradient method to converge, instead, it suffices to guarantee that the stepsize is small enough [11, 49], as shown in theorem 1.

**Theorem 1** (Gradient method with constant stepsize)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be continuously differentiable over $\mathbb{R}^d$ and bounded from below. Let assumption 1 holds. Then, employing the iterate update*

$$w_{k+1} = w_k - \alpha \nabla f(w_k)$$

*with*

$$0 < \alpha < \frac{2}{L_w},$$

*it holds that*

$$\lim_{k \to \infty} \nabla f(w_k) = 0.$$

*Proof.* From assumption 1 and Taylor's expansion it holds

$$\begin{aligned} f(w_{k+1}) - f(w_k) &\leq \nabla f(w_k)^T (w_{k+1} - w_k) + \frac{L_w}{2} \|w_{k+1} - w_k\|^2 \\ &= -\alpha \|\nabla f(w_k)\|^2 + \alpha^2 \frac{L_w}{2} \|\nabla f(w_k)\|^2 \\ &= -\alpha \left(1 - \alpha \frac{L_w}{2}\right) \|\nabla f(w_k)\|^2, \end{aligned}$$

which implies $f(w_{k+1}) \leq f(w_k)$ for all $k$, since $0 < \alpha < \frac{2}{L_w}$. The proof follows by recalling that $f$ is bounded from below. □

Unfortunately, for the reasons outlined in section 1.1 of chapter 1, computing the quantity $d_k^T \nabla f(w_k)$ is often not feasible in the setting of problem 1.3, due to the computational expensiveness of the full gradient $\nabla f(w_k)$. Therefore, a more complicated algorithmic scheme is necessary, as introduced in algorithm 2 that presents a general scheme covering most of the methods proposed in the SG framework. In this scheme, the search direction employed at each iteration $k$ of a standard gradient-based algorithm is split in two components:

(i) a component bringing new (computationally cheap) information at the current iteration, $s_k$, where $s$ stands for short-term;

(ii) a component aggregating past information, $l_k$, where $l$ stands for long-term.

The key of such an approach is to be able to compute the two components, $s_k$ and $d_k$,

(i) in a much cheaper way than if the full negative gradient direction was computed;

(ii) so that the direction $d_k = s_k + l_k$ is as close as possible to the full negative gradient direction.

**Algorithm 2:** Generic algorithmic scheme for finite sum problems

```
for k = 0, 1, ... do
    Compute the 'short-term' component s_k;
    Compute the 'long-term' component l_k;
    Choose a stepsize α_k > 0;
    Update w_{k+1} = w_k + α_k d_k, where d_k = s_k + l_k.
end
```

Algorithm 2 is very general and most of the state-of-the-art methods can be recovered by the definition of $s_k$ and $l_k$, including but not limited to the well-known Stochastic Gradient (SG), Incremental

Gradient (IG), Stochastic Average Gradient (SAG), Incremental Average Gradient (IAG), Stochastic Variance Reduced Gradient (SVRG) methods. These methods can roughly be clustered in two groups: the ones that cyclically choose the indices $i_k$ and those that select $i_k$ at random, based on a probability distribution, at each iteration. The former will be denoted as the *deterministic setting*, the latter as the *stochastic setting*. The convergence properties in the two settings rely, respectively, on two closely related quantities:

1. the error $e_k$ of the approximation with respect to the true gradient $\nabla f$ [15]

$$e_k = d_k + \nabla f(w_k);$$

2. the variance $v_k$ of the search direction [19]

$$v_k = Var[d_k] = E[\|d_k - E[d_k]\|^2].$$

Notice that, since in most of the algorithms in the *stochastic setting* it holds that $E[d_k] = -\nabla f(w_k)$, this is often referred to as the gradient approximation variance. It is clear that asking the error $e_k$ goes to 0 in the deterministic setting is equivalent to asking the gradient approximation variance vanishes in the stochastic setting.

Furthermore, these quantities are strictly connected to the stepsize $\alpha_k$. Indeed, unless a diminishing stepsize is employed, a small, fixed stepsize is not enough to guarantee convergence to a solution (stationary point) of problem 1.3, if the gradient approximation error/ variance does not go to zero. This is somewhat in contrast with the result in theorem 1, which guarantees convergence if a small enough, fixed stepsize is employed. Nevertheless, one should recall that the result in theorem 1 relies on (3.28), which automatically bounds the variance of $d_k$ by imposing an angle condition and in turn guarantees function descent at each iteration.

Indeed, while in theorem 1 one can ensure a function descent condition

$$f(w_{k+1}) - f(w_k) \leq -\alpha \left(1 - \alpha \frac{L_w}{2}\right) \|\nabla f(w_k)\|^2,$$

this is never the case in the methods analyzed in this chapter, which usually can ensure a condition like

$$f(w_{k+1}) - f(w_k) \leq -\alpha \left(1 - \alpha \frac{L_w}{2}\right) \|\nabla f(w_k)\|^2 + C\|e_k\|^2,$$

for some positive constant $C$, where it is clear that $e_k$ must be somewhat controlled, in order for the function to be reduced 'often' (since it is not possible to ensure function descent at any given iteration).

In the following, the convergence properties of the main methods introduced for solving large-scale unconstrained finite sum problems are analyzed, with a focus on the differences between the deterministic and stochastic settings.

## 3.1 Deterministic setting

In the deterministic setting, the methods applied to problem 1.3 are usually referred to as *incremental gradient methods*. The main methods in this category are Incremental Gradient (IG), Incremental Aggregated Gradient (IAG) and Double Incremental Aggregated Gradient (DIAG).

As already introduced above, for incremental gradient methods it is not possible to guarantee any conditions on the full gradient $\nabla f(w_k)$, since this is never computed. Indeed, even if an angle

condition like (3.28) was satisfied over the gradient computed based on one sample $\nabla f_i(w_k)$, this would not imply anything for the full gradient $\nabla f(w_k)$.

At the beginning of this chapter, an open question arouse: what convergence properties can be ensured, when the angle condition (3.1) is not be guaranteed and a fixed stepsize is employed? A first answer to this question was given by Solodov [98], who proved a liminf convergence to a neighborhood of a stationary point, as reported in theorem 2, when the iterate update can be written as

$$w_{k+1} = w_k - \alpha(\nabla f(w_k) + e_k),$$

where the term $e_k = -\alpha\delta$ can be seen as an error term, linearly dependent on the stepsize $\alpha > 0$.

**Theorem 2** (Gradient method with error, fixed stepsize)**.** *Let assumptions 1 and 2 hold, and the iterates be in a bounded set $\mathscr{W} \subseteq \mathbb{R}^d$. Apply the iterate update*

$$w_{k+1} = w_k - \alpha(\nabla f(w_k) + e_k),$$

*where $e_k = -\alpha\delta$ is such that $\|e_k\| \leq \alpha C_1 > 0$, and*

$$0 < \alpha < \frac{2}{L_w}.$$

*Then, there exists an accumulation point $\overline{w}$ and a constant $C > 0$ such that*

$$\|\nabla f(\overline{w})\| \leq C\alpha. \tag{3.3}$$

*Moreover, if $\{f(w_k)\}$ converges, then every accumulation point $\overline{w}$ of $\{w_k\}$ satisfies* (3.3)*.*

*Proof.* Lemma 2 yields

$$\begin{aligned} f(w_k) - f(w_{k+1}) &\geq -\nabla f(w_k)^T(w_{k+1} - w_k) - \frac{L_w}{2}\|w_{k+1} - w_k\|^2 \\ &= \alpha\nabla f(w_k)^T(\nabla f(w_k) - \alpha\delta) - \alpha^2\frac{L_w}{2}\|\nabla f(w_k) - \alpha\delta\|^2 \\ &\geq \alpha\left(1 - \frac{L_w}{2}\alpha\right)\|\nabla f(w_k)\|^2 - \alpha^2(1 + L_w\alpha)C_1\|\nabla f(w_k)\| - \frac{L_w}{2}\alpha^4 C_1^2, \end{aligned}$$

where the last inequality follows from the Cauchy-Schwartz inequality. Defining

$$\phi(w) := \left(1 - \frac{L_w}{2}\alpha\right)\|\nabla f(w)\|^2 - \alpha(1 + L_w\alpha)C_1\|\nabla f(w)\| - \frac{L_w}{2}\alpha^3 C_1^2,$$

it holds that

$$f(w_k) - f(w_{k+1}) \geq \alpha\phi(w_k). \tag{3.4}$$

Now let $\{w_k\}$ denote any sequence generated by the algorithm, and suppose

$$\liminf_{k\to\infty} \phi(w_k) > 0.$$

Then there exists an index $\bar{k}$ and an $\varepsilon > 0$ such that $\phi(w_k) \geq \varepsilon$ for all $k \geq \bar{k}$. Therefore, for all $k \geq \bar{k}$

$$f(w_k) - f(w_{k+1}) \geq \alpha\frac{\varepsilon}{2},$$

and, for all $k > \bar{k}$,

$$\begin{aligned} f(w_{\bar{k}}) - f(w_k) &= \sum_{h=\bar{k}}^{k-1} (f(w_h) - f(w_{h+1})) \\ &\geq \sum_{h=\bar{k}}^{k-1} \alpha \frac{\varepsilon}{2} \\ &= (k-\bar{k})\alpha \frac{\varepsilon}{2} \\ &> 0, \end{aligned}$$

which contradicts the hypothesis that $f$ is continuous and $\mathscr{W}$ bounded, since it implies that $\{f(w_k)\} \to -\infty$. Therefore, it holds that

$$\liminf_{k\to\infty} \phi(w_k) \leq 0.$$

Because $\{w_k\}$ is bounded and $\phi$ continuous, an accumulation point $\overline{w}$ exists such that

$$\phi(\overline{w}) \leq 0,$$

which implies

$$\left(1 - \frac{L_w}{2}\alpha\right) u^2 - \alpha(1 + L_w\alpha)C_1 u - \frac{L_w}{2}\alpha^3 C_1^2,$$

where $u := \|\nabla f(\overline{w})\|$. The solution of the inequality yields

$$u \leq \frac{\alpha}{2 - L_w\alpha} \left( (1 + L_w\alpha)C_1 + \sqrt{(1 + L_w\alpha)^2 C_1^2 + 2L_w\alpha C_1^2 (1 - L_w\frac{\alpha}{2})} \right),$$

which can be rewritten as

$$\|\nabla f(\overline{w})\| \leq C\alpha,$$

for some scalar $C > 0$. The proof of the theorem follows by recalling that, if $\{f(w_k)\}$ converges, (3.4) yields

$$\limsup_{k\to\infty} \phi(w_k) \leq 0.$$

□

Note that theorem 2 assumes that the sequence $w_k$ is bounded, which is not restrictive since this is usually the case in neural networks training [98, 117]. This notwithstanding, there is no guarantee that the whole sequence will converge, since, as opposed to the full gradient method, there is no guarantee that the objective function value decreases at each iteration.

In the seminal paper [15], Bertsekas formalized the framework of *gradient method with error*, where the directions employed by the algorithm are affected by an error, namely

$$w_{k+1} = w_k + \alpha_k(g_k + e_k), \tag{3.5}$$

where $g_k$ is a descent direction, e.g. the anti-gradient $-\nabla f(w_k)$, and $e_k$ is an error term, identifying the error in the gradient estimate at each $k$.

In the above setting, beyond assumptions on the stepsize sequence, the following assumptions are needed to prove convergence [15]:

**Assumption 6.** *$g_k$ is a descent direction at $w_k$ and there exist positive scalars $c_1, c_2$ such that for all $k$*

$$\nabla f(w_k)^T g_k \leq -c_1 \|\nabla f(w_k)\|^2, \qquad \|g_k\| \leq c_2(1 + \|\nabla f(w_k)\|). \tag{3.6}$$

*$e_k$ is an error vector satisfying for all $k$ and for some positive scalars $p$ and $q$*

$$\|e_k\| \leq \alpha_k(q + p\|\nabla f(w_k)\|). \tag{3.7}$$

Observe that the above conditions on the error are a generalization of the one by Solodov, where it was assumed that $\|e_k\| \leq \alpha C_1$.

While the first inequality in (3.6) is the standard angle condition, which is assumed to hold on the $g_k$ term of the direction $d_k$, the second inequality, namely

$$\|g_k\| \leq c_2(1 + \|\nabla f(w_k)\|)$$

is interesting, if read in conjunction with (3.7). Indeed, since $d_k = g_k + e_k$, then it can be written that

$$\begin{aligned} \|d_k\| &= \|g_k + e_k\| \\ &\leq \|g_k\| + \|e_k\| \\ &\leq c_2(1 + \|\nabla f(w_k)\|) + \alpha_k(q + p\|\nabla f(w_k)\|) \\ &= c_2 + q\alpha_k + (c_2 + p\alpha_k)\|\nabla f(w_k)\|, \end{aligned}$$

which, in case $\alpha_k = \alpha > 0$ for all $k$, is a growth condition for the direction $d_k$ of the type

$$\|d_k\| \leq \beta_1 + \beta_2\|\nabla f(w_k)\|,$$

with $\beta_1 = c_2 + q\alpha$ and $\beta_2 = c_2 + p\alpha$.

Furthermore, condition (3.7) asks the norm of the error $e_k$ to be somehow 'controlled' through the stepsize $\alpha_k$. Therefore, it is reasonable to expect that convergence of such a method, under the above assumptions, would be difficult in case $\alpha_k$ did not go to zero. Indeed, the following results from [15] are reported to show that a gradient method with error can only converge if the stepsize goes to zero (unless the error term is controlled and driven to zero).

**Lemma 1.** *Consider three sequences $t_k$, $u_k$ and $v_k$ such that $u_k \geq 0$ for all $k$. If*

$$t_{k+1} \leq t_k - u_k + v_k, \quad \forall\, k,$$

*and $\sum_{k=0}^{K} v_k < \infty$ for $K \to \infty$, then either $t_k \to -\infty$ or else $t_k$ converges to a finite value and $\sum_{k=0}^{\infty} u_k < \infty$.*

**Theorem 3** (Gradient methods with error - diminishing stepsize)**.** *Let the iterate update be*

$$w_{k+1} = w_k + \alpha_k(g_k + e_k)$$

*and the stepsizes sequence $\{\alpha_k\} > 0$ be such that*

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty. \tag{3.8}$$

*Let assumptions 6 hold. Then either $f(w_k) \to -\infty$ or else $f(w_k)$ converges to a finite value and*

$$\lim_{k\to\infty} \nabla f(w_k) = 0.$$

*Furthermore, every limit point of $\{w_k\}$ is stationary for $f$.*

*Proof.* Consider two vectors $w_1, w_2 \in \mathbb{R}^d$ and a scalar $\tau$. Define

$$g(\tau) = f(w_1 + \tau w_2),$$

from which

$$\frac{d}{d\tau} g(\tau) = w_2^T \nabla f(w_1 + \tau w_2).$$

It holds that

$$\begin{aligned}
f(w_1 + w_2) - f(w_1) &= g(1) - g(0) \\
&= \int_0^1 \frac{d}{d\tau} g(\tau) d\tau \\
&= \int_0^1 w_2^T \nabla f(w_1 + \tau w_2) d\tau \\
&\leq \int_0^1 w_2^T \nabla f(w_1) d\tau + \left| \int_0^1 w_2^T (\nabla f(w_1 + \tau w_2) - \nabla f(w_1)) d\tau \right| \\
&\leq w_2^T \nabla f(w_1) + \int_0^1 \|w_2\| \|\nabla f(w_1 + \tau w_2) - \nabla f(w_1)\| d\tau \\
&\leq w_2^T \nabla f(w_1) + \|w_2\| \int_0^1 L\tau \|w_2\| d\tau \\
&= w_2^T \nabla f(w_1) + \frac{L}{2} \|w_2\|^2.
\end{aligned}$$

Applying the above relation with $w_1 = w_k$ and $w_2 = \alpha_k(g_k + e_k)$ yields

$$f(w_{k+1}) \leq f(w_k) + \alpha_k \nabla f(w_k)^T (g_k + e_k) + \frac{\alpha_k^2 L}{2} \|g_k + e_k\|^2,$$

and by using assumption 6 it follows

$$\begin{aligned}
\nabla f(w_k)^T (g_k + e_k) &\leq -c_1 \|\nabla f(w_k)\|^2 + \|\nabla f(w_k)\| \|e_k\| \\
&\leq -c_1 \|\nabla f(w_k)\|^2 + \alpha_k \|\nabla f(w_k)\| + \alpha_k p \|\nabla f(w_k)\|^2.
\end{aligned} \tag{3.9}$$

Moreover, by the assumption on Lipschitz gradients and 6 it can be written that

$$\|g_k\|^2 \leq 2c_2^2 (1 + \|\nabla f(w_k)\|^2)$$

and

$$\|e_k\|^2 \leq 2\alpha_k^2 (q^2 + p^2 \|\nabla f(w_k)\|^2),$$

which together yield

$$\begin{aligned}
\|g_k + e_k\|^2 &\leq 2\|\nabla f(w_k)\|^2 + 2\|e_k\|^2 \\
&\leq 4c_2^2 (1 + \|\nabla f(w_k)\|^2) + 4\alpha_k^2 (q^2 + p^2 \|\nabla f(w_k)\|^2).
\end{aligned} \tag{3.10}$$

Using (3.9) and (3.10),

$$\begin{aligned}
f(w_{k+1}) \leq f(w_k) &- \alpha_k (c_1 - \alpha_k p - 2\alpha_k c_2^2 L - 2\alpha_k^3 p^2 L) \|\nabla f(w_k)\|^2 \\
&+ \alpha_k^2 q \|\nabla f(w_k)\| + 2\alpha_k^2 c_2^2 L + 2\alpha_k^4 q^2 L.
\end{aligned}$$

By the definition of the sequence of the stepsizes, $\alpha_k \to 0$ and therefore a positive constant $c$ exists such that, for $k$ large enough,

$$f(w_{k+1}) \leq f(w_k) - \alpha_k c\|\nabla f(w_k)\|^2 + \alpha_k^2 q\|\nabla f(w_k)\| + 2\alpha_k^2 c_2^2 L + 2\alpha_k^4 q^2 L,$$

and, since $\|\nabla f(w_k)\| \leq 1 + \|\nabla f(w_k)\|^2$,

$$f(w_{k+1}) \leq f(w_k) - \alpha_k(c - \alpha_k q)\|\nabla f(w_k)\|^2 + \alpha_k^2(q + 2c_2^2 L) + 2\alpha_k^4 q^2 L.$$

Again, for sufficiently large $k$, the above can be written as

$$f(w_{k+1}) \leq f(w_k) - \alpha_k \beta_1 \|\nabla f(w_k)\|^2 + \alpha_k^2 \beta_2, \tag{3.11}$$

with $\beta_1, \beta_2 > 0$.

Relation (3.11), coupled with lemma 1 and the assumption on the stepsize (6.7), implies that either $f(w_k) \to -\infty$ or else it converges and

$$\sum_{k=0}^{\infty} \alpha_k \|\nabla f(w_k)\|^2 < \infty. \tag{3.12}$$

Assume by contradiction that there exist an $\varepsilon > 0$ and an index $\bar{k}$ such that $\nabla f(w_k) \geq \varepsilon$ for all $k \geq \bar{k}$. Then, it would follow that

$$\sum_{k=\bar{k}}^{\infty} \alpha_k \|\nabla f(w_k)\|^2 \geq \varepsilon^2 \sum_{k=\bar{k}}^{\infty} \alpha_k = \infty,$$

which contradicts (3.12), from which it holds that

$$\liminf_{k\to\infty} \|\nabla f(w_k)\| = 0.$$

Assume now by contradiction that $\limsup_{k\to\infty} \|\nabla f(w_k)\| > 0$. Consequently, there exists $\varepsilon > 0$ such that $\|\nabla f(w_k)\| < \frac{\varepsilon}{2}$ for an infinite number of iterations $k$. This yields that an infinite set of integers $\widetilde{K}$ exists such that for any $k \in \widetilde{K}$ there is an index $j(k) > k$ such that

$$\begin{cases} \|\nabla f(w_k)\| < \frac{\varepsilon}{2}, \\ \|\nabla f(w_{j(k)})\| > \varepsilon, \\ \frac{\varepsilon}{2} \leq \|\nabla f(w_j)\| \leq \varepsilon, \quad k < j < j(k). \end{cases}$$

The relation

$$\begin{aligned} \|\nabla f(w_{k+1})\| - \|\nabla f(w_k)\| &\leq \|\nabla f(w_{k+1}) - \nabla f(w_k)\| \\ &\leq L_w \|w_{k+1} - w_k\| \\ &= \alpha_k L_w \|g_k\| \\ &\leq \alpha_k L_w c_2 \|1 + \nabla f(w_k)\|, \end{aligned}$$

yields that, for any $k \in \widetilde{K}$ large enough to let $\alpha_k L_w c_2 \leq \frac{\varepsilon}{4}$ holds,

$$\nabla f(w_k) \geq \frac{\varepsilon}{4},$$

since otherwise the condition $\nabla f(w_{k+1}) \geq \frac{\varepsilon}{2}$ would be violated. Now assume without loss of generality that the condition above and (3.11) hold for any $k \in \widetilde{K}$. Then, since $\|g_k\| \leq c_2(1+\|\nabla f(w_k)\|)$ and the Lipschitz assumption 1, it holds for all $k \in \widetilde{K}$ that

$$\begin{aligned} \frac{\varepsilon}{2} &\leq \|\nabla f(w_{j(k)})\| - \|\nabla f(w_k)\| \\ &\leq \|\nabla f(w_{j(k)}) - \nabla f(w_k)\| \\ &\leq L_w \|w_{j(k)} w_k\| \\ &\leq L_w \textstyle\sum_{j=k}^{j(k)-1} \alpha_j (\|g_j\| + \|e_j\|) \\ &\leq L_w c_2 \textstyle\sum_{j=k}^{j(k)-1} \alpha_j (1+\|\nabla f(w_j)\|) + L_w \sum_{j=k}^{j(k)-1} \alpha_j^2 (q + p\|\nabla f(w_j)\|) \\ &\leq L_w c_2 (1+\varepsilon) \textstyle\sum_{j=k}^{j(k)-1} \alpha_j + L_w (q+p\varepsilon) \sum_{j=k}^{j(k)-1} \alpha_j^2. \end{aligned} \tag{3.13}$$

Relation (3.13) yields

$$\frac{1}{2L_w c_2 (1+\varepsilon)} \leq \liminf_{k\to\infty} \sum_{j=k}^{j(k)-1} \alpha_j, \tag{3.14}$$

and by using again (3.11),

$$f(w_{j(k)}) \leq f(w_k) - \beta_1 \left(\frac{\varepsilon}{4}\right)^2 \sum_{j=k}^{j(k)-1} \alpha_j + \beta_2 \sum_{j=k}^{j(k)-1} \alpha_k^2 \quad \forall\, k \in \widetilde{K}.$$

Since $f(w_k)$ converges to a finite value and $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$, the above relation yields

$$\lim_{k\to\infty, k\in\widetilde{K}} \sum_{j=k}^{j(k)-1} \alpha_j = 0,$$

which contradicts (3.14). Therefore, if $\bar{w}$ is a limit point of $\{w_k\}$, then $f(w_k)$ converges to the finite value $f(\bar{w})$. Furthermore, it holds $\nabla f(w_k) \to 0$, and consequently $\nabla f(\bar{w}) = 0$. □

The above theorems show that, in the framework of a gradient method with error, convergence to the solution of problem 1.3 can only be guaranteed in the diminishing stepsize setting. The next section applies the above results to prove convergence of the Incremental Gradient (IG) method both in the constant and diminishing stepsize frameworks.

### 3.1.1 Incremental Gradient method

The Incremental Gradient method (IG) has been used for a long time in a variety of large-scale optimization problems, but the first convergence results date back to the end of the last century/ early 2000s, with the works of Solodov [98] and Bertsekas [15], although the latter is the most well-known.

**Algorithm 3:** IG scheme

**Data:** $w_0 \in \mathbb{R}^d$
**for** $k = 0, 1, \ldots$ **do**
  Choose a stepsize $\alpha_k > 0$;
  Set the inner iterate $v_0 = w_k$;
  **for** $i = 1, \ldots, N$ **do**
    Compute $\nabla f_i(v_{i-1})$;
    Update the inner iterate $v_i = v_{i-1} - \alpha_k \nabla f_i(v_{i-1})$;
  **end**
  Update the outer iterate $w_{k+1} = v_N$;
**end**

Algorithm 3 selects, at each inner iteration, an index $i$, in a cyclic fashion. Then, the direction is computed as the negative gradient of the function $f_i$. It is clear that such direction, i.e. $-\nabla f_i$, can be very far from the negative gradient direction, i.e. $-\frac{1}{N}\sum_{i=1}^{N}\nabla f_i$. For this reason, the convergence of such algorithm was proven in [15] only with respect to the outer iterations $k$, imposing a diminishing updating rule to the stepsizes sequence.

Notice that similar considerations can be made for the minibatch version of algorithm 3, where at each inner iteration a minibatch of consecutive indices $S = \{i, i+1, \dots, i+|S|-1\} \subseteq \{1, \dots, N\}$ is selected, instead of a single index. This case will not be investigated, since it does not change the convergence guarantees.

Since the proof relies on the concept of gradient method with error, the update in algorithm 3 must be seen in the framework of (3.5). First, observe that the $N$ inner iterations performed at each outer iteration $k$ in algorithm 3 can be seen as a unique outer iteration

$$w_{k+1} = w_k - \alpha_k \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(v_{i-1}).$$

Secondly, recall that the standard (full) gradient method can be written in the finite sum setting as

$$w_{k+1} = w_k - \alpha_k \frac{1}{N}\sum_{i=1}^{N}\nabla f_i(w_k).$$

Finally, the error can be defined as

$$e_k = \frac{1}{N}\sum_{i=1}^{N}(\nabla f_i(v_{i-1}) - \nabla f_i(w_k)). \tag{3.15}$$

To relate this method to the general algorithm 2, it is necessary to re-index algorithm 3 in order to have only one iteration loop, i.e. only $k$, where at each $k$ an index $i_k$ is selected cyclically from $\{1, \dots, N\}$. Then, the short-term component would be $s_k = -\nabla f_{i_k}(w_k)$ and the long-term one would be null, i.e.

$$w_{k+1} = w_k + \alpha_k(s_k + l_k) = w_k - \alpha_k \nabla f_{i_k}(w_k).$$

Lemma 2, reported from [98], shows that the iterate in algorithm 3, with a constant stepsize, satisfies the relation

$$w_{k+1} = w_k - \alpha\nabla f(w_k) + \alpha^2\delta,$$

where

$$\|\delta\| \leq C_1$$

for some $C_1 > 0$.

**Lemma 2.** *Let assumptions 1 and 2 hold. Then the IG iterate in algorithm 3, with a constant stepsize $\alpha_k = \alpha > 0$ for any k, satisfies*

$$w_{k+1} = w_k - \alpha\nabla f(w_k) + \alpha^2\delta,$$

*where*

$$\|\delta\| \leq C_1$$

*for some $C_1 > 0$.*

*Proof.*

$$\begin{aligned} w_{k+1} &= w_k - \alpha \tfrac{1}{N} \textstyle\sum_{i=1}^{N} \nabla f_i(v_{i-1}) \\ &= w_k - \alpha \tfrac{1}{N} \textstyle\sum_{i=1}^{N} (\nabla f_i(v_{i-1}) - \nabla f_i(w_k) + \nabla f_i(w_k)) \\ &= w_k - \alpha \tfrac{1}{N} \left( \textstyle\sum_{i=1}^{N} \nabla f_i(w_k) + \sum_{i=1}^{N} (\nabla f_i(v_{i-1}) - \nabla f_i(w_k)) \right) \\ &= w_k - \alpha \nabla f(w_k) + \alpha^2 \delta, \end{aligned} \tag{3.16}$$

where

$$\delta := \frac{1}{\alpha N} \sum_{i=1}^{N} (\nabla f_i(v_{i-1}) - \nabla f_i(w_k)). \tag{3.17}$$

Defining

$$\delta_i := \|\nabla f_i(v_{i-1}) - \nabla f_i(w_k)\|, \quad i = 1, \ldots, N, \tag{3.18}$$

it follows that $\delta_1 = 0$ since $v_0 = w_k$. Now, the aim is to show by induction that

$$\delta_i \leq \alpha L_w \sum_{j=1}^{i-1} (1 + L_w \alpha)^{i-1-j} \|\nabla f_j(w_k)\|, \quad i = 2, \ldots, N, \tag{3.19}$$

It holds

$$\begin{aligned} \delta_2 &= \|\nabla f_2(v_1) - \nabla f_2(w_k)\| \\ &\leq L_w \|v_1 - w_k\| \\ &= \alpha L_w \|\nabla f_1(w_k)\|. \end{aligned}$$

Therefore, (3.19) was proved for $i = 2$. Suppose it holds for $i = 2, \ldots, m$, with $m < N$. By using (3.18) and the triangle inequality,

$$\|\nabla f_i(v_{i-1})\| \leq \|\nabla f_i(w_k)\| + \delta_i,$$

which combined with (3.19) yields

$$\|\nabla f_i(v_{i-1})\| \leq \|\nabla f_i(w_k)\| + \alpha L_w \sum_{j=1}^{i-1} (1 + L_w \alpha)^{i-1-j} \|\nabla f_j(w_k)\|, \quad i = 2, \ldots, N. \tag{3.20}$$

Taking in consideration $i = m + 1$, one gets

$$\begin{aligned} \delta_{m+1} &= \|\nabla f_{m+1}(v_m) - \nabla f_{m+1}(w_k)\| \\ &\leq L_w \|v_m - w_k\| \\ &= L_w \left\| \sum_{j=1}^{m} (v_j - v_{j-1}) \right\| \\ &\leq L_w \sum_{j=1}^{m} \|v_j - v_{j-1}\| \\ &= \alpha L_w \sum_{j=1}^{m} \|\nabla_j(v_{j-1})\|, \end{aligned}$$

which, together with (3.20), yields

$$\begin{aligned} \delta_{m+1} &\leq \alpha L_w \sum_{j=1}^{m} \left( \|\nabla f_j(w_k)\| + \alpha L_w \sum_{l=1}^{j-1} (1 + L_w \alpha)^{j-1-l} \|\nabla f_l(w_k)\| \right) \\ &= \alpha L_w \sum_{j=1}^{m} (1 + L_w \alpha)^{m-j} \|\nabla f_j(w_k)\|, \end{aligned}$$

which is exactly (3.19) with $i = m+1$. Therefore, (3.19) is true.

Finally, using (3.17)-(3.19), it holds that

$$\begin{aligned}\|\delta\| &\leq \frac{1}{\alpha}\sum_{i=2}^{N}\delta_i \\ &\leq L_w\sum_{i=2}^{N}\sum_{j=1}^{i-1}(1+L_w\alpha)^{i-1-j}\|\nabla f_j(w_k)\| \\ &\leq c\sum_{i=1}^{N}\|\nabla f_i(w_k)\| \\ &\leq cNB,\end{aligned}$$

for some $c > 0$. The proof follows with

$$C_1 := cNB.$$

□

Lemma 2 can be seen as a bound on the error when an incremental method is employed. In fact, one can write

$$w_{k+1} = w_k - \alpha(\nabla f(w_k) - \alpha\delta) := w_k - \alpha(\nabla f(w_k) + e_k),$$

where the norm of the error $e_k = -\alpha\delta$ is bounded by the quantity $\alpha C_1 > 0$. From the above lemma, thanks to theorem 2, theorem 4 directly follows.

**Theorem 4.** *Let assumptions 1 and 2 hold. Apply the IG iterate in algorithm 3, with a constant stepsize $\alpha_k = \alpha$ for all k such that*

$$0 < \alpha < \frac{2}{L_w}.$$

*Then, there exists an accumulation point $\overline{w}$ and a constant $C > 0$ such that*

$$\|\nabla f(\overline{w})\| \leq C\alpha. \tag{3.21}$$

*Moreover, if $\{f(w_k)\}$ converges, then every accumulation point $\overline{w}$ of $\{w_k\}$ satisfies* (3.21).

Theorem 3, presented in the previous section, proved convergence of a generic algorithm with a diminishing stepsize, where the direction is affected by a deterministic error. Theorem 5 shows how this framework can be easily applied to the setting of algorithm 3, where the error can be written as in (3.15).

**Theorem 5.** *Let $\{w_k\}$ be the sequence generated by algorithm 3 under the assumptions that*

- *$C, D > 0$ exist such that*

$$\|\nabla f_i(w)\| \leq C + D\|\nabla f(w)\| \quad \forall\, w \in \mathbb{R}^d; \tag{3.22}$$

- *the stepsize $\alpha_k$ satisfies*

$$\sum_{k=0}^{\infty}\alpha_k = \infty, \quad \sum_{k=0}^{\infty}\alpha_k^2 < \infty.$$

*Then either $f(w_k) \to -\infty$ or else $f(w_k)$ converges to a finite value and*

$$\lim_{k\to\infty} \nabla f(w_k) = 0.$$

*Furthermore, every limit point of $\{w_k\}$ is stationary for $f$.*

*Proof.* Without loss of generality, assume $N = 2$.

$$v_1 = w_k - \alpha_k \nabla f_1(w_k),$$

$$w_{k+1} = v_1 - \alpha_k \nabla f_2(v_1).$$

Summing,

$$w_{k+1} = w_k + \alpha_k(-\nabla f(w_k) + e_k),$$

where

$$e_k = \nabla f_2(w_k) - \nabla f_2(v_1).$$

Therefore

$$\|e_k\| \leq L_w \|w_k - v_1\| = \alpha_k L_w \|\nabla f_1(w_k)\| \leq \alpha_k (L_w C + L_w D \|\nabla f(w_k)\|),$$

and by using theorem 3 the proof follows. □

Observe that assuming relation (3.22) is inherently asking the error $e_k$ be dependent on the value of the true gradient $\nabla f(w_k)$. Furthermore, this assumption is very similar to the bounded gradient approximation variance assumption required for the convergence analysis in the stochastic setting, as will be shown in the next section. Nevertheless, remark that condition (3.22) do not ask the gradients of the component functions $\nabla f_i$ to be all 0 at a stationary point (where the norm of $\nabla f$ is 0), which is often assumed and is a strong assumption, rarely true in real applications.

The assumption on the stepsize is also quite strong. Indeed, the sequence $\{\alpha_k\}$ is required to be squared summable, but not summable, which hinders the rate of convergence. More advanced results, like IAG and DIAG, rely on the gradient method with errors setting, but at the same time are able to prove some results on the rate of convergence.

As a final remark, observe that the analysis by Solodov [98] requires the error term $e_k$ to satisfy a stronger condition, i.e. $\|e_k\| \leq \alpha C_1$, than the one required by Bertsekas [15], where the error is required to be 'controlled' by the sum of a constant and a quantity linear dependent on the norm of the gradient. Indeed, in order for the condition required by Solodov to hold, a uniform bound on the gradient norm is necessary, while this is not required in the analysis by Bertsekas.

To the best of the author's knowledge, these are the first convergence results for an incremental gradient method. Furthermore, no convergence rate was established. It must be remarked that both theorems 5 and 2 do not assume anything more than Lipschitz-continuity of the gradients. If (strong) convexity of the function was assumed, then it would be easy to get similar bounds on the distance of the function from its optimal value.

Bertsekas [12] more recently proved convergence of IG with a constant stepsize to a neighborhood of the solution, quantifying the neighborhood size, namely proving that, if the function $f$ is bounded from below and the component functions $f_i$ are convex, then

$$\liminf_{k\to\infty} f(w_k) \leq f^\star + \frac{\alpha(4N+1)L_w^2 N}{2}, \tag{3.23}$$

where $f^\star = f(w^\star)$

Regarding the analysis of the convergence rate, it is well-known [11] that methods with a diminishing stepsize like the one in theorem 3 achieve sublinear convergence. Nevertheless, IG algorithms with a constant, small enough, stepsize can be proven to have linear convergence to a neighborhood of the solution [11, Proposition 5]. In [12], a bound on such convergence rate was computed, i.e. proving that to reach the neighborhood in (3.23) with an $\varepsilon$-accuracy one needs to run $K$ iterations, with

$$K = N \left\lfloor \frac{dist(w_0; W^\star)^2}{\alpha\varepsilon} \right\rfloor, \tag{3.24}$$

with $W^\star = \{w \in \mathbb{R}^d : f(w) = f^\star\}$.

In the next subsection a more advanced method, which is a first step to a provable fast convergence to the solution of problem 1.3 with a constant stepsize, will be introduced.

### 3.1.2 Incremental Aggregated Gradient method

**Algorithm 4:** IAG scheme

**Data:** $\nabla f_1(w_0), \nabla f_2(w_1), \ldots, \nabla f_N(w_{N-1})$, $w_N \in \mathbb{R}^d$
Set $i_k = 1$;
**for** $k = N, N+1, \ldots$ **do**
  Compute $\nabla f_{i_k}(w_k)$ and store it;
  Choose a stepsize $\alpha_k > 0$;
  Update the iterate $w_{k+1} = w_k - \alpha_k \frac{1}{N} \sum_{i=1}^{N} \nabla f_{mod(k-i,N)}(w_{k-i})$;
  If $i_k = N$ then $i_k \leftarrow 1$ else $i_k \leftarrow i_k + 1$
**end**

The Incremental Aggregated Gradient (IAG) [17] was proposed in 2007 to answer the question: is it possible to relax the assumptions on the stepsizes sequence made in 5 without compromising its convergence properties, maintain its efficiency in the computation of the search direction and get faster convergence?

The idea behind IAG is pretty simple, namely to compute the gradient on one sample only, $\nabla f_{i_k}(w_k)$, at each iteration, but improving the search direction by using all the information gathered from gradients computed in past iterations. In the framework of algorithm 2,

$$s_k = -\frac{1}{N} \nabla f_{i_k}(w_k) \quad \text{and} \quad l_k = -\frac{1}{N} \sum_{i \neq i_k} \nabla f_{mod(k-i,N)}(w_{k-i}),$$

which corresponds to an iterate that can be written as

$$w_{k+1} = w_k + \alpha_k d_k = w_k - \alpha_k \frac{1}{N} \sum_{i=1}^{N} \nabla f_{mod(k-i,N)}(w_{k-i}). \tag{3.25}$$

The objective of an iterate like (3.25) is clear: compute the gradient of one only component function, but also leverage past information, although that is not computed at the current iterate and therefore is a source of error. Indeed, if $\|w_{k+1} - w_k\| \to 0$, then one would have $\frac{1}{N} \sum_{i=1}^{N} \nabla f_{mod(k-i,N)}(w_{k-i}) \to \nabla f(w_k)$.

Theorems 6-8 report the main convergence results of the method just outlined. These consider two common settings, namely when the functions $f_i$ are Lipschitz-continuous and when those are convex quadratic.

**Theorem 6.** *Assume the iterate update* (3.25) *is executed. Let assumptions 1 and 2 hold, $f(w)$ be bounded and the stepsize be $\alpha_k = \alpha > 0$ for all k, satisfying*

$$\alpha < \frac{1}{2L_w}.$$

*Then*

$$\liminf_{k\to\infty} \|\nabla f(w_k)\| \leq \frac{2L_w B}{1-2\alpha L_w}.$$

Theorem 6 guarantees convergence of a subsequence of $\{w_k\}$, just like theorem 2 for IG method. Before introducing theorem 7, which improves the convergence properties of IAG when the function has better properties, two stronger assumptions are needed.

**Assumption 7.** *$f : \mathbb{R}^d \to \mathbb{R}$ has a unique global solution $w^\star$ and $\nabla^2 f(w)$ is continuous and positive definite at $w^\star$. It follows that a neighborhood $\mathscr{B}(w^\star)$ and positive constants $A_1, A_2, B_1, B_2$ exist such that for any $w \in \mathscr{B}(w^\star)$ it holds*

$$A_1\|w-w^\star\|^2 \leq f(w) - f(w^\star) \leq B_1\|w-w^\star\|^2,$$

$$A_2\|w-w^\star\|^2 \leq \|\nabla f(w^\star)\|^2 \leq B_2\|w-w^\star\|^2.$$

**Assumption 8.** *For any sequence $\{w_k\}$, if $\lim_{k\to\infty} f(w_k) = f(w^\star)$ or $\lim_{k\to\infty} \|\nabla f(w_k)\| = 0$, then $\lim_{k\to\infty} w_k = w^\star$.*

Observe that the above assumptions yield that there exists an $\varepsilon > 0$ such that $w \in \mathscr{B}(w^\star)$ if $f(w) - f(w^\star) < \varepsilon$ or $\|\nabla f(w^\star)\| < \varepsilon$.

One final remark is that if assumption 7 is satisfied and the function is strictly convex, then assumption 8 holds. Furthermore, there exist nonconvex functions that satisfy assumptions 7-8.

**Theorem 7.** *Let assumptions 1, 2, 7 and 8 hold. If the fixed stepsize $\alpha_k = \alpha > 0$ for all k satisfies*

$$\alpha < \min\left\{ \frac{1}{9L_w}, \frac{1}{L_w \max\left\{3\sqrt{\frac{B_1B_2}{A_1A_2}}, \frac{2}{1-2\alpha L_w}\right\}}, \frac{\varepsilon}{3L_wB}, \frac{1}{3L_wB}\sqrt{\frac{A_2\varepsilon}{B_1}} \right\},$$

*then*

$$\lim_{k\to\infty} w_k = w^\star.$$

**Theorem 8.** *Let the functions $f_i$ be defined as*

$$f_i(w) = \frac{1}{2} w^T Q_i w - c_i^T w, \quad i = 1, \ldots, N,$$

*where $\sum_{i=1}^N Q_i$ is positive definite. Then a constant stepsize $\alpha > 0$ exists such that*

$$\lim_{k\to\infty} w_k = w^\star$$

*at a linear rate.*

Theorem 7 is the first convergence result, to the best of the author's knowledge, proving convergence to the solution of a finite sum problem by employing an incremental method with a constant stepsize. This is done at the cost of storing $N$ $d$-dimensonal vectors, which can be expensive from a memory point of view. The key of such a result lies in the aggregation of past computed $\nabla f_i$ that, although computed in different iterates, bring enough information to let the error in the search direction be controlled. A convergence rate result, instead, was developed by Blatt et al. only in the strictly convex quadratic case, as shown in theorem 8. Observe that the positive definiteness assumption on $Q_i$ implies that assumptions 7 and 8 hold.

The work in [110] proved linear convergence rate to a solution using a strong-convexity-like assumption, but without the computation of an explicit linear rate constant. This was derived in the strongly convex case in [50], from which the following theorem is reported

**Theorem 9.** *Let assumptions 1 and 3 hold. Then the IAG iterate $w_k$ with a constant stepsize $0 < \alpha < \overline{\alpha}$, where*

$$\overline{\alpha} = \left(\frac{8\mu}{25NL_w}\right)\frac{1}{\mu + L_w},$$

*is globally linearly convergent to the unique solution of problem 1.3. Furthermore, if $\alpha = \frac{\overline{\alpha}}{2}$ the following hold*

$$\|w_k - w^\star\| \leq \left(1 - \frac{c_N}{(\kappa+1)^2}\right)^k \|w_0 - w^\star\|,$$

$$f(w_k) - f(w^\star) \leq \frac{L_w}{2}\left(1 - \frac{c_N}{(\kappa+1)^2}\right)^{2k} \|w_0 - w^\star\|^2,$$

*where $\kappa \geq 1$ is the condition number as defined in definition 1 and*

$$c_N = \frac{2}{25N(2N+1)}.$$

### 3.1.3 Double Incremental Aggregated Gradient method

**Algorithm 5:** DIAG scheme

**Data:** $w_0, w_1, \ldots, w_{N-1} \in \mathbb{R}^d$, $\nabla f_1(w_0), \nabla f_2(w_1), \ldots, \nabla f_N(w_{N-1})$

Set $i_k = 1$;

**for** $k = N, N+1, \ldots$ **do**

  Compute $\nabla f_{i_k}(w_k)$ and store the pair $\{w_k, \nabla f_{i_k}(w_k)\}$;

  Choose a stepsize $\alpha_k > 0$;

  Update the iterate $w_{k+1} = \frac{1}{N}\sum_{i=1}^N w_{k-i+1} - \alpha_k \frac{1}{N}\sum_{i=1}^N \nabla f_{mod(k-i,N)}(w_{k-i+1})$;

  If $i_k = N$ then $i_k \leftarrow 1$ else $i_k \leftarrow i_k + 1$

**end**

DIAG algorithm [74] was introduced to answer the even harder question: is it possible to design an incremental gradient algorithm, i.e. computing the gradient over only one sample $i_k$ at each iteration, with a convergence rate that is provably better than the one of the full gradient method?

Algorithm 5 answers this question by setting the short- and long-term components of the search direction, respectively, to

$$s_k = \frac{1}{N}\left(w_k - \nabla f_{i_k}(w_k)\right) \quad \text{and} \quad l_k = \frac{1}{N\alpha}\sum_{i=2}^{N} w_{k-i+1} - \frac{1}{N}\sum_{i \neq i_k} \nabla f_{mod(k-i,N)}(w_{k-i}).$$

Therefore, the average is computed over both the last $N$ iterates and the last $N$ component gradients $\nabla f_i$ computed. The authors of [74] justify the idea by comparing DIAG iterate to IAG one. Indeed, IAG iterate can be seen as the solution of the optimization problem

$$w_{k+1} = \arg\min_{w\in\mathbb{R}^d} \left\{ \frac{1}{N}\sum_{i=1}^{N} f_i(y_k^i) + \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(y_k^i)^T (w - y_k^i) + \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\alpha}\|w - w_k\|^2 \right\},$$

where $y$ is updated as

$$y_{k+1}^i = \begin{cases} w_{k+1}, & i = i_k, \\ y_k^i, & \text{otherwise.} \end{cases}$$

This shows that IAG uses, at each iteration $k$, an approximation of each component function that can be written as

$$f_i(w) \approx f_i(y_k^i) + \nabla f_i(y_k^i)^T (w - y_k^i) + \frac{1}{2\alpha}\|w - w_k\|^2,$$

where the first two terms are a first-order approximation of $f_i$ around $y_k^i$ and the last one is a proximal term. This is not customary for proximal algorithms, indeed the proximal term is evaluated around $w_k$ instead of $y_k^i$. Thus, the idea of the authors in [74] was to modify the approximation to get

$$f_i(w) \approx f_i(y_k^i) + \nabla f_i(y_k^i)^T (w - y_k^i) + \frac{1}{2\alpha}\|w - y_k^i\|^2,$$

which corresponds to an iterate that is the solution of the optimization problem

$$w_{k+1} = \arg\min_{w\in\mathbb{R}^d} \left\{ \frac{1}{N}\sum_{i=1}^{N} f_i(y_k^i) + \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(y_k^i)^T (w - y_k^i) + \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2\alpha}\|w - y_k^i\|^2 \right\}.$$

Solving the above, and recalling the definition of $y_k^i$, it follows that DIAG iterate can be written as

$$w_{k+1} = \frac{1}{N}\sum_{i=1}^{N} w_{k-i+1} - \alpha_k \frac{1}{N}\sum_{i=1}^{N} \nabla f_{mod(k-i,N)}(w_{k-i+1}).$$

The convergence results depend on the following fundamental lemma.

**Lemma 3.** *Consider the method defined in algorithm 5. Let assumptions 1 and 3 hold and let the stepsize be $\alpha_k = \alpha = \frac{2}{\mu + L_w}$ for all $k$. Then the sequence $\{w_k\}$ satisfies*

$$\|w_{k+1} - w^\star\| \leq \left(\frac{\kappa - 1}{\kappa + 1}\right) \left(\frac{\|w_k - w^\star\| + \cdots + \|w_{k-N+1} - w^\star\|}{N}\right), \tag{3.26}$$

*where $\kappa$ is the condition number of the function.*

Lemma 3 makes a fundamental statement: the new iterate $w_{k+1}$ is always closer (or has the same distance) to the solution of the problem than the average of the last $N$ iterates. This notwithstanding, observe that such a result holds for an exact value of the stepsize $\alpha$: an open question is if the result would hold if the stepsize was smaller than the given quantity.

**Theorem 10.** *Assume the DIAG algorithm 5 is executed with constant stepsize $\alpha = \frac{1}{\mu + L_w}$. Let assumptions 1 and 3 hold. Then it holds that*

$$\|w_k - w^\star\| \leq a_0 \gamma_0^k \|w_0 - w^\star\|, \tag{3.27}$$

*where*

$$a_0 = \max_{i\in\{1,\dots,N\}} \rho \left(1 - \frac{(i-1)(1-\rho)}{N}\right) \gamma_0^{-i},$$

$\rho = \frac{\kappa-1}{\kappa+1}$ *and* $\gamma_0$ *is the unique solution of*

$$\gamma^{N+1} - \left(1+\frac{\rho}{N}\right)\gamma^N + \frac{\rho}{N} = 0.$$

In [74, Theorem 9], which is not reported for its technicality, the authors prove that the above convergence rate is better than the full gradient descent one in the worst case. This is an unprecedented result, since it states that, at the cost of storing $N$ vectors, the unconstrained finite sum optimization problem can be solved by a method computing only the gradient of one component function at each iteration, with a convergence rate in the strongly convex case that is faster than the one of the full gradient method, which computes the gradients of the $N$ component functions at each iteration!

With the introduction and analysis of the DIAG method, the section dedicated to the incremental gradient methods in the deterministic setting is concluded.

## 3.2 Stochastic setting

In the stochastic setting, methods that rely on the computation of only one component function (or a minibatch of component functions) are known as *stochastic gradient methods* (SG methods), mainly due to (i) the selection criteria for the index $i_k$ at each iteration $k$ and (ii) the techniques adopted for convergence analysis.

As for IG methods, SG methods can not rely on any deterministic angle condition to hold at each iteration. Nonetheless, as opposed to IG methods, SG ones usually rely on an angle condition in expectation, as will be clearer in the next lines. This considerably simplifies the convergence analysis of such methods, and is the reason why SG methods are more widely known to the ML community.

The standard conditions that a stochastic search direction $d_k$ is usually asked to guarantee [19] are:

**Assumption 9.** *There exist constants* $c_g \geq c > 0$, $M, M_v \geq 0$ *such that for all* $k \geq 0$

$$\nabla f(w_k)^T E[d_k] \leq -c\|\nabla f(w_k)\|^2, \tag{3.28}$$

$$\|E[d_k]\| \leq c_g\|\nabla f(w_k)\|, \tag{3.29}$$

$$Var[d_k] \leq M + M_v\|\nabla f(w_k)\|^2, \tag{3.30}$$

where $Var[d_k]$ is the variance of the vector $d_k$, namely

$$Var[d_k] := E\left[\|d_k - E[d_k]\|^2\right] = E\left[\|d_k\|^2\right] - E\left[\|d_k\|\right]^2.$$

**Remark 4.** *Let assumption 9 hold. Then*

$$E\left[\|d_k\|^2\right] \leq M + M_G\|\nabla f(w_k)\|^2,$$

*where* $M_G = M_v + c_g \geq c^2 > 0$.

Condition (3.28) is the stochastic counterpart of the classical angle condition (3.6), assumed in expectation, while condition (3.29) asks the first moment of the stochastic direction to be proportional to the norm of the true gradient. The inequality (3.30), instead, imposes a growth condition to the variance of the direction $d_k$, asking it does not grow more than linearly with respect to the squared norm of the true gradient. Observe that (3.30), in the case $d_k = -\nabla f_{i_k}(w_k)$, lets the gradients of the component functions be bounded away from zero even when at the solution (where the squared norm of the true gradient, $\|\nabla f(w_k)\|^2$ is equal to 0).

Sometimes, a stronger condition like

$$E[d_k] = -\nabla f(w_k)$$

is assumed. This, unless needed, will not be assumed in this dissertation. Observe that assuming such a condition directly implies (3.28) (for any $c \leq 1$), (3.29) (for any $c_g \geq 1$) and also $E[e_k] = 0$, where $e_k$ is the error term of the search direction w.r.t. the true gradient, as introduced in the previous section.

Controlling the variance of the direction employed is key in the convergence analysis of stochastic methods, as the following theorem from [19] shows.

**Theorem 11** (Strongly convex case, variance reduced at geometric rate)**.** *Assume assumptions 1, 3 and 9 hold. Moreover, assume that there exist $C \geq 0$, $\tau \in (0,1)$ such that, for all k, it holds that*

$$Var[d_k] \leq C\tau^{k-1}.$$

*Then, employing algorithm 2 with $\alpha_k = \alpha$, for all k, such that*

$$0 < \alpha \leq \min\left\{\frac{c}{L_w c_g^2}, \frac{1}{\mu c}\right\},$$

*it follows that*

$$E[f(w_k)] - f^\star \leq \omega\rho^{k-1},$$

*where*

$$\omega := \max\left\{\frac{\alpha L_w C}{\mu c}, f(w_0) - f^\star\right\}$$

*and*

$$\rho := \max\left\{1 - \frac{\alpha\mu c}{2}, \tau\right\} < 1.$$

*Therefore,*

$$\lim_{k\to\infty} E[f(w_k)] = f^\star,$$

*and the convergence rate is linear.*

As shown by the results in theorem 11, letting the variance of the direction go to zero at a geometric rate yields linear convergence to the solution of problem 1.3, in the strongly convex case. This notwithstanding, many practitioners employ SG methods without any control on the variance of the direction. The following analysis explains what one should expect from such 'naive' application of SG methods.

For the analysis in the stochastic setting, Nocedal et al. [19] assume that the direction $d_k$ employed is a function of both the iterate $w_k$ and a random variable $\zeta_k$, i.e.

$$d_k = -g(w_k, \zeta_k),$$

from which the generic SG iterate can be written as

$$w_{k+1} = w_k - \alpha_k g(w_k, \zeta_k), \tag{3.31}$$

where $\zeta$ is a generic random variable defined by a generic probability distribution $\mathscr{P}_\zeta$. This random variable can be seen, in a SG algorithm, as the seed for the selection of the index $i_k$. Nevertheless, the subsequent analysis holds for any other algorithm satisfying assumptions 9.

The convergence analysis relies on the following two fundamental lemmas, which are reported from [19]. Observe that all the convergence results presented below hold for any method satisfying the conditions in assumption 9.

**Lemma 4.** *Let assumption 1 holds. Then the iterate* (3.31) *satisfies, for all k,*

$$E_{\zeta_k}[f(w_{k+1})] - f(w_k) \leq -\alpha_k \nabla f(w_k)^T E_{\zeta_k}[g(w_k, \zeta_k)] + \frac{1}{2}\alpha_k^2 L_w E_{\zeta_k}[\|g(w_k, \zeta_k)\|^2]. \tag{3.32}$$

*Proof.* Assumption 1 yields

$$f(w_{k+1}) - f(w_k) \leq \nabla f(w_k)^T (w_{k+1} - w_k) + \frac{1}{2} L_w \|w_{k+1} - w_k\|^2 \tag{3.33}$$

$$= -\alpha_k \nabla f(w_k)^T g(w_k, \zeta_k) + \frac{1}{2}\alpha_k^2 L_w \|g(w_k, \zeta_k)\|^2. \tag{3.34}$$

The proof follows by taking the expectation of the above inequality with respect to $\zeta_k$. □

From lemma 4, the reason why assumptions 9 are necessary is immediately clear. Indeed, condition 3.28 ensures that the first term in (3.33) is negative, while 3.30 gives a bound on the second term, which depends on the true gradient $\nabla f$.

**Lemma 5.** *Let assumptions 1 and 9 hold. Then iterate* (3.31) *yields, for all k,*

$$E_{\zeta_k}[f(w_{k+1})] - f(w_k) \leq -\left(c - \frac{1}{2} L_w M_G\right) \alpha_k \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w M. \tag{3.35}$$

*Proof.* Using lemma 4 and assumptions 9, it holds

$$\begin{aligned} E_{\zeta_k}[f(w_{k+1})] - f(w_k) &\leq -\alpha_k \nabla f(w_k)^T E_{\zeta_k}[g(w_k, \zeta_k)] + \frac{1}{2}\alpha_k^2 L_w E_{\zeta_k}[\|g(w_k, \zeta_k)\|^2] \\ &\leq -c\alpha_k \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w E_{\zeta_k}[\|g(w_k, \zeta_k)\|^2] \\ &\leq -\left(c - \frac{1}{2} L_w M_G\right) \alpha_k \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w M. \end{aligned}$$

□

Observe that, if $Var[d_k] \to 0$, then the constants $M, M_v$ can be taken small enough in order for condition (3.35) to imply a function descent (in expectation) condition on the function $f$.

Based on the above lemmas, which proof is straightforward thanks to the nice properties of the expectation operator, convergence results can be derived for both the nonconvex and convex cases, and both the constant and diminishing stepsize settings. These are reported, again, from [19]. In the following, the expected value with respect to $\zeta_k$ may be omitted, in which case it will hold

$$E[\cdot] := E_{\zeta_k}[\cdot].$$

Theorems 12 and 13, reported from [19], give the convergence properties of a SG method applied to general (nonconvex) objective functions. When a constant stepsize is employed, the result is very weak, only guaranteeing a lim inf convergence to a neighborhood of a stationary point.

**Theorem 12** (Nonconvex, fixed stepsize case)**.** *Let assumptions 1 and 9 hold. Let $f$ be bounded from below by $f_{inf}$ and suppose to apply iterate* (3.31) *with $\alpha_k = \alpha$ for all $k$ such that*

$$0 < \alpha \leq \frac{c}{L_w M_g}.$$

*Then it holds*

$$\lim_{K\to\infty} E\left[\frac{1}{K}\sum_{k=0}^{K} \|\nabla f(w_k)\|^2\right] \leq \frac{\alpha L_w M}{c}. \tag{3.36}$$

*Proof.* Using lemma 5 and the assumption on the stepsize, taking total expectation,

$$\begin{aligned} E[f(w_{k+1}] - E[f(w_k)] &\leq -\left(c - \frac{1}{2}\alpha L_w M_G\right)\alpha E[\|\nabla f(w_k)\|^2] + \frac{1}{2}\alpha^2 L_w M \\ &\leq -\frac{1}{2}c\alpha E[\|\nabla f(w_k)\|^2] + \frac{1}{2}\alpha^2 L_w M. \end{aligned}$$

Summing the above inequality for $k = 0, \ldots, K-1$ yields

$$f_{inf} - f(w_0) \leq E[f(w_K)] - f(w_0) \leq -\frac{1}{2}c\alpha \sum_{k=0}^{K-1} E[\|\nabla f(w_k)\|^2] + \frac{1}{2}K\alpha^2 L_w M.$$

Rearranging and dividing by $K$, the result follows. □

Observe that the result in (3.36) implies that

$$P\left\{\lim_{k\to\infty} \|\nabla f(w_k)\|^2 \leq \frac{\alpha L_w M}{c}\right\} \leq 1 - \varepsilon,$$

for any $\varepsilon > 0$.

In the nonconvex setting, even with a diminishing stepsize, the best one can ensure is that

$$\liminf_{k\to\infty} E[\|\nabla f(w_k)\|^2] = 0,$$

as proven by the following theorem. Notice that this is a worse result than the one in the deterministic setting, where convergence of the whole sequence was proved in theorem 3.

**Theorem 13** (Nonconvex, diminishing stepsize case)**.** *Let assumptions 1 and 9 hold. Let $f$ be bounded from below by $f_{inf}$ and suppose to apply iterate* (3.31) *with $\alpha_k$ satisfying*

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

*Then it holds*

$$\lim_{K\to\infty} E\left[\frac{1}{A_K}\sum_{k=0}^{K-1} \alpha_k \|\nabla f(w_k)\|^2\right] = 0,$$

*where*

$$A_K = \sum_{k=0}^{K-1} \alpha_k.$$

*Proof.* Since $\{\alpha_k\} \to 0$, without loss of generality let the following holds for all $k$

$$\alpha_k L_w M_G \leq c.$$

Using lemma 5 and taking the total expectation,

$$E[f(w_{k+1}] - E[f(w_k)] \leq -\left(c - \frac{1}{2}\alpha L_w M_G\right) \alpha E[\|\nabla f(w_k)\|^2] + \frac{1}{2}\alpha^2 L_w M$$

$$\leq -\frac{1}{2}c\alpha E[\|\nabla f(w_k)\|^2] + \frac{1}{2}\alpha^2 L_w M.$$

Summing the above inequality for $k = 0, \ldots, K-1$ yields

$$f_{inf} - f(w_0) \leq E[f(w_K)] - f(w_0) \leq -\frac{1}{2}c\sum_{k=0}^{K-1} \alpha_k E[\|\nabla f(w_k)\|^2] + \frac{1}{2}L_w M \sum_{k=0}^{K-1} \alpha_k^2.$$

Multiplying by 2 and dividing by $c$, the above inequality yields

$$\sum_{k=0}^{K-1} \alpha_k E[\|\nabla f(w_k)\|^2] \leq \frac{2(f(w_0) - f_{inf})}{c} + \frac{L_w M}{c} \sum_{k=0}^{K-1} \alpha_k^2.$$

Letting $K$ go to $\infty$, the right hand side is finite by the assumption on the stepsize sequence. Furthermore,

$$\lim_{K\to\infty} A_K = \sum_{k=0}^{K-1} \alpha_k = \infty,$$

from which the proof follows. □

As in the deterministic setting, stronger convergence results can be proved in the (strongly) convex case.

**Theorem 14** (Strongly convex, fixed stepsize case)**.** *Let assumptions 1, 3 and 9 hold. Let the optimal value of $f$ be $f^\star$ and suppose to apply iterate* (3.31) *with $\alpha_k = \alpha$ for all $k$, such that*

$$0 < \alpha \leq \frac{c}{L_w M_g}. \tag{3.37}$$

*Then it holds that*

$$\lim_{k\to\infty} E[f(w_k)] \leq f^\star + \frac{\alpha L_w M}{2\mu c}. \tag{3.38}$$

*Proof.* Recalling lemma 5 and remark 2, it holds that

$$\begin{aligned} E[f(w_{k+1})] - f(w_k) &\leq -\left(c - \frac{1}{2}L_w M_G\right) \alpha_k \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w M \\ &\leq -\frac{1}{2}\alpha c \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w M \\ &\leq -\alpha\mu c(f(w_k) - f^\star) + \frac{1}{2}\alpha_k^2 L_w M, \end{aligned}$$

where the second inequality follows from the stepsize (3.37) and the last inequality from remark 2. Taking the expectation of the above inequality and subtracting $f^\star$ from both sides, it follows that

$$E[f(w_{k+1})] - f^\star \leq (1 - \alpha\mu c)\,(E[f(w_k)] - f^\star) + \frac{1}{2}\alpha_k^2 L_w M.$$

Now, subtract the quantity $\frac{\alpha L_w M}{2\mu c}$ from both sides, obtaining

$$\begin{aligned} E[f(w_{k+1})] - f^\star - \frac{\alpha L_w M}{2\mu c} &\leq (1-\alpha\mu c)\left(E[f(w_k)] - f^\star\right) + \frac{1}{2}\alpha_k^2 L_w M - \frac{\alpha L_w M}{2\mu c} \\ &\leq (1-\alpha\mu c)\left(E[f(w_k)] - f^\star - \frac{\alpha L_w M}{2\mu c}\right), \end{aligned}$$

and by iterating the above one gets

$$E[f(w_{k+1})] - f^\star - \frac{\alpha L_w M}{2\mu c} \leq (1-\alpha\mu c)^k \left(f(w_0) - f^\star - \frac{\alpha L_w M}{2\mu c}\right).$$

Recalling that

$$0 < \alpha\mu c \leq \frac{\mu c^2}{L_w M_G} \leq \frac{\mu c^2}{L_w c^2} = \frac{\mu}{L_w} \leq 1,$$

the result follows. □

The above result is very similar to the one in the deterministic setting for the IG method. Indeed, theorem 14 proves that the SG method, applied with a constant, small enough stepsize, ensures linear convergence to a neighborhood of the solution. Unfortunately, there is no guarantee on what could happen after the neighborhood has been reached. Indeed, one could expect instability in the iterate sequence $\{w_k\}$, which may oscillate in and out of the neighborhood with no control.

The proof of the convergence in the diminishing stepsize case is reported again from [19], where the stepsize is assumed to satisfy

$$\alpha_k = \frac{\beta}{\gamma + k}, \tag{3.39}$$

with $\beta, \gamma > 0$ such that $\alpha_0 \leq \frac{c}{L_w M_G}$. Note that requiring $\alpha_0 \leq \frac{c}{L_w M_G}$ is not restrictive, since even if that did not hold for $k = 0$, then a $\bar{k} > 0$ would exist such that it would hold for all $k \geq \bar{k}$. Furthermore, observe that (3.39) guarantees the standard requirement for a diminishing stepsize, i.e.

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

**Theorem 15** (Strongly convex, diminishing stepsize case)**.** *Let assumptions 1, 3 and 9 hold. Let the optimal value of $f$ be $f^\star$ and suppose to apply iterate* (3.31) *with $\alpha_k$ such that* (3.39) *holds. Then*

$$\lim_{k\to\infty} E[f(w_k)] = f^\star. \tag{3.40}$$

*Proof.* Using the strong convexity, lemma 5 and (3.39), the following holds

$$\begin{aligned} E[f(w_{k+1})] - f(w_k) &\leq -\left(c - \frac{1}{2}\alpha_k L_w M_G\right)\alpha_k \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w M \\ &\leq -\frac{1}{2}\alpha_k c \|\nabla f(w_k)\|^2 + \frac{1}{2}\alpha_k^2 L_w M \\ &\leq -\alpha_k \mu c (f(w_k) - f^\star) + \frac{1}{2}\alpha_k^2 L_w M. \end{aligned}$$

Subtracting $f^\star$ from both sides and rearranging, it yields

$$E[f(w_{k+1})] - f^\star \leq (1-\alpha_k\mu c)\left(E[f(w_k)] - f^\star\right) + \frac{1}{2}\alpha_k^2 L_w M.$$

Defining $\nu > 0$ as

$$\nu := \max\left\{\frac{\beta^2 L_w M}{2(\beta\mu c - 1)}, (\gamma+1)(f(w_0) - f^\star)\right\},$$

for $k = 0$ it holds that

$$E[f(w_k)] - f^\star \leq \frac{\nu}{\gamma + k}.$$

To prove the above inequality by induction, it is necessary to prove it for some $k \geq 0$. Setting $\widetilde{k} = \gamma + k$ and recalling that $\widetilde{k}^2 \geq (\widetilde{k}+1)(\widetilde{k}-1)$,

$$\begin{aligned} E[f(w_k)] - f^\star &\leq \left(1 - \frac{\beta\mu c}{\widetilde{k}}\right)\frac{\nu}{\widetilde{k}} + \frac{\beta^2 L_w M}{2\widetilde{k}^2} \\ &= \left(\frac{\widetilde{k} - \beta\mu c}{\widetilde{k}^2}\right)\nu + \frac{\beta^2 L_w M}{2\widetilde{k}^2} \\ &= \left(\frac{\widetilde{k}-1}{\widetilde{k}^2}\right)\nu - \left(\frac{\beta\mu c - 1}{\widetilde{k}^2}\right)\nu + \frac{\beta^2 L_w M}{2\widetilde{k}^2} \\ &\leq \frac{\nu}{\widetilde{k}+1}, \end{aligned}$$

where in the last inequality it was used that

$$-\left(\frac{\beta\mu c - 1}{\widetilde{k}^2}\right)\nu + \frac{\beta^2 L_w M}{2\widetilde{k}^2} \leq 0,$$

by the definition of $\nu$. Therefore, the inequality

$$E[f(w_k)] - f^\star \leq \frac{\nu}{\gamma + k}$$

holds for all $k$. The proof follows by taking the limit of the above inequality for $k \to \infty$. □

As expected, the introduction of a diminishing stepsize ensures convergence to the exact solution of the problem. The cost, exactly as in the deterministic setting, is a sublinear rate of convergence, instead of a linear one.

It is interesting to compare the above results from [19] with the earlier ones presented in [15], where, beyond all the analysis in the deterministic setting, Bertsekas analyzed the convergence properties of a gradient method with error, when the error is of a stochastic nature. In [15], only the nonconvex setting was analyzed, with assumptions very similar to the ones by Nocedal et al. in [19]. Indeed, the assumptions in [15] are:

- Iterate update

  $$w_{k+1} = w_k + \alpha_k(g_k + e_k),$$

  where $e_k$ is an error term of stochastic nature, and $g_k$ is a deterministic descent direction;

- A diminishing, squared summable but not summable, stepsize sequence $\{\alpha_k\}$ such that

  $$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty;$$

- Assumptions 6 hold;
- For all $k$, it holds with probability 1 that

$$E[e_k] = 0, \tag{3.41}$$

which implies

$$E[(g_k + e_k)^T \nabla f(w_k)] \geq c_3 \|\nabla f(w_k)\|^2,$$

for some $c_3 > 0$, and

$$E[\|e_k\|^2] \leq A(1 + \|\nabla f(w_k)\|^2), \tag{3.42}$$

for some $A > 0$.

Observe that condition (3.42) from [15] is indeed very similar to the last condition in assumptions 9 from [19], namely a condition bounding the growth of the variance in the gradient approximation error at each iteration. Under the above assumptions, convergence with probability 1 of any accumulation point of the sequence $\{w_k\}$ to a stationary point was proven.

Now that the main assumptions and results for a generic gradient method with a stochastic search direction have been presented, the basic Stochastic Gradient method can be introduced.

### 3.2.1 Stochastic Gradient method

The most basic way of computing $d_k$ is by choosing an index $i_k$ at random from $\{1, \ldots, N\}$ and setting

$$d_k = -\nabla f_{i_k}(w_k). \tag{3.43}$$

**Algorithm 6:** SG scheme

**Data:** $w_0 \in \mathbb{R}^d$

**for** $k = 0, 1, \ldots$ **do**
- Compute $\nabla f_{i_k}(w_k)$, where $i_k$ is chosen uniformly at random from $\{1, \ldots, N\}$;
- Choose a stepsize $\alpha_k > 0$;
- Update the iterate $w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$;

**end**

This is the stochastic counterpart of the IG method. The convergence analysis of algorithm 6 directly follows from the analysis above, by assuming that the direction $d_k$ in (3.43) satisfies assumption 9.

Thanks to the analysis in the previous section, the convergence of the basic SG method and of any other algorithms satisfying assumption 9 can be proven. Nevertheless, to overcome the diminishing stepsize, and consequently the sublinear convergence, issue, more sophisticated methods must be introduced. These use aggregated gradient information to reduce the gradient approximation variance. The aim is to get convergence to a solution, without the need for a diminishing stepsize.

### 3.2.2 Stochastic Average Gradient method

**Algorithm 7:** SAG scheme

**Data:** $y_0^1 = y_0^2 = \cdots = y_0^N = 0, w_0 \in \mathbb{R}^d$

**for** $k = 0, 1, \ldots$ **do**

  Choose a stepsize $\alpha_k > 0$;

  Compute $\nabla f_{i_k}$, where $i_k$ is chosen uniformly at random from $\{1, \ldots, N\}$, and store it;

  Compute $\nabla \widetilde{f}(w_k) = \frac{1}{N} \sum_{i=1}^N y_k^i, \quad y_k^i = \begin{cases} \nabla f_i(w_k), & i = i_k, \\ y_{k-1}^i & \text{otherwise;} \end{cases}$

  Update the iterate $w_{k+1} = w_k - \alpha_k \nabla \widetilde{f}(w_k)$;

**end**

The stochastic counterpart of IAG is the Stochastic Average Gradient (SAG) [89], where the iterate update can be written as

$$w_{k+1} = w_k - \frac{\alpha}{N} \sum_{i=1}^N y_k^i, \quad y_k^i = \begin{cases} \nabla f_i(w_k), & i = i_k, \\ y_{k-1}^i & \text{otherwise,} \end{cases} \tag{3.44}$$

in which the index $i_k$ is selected uniformly at random from $\{1, \ldots, N\}$ at every iteration $k$.

To interpret SAG in the framework of algorithm 2, it suffices to set the short- and long-term components of the search direction to, respectively,

$$s_k = -\frac{1}{N} \nabla f_{i_k}(w_k) \quad \text{and} \quad l_k = -\frac{1}{N} \sum_{i \neq i_k} y_k^i.$$

The SAG method is interestingly tightly connected to the well-known SG method with momentum. SG with momentum is very often used in real applications, and employs an iterate update of the type

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k) + \beta_k (w_k - w_{k-1}),$$

where $\alpha_k, \beta_k > 0$. The above can be rewritten, in the common case where $\beta_k = \beta > 0$ for all $k$, as

$$w_{k+1} = w_k - \sum_{h=1}^k \alpha_k \beta^{k-h} \nabla f_{i_h}(w_h),$$

which is a geometric weighting of past computed gradients of the component functions. SAG, instead, can be written as

$$w_{k+1} = w_k - \sum_{h=1}^k \alpha_k S(h, i_{1:k}) \nabla f_{i_h}(w_h),$$

where $S(h, i_{1:k})$ is equal to $\frac{1}{N}$ if $h$ is the closest iteration where $i_h$ was selected, 0 otherwise. This is an average of the most recent evaluations of the gradients of each component function. Observe that SG with momentum, instead, does not guarantee that the gradient of each component function has the same weight in the weighted average. Indeed, SG with momentum has no provable convergence guarantees with a constant stepsize, and is therefore usually employed with a diminishing one.

Theorem 16 proves convergence of SAG to the solution, at a linear rate, with a small, constant stepsize, in the convex case.

**Theorem 16.** *Let assumptions 1 and 5 hold for each composite function $f_i$. Let $y_0^i = 0$ for all $i$ and*

$$\alpha_k = \alpha = \frac{1}{16L_w} \quad \forall\, k.$$

*Then it holds for all $k$*

$$E[f(\overline{w}_k)] - f^\star \leq \frac{32N}{k} C_0, \tag{3.45}$$

*where*

$$\overline{w}_k = \frac{1}{k}\sum_{h=0}^{k-1} w_h$$

*and*

$$C_0 = f(w_0) - f^\star + \frac{4L_w}{N}\|w_0 - w^\star\|^2 + \frac{\sigma^2}{16L_w},$$

*with $\sigma^2 = \frac{1}{N}\sum_{i=1}^N \|\nabla f_i(w^\star)\|^2$. It follows that*

$$\liminf_{k\to\infty} E[f(w_k)] = f^\star. \tag{3.46}$$

*Furthermore, if assumption 3 holds for $f$, then*

$$E[f(w_k)] - f^\star \leq \left(1 - \min\left\{\frac{\mu}{16L_w}, \frac{1}{8N}\right\}\right)^k C_0, \tag{3.47}$$

*from which*

$$\lim_{k\to\infty} E[f(w_k)] = f^\star. \tag{3.48}$$

Observe that the authors in [89] numerically tested SAG vs IAG, noticing that IAG usually needs a stepsize of around $\frac{1}{NL_w}$ to converge, compared to SAG's that can be $N$ times larger. This is in line with the theoretical results of IAG and SAG.

The last remark on the SAG method is that, to the best of the author's understanding, no results have been proved in the nonconvex setting. Whether some mild convergence guarantees, like the ones obtained for IAG in the deterministic setting, can be given is still an open question.

### 3.2.3 Finito method

**Algorithm 8:** Finito scheme

**Data:** $y_0^1 = y_0^2 = \cdots = y_0^N = w_0 \in \mathbb{R}^d$, $\nabla f_1(y_0^1), \nabla f_2(y_0^2), \ldots, \nabla f_N(y_0^N)$

**for** $k = 0, 1, \ldots$ **do**

- Compute $\nabla f_{i_k}(y_k^i)$, where $i_k$ is selected uniformly at random from $\{1, \ldots, N\}$, and store it;
- Choose a stepsize $\alpha_k > 0$;
- Update the iterate $w_{k+1} = \frac{1}{N}\sum_{i=1}^N y_k^i - \alpha_k \frac{1}{N}\sum_{i=1}^N \nabla f_i(y_k^i)$;
- Set $y_k^{i+1} = \begin{cases} w_{k+1}, & i = i_k, \\ y_k^i & \text{otherwise;} \end{cases}$

**end**

Finito method [35] is the stochastic counterpart of DIAG method. As the other methods, also Finito can be put in the framework of algorithm 2, by setting $s_k$ and $l_k$ to

$$s_k = \frac{1}{N}\left(w_k - \nabla f_{i_k}(w_k)\right) \quad \text{and} \quad l_k = \frac{1}{\alpha_k N}\sum_{i=1}^N y_k^i - \frac{1}{N}\sum_{i\neq i_k} \nabla f_i(y_k^i),$$

where $y_k^i$, $i = 1, \dots, N$, are updated as

$$y_k^{i+1} = \begin{cases} w_{k+1}, & i = i_k, \\ y_k^i & \text{otherwise.} \end{cases}$$

Proposed in 2014, this method aims at finding a better linear rate of convergence with respect to SAG. The convergence rate of Finito is, indeed, better than SAG one, as shown in theorem 18. Nevertheless, such convergence rate can be proved in the regime the authors in [35] call *big data regime*, where they assume

$$N \geq 2\frac{L_w}{\mu}.$$

When this does not hold, then theorem 17 only proves convergence of the function computed at the average iterate $f(\overline{w})$ to the optimal value $f^\star$.

**Theorem 17.** *Let assumptions 1 and 3 hold. Assume $c \geq 2$, $\beta \geq 2$ and*

$$\frac{2}{c} - \frac{1}{c^2} - \beta + \frac{\beta}{c} \leq 0.$$

*If $y_0^1 = y_0^2 = \cdots = y_0^N = w_0$ and $\alpha_k = \alpha = \frac{1}{c\mu}$ for all k, then*

$$E[f(\overline{w}_k)] - f^\star \leq \frac{C}{\mu}\left(1 - \frac{1}{cN}\right)^k \|\nabla f(w_0)\|^2,$$

*where $C = 1 - \frac{1}{2c}$, from which*

$$\lim_{k\to\infty} E[f(\overline{w}_k)] = f^\star.$$

**Theorem 18.** *Let assumptions 1 and 3 hold. Assume $N \geq 2\frac{L_w}{\mu}$. Therefore, $c = 2$ is admissible. If $y_0^1 = y_0^2 = \cdots = y_0^N = w_0$ and $\alpha_k = \alpha = \frac{1}{2\mu}$ for all k, then*

$$E[f(\overline{w}_k)] - f^\star \leq \frac{3}{4\mu}\left(1 - \frac{1}{2N}\right)^k \|\nabla f(w_0)\|^2.$$

To compare the convergence rate of Finito with SAG one, note that while for Finito it depends on the quantity

$$1 - \frac{1}{2N},$$

for SAG it depends on

$$1 - \frac{1}{8N}.$$

Since $\frac{1}{8N}$ is smaller than $\frac{1}{2N}$, then SAG rate is worse than Finito one. As a last comment, observe that both SAG and Finito converge under the assumption of a stepsize exactly equal to a small constant quantity, which is difficult to compute. Therefore, it seems to be an open question whether the convergence results are maintained if the stepsize is smaller (which is usually the case when, in real applications, the constant stepsize must be guessed).

### 3.2.4 Stochastic Variance Reduced Gradient method

**Algorithm 9:** SVRG scheme

**Data:** $m > 0, s = 0, \widetilde{w} = \overline{w}_s = w_0 \in \mathbb{R}^d, \nabla f(\widetilde{w})$
**for** $k = 0, 1, \ldots$ **do**
  Compute $\nabla f_{i_k}(w_k)$, where $i_k$ is selected at random from $\{1, \ldots, N\}$;
  Choose a stepsize $\alpha_k > 0$;
  Update the iterate $w_{k+1} = w_k - \alpha_k(\nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w}) + \nabla f(\widetilde{w}))$;
  **if** $mod(k+1, m) = 0$ **then**
    Compute $\nabla f(\widetilde{w})$, where $\widetilde{w} = w_{k+1}$;
    Set $\overline{w}_{s+1} = \begin{cases} w_{k+1}, & \text{option I,} \\ w_{k+1-j},\ j \text{ random from } \{1, \ldots, m\} & \text{option II;} \end{cases}$
    $s \leftarrow s + 1$;
  **end**
**end**

In the SVRG scheme, the short-term component $s_k$ of the direction $d_k$ is the negative gradient of the component function $f_{i_k}$, while the long-term one, $l_k$, is the correction based on the full gradient computed in the 'snapshot' iterate $\widetilde{w}$, namely $\nabla f_{i_k}(\widetilde{w}) - \nabla f(\widetilde{w})$:

$$s_k = -\nabla f_{i_k}(w_k) \quad \text{and} \quad l_k = \nabla f_{i_k}(\widetilde{w}) - \nabla f(\widetilde{w}).$$

Analyzing the search direction $d_k$ in algorithm 9, one can recognize that it is an approximation of the negative gradient, i.e.

$$d_k = -\nabla \widetilde{f}(w_k),$$

where

$$\begin{aligned}
\nabla \widetilde{f}(w_k) &= \nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w}) + \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\widetilde{w}) \\
&= \frac{1}{N}\sum_{i=1}^{N} \nabla f_{i_k}(w_k) - \frac{1}{N}\sum_{i=1}^{N} \nabla f_{i_k}(\widetilde{w}) + \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\widetilde{w}) \\
&= \frac{1}{N}\sum_{i=1}^{N} [\nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w}) + \nabla f_i(\widetilde{w})] \\
&= \frac{1}{N}\sum_{i=1,\ i\neq i_k}^{N} [\nabla f_i(\widetilde{w}) + (\nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w}))] + \frac{1}{N}\nabla f_{i_k}(w_k) \\
&= \sum_{i=1}^{N} y_k^i
\end{aligned}$$

where

$$y_k^i = \begin{cases} \nabla f_{i_k}(w_k), & i = i_k, \\ \nabla f_i(\widetilde{w}) + (\nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w})), & \text{otherwise.} \end{cases}$$

The idea behind SVRG is therefore the following: given the full gradient evaluated in the snapshot $\nabla f(\widetilde{w})$, when at iteration $k$ a new component $\nabla f_{i_k}(w_k)$ of the summation composing the full gradient at $w_k$ is computed, then do the following:

1. substitute the $i_k$-th sum component of the snapshot with the new one;

2. correct every other sum component $i: \ i \neq i_k$ with the quantity $\nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w})$,

where the correction term is an approximation of the error in the sum component $\nabla f_i$ due to its evaluation in the old snapshot of the iterate $\widetilde{w}$. SVRG algorithm (9) is proven to converge to the optimal solution of problem (1.3) with a constant stepsize if $f$ is convex. Such convergence was extended to a nonconvex setting in [3]. It must be underlined nonetheless that such linear convergence results are proven on the iterates $\overline{w}_s$, which are updated every $m$ iterations, and therefore it can be slower than expected.

The convergence results in the strongly convex case are reported from [58].

**Theorem 19** (Strongly convex case)**.** *Let assumptions 1 and 3 hold for $f$, and assumption 5 for all the $f_i$. Let SVRG be run with option II, $\alpha_k = \alpha \in \left(0, \frac{2}{L_w}\right)$ for all $k$ and $m > 0$ large enough to guarantee*

$$c = \frac{1}{\mu\alpha(1-2L_w\alpha)m} + \frac{2L_w\alpha}{1-2L_w\alpha} < 1.$$

*Then*

$$E[f(\overline{w}_s)] \leq f^\star + c^s(f(w_0) - f^\star),$$

*where $f^\star$ is the optimal value of $f$, and it follows that*

$$\lim_{s\to\infty} E[f(\overline{w}_s)] = f^\star.$$

Theorem 19 shows that the SVRG method guarantees, in the strongly convex case, linear convergence of the sequence $\{f(\overline{w}_s)\}_{s=0}^{\infty}$ to the optimal value of the objective. The SVRG method is the only stochastic method that enjoys convergence, although in expectation and at a sublinear rate, to a stationary point in the nonconvex case. Such convergence result is reported from [83].

**Theorem 20** (Nonconvex case)**.** *Let assumption 1 hold, $f$ be bounded from below by $f_{inf}$ and $c_m, \beta > 0$ be constants such that*

$$\Gamma_i = \alpha - \frac{c_{i+1}\alpha}{\beta} - \alpha^2 L_w - 2c_{i+1}\alpha^2 > 0, \quad i = 1, \ldots, m$$

*where*

$$c_i = c_{i+1}(1 + \alpha\beta + 2\alpha^2 L_w^2) + \alpha^2 L_w^3, \quad i = 1, \ldots, m.$$

*Define $\gamma := \min_{i=1,\ldots,m} \Gamma_i$ and let algorithm 9 be run with option II. Then it holds that*

$$E[\|\nabla f(\overline{w}_s)\|^2] \leq \frac{f(w_0) - f_{inf}}{s\gamma},$$

*which yields*

$$\lim_{s\to\infty} E[\|\nabla f(\overline{w}_s)\|^2] = 0.$$

### 3.2.5 SAGA method - a bridge between SAG and SVRG

**Algorithm 10:** SAGA scheme

**Data:** $y_0^1, y_0^2, \ldots, y_0^N, w_0 \in \mathbb{R}^d$

**for** $k = 0, 1, \ldots$ **do**

  Choose a stepsize $\alpha_k > 0$;

  Compute $\nabla f_{i_k}$, where $i_k$ is chosen uniformly at random from $\{1, \ldots, N\}$, and store it;

  Compute $\nabla \widetilde{f}(w_k) = \nabla f_{i_k}(w_k) - \nabla f_{i_k}(y_{k-1}^{i_k}) + \frac{1}{N}\sum_{i=1}^N y_k^i$, $\quad y_k^i = \begin{cases} \nabla f_i(w_k), & i = i_k, \\ y_{k-1}^i & \text{otherwise;} \end{cases}$

  Update the iterate $w_{k+1} = w_k - \alpha_k \nabla \widetilde{f}(w_k)$;

**end**

The last method analyzed in the stochastic setting is SAGA [34]. SAGA was introduced in 2014 and aims at combining the good properties of SAG and SVRG. The method employs the short-term and long-term components

$$s_k = -\nabla f_{i_k}(w_k) \quad \text{and} \quad l_k = \nabla f_{i_k}(y_k^i) - \nabla f(y_k^i),$$

therefore having an iterate update

$$w_{k+1} = w_k - \nabla f_{i_k}(w_k) + \nabla f_{i_k}(y_{k-1}^{i_k}) - \frac{1}{N}\sum_{i=1}^N y_k^i,$$

where

$$y_k^i = \begin{cases} \nabla f_i(w_k), & i = i_k, \\ y_{k-1}^i & \text{otherwise.} \end{cases}$$

Just like for SVRG, the search direction $d_k$ in algorithm 10 can be seen as an approximation of the negative gradient, i.e.

$$d_k = -\nabla \widetilde{f}(w_k),$$

where

$$\begin{aligned} \nabla \widetilde{f}(w_k) &= \nabla f_{i_k}(w_k) - \nabla f_{i_k}(y_k^i) + \frac{1}{N}\sum_{i=1}^N \nabla f_i(y_k^i) \\ &= \frac{1}{N}\sum_{i=1}^N \nabla f_{i_k}(w_k) - \frac{1}{N}\sum_{i=1}^N \nabla f_{i_k}(y_k^i) + \frac{1}{N}\sum_{i=1}^N \nabla f_i(y_k^i) \\ &= \frac{1}{N}\sum_{i=1}^N \left[\nabla f_{i_k}(w_k) - \nabla f_{i_k}(y_k^i) + \nabla f_i(y_k^i)\right] \\ &= \frac{1}{N}\sum_{i=1,\ i\neq i_k}^N \left[\nabla f_i(y_k^i) + \left(\nabla f_{i_k}(w_k) - \nabla f_{i_k}(y_k^i)\right)\right] + \frac{1}{N}\nabla f_{i_k}(w_k), \end{aligned}$$

where the correction is not based on the last 'snapshot' $\widetilde{w}$, as in SVRG, but based on $y_k^i$, which is equal, for each $i = 1, \ldots, N$, to the most recent gradient of the component function $f_i$. Observe that this modification allows SAGA to never compute anything else than the gradient of the currently selected component function $f_{i_k}$.

The convergence results of SAGA are reported from [34]. As can be seen from theorems 21 and 22, linear convergence in expectation to the solution of problem 1.3 can be proven both in the strongly

convex and general convex cases. Nevertheless, in the general convex case the convergence can be proven over the average iterate $\overline{w}_k$ only.

**Theorem 21** (Convergence in the strongly convex case)**.** *Let assumptions 1 and 3 hold. Then, running algorithm 10 with a stepsize $\alpha_k = \alpha$ for all k such that*

$$\alpha = \frac{1}{2(\mu N + L_w)}.$$

*Then*

$$E[\|w_k - w^\star\|^2] \leq \left(1 - \frac{\mu}{2(\mu N + L_w)}\right)^k \left(\|w_0 - w^\star\|^2 + \frac{N}{\mu N + L_w}(f(w_0) - \nabla f(w^\star)^T(w_0 - w^\star) - f(w^\star))\right).$$

**Theorem 22** (Convergence in the general convex case)**.** *Let assumptions 1 and 5 hold. Then, running algorithm 10 with a stepsize $\alpha_k = \alpha$ for all k such that*

$$\alpha = \frac{1}{3L_w}.$$

*Then*

$$E[\|\overline{w}_k - w^\star\|^2] \leq \left(1 - \min\left\{\frac{1}{4N}, \frac{\mu}{3L_w}\right\}\right)^k \left(\|w_0 - w^\star\|^2 + \frac{2N}{3L_w}(f(w_0) - \nabla f(w^\star)^T(w_0 - w^\star) - f(w^\star))\right),$$

*where*

$$\overline{w}_k = \frac{1}{k}\sum_{h=1}^{k} w_h.$$

Now that the main stochastic methods to solve problem 1.3 have also been introduced, a comparison between their convergence properties and those of the incremental methods is presented in the next section.

## 3.3 Comments on the convergence properties of the two settings

In the previous two sections, some of the most well-known methods to solve large-scale unconstrained finite sum problems are introduced and analyzed. All of them have the aim of getting convergence guarantees in the setting where only the gradient of a single component function is computed at each iteration (except for SVRG, which needs the computation of a full gradient every $m > 0$ iterations). It is interesting to notice that the convergence properties of such methods are quite different, based on whether the analysis is carried on in the deterministic or the stochastic settings. Nevertheless, most of them are just the same method, calculating the gradient of a component function $\nabla f_{i_k}$ at each iteration, and differing only in the criterion of selection for the index $i_k$ (i.e. cyclic vs random) and the analysis setting (i.e. deterministic vs stochastic). This is the case, e.g., of IG vs SG methods, IAG vs SAG methods, and DIAG vs Finito methods. Tables 3.1-3.3 report a summary of the convergence properties of such methods in the two settings. Observe that the following tables are to give a quick view of the convergence properties of the algorithms. Nevertheless, small theoretical differences are not reported, e.g., when the convergence is a liminf-type one vs a lim-type one.

| Method | Type | Stepsize | Convergence | Rate |
|---|---|---|---|---|
| IG | Deterministic | fixed | neighborhood | ? |
| IG | Deterministic | diminishing | OK | sublinear |
| SG | Stochastic** | fixed | neighborhood | linear |
| SG | Stochastic** | diminishing | OK | sublinear |
| IAG | Deterministic | fixed | neighborhood* | ? |
| SAG | Stochastic** | fixed | ? | ? |
| DIAG | Deterministic | fixed | ? | ? |
| Finito | Stochastic** | fixed | ? | ? |
| SVRG | Stochastic** | fixed | OK | sublinear |
| SAGA | Stochastic** | fixed | ? | ? |

**Table 3.1.** Nonconvex case.
* lim inf result
** All the stochastic methods' convergence results hold in expectation.

| Method | Type | Stepsize | Convergence | Rate |
|---|---|---|---|---|
| IG | Deterministic | fixed | neighborhood | linear |
| IG | Deterministic | diminishing | OK | sublinear |
| SG | Stochastic** | fixed | neighborhood | linear |
| SG | Stochastic** | diminishing | OK | sublinear |
| IAG | Deterministic | fixed | OK | linear* |
| SAG | Stochastic** | fixed | OK*** | linear |
| DIAG | Deterministic | fixed | ? | ? |
| Finito | Stochastic** | fixed | ? | ? |
| SVRG | Stochastic** | fixed | OK | ? |
| SAGA | Stochastic** | fixed | OK*** | linear*** |

**Table 3.2.** Convex case.
* Only in the quadratic case.
** All the stochastic methods' convergence results hold in expectation.
*** The result holds over the average iterate.

One of the most interesting comparisons is the one between assumption 6 for the deterministic analysis of incremental methods and assumption 9 for the stochastic analysis of stochastic methods. In particular, the last condition in assumption 6 asks the following to hold for all iterations $k$:

$$e_k \leq \alpha_k(q + p\|\nabla f(w_k)\|),$$

for some $p, q > 0$, where $e_k$ is the error in the direction employed by a gradient method with error, namely

$$w_{k+1} = w_k + \alpha_k d_k = w_k + \alpha_k(g_k + e_k).$$

In contrast, the last condition in assumption 9 asks the stochastic direction $d_k$ to satisfy for each iteration $k$:

$$E[\|d_k\|^2] \leq M + M_G\|\nabla f(w_k)\|^2,$$

for some $M, M_G > 0$. Both the conditions can be read as a bound on the variance of the direction employed by the method. Nevertheless, in the deterministic setting the assumption is stronger, namely

| Method | Type | Stepsize | Convergence | Rate |
|---|---|---|---|---|
| IG | Deterministic | fixed | neighborhood | linear |
| IG | Deterministic | diminishing | OK | sublinear |
| SG | Stochastic** | fixed | neighborhood | linear |
| SG | Stochastic | diminishing | OK | sublinear |
| IAG | Deterministic | fixed | OK | linear |
| SAG | Stochastic** | fixed | OK | linear |
| DIAG | Deterministic | fixed | OK | linear |
| Finito | Stochastic** | fixed | OK | linear |
| SVRG | Stochastic** | fixed | OK | linear |
| SAGA | Stochastic** | fixed | OK | linear |

**Table 3.3.** Strongly convex case
** All the stochastic methods' convergence results hold in expectation.

the error term $e_k$ is asked to be somehow 'controlled' through the stepsize $\alpha_k$. Notice that this condition is satisfied [15] by an IG method applied to problem 1.3, if the following holds for all indices $i$:

$$\|\nabla f_i(w_k)\| \leq C + D\|\nabla f(w_k)\|^2,$$

which is very similar to the stochastic condition on the squared norm of the direction. Using this assumption, in the deterministic setting a deterministic convergence to the solution, with a diminishing stepsize, can be obtained, as shown in the previous sections. In the stochastic setting, instead, only a convergence in expectation can be proved.

Another difference, which deserves to be discussed, between the stochastic analysis and the deterministic one is the following. Taking in consideration the Incremental Gradient and the Stochastic Gradient recall how the algorithms 3 and 6 are defined. The convergence results are proven for the sequences $\{w_k\}$, which are updated based on the rules, respectively,

$$w_{k+1} = w_k - \alpha_k \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(v_{i-1})$$

for IG, and

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k), \quad i_k \text{ random from } \{1,\dots,N\}$$

for SG. Therefore, although the convergence results of the two methods are with respect to $w_k$, one fundamental difference is that the update of the iterate $w_k$ in the deterministic setting has the same cost of $N$ iterations in the stochastic setting. One would therefore expect SG methods to be much more efficient when applied to real, large-scale finite sum problems. Nevertheless, this is not what happens in real applications. Indeed, numerically the index selection criterion that seems to work consistently better than any other is the so-called *reshuffling* one, where the indices are selected cyclically, but the order is reshuffled after each epoch (i.e., after all the $N$ component functions have been employed exactly once). This implementation actually falls under the deterministic setting, where all the indices are employed exactly one time every $N$ iterations, although the reshuffling of the order in which the indices are selected makes it similar to the stochastic one. Below, an extract from [12] is reported, which supports the above point.

*Bertsekas' comment on reshuffling the order of the indices*

Another technique for incremental methods, popular in neural network training practice, is to reshuffle randomly the order of the component functions after each cycle. This alternative order selection scheme leads to convergence, like the preceding two. Moreover, this scheme has the nice property of allocating exactly one computation slot to each component in an m-slot cycle (m incremental iterations). By comparison, choosing components by uniform sampling allocates one computation slot to each component on the average, but some components may not get a slot while others may get more than one. A nonzero variance in the number of slots that any fixed component gets within a cycle, may be detrimental to performance, and indicates that reshuffling randomly the order of the component functions after each cycle may work better; this is consistent with experimental observations shared with the author by B. Recht (private communication). However, establishing this fact analytically seems difficult, and remains an open question.

A fundamental difference between any deterministic method and its stochastic counterpart is that all the results in the stochastic setting hold in expectation, while the ones in the deterministic setting hold deterministically. This notwithstanding, the stochastic results are usually tighter, e.g., better rates can be achieved, mainly due to the characteristic that the expected value of the search direction is an unbiased estimate of the negative gradient direction.

Considering the Incremental Aggregated Gradient (IAG) method and its stochastic counterpart, the Stochastic Average Gradient (SAG) method,

- IAG employs a fixed stepsize
$$\alpha < \left(\frac{8}{25N\kappa}\right)\frac{1}{\mu + L_w},$$
achieving a convergence rate to the optimal value $f^\star$ of
$$(1 - \frac{c_N}{(\kappa+1)^2})^2,$$
where $c_N = \frac{2}{25N(2N+1)}$;
- SAG employs a fixed stepsize
$$\alpha = \frac{1}{16L_w},$$
achieving a convergence rate to the optimal value $f^\star$ of
$$1 - \min\{\frac{1}{16\kappa}, \frac{1}{8N}\}.$$

Observe that the IAG method employs a stepsize that depends, beyond $L_w$, on a quantity, $\mu$, that is hardly known in practice, whereas SAG only needs the Lipschitz constant $L_w$. Furthermore, IAG's rate has a quadratic dependence on the condition number, while SAG has linear dependence, since the expected value of the direction is the true gradient and therefore $E[e_k] = 0$. Last but not least, it must be remarked that SAG's convergence results rely on the exact stepsize, which depends on not known constants, while IAG's ones hold for any stepsize smaller than a constant.

DIAG and Finito methods were introduced to improve the convergence rates of IAG and SAG. In summary,

- DIAG employs a fixed stepsize
$$\alpha = \frac{2}{\mu + L_w},$$
achieving a convergence rate to the optimal value $f^\star$ better than the full gradient one, namely
$$1 - \frac{1}{\kappa};$$
- Finito employs a fixed stepsize
$$\alpha = \frac{1}{c\mu},$$
with $c \geq 2$, achieving a convergence rate to the optimal value $f^\star$ of
$$1 - \frac{1}{cN}.$$

Observe that, in this case, the DIAG method relies on an exact stepsize that depends on both the strong convexity constant $\mu$ and the Lipschitz constant $L_w$, while Finito's results hold for any stepsize smaller than a threshold, which depends on the Lipschitz constant only. Nevertheless, it must be underlined that the Finito method assumes the 'big data assumption', as stated by the authors, which could not hold in real applications.

Finally, to the best of this dissertation author's understanding, a deterministic counterpart to the SVRG method has not been proposed yet, and it is not clear whether similar convergence results could be proved, or not.

# Chapter 4

# New dynamic batching techniques based on the Fisher test

## 4.1 Motivation

Dynamic minibatching gradient methods to solve problem 1.3 employ the iteration

$$w_{k+1} = w_k - \alpha_k \nabla f_{S_k}(w_k),$$

with

$$\nabla f_{S_k}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k),$$

where $S_k \subseteq \{1, \dots, N\}$ is called a minibatch, i.e. a subset of indices drawn from $\{1, \dots, N\}$, and $|S_k|$ is its cardinality.

The aim of this chapter is to introduce a new test to determine whether the minibatch size $|S_k|$ of a dynamic minibatching gradient method should be increased, or not. Indeed, when developing a dynamic batching technique, the main question that needs be answered is whether the gradient estimate $\nabla f_{S_k}(w_k)$, where $S_k$ is a minibatch of samples, is a good approximation of the true gradient $\nabla f(w_k)$. Furthermore, one may want to answer the even more difficult question: is the estimate $\nabla f_{S_k}(w_k)$ consistently a good approximation of the true gradient $\nabla f(w_k)$, as the algorithm updates $w_k \in \mathbb{R}^d$? In the literature, significant effort has been put in defining a good test in order to determine whether the current gradient estimate is similar enough to the true gradient [22, 18], although the true gradient is usually unknown in the SG/ IG settings.

Looking at the above question from a different angle, one may recognize that it may be reformulated as: are the gradient estimates produced by the algorithm similar to each other? In fact, in the extreme case where all the estimates are produced similar to each other at any iteration, one would have that the probability of having consistently estimated the true gradient with a good level of accuracy is high. Indeed, this second question may be seen as: is the current estimate of the mean of the population, based on a subsample (i.e. the minibatch gradient), a good approximation of the true mean of the population (i.e. the true gradient)? Therefore, in a way, the problem of estimating the gradient based on a minibatch may be considered as a mean estimate problem, which is a well-known problem. Indeed, it has been addressed by statisticians for decades, in a slightly different manner, by employing the Fisher test (aka ANOVA - Analysis of Variance) [88]. The Fisher test was designed to determined whether the means within the groups of a population were statistically different from the mean of the population.

The comparison with dynamic batching techniques is evident: the objective is to determine whether the mean of a group, i.e. a minibatch gradient $\nabla f_{S_k}(w_k)$, is representative of the mean of the entire population, i.e. the full gradient $\nabla f(w_k)$.

Although the Fisher test seems a natural way of developing a test to check the quality of the minibatch gradient approximations, it must be noticed that an exact Fisher test can not be applied in the SGD regime, since the minibatch gradients are computed at a different point $w_k \in \mathbb{R}^n$ at each iteration $k$ of the algorithm. Therefore, comparing minibatch gradients computed at different iterations may be misleading, since those may be different due to the combination of two effects: (i) the effect of sampling in the data space and (ii) the effect of computing the gradients in different points in the variable space.

These and other issues will be addressed in the next sections.

## 4.2 Introduction and literature review

**Algorithm 11:** Generic SG/ IG method with minibatch size control

**for** $k = 0, 1, \ldots$ **do**
  Identify a minibatch $S_k \subseteq \{1, \ldots, N\}$ of size $|S_k|$;
  Compute a search direction $d_k$ based on $S_k$;
  Choose a stepsize $\alpha_k > 0$;
  Update $w_{k+1} = w_k + \alpha_k d_k$;
  Determine $|S_{k+1}| \geq |S_k|$.
**end**

*Dynamic minibatching* methods employ a minibatching scheme of the form

$$w_{k+1} = w_k + \alpha_k d_k, \qquad d_k = -\nabla f_{S_k}(w_k) = -\frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(w_k), \tag{4.1}$$

and control the minibatch dimension and composition throughout the optimization procedure. Observe that a framework like algorithm 11 can be analyzed both from the stochastic and deterministic viewpoint, i.e. by considering the computation of a minibatch of gradients of the component functions as a means to get gradient approximation variance/ error reduction.

Observe that, when employing an iterate like (4.1), it holds that

- in the stochastic setting, where the indices included in $S_k$ are selected at random from $\{1, \ldots, N\}$,

$$E[d_k] = -\nabla f(w_k);$$

- in the deterministic setting, where the indices included in $S_k$ are selected cyclically from $\{1, \ldots, N\}$,

$$d_k = g_k + e_k,$$

  where

$$g_k = -\nabla f(w_k) \quad \text{and} \quad e_k = \nabla f(w_k) - \nabla f_{S_k}(w_k).$$

Therefore, both the analyses carried on in the previous chapter for the deterministic and stochastic settings apply. Remark that, below the two facts above, assumptions on the gradient approximation error/ variance must hold. Indeed in the stochastic setting, it was proved, see e.g. [44, 19], that an

increase in the minibatch size at geometric rates not only guarantees convergence without the need of a *diminishing stepsize* like (3.8), but preserves the linear convergence rate of the full gradient method. Unfortunately this hardly works well in practice, since such a speed of increase in the minibatch size would imply high computational costs after a few iterations.

For these reasons several works on how to adaptively determine the minibatch size [18, 22, 67] and composition [113] were proposed in order to let the variance/ error of the search direction go to zero. In particular, in [22] the test

$$E\left[\|\nabla f_{S_k}(w_k) - \nabla f(w_k)\|\right] \leq \delta \|\nabla f(w_k)\|, \qquad \delta \in [0,1), \tag{4.2}$$

later named *norm test*, was proposed to check if the search direction given the current size of the batch would be close to the steepest one, i.e. the anti-gradient, "with high probability". It must be remarked that

$$E\left[\|\nabla f_{S_k}(w_k) - \nabla f(w_k)\|^2\right] = \|Var(\nabla f_{S_k}(w_k))\|_1,$$

based on which, the following approximation of (4.2) was proposed:

$$\frac{\|Var_{i\in S_k}(\nabla f_i(w_k))\|_1}{|S_k|} \leq \delta^2 \|\nabla f_{S_k}(w_k)\|^2, \qquad \delta \in [0,1). \tag{4.3}$$

Based on this test, Byrd et al. [22] proved the following result in the strongly convex case.

**Theorem 23.** *Let assumptions 1 and 3 hold. Let the iterate update be*

$$w_{k+1} = w_k - \alpha \nabla f_{S_k}(w_k),$$

*where*

$$0 < \alpha < \frac{1}{L_w}$$

*and*

$$\|\nabla f_{S_k}(w_k) - \nabla f(w_k)\| \leq \theta \|\nabla f_{S_k}(w_k)\|,$$

*for some* $\theta \in (0,1)$. *Then*

$$f(w_{k+1}) \leq \left(1 - \frac{\beta\mu}{L_w}\right) f(w_k),$$

*with* $\beta = \frac{(1-\theta)^2}{2(1+\theta)^2}$, *and*

$$\lim_{k\to\infty} w_k = w^\star,$$

*where* $w^\star$ *is the unique solution of problem 1.3. Furthermore, the number of iterations k to get* $f(w_k) \leq f(w^\star) + \varepsilon$ *are*

$$\frac{L_w}{\beta\mu}\left[\log\left(\frac{1}{\varepsilon}\right) + log(f(w_0))\right],$$

*and*

$$\|\nabla f_{S_k}(w_k)\|^2 \leq \frac{1}{(1-\theta)^2}\frac{2L_w^2 f(w_0)}{\mu}\left(1 - \frac{\beta\mu}{L_w}\right)^k.$$

Theorem 23 shows that linear convergence to a solution can also be ensured by requiring the gradient estimate $\nabla f_{S_k}(w_k)$ satisfy a norm condition, at each iteration $k$.

In [18], the *norm test* was substituted by a weaker one named *inner product test*

$$E\left[\left(\nabla f_{S_k}(w_k)^T \nabla f(w_k) - \|\nabla f(w_k)\|^2\right)^2\right] \leq \delta^2 \|\nabla f(w_k)\|^4, \tag{4.4}$$

an approximation of which can be written as

$$\frac{Var_{i\in S_k}(\nabla f_i(w_k)^T \nabla f_{S_k}(w_k))}{|S_k|} \leq \delta^2 \|\nabla f(w_k)\|^4, \tag{4.5}$$

where

$$Var_{i\in S_k}(\nabla f_i(w_k)^T \nabla f_{S_k}(w_k)) = \frac{1}{|S_k|-1} \sum_{i\in S_k} \left(\nabla f_i(w_k)^T \nabla f_{S_k}(w_k) - \|\nabla f_{S_k}(w_k)\|^2\right)^2.$$

In order to get linear convergence of a gradient method employing the inner product test, it must be ensured that the gradient estimate $\nabla f_{S_k}(w_k)$ is never too close to orthogonality w.r.t. the true gradient $\nabla f(w_k)$. This is done by asking the gradient estimate to satisfy another test, i.e. the *orthogonality test*

$$E\left[\left\|\nabla f_{S_k}(w_k) - \frac{\nabla f_{S_k}(w_k)^T \nabla f(w_k)}{\|\nabla f(w_k)\|^2} \nabla f(w_k)\right\|^2\right] \leq \tau^2 \|\nabla f(w_k)\|^2.$$

Just like the inner product test, this can be approximated, by substituting the quantities $\nabla f_{S_k}(w_k)$ and $\nabla f(w_k)$ by their sample approximations, to be actually computable.

Bollapragada et al. proved [18] convergence of a gradient method employing the *exact* inner product and orthogonality tests to a solution of problem 1.3, in the stochastic setting, as shown in theorem 24 for the nonconvex case, in theorem 25 for the strongly convex one.

**Theorem 24.** *Let assumption 1 hold and let $f$ be bounded from below by $f_{inf}$. Let the iterate update be*

$$w_{k+1} = w_k - \alpha \nabla f_{S_k}(w_k),$$

*where*

$$0 < \alpha \leq \frac{1}{(1+\delta^2+\tau^2)L_w}$$

*and the exact inner product and orthogonality tests are satisfied, with some $\delta, \tau > 0$, at each iteration. Then*

$$\lim_{k\to\infty} E[\|\nabla f(w_k)\|^2] = 0.$$

*Furthermore,*

$$\min_{0\leq k\leq K-1} E[\|\nabla f(w_k)\|^2] \leq \frac{2}{\alpha K}(f(w_0) - f_{inf}).$$

**Theorem 25.** *Let assumptions 1 and 3 hold. Let the iterate update be*

$$w_{k+1} = w_k - \alpha \nabla f_{S_k}(w_k),$$

*where*

$$0 < \alpha \leq \frac{1}{(1+\delta^2+\tau^2)L_w}$$

*and the exact inner product and orthogonality tests are satisfied, with some $\delta, \tau > 0$, at each iteration. Then*

$$E[f(w_k)] - f(w^\star) \leq \rho^k (f(w_0) - f(w^\star)),$$

*where*

$$\rho = 1 - \mu\alpha.$$

It is interesting to study the similarities between the norm and inner product tests. In fact one can write:

$$\begin{aligned} Var_{i\in S_k}(\nabla f_i(w_k)^T \nabla f_{S_k}(w_k)) &= E\left[\left(\nabla f_i(w_k)^T \nabla f_{S_k}(w_k) - \|\nabla f_{S_k}(w_k)\|^2\right)^2\right] = \\ &= E\left[\left(\nabla f_i(w_k)^T \nabla f_{S_k}(w_k)\right)^2\right] - \|\nabla f_{S_k}(w_k)\|^4 \leq \\ &\leq E\left[\|\nabla f_i(w_k)\|^2 \|\nabla f_{S_k}(w_k)\|^2\right] - \|\nabla f_{S_k}(w_k)\|^4 = \\ &= \|\nabla f_{S_k}(w_k)\|^2 \left(E\left[\|\nabla f_i(w_k)\|^2\right] - \|\nabla f_{S_k}(w_k)\|^2\right) = \\ &= \|\nabla f_{S_k}(w_k)\|^2 Var_{i\in S_k}(\|\nabla f_i(w_k)\|). \end{aligned}$$

Therefore the test

$$\frac{\|\nabla f_{S_k}(w_k)\|^2 Var_{i\in S_k}(\|\nabla f_i(w_k)\|)}{|S_k|} \leq \delta^2 \|\nabla f(w_k)\|^4,$$

that by simplifying equals the norm test (4.3), i.e.,

$$\frac{Var_{i\in S_k}(\|\nabla f_i(w_k)\|)}{|S_k|} \leq \delta^2 \|\nabla f(w_k)\|^2,$$

implies the inner product test (4.5), which is less restrictive.

All of the methods to reduce gradient approximation variance described up to now are based on the function structure, namely the fact that $f$ has a finite sum structure. Nevertheless, recalling that in ML the function components are defined as

$$f_i(w) = \phi(w; x^i, y^i),$$

another approach to control the gradient approximation error/ variance is to leverage the dataset structure, i.e. how the input-output pairs $\{z^i\}_{i=1}^N = \{x^i, y^i\}_{i=1}^N$ are structured in the input-output space.

In [52] a neighboring system is introduced to give aggregation methods, like the ones introduced in chapter 3, a higher degree of information when updating the sum components $\nabla f_i$. In fact, if an index $i_k$ is extracted at iteration $k$ of, say, SAG, then the newly computed $\nabla f_{i_k}(w_k)$ can be used to update not only the $i_k$-th sum term, but also the sum terms of the indices in the neighborhood of $z^{i_k}$. In [52], the dimension of the neighborhood must be fixed a priori.

In [4], the concept of *raw clustering* is introduced and applied to improve SVRG [58]. Given a clustering $C_1 \cup \cdots \cup C_{N_c}$ of $\{1, \ldots, N\}$ and denoting by $c_i \in \{1, \ldots, N_c\}$ the cluster that index $i$ belongs to, the following iterate update is performed

$$w_{k+1} = w_k - \eta \widetilde{\nabla} f(w_k), \qquad \widetilde{\nabla} f(w_k) = \frac{1}{N}\sum_{i=1}^{N} (\nabla f_i(\widetilde{w}) + \chi_{c_i}) + \nabla f_{i_k}(w_k) - (\nabla f_{i_k}(\widetilde{w}) + \chi_{c_{i_k}}), \quad (4.6)$$

where $\widetilde{w}$ is a snapshot of the iterate, as introduced in algorithm (9), and $i_k$ is chosen uniformly at random from $\{1, \ldots, N\}$. The authors propose two different options for updating $\chi_{c_{i_k}}$, one of which being

$$\chi_{c_{i_k}} = \nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w}). \quad (4.7)$$

To better understand how (4.6) works, assume that at iteration $k$ the selected index is $i_k$ and $z^{i_k}$ belongs to cluster $C_{j_k} : j_k \in \{1, \ldots, N_c\}$. Then the summation in (4.6) has three types of terms:

1. those which index $i \neq i_k$ represents input-output pairs $z^i$ in clusters $C_j : j \in \{1, \ldots, N_c\}$, $j \neq j_k$. In this case the term in the summation becomes

$$(\nabla f_i(\widetilde{w}) + \nabla f_i(\overline{w}) - \nabla f_i(\widetilde{w})) + \nabla f_{i_k}(w_k) - (\nabla f_{i_k}(\widetilde{w}) + \nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w})),$$

that simplified turns out to be

$$\nabla f_i(\overline{w}),$$

where $\overline{w}$ is the value of the iterate at the last iteration in which an index from the same cluster was selected;

2. those indexed by $i \neq i_k$, representing input-output pairs $z^i$ in the same cluster $C_{j_k}$. In this case the term in the summation becomes

$$(\nabla f_i(\widetilde{w}) + \nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w})) + \nabla f_{i_k}(w_k) - (\nabla f_{i_k}(\widetilde{w}) + \nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w})),$$

which yields

$$\nabla f_i(\widetilde{w}) + \nabla f_{i_k}(w_k) - \nabla f_{i_k}(\widetilde{w});$$

3. the one with index $i = i_k$, which term in the summation turns out to be

$$(\nabla f_i(\widetilde{w}) + \nabla f_i(w_k) - \nabla f_i(\widetilde{w})) + \nabla f_i(w_k) - (\nabla f_i(\widetilde{w}) + \nabla f_i(w_k) - \nabla f_i(\widetilde{w})) = \nabla f_i(w_k).$$

Therefore, the *raw clustering* information is leveraged to update the snapshot gradient more frequently, namely a sum component $i_k$ is updated every time an input-output pair in the same cluster is selected. Obviously, this update is affected by a certain degree of noise. Indeed, the gradient computed at an input-output pair in the same cluster is different, up to a certain degree, depending on the number of clusters and the quality of the clustering, from the gradient computed at the input-output pair under consideration.

In the following section, a Fisher test-based dynamic batching technique is introduced and adapted to the SG/ IG settings to solve problem 1.3. Then, the convergence properties of the introduced method are analyzed in section 4.4. Finally, some implementation details and numerical results on standard ML tasks are presented.

## 4.3 The Fisher test for dynamic batching

The algorithm proposed in this section employs a gradient-based iterate update

$$w_{k+1} = w_k - \alpha_k \nabla f_{S_k^{j_k}}, \tag{4.8}$$

where $\alpha_k > 0$ is a step length and $\nabla f_{S_k^{j_k}}(w_k)$ is a gradient estimate based on the minibatch $S_k^{j_k} \subseteq \{1, \ldots, N\}$, namely

$$\nabla f_{S_k^{j_k}}(w_k) = \frac{1}{|S_k|} \sum_{i \in S_k^{j_k}} \nabla f_i(w_k).$$

Let $j_k = 1, \ldots, M_k$ be the indices of a partition of $\{1, \ldots, N\}$, i.e.

$$S_k^1, S_k^2, \ldots, S_k^{M_k} : \bigcup_{j_k=1}^{M_k} S_k^{j_k} = \{1, \ldots, N\} \; \forall k \quad \text{and} \quad S_k^{j_k^1} \bigcap S_k^{j_k^2} = \varnothing \; \forall j_k^1 \neq j_k^2.$$

One of the crucial points of the algorithm is how to update the size of the minibatch $|S_k|$, when strictly necessary. Ideally, one could do so by comparing the "within-minibatch" variance to the "between-minibatches" one, like in the well-known *F-test* [32, 88], which aim is to verify if the variance of the values within a group of samples is significant when compared to the variance between the means of the groups. The *F-test*, also known as *Analysis of Variance (ANOVA)*, is based on the decomposition of the total variance of a population divided in groups in the variance within the groups and the variance between the means of the groups. In the above setting, if the aim is to control the variance of the distance of the gradient approximation to the true gradient, the total variance at epoch $k$, $T_k$, can be decomposed in:

- $V_k$, the variance within all the minibatches of samples $S_k^1, S_k^2, \dots, S_k^{M_k}$ that could be employed at a given iteration;
- $W_k$, the variance between the means (i.e. the gradient estimates) of such minibatches.

Therefore the following can be written ([32])

$$T_k = V_k + W_k, \tag{4.9}$$

where

$$T_k = \frac{1}{N}\sum_{i=1}^{N} \|\nabla f_i(w_k) - \nabla f(w_k)\|^2 = \frac{1}{N}\sum_{j_k=1}^{M_k}\sum_{h\in S_k^{j_k}} \left\|\nabla f_{j_k h}(w_k) - \nabla f(w_k)\right\|^2, \tag{4.10}$$

$$V_k = \frac{1}{N}\sum_{j_k=1}^{M_k}\sum_{h\in S_k^{j_k}} \left\|\nabla f_{j_k h}(w_k) - \nabla f_{S_k^{j_k}}(w_k)\right\|^2, \tag{4.11}$$

and

$$W_k = \frac{1}{N}\sum_{j_k=1}^{M_k} |S_k^{j_k}| \left\|\nabla f_{S_k^{j_k}}(w_k) - \nabla f(w_k)\right\|^2. \tag{4.12}$$

Similarly to the above, where a norm condition is enforced, one could be interested [18] in controlling an angle condition between the gradient approximation and the true gradient, in order to guarantee that the search direction in (4.8) is a descent direction. In this framework, the total variance of the angle can be decomposed as

$$T_k = V_k + W_k, \tag{4.13}$$

where

$$T_k = \frac{1}{N}\sum_{i=1}^{N} \left|\nabla f_i(w_k)^T \nabla f(w_k) - \|\nabla f(w_k)\|^2\right|^2 = \frac{1}{N}\sum_{j_k=1}^{M_k}\sum_{h\in S_k^{j_k}} \left|\nabla f_{j_k h}(w_k)^T \nabla f(w_k) - \|\nabla f(w_k)\|^2\right|^2, \tag{4.14}$$

$$V_k = \frac{1}{N}\sum_{j_k=1}^{M_k}\sum_{h\in S_k^{j_k}} \left|\nabla f_{j_k h}(w_k)^T \nabla f_{S_k^{j_k}}(w_k) - \|\nabla f_{S_k^{j_k}}(w_k)\|^2\right|^2, \tag{4.15}$$

and

$$W_k = \frac{1}{N}\sum_{j_k=1}^{M_k} |S_k^{j_k}| \left|\nabla f_{S_k^{j_k}}(w_k)^T \nabla f(w_k) - \|\nabla f(w_k)\|^2\right|^2. \tag{4.16}$$

In the following, the analysis will be carried on the norm condition and the total variance as defined in (4.10). Under the assumptions of data normality and homoscedasticity, the *exact Fisher test* can be defined as

$$\frac{\frac{W_k}{d_1}}{\frac{V_k}{d_2}} \geq F_{d_1,d_2,\varepsilon_k}, \tag{4.17}$$

where $F_{d_1,d_2,\varepsilon_k}$ is the $\varepsilon_k$-percentile of a Fisher distribution with degrees of freedom $d_1 = M_k - 1$ and $d_2 = N - M_k$. If the test is satisfied, then the minibatch gradient estimates are significantly different one to another (i.e. at least one is different from the others). Therefore no minibatch gradient can be considered as a reliable estimate of the true gradient. Indeed, this decision might be wrong with probability $\varepsilon_k$, which defines the test significance level. If the normality and homoscedasticity assumptions are violated, the test remains valid providing that the minibatches have sufficiently large size and are balanced (i.e. the sizes are similar) [90, 46]. If (4.17) is satisfied, the size $|S_k|$ of the minibatch should be increased. The rationale behind such a choice is that in order for a minibatch to be representative of the whole population, the minibatch should include most of the total variance. It must be observed that, for fixed $d_1, d_2 > 0$, the percentile $F_{d_1,d_2,\varepsilon_k}$ goes to 0 as $\varepsilon_k$ goes to 1 (see e.g. [32] for the theoretical properties of the Fisher distribution). Letting $F_{d_1,d_2,\varepsilon_k}$ go to 0 means that the test is more easily satisfied and therefore the minibatch size is increased until $W_k$ becomes small enough to fail the test. As a consequence, as one gets further with the optimization, the minibatch size is increased implying that the gradient estimates get closer to the true gradient.

It is then convenient to reformulate the test in the following form

$$\frac{W_k}{V_k} \geq \eta_k \|\nabla f(w_k)\|^2, \tag{4.18}$$

where $\eta_k > 0$, and the term $\|\nabla f(w_k)\|^2$ is multiplied to the right hand side for scaling $\eta_k$ with the gradient norm. Note that $\eta_k$ can be chosen either fixed, e.g. such that the significance of the $F$-test is fixed, or diminishing.

Algorithm 12 implements the above rationale to dynamically control the minibatch size, by the use of the *exact Fisher test* (4.18).

**Algorithm 12:** Exact F-test cyclic dynamic batching algorithm

1 Data: $w_0 \in \mathbb{R}^n$, $\Delta_0, \eta_0 > 0$;
2 **while** *a convergence test is not satisfied* **do**
3 Compute $d_k = -\nabla f_{S_k^{j_k}}(w_k)$;
4 Compute a step length $\alpha_k > 0$;
5 New iterate: $w_{k+1} = w_k + \alpha_k d_k$;
6 Compute the between- and within-minibatches variance as in (4.11) and (4.12);
7 If condition (4.18) is satisfied, $|S_{k+1}| > |S_k|$;
8 Compute $\eta_{k+1} \leq \eta_k$;
9 $k \leftarrow k+1$.
10 **end**

At step 3, the gradient estimate is computed and the search direction is set to the negative gradient estimate, while at step 4 a stepsize is computed: note that the procedure to compute the stepsize is not explicitely mentioned, thus a small, fixed stepsize may be employed. At step 5 the iterate is updated and at step 6 the quantities $W_k$ and $V_k$ are computed. Finally, at step 7 the Fisher test is employed to

decide wether the minibatch size should be increased or not, and at step 8 the parameter $\eta_k$ is updated. Note that steps 7 and 8 are vague, just like step 4, and will be further developed in section 4.5.

## 4.4 Theoretical analysis

In this section, the convergence properties of Algorithm 12 are investigated. First, an assumption on the total variance of the gradients is reported.

**Assumption 10** (Bound on total variance)**.** *For all $w \in \mathbb{R}^d$, there exist constants $C, D > 0$ such that $T \leq C + D\|\nabla f(w)\|^2$, with $T$ defined as in* (4.10)*.*

Theorem 28 proves convergence of Algorithm 12 with a dynamic, automatically updated stepsize to a solution of problem 1.3 when assumptions 1, 3 and 10 hold. The following lemma is useful in the proof of the theorem.

**Lemma 6.** *Let $\theta_k = \frac{\eta_k T_k}{1+\eta_k\|\nabla f(w_k)\|^2}$ as defined in* (4.10) *and let Assumptions 2 and 10 hold. Let*

$$\eta_0 \leq \frac{1}{C + (D-1)B^2}. \tag{4.19}$$

*Then we have*

$$\theta_k < 1 \quad \forall\, k. \tag{4.20}$$

*Proof.* By the definition of $\theta_k$, it holds that

$$\begin{aligned}\theta_k &= \frac{\eta_k T_k}{1+\eta_k\|\nabla f(w_k)\|^2} \\ &\leq \frac{\eta_k(C + D\|\nabla f(w_k)\|^2)}{1+\eta_k\|\nabla f(w_k)\|^2} \\ &\leq \frac{\eta_k(C + DB^2)}{1+\eta_k B^2}.\end{aligned}$$

Recalling that $\eta_{k+1} \leq \eta_k$ for all $k$ and (4.19), we get (4.20). □

**Theorem 26** (*Exact Fisher test - function descent*)**.** *Let assumptions 1, 2, 3 and 10 hold. Let iteration $k$ be such that test* (4.18) *is not satisfied, $\eta_k = \overline{\eta} \leq \frac{1}{C+(D-1)B^2}$ and $\alpha_k = \frac{1-\theta_k}{L_w}$, where*

$$\theta_k = \frac{\overline{\eta} T_k}{1+\overline{\eta}\|\nabla f(w_k)\|^2}, \tag{4.21}$$

*with $T_k$ defined as in* (4.10)*. Then*

$$f(w_{k+1}) \leq f(w_k) - \frac{\beta_k}{L_w}\|\nabla f(w_k)\|^2, \tag{4.22}$$

*where $\beta_k = \frac{(1-\theta_k)^3}{2}$.*

*Proof.* If test (4.18) is not true, this means that

$$\frac{W_k}{V_k} \leq \overline{\eta}\|\nabla f(w_k)\|^2. \tag{4.23}$$

Recalling (4.9), the following can be written

$$\frac{W_k}{T_k - W_k} \leq \overline{\eta}\|\nabla f(w_k)\|^2,$$

where $T_k$ is the total variance, i.e. $T_k = \frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(w_k) - \nabla f(w_k)\|^2$. By simple calculation it holds

$$W_k \leq \frac{\overline{\eta} T_k \|\nabla f(w_k)\|^2}{1 + \overline{\eta}\|\nabla f(w_k)\|^2}, \tag{4.24}$$

from which

$$\|\nabla f_{S_k^{j_k}}(w_k) - \nabla f(w_k)\|^2 \leq \frac{\overline{\eta} T_k \|\nabla f(w_k)\|^2}{1 + \overline{\eta}\|\nabla f(w_k)\|^2} - \sum_{h \neq j_k}^{M_k} \|\nabla f_{S_k^{j_k}}(w_k) - \nabla f(w_k)\|^2 \tag{4.25}$$

$$\leq \frac{\overline{\eta} T_k \|\nabla f(w_k)\|^2}{1 + \overline{\eta}\|\nabla f(w_k)\|^2} \tag{4.26}$$

$$\leq \theta_k \|\nabla f(w_k)\|^2, \tag{4.27}$$

for any $j_k = 1, \ldots, M_k$, where

$$\theta_k = \frac{\overline{\eta} T_k}{1 + \overline{\eta}\|\nabla f(w_k)\|^2}.$$

Observing that, by Assumption 10, $\theta_k < 1$, one gets

$$\|\nabla f_{S_k^{j_k}}(w_k)\|^2 - \|\nabla f(w_k)\|^2 \leq \|\nabla f_{S_k^{j_k}}(w_k) - \nabla f(w_k)\|^2 \leq \theta_k \|\nabla f(w_k)\|^2$$

and

$$\|\nabla f(w_k)\|^2 - \|\nabla f_{S_k^{j_k}}(w_k)\|^2 \leq \|\nabla f_{S_k^{j_k}}(w_k) - \nabla f(w_k)\|^2 \leq \theta_k \|\nabla f(w_k)\|^2.$$

Therefore, from (4.25) and the relations above, the following holds

$$(1 - \theta_k)\|\nabla f(w_k)\|^2 \leq \|\nabla f_{S_k^{j_k}}(w_k)\|^2 \leq (1 + \theta_k)\|\nabla f(w_k)\|^2. \tag{4.28}$$

Furthermore,

$$\|\nabla f_{S_k^{j_k}}(w_k) - \nabla f(w_k)\|^2 = \|\nabla f(w_k)\|^2 + \|\nabla f_{S_k^{j_k}}(w_k)\|^2 - 2\nabla f(w_k)^T \nabla f_{S_k^{j_k}}(w_k) \leq \theta_k \|\nabla f(w_k)\|^2,$$

from which, using the l.h.s. of (4.28), it can be written

$$2\nabla f(w_k)^T \nabla f_{S_k^{j_k}}(w_k) \geq (1 - \theta_k)\|\nabla f(w_k)\|^2 + \|\nabla f_{S_k^{j_k}}(w_k)\|^2 \geq 2(1 - \theta_k)\|\nabla f(w_k)\|^2,$$

which yields

$$\nabla f(w_k)^T \nabla f_{S_k^{j_k}}(w_k) \geq (1 - \theta_k)\|\nabla f(w_k)\|^2. \tag{4.29}$$

Combining (4.28) and (4.29), recalling Taylor's theorem and Assumption 3,

$$\begin{aligned}
f(w_{k+1}) &\leq f(w_k) - \frac{(1-\theta_k)}{L_w}\nabla f(w_k)^T \nabla f_{S_k^{j_k}}(w_k) + \frac{L_w}{2}\left(\frac{1-\theta_k}{L_w}\right)^2 \|\nabla f_{S_k^{j_k}}(w_k)\|^2 \\
&\leq f(w_k) - \frac{(1-\theta_k)}{L_w}(1-\theta_k)\|\nabla f(w_k)\|^2 + \frac{L_w}{2}\left(\frac{1-\theta_k}{L_w}\right)^2 \|\nabla f_{S_k^{j_k}}(w_k)\|^2 \\
&\leq f(w_k) - \frac{(1-\theta_k)^2}{L_w}\|\nabla f(w_k)\|^2 + \frac{L_w}{2}\left(\frac{1-\theta_k}{L_w}\right)^2 (1+\theta_k)\|\nabla f(w_k)\|^2 \\
&\leq f(w_k) - \left(\frac{(1-\theta_k)^2}{L_w} - \frac{(1+\theta_k)(1-\theta_k)^2}{2L_w}\right)\|\nabla f(w_k)\|^2 \\
&\leq f(w_k) - \frac{(1-\theta_k)^3}{2L_w}\|\nabla f(w_k)\|^2 \\
&\leq f(w_k) - \frac{\beta_k}{L_w}\|\nabla f(w_k)\|^2,
\end{aligned}$$

where $\beta_k = \frac{(1-\theta_k)^3}{2}$, which completes the proof. □

Theorem 27 proves convergence of algorithm 12 to stationary points in the nonconvex case.

**Theorem 27** (*Exact Fisher test - convergence in the nonconvex case*)**.** *Let assumptions 1, 2 and 10 hold. Let $\{w_k\}$ be the sequence generated by Algorithm 12 with $\eta_k = \overline{\eta} \leq \frac{1}{C+(D-1)B^2}$ and $\alpha_k = \frac{1-\theta_k}{L_w}$, where*

$$\theta_k = \frac{\overline{\eta} T_k}{1+\overline{\eta}\|\nabla f(w_k)\|}, \tag{4.30}$$

*with $T_k$ as defined in* (4.10)*. Let $f$ be bounded from below. Then*

$$\lim_{k\to\infty} \|\nabla f(w_k)\| = 0. \tag{4.31}$$

*Proof.* Like in theorem 28, assume there exists a $\overline{k} > 0$ such that $|S_k| < N$ for all $k \geq \overline{k}$. Then, recalling 4.22, it holds that

$$f(w_{k+1}) \leq f(w_k) \qquad \forall\, k \geq \overline{k},$$

from which, recalling the continuity and boundedness from below of $f$,

$$\lim_{k\to\infty} f(w_k) = \overline{f}.$$

By assumption $\theta_k$ is bounded away from 1 and therefore $\beta_k$ is bounded away from 0. This yields

$$\lim_{k\to\infty} \|\nabla f(w_k)\|^2 = 0,$$

which completes the proof. □

The convergence properties of Algorithm 12 for strongly convex objectives are stated in the next theorem.

**Theorem 28** (*Exact Fisher test - convergence in the strongly convex case*)**.** *Let assumptions 1, 2, 3 and 10 hold. Let* $\{w_k\}$ *be the sequence generated by Algorithm 12 with* $\eta_k = \overline{\eta} \leq \frac{1}{C+(D-1)B^2}$ *and* $\alpha_k = \frac{1-\theta_k}{L_w}$*, where*

$$\theta_k = \frac{\overline{\eta} T_k}{1+\overline{\eta}\|\nabla f(w_k)\|^2},$$

*with* $T_k$ *as defined in* (4.10)*. Then*

$$\lim_{k\to\infty} f(w_k) = f(w^\star), \tag{4.32}$$

*and*

$$\lim_{k\to\infty} w_k = w^\star, \tag{4.33}$$

*where* $w^\star$ *is a solution of problem 1.3.*
*Furthermore, if a* $\overline{k} > 0$ *exists such that* $|S_k| < N$ *for all* $k \geq \overline{k}$*, the number of iterations K to obtain* $f(w_K) \leq f(x^\star) + \varepsilon$ *is at most*

$$\frac{L_w}{\lambda\overline{\beta}}\left(\log f(w_{\overline{k}}) + \log\left(\frac{1}{\varepsilon}\right)\right), \tag{4.34}$$

*where*

$$\overline{\beta} = \min_{k\geq\overline{k}} \beta_k \quad \textit{and} \quad \beta_k = \frac{(1-\theta_k)^3}{2}.$$

*Proof.* There exist two cases:

(i) there exists a $\widehat{k} > 0$ such that $|S_k| = N$ for all $k \geq \widehat{k}$;

(ii) there exists a $\overline{k} > 0$ such that $|S_k| < N$ for all $k \geq \overline{k}$.

In case (i), the iteration reduces to a standard full gradient iteration, i.e.

$$w_{k+1} = w_k - \alpha_k \nabla f(w_k),$$

where $\alpha_k = \frac{1-\theta_k}{L_w}$ with $\theta_k \in (0,1)$, which trivially yields convergence (see e.g. [11]).

The next lines will focus on case (ii), and in particular iterations with $k \geq \overline{k}$, where $|S_k|$ is not increased. Following the lines of [22, Section 4.1], by Assumption 1,

$$\|\nabla f(w_k)\|^2 \geq \lambda\left(f(w_k) - f(w^\star)\right),$$

where $w^\star$ is a solution of problem 1.3. Combining this with (4.22), it holds that

$$f(w_{k+1}) \leq f(w_k) - \frac{\lambda\beta_k}{L_w}\left(f(w_k) - f(w^\star)\right). \tag{4.35}$$

Now, assuming that test 4.18 returns *False* at every iteration, i.e. (4.23) holds at any $k$ it can be written at iteration $k$

$$f(w_k) - f(w^\star) = \prod_{k-\overline{k}+1}^{k}\left(1 - \frac{\lambda\beta_k}{L_w}\right)\left(f(w_{\overline{k}}) - f(w^\star)\right) \tag{4.36}$$

$$\leq \left(1 - \frac{\lambda\overline{\beta}}{L_w}\right)^{k-\overline{k}}\left(f(w_{\overline{k}}) - f(w^\star)\right), \tag{4.37}$$

where $\overline{\beta} = \min_{k \geq \overline{k}} \beta_k$. Since the term inside the brackets is less than 1, this implies $f(w_k) \to f(w^\star) = 0$ and, by uniform convexity, $\{w_k\} \to w^\star$.

Assume now, without loss of generality, that $\overline{k} = 0$. By taking the log on both sides of (4.36), one gets

$$\log(f(w_k)) \leq k \log\left(1 - \frac{\lambda \overline{\beta}}{L_w}\right) + \log f(w_{\overline{k}}) \tag{4.38}$$

$$\leq -k \frac{\lambda \overline{\beta}}{L_w} + \log f(w_{\overline{k}}), \tag{4.39}$$

where the last inequality uses the concavity of the log function, from which $\log(1-x) \leq -x$. Therefore, it follows for all $k \geq \overline{k}$ that

$$k \leq \frac{L_w}{\lambda \overline{\beta}} \left(\log f(w_{\overline{k}}) - \log(f(w_k))\right). \tag{4.40}$$

If one wants $f(w_k) < \varepsilon$, this yields

$$k > \frac{L_w}{\lambda \overline{\beta}} \left(\log f(w_{\overline{k}}) + \log\left(\frac{1}{\varepsilon}\right)\right),$$

which completes the proof. □

The theorems just stated show that test (4.18), if applied at each iteration of algorithm 12 to determine whether the minibatch size $|S_k|$ should be increased or not, yields convergence in both the strongly convex and nonconvex cases. It must be noticed, nonetheless, that test (4.18) is 'exact', in the sense that computing the quantities in (4.18) equals to computing the full gradient at each iteration. Since this is not feasible in practice, in the next section some an 'inexact' approximation of test (4.18) will be introduced.

## 4.5 Implementation details

Unfortunately, the quantities $\nabla f(w_k)$, $Var_{i \in S_k^{j_k}} \nabla f_{S_k^{j_k}}(w_k)$ and $\nabla f_{S_k^{j_k}}(w_k)$, $j_k = 1, \ldots, M_k$, are not available in practice. These "ideal" quantities would imply computing, at each iteration $k$, the full gradient, but this is not feasible in the SG/ IG regime. For this reason, in algorithm 13 $\nabla \widetilde{f}(w_k)$, $\widetilde{W}_k$ and $\widetilde{V}_k$ are computed based on a memory $m$ of past iterations. In particular,

$$\nabla \widetilde{f}(w_k) = \frac{1}{\sum_{h=k-m+1}^{k} |S_h|} \sum_{h=k-m+1}^{k} |S_h| \nabla f_{S_h}(w_h), \tag{4.41}$$

and the *approximate Fisher test* is

$$\frac{\widetilde{W}_k}{\widetilde{V}_k} > \eta_k \|\nabla \widetilde{f}(w_k)\|^2, \tag{4.42}$$

where

$$\widetilde{W}_k = \frac{1}{\sum_{h=k-m+1}^{k} |S_h|} \sum_{h=k-m+1}^{k} |S_h| \left\| \nabla f_{S_h}(w_h) - \nabla \widetilde{f}(w_h) \right\|^2, \tag{4.43}$$

and

$$\widetilde{V}_k = \sum_{h=k-m+1}^{k} |S_h| Var_{i\in S_h}(\nabla f_{S_h}(w_h)). \tag{4.44}$$

Consequently, the approximate total variance is

$$\widetilde{T}_k = \widetilde{W}_k + \widetilde{V}_k, \tag{4.45}$$

which will be used in computing the adaptive stepsize, by computing

$$\widetilde{\theta}_k = \frac{\overline{\eta}\widetilde{T}_k}{1+\overline{\eta}\|\widetilde{\nabla} f(w_k)\|}. \tag{4.46}$$

Algorithm 13, where $\|\widetilde{\nabla} f(w_k)\|$, $\widetilde{W}_k$ and $\widetilde{V}_k$ are computed by keeping memory of only $m$ past observations of the minibatch gradients and within-minibatch variances, can now be proposed. Algorithm

**Algorithm 13:** Approximate F-test cyclic dynamic batching

Data: $w_0,\dots,w_m \in \mathbb{R}^n$, $\alpha_m > 0$, $\nabla\widetilde{f}(w_m) \in \mathbb{R}^n$, $\widetilde{W}_m \in \mathbb{R}$, $\widetilde{V}_m \in \mathbb{R}$, $\theta_m, \Delta_m, \eta_m > 0$;
**for** $k = m+1,\dots$ **do**
  Select $S_k \subseteq \{1, \dots, N\}$;
  Compute $\nabla f_{S_k}(w_k)$ and $Var_{i\in S_k}(w_k)$;
  Update $\nabla\widetilde{f}(w_k)$, $\widetilde{W}_k$ and $\widetilde{V}_k$ as in (4.41),(4.43),(4.44);
  Compute $\widetilde{\theta}_k = \frac{\overline{\eta}\widetilde{T}_k}{1+\overline{\eta}\|\widetilde{\nabla} f(w_k)\|}$;
  Compute $\alpha_k = (1-\theta_k)\alpha_m$;
  Update $w_{k+1} = w_k - \alpha_k \nabla f_{S_k}(w_k)$;
  **if** $\frac{\widetilde{W}_k}{\widetilde{V}_k} > \eta_k \|\nabla\widetilde{f}(w_k)\|^2$ **then**
    Compute $|S_{k+1}| > |S_k|$
  **end**
  Compute $\eta_{k+1}$.
**end**

13 relies on two main parameters that must be determined, i.e. the memory $m$ and $\overline{\eta}$. Section 4.6 will show how those were chosen in the numerical experiments.

One important open question is how to compute the new minibatch size $|S_{k+1}|$ once test (4.42) is violated. A heuristic to do so is to make $|S_k|$ grow by a quantity proportional to the ratio between the sum of the within-minibatch variances and the variance between the means of the minibatches, namely by employing the formula

$$|S_{k+1}| = \left(1 + \frac{\widetilde{W}_k}{\widetilde{V}_k}\right)^m |S_k|, \tag{4.47}$$

which represents the amount of violation of test (4.42).

### 4.5.1 How to leverage input data information

This section will focus, for simplicity and without loss of generality, to a nonlinear regression problem of the type

$$y = \phi(w, x),$$

where $w \in \mathbb{R}^d$ is a parameter vector and $\phi$ is a generic function that needs be identified based on the available data samples

$$\{x^i, y^i\}_{i=1}^N,$$

where $x^i \in \mathbb{R}^d$ and $y^i \in \mathbb{R}$ for all $i$.

The aim is to show how information on the input samples can be leveraged to reduce the gradient approximation error/ variance in an SG/ IG setting.

In this new setting, problem 1.3 can be reformulated as

$$\min_{w \in \mathbb{R}^d} f(w) = \frac{1}{N}\sum_{i=1}^{N} \phi(w, z^i) = \frac{1}{N}\sum_{i=1}^{N} f_i(w), \tag{4.48}$$

where $w \in \mathbb{R}^d$ represents the vector of parameters of the model. Lipschitz-continuity is assumed both with respect to the parameters vector $w$ and with respect to inputs $z^i$, $i = 1, .., N$, with constant bounded by $L_z$ for all $i$, namely the following relation, beyond the classical one stated in assumption 1, holds:

$$|\nabla\phi(w, z^{i_1}) - \nabla\phi(w, z^{i_2})| \leq L_z \|z^{i_1} - z^{i_2}\|, \qquad \forall\, i_1,\, i_2 \in \{1, \ldots, N\},\, \forall w \in \mathbb{R}^d, \tag{4.49}$$

A typical example where problem 4.48 must be solved is the training of NN.

A further assumption is that data samples are clustered in $N_c$ clusters $C_h$, $h \in \{1, \ldots, N_c\}$, where each cluster $C_h$ contains $|C_h| = N_h$ data samples and that an $\eta(N_c) \geq 0$ exists such that

$$\|z^1 - z^2\| \leq \eta(N_c), \quad \forall\, z^1,\, z^2 \in C_h,\, h \in \{1, \ldots, N_c\}, \tag{4.50}$$

and the maximum distance between two data samples can be defined as

$$\eta^{max} = \max_{i_1, i_2 \in \{1,\ldots,N\}} \{\|z^{i_1} - z^{i_2}\|\}. \tag{4.51}$$

Obviously, it holds that

$$\eta(N_c) \leq \eta^{max} \quad \forall\, N_h \geq 1. \tag{4.52}$$

Observe that condition (4.50) is trivially satisfied by any finite data set.

Assuming $N_c$ clusters $C_h$, $h = 1, \ldots, N_c$, exist with $|C_h| = N_h$ and $j_h = 1, \ldots, N_h$ for all $h$, the function can be reindexed to get

$$f(w) = \sum_{i=1}^{N} \phi(w, z^i) = \sum_{h=1}^{N_c} \frac{N_h}{N} \frac{1}{N_h} \sum_{j_h=1}^{N_h} \phi(w, z^{h j_h}).$$

Similarly, the gradient at a given $w$ is

$$\nabla f(w) = \sum_{h=1}^{N_c} \frac{N_h}{N} \frac{1}{N_h} \sum_{j_h=1}^{N_h} \nabla\phi(w, z^{h j_h}).$$

The straightforward strategy to leverage clustering information may be to set the dimension of the minibatches to $|S_k| = N_c$ for all $k = 0, 1, \ldots$, and employ a procedure building each minibatch such that it contains exactly one point from each cluster:

$$w_{k+1} = w_k - \alpha_k \nabla f_{S_k}(w_k), \tag{4.53}$$

where

$$\nabla f_{S_k}(w) = \sum_{h=1}^{N_c} \frac{N_h}{N} \nabla \phi(w, z^{h j_h^k}), \quad j_h^k \in \{1, \ldots, N_h\}. \tag{4.54}$$

Above, $j_h^k$ represents the index of the sample chosen at iteration $k$ from cluster $C_h$. Therefore

$$\|\nabla f_{S_k}(w) - \nabla f(w)\| = \left\| \sum_{h=1}^{N_c} \frac{N_h}{N} \left[ \nabla \phi(w, z^{h j_h^k}) - \frac{1}{N_h} \sum_{j_h=1}^{N_h} \nabla \phi(w, z^{h j_h}) \right] \right\|,$$

from which it can be easily proved that an $M(L_z, \eta(N_c)) = M(L_z, N_c) > 0$ exists s.t.

$$\|\nabla f_{S_k}(w) - \nabla f(w)\| \leq M(L_z, N_c), \tag{4.55}$$

as shown in theorem 29.

**Theorem 29.** *Given $L_z, \eta(N_c) > 0$ defined in* (4.49) *and* (4.50)*, an $M(L_z, \eta(N_c)) = M(L_z, N_c) > 0$ exists such that*

$$\|\nabla f_{S_k}(w) - \nabla f(w)\| \leq M(L_z, N_c).$$

*Proof.*

$$\begin{aligned}
\|\nabla f_{S_k}(w) - \nabla f(w)\| &= \left\| \sum_{h=1}^{N_c} \frac{N_h}{N} \left[ \nabla \phi(w, z^{h j_h^k}) - \frac{1}{N_h} \sum_{j_h=1}^{N_h} \nabla \phi(w, z^{h j_h}) \right] \right\| \\
&\leq \sum_{h=1}^{N_c} \frac{N_h}{N} \left\| \nabla \phi(w, z^{h j_h^k}) - \frac{1}{N_h} \sum_{j_h=1}^{N_h} \nabla \phi(w, z^{h j_h}) \right\| \\
&= \sum_{h=1}^{N_c} \frac{N_h}{N} \left\| \frac{1}{N_h} \sum_{j_h=1}^{N_h} \left[ \nabla \phi(w, z^{h j_h^k}) - \nabla \phi(w, z^{h j_h}) \right] \right\| \\
&\leq \sum_{h=1}^{N_c} \frac{N_h}{N} \frac{1}{N_h} \sum_{j_h=1}^{N_h} \left\| \nabla \phi(w, z^{h j_h^k}) - \nabla \phi(w, z^{h j_h}) \right\|,
\end{aligned}$$

where $j_h^k$ is the index selected at iteration $k$ from the indices $j_h \in \{1, \ldots, N_h\}$. Therefore, recalling (4.49) and (4.50), we can write

$$\begin{aligned}
\|\nabla f_{S_k}(w) - \nabla f(w)\| &\leq \sum_{h=1}^{N_c} \frac{N_h}{N} \frac{1}{N_h} \sum_{j_h=1}^{N_h} \left\| \nabla \phi(w, z^{h j_h^k}) - \nabla \phi(w, z^{h j_h}) \right\| \\
&\leq \sum_{h=1}^{N_c} \frac{N_h}{N} \frac{1}{N_h} \sum_{j_h=1}^{N_h} L_z \left\| z^{h j_h^k} - z^{h j_h} \right\| \\
&\leq \sum_{h=1}^{N_c} \frac{N_h}{N} \frac{1}{N_h} \sum_{j_h=1}^{N_h} L_z \eta(N_c) \\
&= M(L_z, N_c),
\end{aligned}$$

which completes the proof. □

Clearly, $M(L_z, N_c)$ decreases with the increase of the number of clusters. Furthermore, it can be proved that

$$\nabla f(w)^T (-\nabla f_{S_k}(w)) \leq 0 \Leftrightarrow \|\nabla f(w)\| \geq M(L_z, N_c),$$

i.e. the direction is a descent direction, as shown in Proposition 30.

**Theorem 30.** *Given $M(L_z,N_c) > 0$ as defined in theorem 29, the following holds for all $w \in \mathbb{R}^n$:*

$$\nabla f(w)^T(-\nabla f_{S_k}(w)) \leq 0 \Leftrightarrow \|\nabla f(w)\| \geq M(L_z,N_c).$$

*Proof.* It holds that

$$\begin{aligned}\nabla f(w)^T(-\nabla f_{S_k}(w)) &= \nabla f(w)^T\left[(\nabla f(w) - \nabla f_{S_k}(w)) - \nabla f(w)\right] = \\ &= \nabla f(w)^T\left[(\nabla f(w) - \nabla f_{S_k}(w))\right] - \|\nabla f(w)\|^2 \leq \\ &\leq \|\nabla f(w)\|\|\nabla f(w) - \nabla f_{S_k}(w)\| - \|\nabla f(w)\|^2 \leq \\ &\leq \|\nabla f(w)\|(M(L_z,N_c) - \|\nabla f(w)\|).\end{aligned}$$

Imposing the condition

$$\nabla f(w)^T(-\nabla f_{S_k}(w)) \leq 0,$$

yields

$$\|\nabla f(w)\| \geq M(L_z,N_c).$$

□

This means such 'approximate' direction is a descent direction up to a certain distance from a stationary point.

The results presented so far express "local" (in time) bounds on the approximation error of the gradient estimate with respect to the true gradient. It can also be proved that similar bounds can be computed for the variance of these approximations in the evolution of the iterates. This has important impacts on the final quality of the solutions found, as seen in assumptions 9 and 6. In fact, if the approximation variance can not be led to zero, then a diminishing stepsize, namely not summable but squared-summable, is needed to prove convergence.

The next theorem goes in this direction, by proving a bound on the gradient approximation variance which depends on the clustering quality, i.e. the maximum intra-cluster distance (4.50).

**Theorem 31.** *Given $M(L_z,N_c) > 0$ as defined in theorem 29, assume the number of samples $N$ is finite. Then it holds*

$$Var[\|\nabla f_{S_k}(w_k)\|] \leq M^2(L_z,N_c).$$

*Proof.*

$$\begin{aligned}Var[\|\nabla f_{S_k}(w_k)\|] &= E\left[\|\nabla f_{S_k}(w_k) - \nabla f(w_k)\|^2\right] \\ &= \frac{1}{K}\sum_{k=1}^{K}\|\nabla f_{S_k}(w_k) - \nabla f(w_k)\|^2,\end{aligned}$$

where $K$ is the number of different minibatches of dimension $|S_k|$ that can be sampled from $\{1,\dots,N\}$. This number is finite and it follows from theorem 29 that

$$\begin{aligned}Var[\nabla f_{S_k}(w_k)] &= \frac{1}{K}\sum_{k=1}^{K}\|\nabla f_{S_k}(w_k) - \nabla f(w_k)\|^2 \\ &\leq \frac{1}{K}\sum_{k=1}^{K}M^2(L_z,N_c) \\ &\leq M^2(L_z,N_c).\end{aligned}$$

□

| Dataset | Task | N | d |
|---|---|---|---|
| Covertype | Binary classification | 581,012 | 54 |
| ijcnn1 | Binary classification | 35,000 | 22 |
| SUSY | Binary classification | 5,000,000 | 18 |
| SkinNonskin | Binary classification | 245,057 | 3 |
| YearPred | Regression | 463,715 | 90 |
| CaliforniaHousing | Regression | 20,640 | 8 |

**Table 4.1.** Datasets. Source: https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

The analysis above obviously applies also to dynamic batching strategies, like the one proposed in Section 4.3. Indeed, once the decision to increase the dimension of the minibatch is taken, one needs to determine which samples will be included in the next minibatch. This is usually done in a random/ cyclic way, but the results in this section suggest that leveraging the information on clusters in data samples can bring improvements in dynamic batching schemes, namely it allows to increase the dimension of the minibatch less frequently and less heavily.

## 4.6 Numerical experiments

In this section, some numerical results on real ML data sets are reported. In particular, in section 4.6.1 the experimental setup is described, in section 4.6.2 a summary of the results is presented and in section 4.7 some conclusions and potential next steps are explained. Finally, in section 4.8 the complete numerical results are reported.

### 4.6.1 Experimental setup

The experimental setup was carried on the datasets listed in table 4.1, a mix of binary classification and regression problems. For linear binary classification (LBC), when the labels are in $\{-1,1\}$, the training problem formulation can be written as

$$\min_{w\in\mathbb{R}^d} L(w) = \frac{1}{N}\sum_{i=1}^{N} \log(1+e^{-y^i w^T x^i}).$$

For regression, both Linear (LLS) and Nonlinear Least Squares (NLS) were tested, with the training problem formulation being

$$\min_{w\in\mathbb{R}^d} L(w) = \frac{1}{N}\sum_{i=1}^{N} (\phi(w^T x^i) - y_i)^2,$$

where the function $\phi(\cdot)$ determines whether it is LLS or NLS.

All the experiments were run for a maximum of 100 epochs (i.e. $100N$ single gradient evaluations were computed), unless a given accuracy was reached, namely

$$f(w_k) - f(w^\star) \leq 10^{-6}$$

for convex problems (i.e. LBC and LLS), or

$$\|\nabla f(w_k)\|_\infty \leq 10^{-6}$$

| Parameter | Grid |
|---|---|
| $\alpha$ | $\{2^{-10},\dots,2^{10}\}$ |
| $m$ | $\{10,100,1000\}$ |
| $\eta$ | $\{10^{-4},\dots,10^{-1}\}$ |

**Table 4.2.** Parameters tuning grid

for nonconvex ones (i.e. NLS).

All the tests were run with a constant stepsize, which was tuned for both the algorithms together with the other parameters, as detailed in table 4.2. For the norm test, the parameter $\theta$ was set to 0.9, as proposed in [22].

### 4.6.2 Results and discussion

In this section, a fixed-stepsize SGD method equipped with the Fisher test is compared to a fixed-stepsize SGD equipped with the Norm test [22]. As explained in the previous section, both the algorithms' parameters were tuned to their (locally) optimal values by means of a grid search. Figures 4.1-4.4 show the results on two convex problems defined by two of the datasets introduced in table 4.1, namely *covertype* (logistic regression) and *californiaHousing* (linear least squares).

Figures 4.1 and 4.3 plot the distance of the objective function to the optimum vs the number of single gradient evaluations, where a single gradient evaluation represents the gradient evaluated over one sample, while figures 4.3 and 4.4 plot the relative minibatch size $|Sk|\%$ vs the iterations, where the relative minibatch size is defined as

$$Sk\% = \frac{|Sk|}{N}.$$

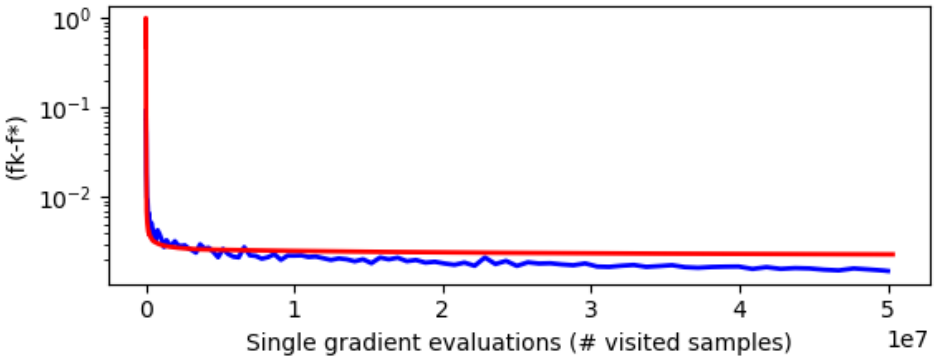


**Figure 4.1.** Covertype dataset. Function value distance from the optimum with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

As the figures show, controlling the minibatch size growth, and letting it grow slowly, can be efficient in terms of total single gradient evaluations. In fact, a trade-off between computing cheap but poor-quality gradient estimates and expensive high-quality ones must be found. In many machine learning applications, letting the minibatch size grow slowly and exploiting the properties of SGD is

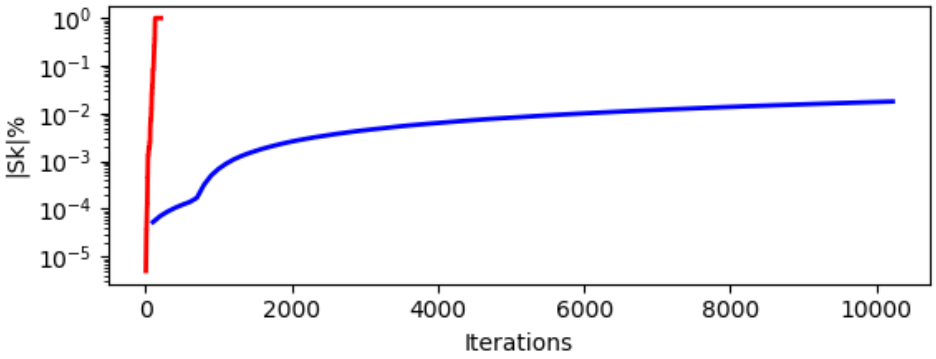


**Figure 4.2.** Covertype dataset. Minibatch size $|S_k|$ as percentage of the population size $N$, with respect to iterations. In blue the Fisher test, in red the Norm test.

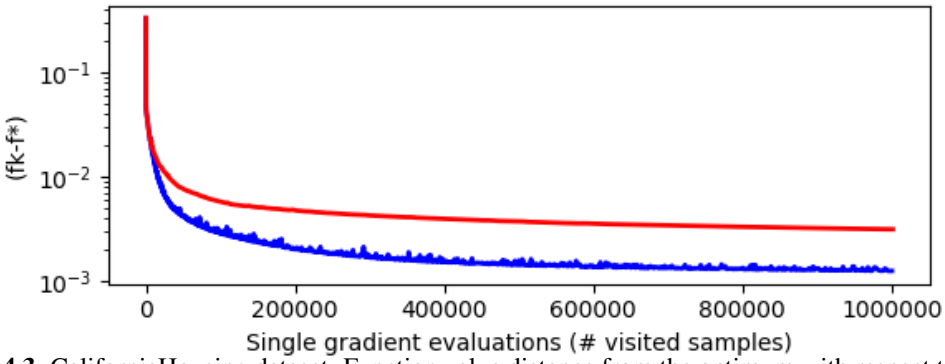


**Figure 4.3.** CaliforniaHousing dataset. Function value distance from the optimum with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

often beneficial. Nevertheless, it is not clear if this is always beneficial, or if it depends on the dataset at hand. In fact, recalling the norm test definition

$$\frac{\|Var_{i\in S_k}(\nabla f_i(w_k))\|_1}{|S_k|} \leq \delta^2 \|\nabla f_{S_k}(w_k)\|^2,$$

the rhs of the test is proportional to the square of the gradient norm. In many machine learning applications, where the optimization landscape is characterized by *flat regions* (i.e. regions where the objective function is almost flat and therefore its gradient close to zero), this may cause the norm test to fail too often, too early.

## 4.7 Conclusions and future developments

In conclusion, coupling a norm test with a statistically sound test, i.e. the Fisher test 4.17, can bring stability to the growth of the minibatch size in a SG/ IG setting. Indeed, slowing down the minibatch

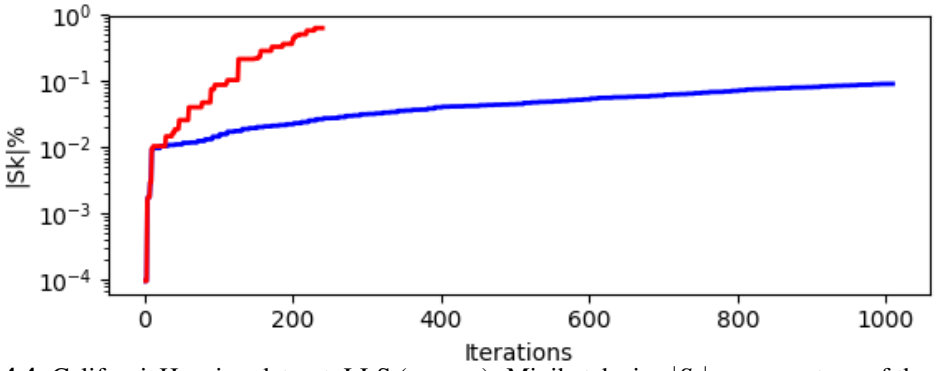


**Figure 4.4.** CaliforniaHousing dataset, LLS (convex). Minibatch size $|S_k|$ as percentage of the population size $N$, with respect to iterations. In blue the Fisher test, in red the Norm test.

size growth showed to be beneficial in most of the datasets, in terms of total computational complexity, namely, the total number of visited data samples to reach a given accuracy are lower, or the accuracy reached given a maximum number of visited data samples is higher.

Nevertheless, it is not clear whether this behavior is true for any dataset, or if there exist datasets where a faster growth is necessary and datasets where a slower one is advised. Further tests on other applications where the optimization problem has a finite sum structure may shed light on this open question.

Potential next steps of this line of research may be:

- the extension and theoretical analysis of the Fisher test coupled with the inner product test, like already sketched in (4.14),(4.15),(4.16);
- a formal characterization of the structure of the problems where a slower growth of the minibatch size is desirable and beneficial, as opposed to problems with a structure that needs a steady growth of the minibatch from the beginning of the iterations;
- the definition of a dynamic batching algorithm that leverages the structure in the data set, i.e. by employing clustering information over the data samples in order to reduce the gradient approximation error/ variance.

## 4.8 Appendix - Complete numerical results

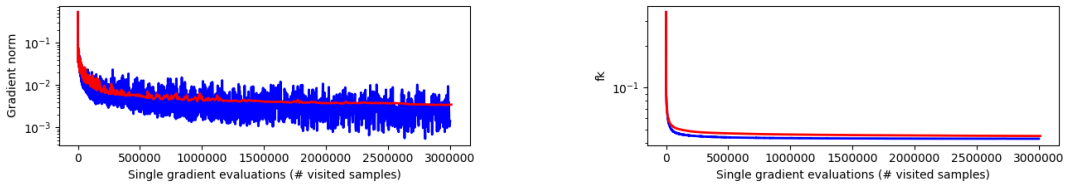


**Figure 4.5.** CaliforniaHousing dataset, NLS (nonconvex). Function value (right) and gradient norm (left) with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

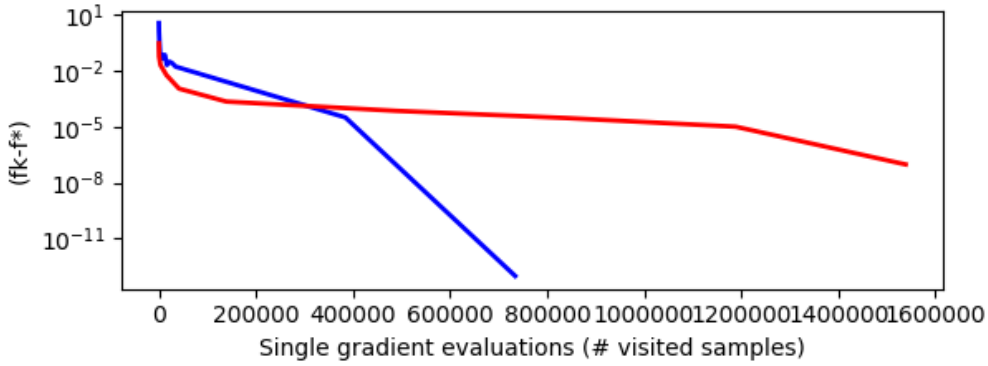


**Figure 4.6.** Ijcnn1 dataset. Function value distance from the optimum with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

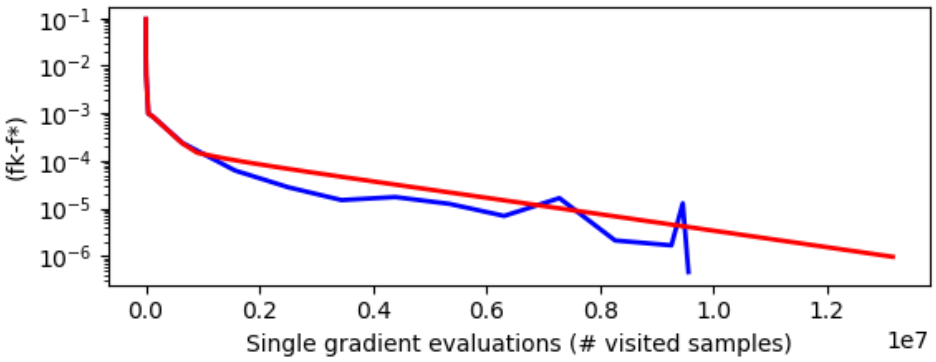


**Figure 4.7.** SkinNonSkin dataset. Function value distance from the optimum with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

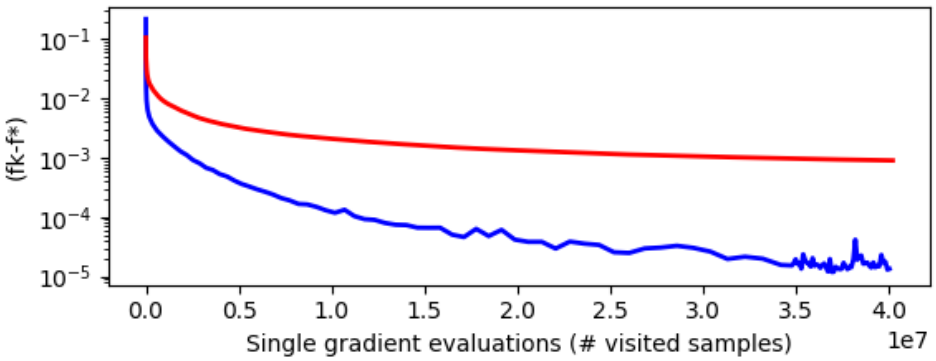


**Figure 4.8.** Yearpred dataset, LLS (convex). Function value distance from the optimum with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

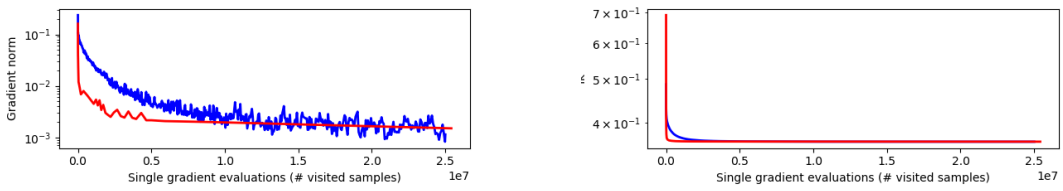


**Figure 4.9.** YearPred dataset, NLS (nonconvex). Function value (right) and gradient norm (left) with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

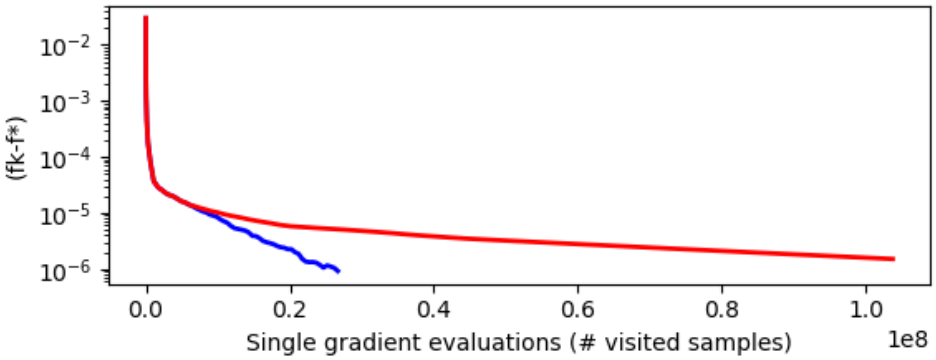


**Figure 4.10.** SUSY dataset. Function value distance from the optimum with respect to single gradient evaluations. In blue the Fisher test, in red the Norm test.

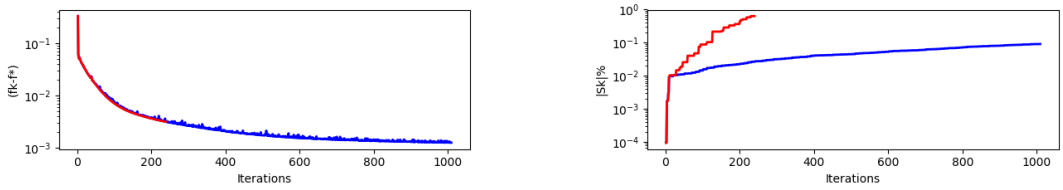


**Figure 4.11.** CaliforniaHousing dataset, LLS (convex). Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and function value (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

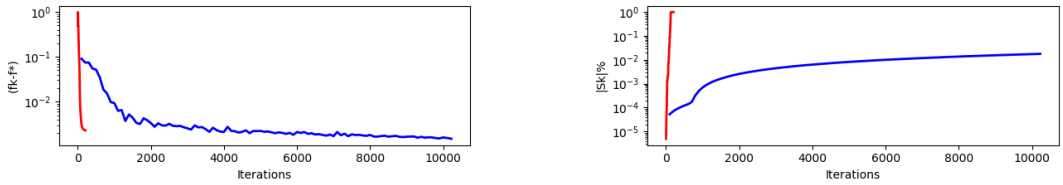


**Figure 4.12.** Covertype dataset. Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and function value (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

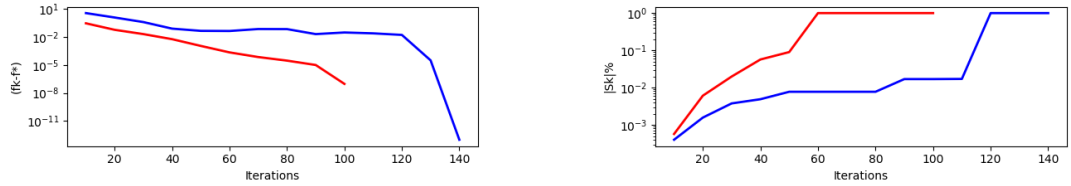


**Figure 4.13.** Ijcnn1 dataset. Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and function value (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

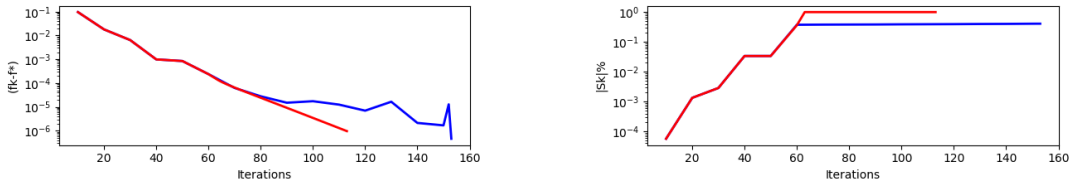


**Figure 4.14.** SkinNonSkin dataset. Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and function value (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

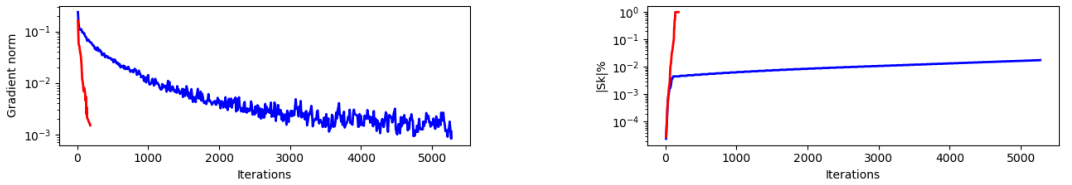


**Figure 4.15.** YearPred dataset, NLS (nonconvex). Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and gradient norm (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

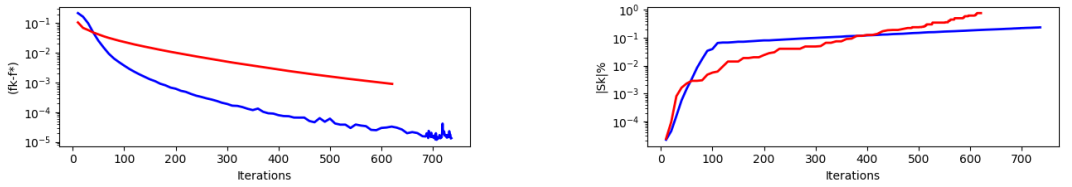


**Figure 4.16.** YearPred dataset, LLS (convex). Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and function value (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

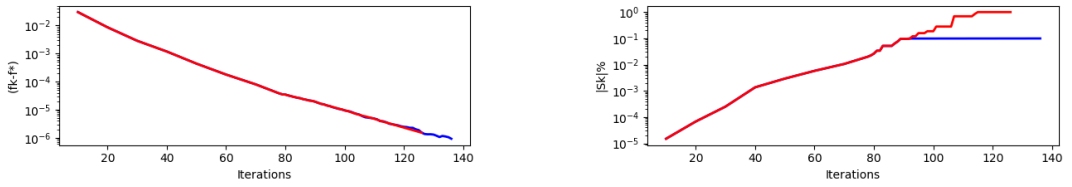


**Figure 4.17.** SUSY dataset. Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and function value (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

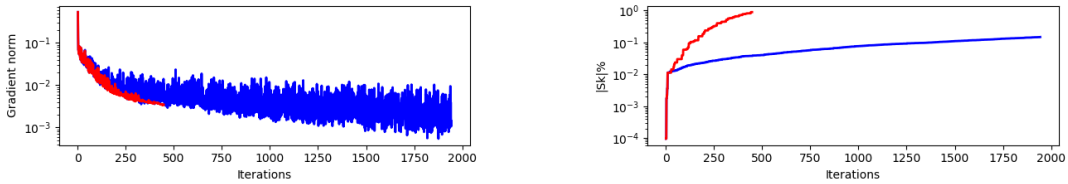


**Figure 4.18.** CaliforniaHousing dataset, NLS (nonconvex). Minibatch size $|S_k|$ as percentage of the population size $N$ (right) and gradient norm (left) with respect to iterations. In blue the Fisher test, in red the Norm test.

# Chapter 5

# DFO approaches for policy optimization in RL

This chapter is the output of the 6-month visiting period spent at Northwestern University (Chicago, US), in collaboration with prof. Jorge Nocedal.

## 5.1 Introduction and motivation

### 5.1.1 Brief outline of reinforcement learning

Figure 5.1 shows a block diagram representing the reinforcement learning framework. In this case, the probability distribution mapping a state to the action, i.e. $\pi$, is encoded by a deep neural network (aka policy network).
In the following, an *episode* will represent the simulation of the environment, starting from state $s_0$, from time step $t = 0$ to time step $t = T$. An episode generates a *trajectory* $\{s_0, a_0, s_1, a_1, \ldots, s_t, a_t, \ldots, s_{T-1}, a_{T-1}, s_T\}$, where each action $a_t$ is generated by $\pi$, given the current state $s_t$.

For continuous control tasks like the ones object of this chapter, the above framework can be represented by a MDP $(\mathscr{S}, \mathscr{A}, r, \mathscr{P})$, where:

- the state space is real-valued, i.e. $\mathscr{S} \subseteq \mathbb{R}^s$;
- the action space is real-valued, i.e. $\mathscr{A} \subseteq \mathbb{R}^a$;

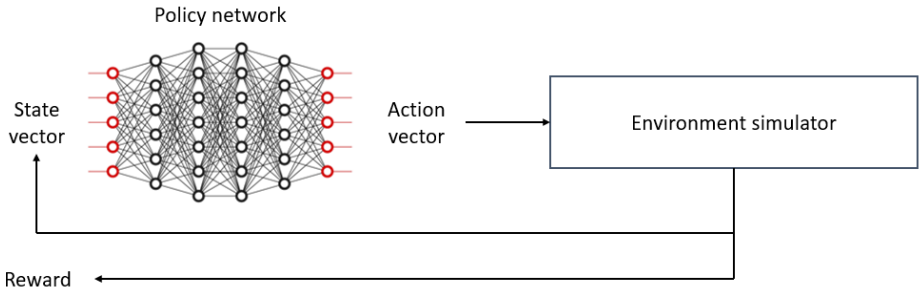


**Figure 5.1.** DeepRL block diagram

- $r : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ is the reward function, through which the total reward of an episode can be written as

$$R = \sum_{t=1}^{T} r_t,$$

  where $r_t \in \mathbb{R}$ are the rewards received from the environment at each time step $t$;

- the state-transition probabilities, modeling the probability of transitioning to a state $s'$ from the current state $s$ taking action $a$, i.e. $\mathscr{P} : \mathscr{S} \times \mathscr{A} \to \mathscr{S}$, satisfy the Markov property.

Observe that the state transition probabilities $\mathscr{P}$ can be seen, in the framework of figure 5.1, as the composition of (i) the probability of taking an action $a$, based on the current state $s$, modeled by $\pi$ and (ii) the probability of transitioning to state $s'$, given the current state $s$ and the action taken $a$.

If $\pi$ is encoded by a (deep) neural network, as is often the case in the RL community, then $\pi : \mathbb{R}^s \to \mathbb{R}^a$ determines a policy, i.e. the function that given a state $s$ returns the action $a$ to be taken by the agent. A common way of modeling it is to consider the output of the *policy network* as

$$\mu_A = \pi(w; s_t), \quad \text{for all } t = 1, \ldots, T,$$

where $\mu_A$ is the mean of the normally distributed random variable $A_t$ representing the stochastic actions, $\sigma_A$ is its standard deviation, and $w \in \mathbb{R}^d$ is the vector of parameters (i.e. the weights of the policy network) to be learned. Notice that the standard deviation is often taken as fixed, i.e. $\sigma_A = 1$, or diminishing during the optimization, i.e. $\sigma_A^k \sim \frac{1}{k}$, where $k$ is the index of the optimization iterations.

The aim of policy optimization for reinforcement learning is to find an optimal policy $\pi^\star$ and, since the policy is parameterized by $w$, the objective is to find $w^\star$ such that the expected total reward is maximized. In this setting, the expected reward function $\mathscr{R}$ can be written as

$$\mathscr{R}(w) = E_{\mathscr{P}}[R],$$

where $R = \sum_{t=1}^{T} r_t$. If, with some abuse of notation, all the randomicity is indicated by a unique random variable $\zeta$, the objective of policy optimization is therefore to solve the problem

$$\max_{w \in \mathbb{R}^d} \mathscr{R}(w), \tag{5.1}$$

where $\mathscr{R}(w) = E_{\mathscr{P}}[R]$.

Based on the well-known policy gradient theorem [105]

$$\nabla \mathscr{R}(w) = E_{\mathscr{P}}\left[R \nabla \log \pi(w)\right], \tag{5.2}$$

policy gradient (PG) algorithms employ gradient-based iterations, where the gradient is estimated through (5.2). Observe that (5.2) gives an estimate of the gradient of the cumulative reward with respect to the policy network parameters $w$, which can be computed even if the 'simulation block' in figure 5.1 is nonsmooth/ noncontinuous, which is often the case in policy optimization for reinforcement learning. Therefore, PG algorithms are based on sample estimates of the gradient defined by (5.2) and then perform gradient ascent in the gradient estimate direction. Notice that PG algorithms are usually online algorithms, meaning that the policy network parameters $w$ are updated at each time

step $t$ of an episode. One of the most basic, yet much used, example is the REINFORCE algorithm 14.

**Algorithm 14:** REINFORCE algorithm

**for** $k = 0, 1, \dots$ **do**
  Starting from $w_k$, set $w_k^1 = w_k$;
  Generate a trajectory based on $\pi_{w_k}$;
  **for** $t = 1, 2, \dots, T$ **do**
    Estimate the gradient of the expected reward through (5.2);
    Update $w_k^{t+1}$ in the gradient direction;
  **end**
  Set $w_{k+1} = w_k^T$;
**end**

PG methods will not be analyzed further, since those are not the target of this chapter. See, e.g., [107, 92, 96] for the details of some of the most common policy gradient algorithms. In [40], many state-of-the-art algorithms are tested on several continuous control tasks.

Given the function $\mathscr{R}$ defined above, one could easily apply a derivative free optimization (DFO) algorithm to maximize it. Indeed, $\mathscr{R}$ inherently depends on a black box computation, i.e. the simulation of the episode(s), and therefore its derivatives are not directly computable. This setting will be further investigated in the following section.

### 5.1.2 DFO algorithms for policy optimization

Recalling figure 5.1, note that the optimization process can be read in different ways, depending on the online vs batch policy optimization setting. Indeed, in the online setting the process can be described by the repetition of the following steps:

(i) the state $s_t$ is observed;

(ii) the action $a_t$ is generated through the policy, parameterized by the network weights $w_k^t$;

(iii) the reward $r_t$ and new state $s_{t+1}$ are observed;

(iv) the new network weights $w_k^{t+1}$ are computed in order to maximize the expected reward;

(v) repeat.

Above, the index $k$ represented an episode, while the index $t$ represented a time step during the episode. In the batch setting, instead,

(i) an entire episode is run with network weights $w_k$ the state vector $s_t$ is observed, the rewards sequence $\{r_t\}_{t=1}^T$ is output and the cumulative reward $R$ is computed;

(ii) the new network weights $w_{k+1}$ are computed in order to maximize the expected reward;

(iii) repeat.

In the analysis of the following sections, it will be assumed that the policy network weights are updated only at the end of an entire episode. Observe that the exact same distinction can be made in the case a minibatch of episodes is run at each iteration $k$.

The problem can therefore be considered in the simulation-optimization framework, where at each iteration of the solution algorithm some simulations are run, the objective functions are observed (and averaged), a suitable search direction is chosen and the parameters are updated. The objective of the optimization is to maximize the expected cumulative reward $\mathscr{R}$. To put the problem in the standard optimization framework, problem 5.3 is the minimization of the expected value of a function $f$ of a generic $d$-dimensional variable $w$.

$$\min_{w \in \mathbb{R}^d} F(w), \tag{5.3}$$

where $F(w) = E_\zeta[f^0(w,\zeta)] := -E_{\mathscr{P}}[R]$ and $\zeta$ is a random variable.

From now on, the analysis will focus on the function

$$f(w,\zeta),$$

where the random variable $\zeta$ represents, with some abuse of notation, the stochasticity of the environment and of the potential subsampling, i.e. the estimation of the function based on a finite number of episodes. Therefore, $f$ is a sample estimate of the function $f^0$.

In example 1 below, the calculation of the reward function from an environment of the mujoco [109] library from OpenAI Gym [21] is reported. The aim of the example is to show how the reward, i.e. the objective function of the DFO algorithm, is computed, and how there are no guarantees on its smoothness properties.

**Example 1** (HalfCheetah environment example)**.** *The goal of the HalfCheetah environment is to teach a halfcheetah to run in the 2-dimensional space. Therefore, given a state (including, e.g., position, velocity, ...), the aim is to design a policy letting the halfcheetah run.*

*The reward at any time step t of an episode is*

$$r(t) = c_1 \frac{x_{t+1} - x_t}{dt} - c_2 \|a_t\|^2,$$

*where $c_1, c_2$ are two positive constants, $dt > 0$ is the duration of a time step in the environment simulator (i.e. time is discretized), $x_t$ is the current position (one component of the state vector $s_t$) and $a_t$ is the action vector.*

*Since the policy is parameterized by a neural network with weight vector $w \in \mathbb{R}^d$, the function $\pi$ s.t.*

$$a_t = \pi(w; s_t)$$

*is continuously differentiable w.r.t. $w$ if the activation functions of the neural network are continuously differentiable.*

*Unfortunately, the continuity of the policy function $\pi$ does not guarantee continuity of the reward function. Indeed, the function $\phi$ s.t.*

$$x_{t+1} = \phi(a_t, s_t; \zeta) = \phi(\pi(w; s_t), s_t; \zeta)$$

*is not fully specified and, in general, depends on a random variable $\zeta$, which depends on how the environment simulator is built. Therefore, the function $r(t)$, which depends on $x_{t+1}$, can not be ensured to be smooth.*

*Furthermore, even if the function $\phi$ was assumed to be continuously differentiable, which would imply $r(t)$ to be continuously differentiable for each $t$, this would not imply smoothness of the total reward $R$! Indeed, given the definition of the total reward*

$$R = \sum_{t=0}^{T} r(t)$$

*this is not clear, since the time is discretized and the total reward function is the sum of $T$ rewards, with $T$ varying based on the simulator. In fact, if close to instability, a small variation in the parameters $w$ may cause a large variation in the total reward - for example in the case the simulation would end earlier (therefore having a smaller $T$) because of the instability of the controlled system.*

In DFO algorithms, the gradient is estimated based on function evaluations only. Being the function to minimize the negative total reward on the whole episode (or minibatch of episodes), such algorithms are inherently batch and can be sketched as in algorithm 15.

**Algorithm 15:** DFO RL algorithm

**for** $k = 0, 1, \ldots$ **do**
  Generate $m_k$ perturbations of $w_k$;
  Run $m_k$ episodes based on the perturbed policy and collect the returns $f_j(w_k, \zeta), j = 1, \ldots, m_k$;
  Estimate the gradient direction $g(w_k, \zeta)$ based on $f_j(w_k, \zeta), j = 1, \ldots, m_k$;
  Update $w_{k+1} = w_k - \alpha_k g(w_k, \zeta)$;
**end**

Derivative-free optimization algorithms are sometimes referred to, in the RL community, as *evolution strategies*. In [87, 27], such algorithms have been applied to several continuous control problems, showing promising results. In [9], instead, several DFO algorithms were compared on a small set of continuous control problems. Among others, the authors applied finite differences (FD), Gaussian Smoothing (GS) and Linear Interpolation (LI) to estimate gradients based on function evaluations only, then employed a linesearch "with error" to compute the stepsize $\alpha_k$. It must be noted that in [9] the number of function evaluations to estimate the gradients was determined such that a *norm condition* was satisfied with high-enough probability, therefore in contrast with the common belief (in the RL community), that approximation methods like GS perform well even with a small number of function evaluations per iteration.

One of the most interesting open questions in optimization applied to reinforcement learning is why, given a highly nonlinear, nonconvex, nonsmooth problem can be solved at good-enough accuracy by gradient-based methods, where, furthermore, the gradients are approximated via a small number (e.g., only one) of function evaluations. The seminal work [42] tries to shed light on this open question, analyzing a simple, theoretical continuous control problem, the linear quadratic regulator (LQR), and proving that such problem, although nonconvex, can be solved to optimality by gradient methods (even if the gradient is approximated).

In [70], instead, a numerical investigation was carried out on several reinforcement learning tasks. The goal was to show that, even with simple models, i.e. a linear mapping instead of a neural network to parameterize the policy $\pi$, good results could be achieved on many state-of-the-art tasks. Interestingly, the results were surprisingly good. Nevertheless, the method employed, a DFO based on Gaussian Smoothing gradient approximation, was enhanced by several heuristics. Therefore, it is not clear if the good results were due to the simplicity of the linear mapping to encode the policy, or to the heuristics employed to improve the quality of the search directions.

Now, gradient approximation techniques will be further detailed, in the setting where only function values are available. In particular, the analysis will refer to a generic function $\phi : \mathbb{R}^d \to \mathbb{R}$. Observe that the main difficulty in the RL framework is that not only the function can not be computed in a deterministic manner, i.e. the function could be seen as affected by noise, but every function calculation can be computationally very expensive. In particular, employing gradient-based methods can be highly ineffective, since estimating a gradient based on function evaluations only can (i) be very

expensive and (ii) give poor approximation quality, due to noisy function evaluations. Nevertheless, much attention has been lately put in this field.

In [7], the gradient approximation method first proposed in [100, 101] and later named Gaussian Smoothing, was applied to the training of overparameterized Machine Learning problems, with the aim of exploiting some kind of sparsity in the function mapping. Gaussian Smoothing is based on the so-called Stein identity, i.e.

$$\nabla\phi^{\nu}(w) = E_u\left[\frac{\phi(w+\nu u)}{\nu}u\right], \tag{5.4}$$

where

$$\phi^{\nu}(w) = E_u[\phi(w+\nu u)] \tag{5.5}$$

is a *smoothed function*, $u \sim N(0, I_d)$ is a normally distributed random vector and $\nu > 0$. By the fact that $E[u] = 0$ and simple calculations, the (Forward) Gaussian Smoothing (FGS) gradient estimate can be computed by

$$\nabla\phi^{FGS}(w_k) = \frac{1}{m_k}\sum_{j=1}^{m_k}\nabla\phi^{FGS}(w_k) = \frac{1}{m_k}\sum_{j=1}^{m_k}\frac{\phi(w_k+\nu u_k^j)-\phi(w_k)}{\nu}u_k^j, \tag{5.6}$$

where $u_k^j \sim N(0, I_d)$ for all $j,k$. The appeal of the Gaussian Smoothing method is that a (biased) gradient estimate can be computed by as few as just one function evaluation. Obviously, the quality of the estimate gets better as the number of function evaluations, i.e. $m_k$, employed in (5.6) increases. In contrast, the well-known (Forward) Finite Difference (FFD) gradient estimator needs $d+1$ function evaluations

$$\nabla\phi^{FFD}(w_k) = \left[\frac{\phi(w_k+\nu e_i)-\phi(w_k)}{\nu}\right]_{i=1}^{d}, \tag{5.7}$$

where $e_i \in \mathbb{R}^d$ represents the $i$-th coordinate direction. Although more expensive than Gaussian Smoothing, Forward Differences gives very accurate gradient estimates. Indeed, in [10] a bound on the number of function evaluations necessary to get the norm condition

$$\|g_k - \nabla\phi(w_k)\| \leq \theta\|\nabla\phi(w_k)\| \tag{5.8}$$

verified with more than 50% probability was given, showing that FGS needs $O(3d)$ function evaluations, compared to the $d+1$ of FFD. Central versions of the two methods exist, at the cost of doubling the number of function evaluations,

$$\nabla\phi^{CGS}(w_k) = \frac{1}{m_k}\sum_{j=1}^{m_k}\nabla\phi^{CGS}(w_k) = \frac{1}{m_k}\sum_{j=1}^{m_k}\frac{\phi(w_k+\nu u_k^j)-\phi(w_k-\nu u_k^j)}{2\nu}u_k^j, \tag{5.9}$$

$$\nabla\phi^{CFD}(w_k) = \left[\frac{\phi(w_k+\nu e_i)-\phi(w_k-\nu e_i)}{2\nu}\right]_{i=1}^{d}. \tag{5.10}$$

A method to compute computationally cheap search directions, which can be thought of as a bridge between GS and FD, is the one employed in Random Coordinate Descent (RCD) methods, where the gradient approximation is computed as

$$\nabla\phi^{RCD}(w_k) = \left[\frac{\phi(w_k+\nu e_i)-\phi(w_k)}{\nu}\right]_{i\in S_k}, \tag{5.11}$$

where $S_k$ is a subset of the indices $\{1,\dots,d\}$ of size $m_k$. What the RCD method does is to randomly select $m_k$ coordinate directions and then compute a finite difference along each of those. Therefore, given a budget $m_k$ of function evaluations, its computational cost is exactly the same of Gaussian Smoothing.

Another gradient approximation method is the Simultaneous Perturbation Stochastic Approximation (SPSA) method, aka Spall's method, [99], which is very well-known in the Stochastic Optimization community, since it allows to estimate gradients by only two function evaluations.

$$\nabla\phi^{SPSA}(w_k) = \frac{1}{m_k}\sum_{j=1}^{m_k}\nabla\phi^{SPSA}(w_k) = \frac{1}{m_k}\sum_{j=1}^{m_k}\left[\frac{\phi(w_k+\nu u_j)-\phi(w_k-\nu u_j)}{2\nu[u_j]_i}\right]_{i=1}^{d}, \tag{5.12}$$

where $u_j, j = 1,\dots,m_k$ are random variables which expected value must be bounded away from zero. Although the formula is very similar to GS, the reasoning behind it is quite different: Spall's method is based on the conjecture that the gradient estimated in a random direction can be representative of all the gradient components.

It is clear that Gaussian Smoothing and Spall's method are not the go-to approaches when in need of an accurate gradient estimation. Nevertheless, applying an orthogonalization technique [9] to the $m_k$ random directions generated at each iteration may be beneficial, since it would avoid to waste function evaluations in directions that may not bring any further information.

Another common gradient approximation method is based on interpolation. This is done by building a model, which is usually linear or quadratic, of the objective function around the current iterate [10]. A linear model centered in $x \in \mathbb{R}^d$ can be written as

$$M(y) = \phi(x) + g(x)^T(y-x), \tag{5.13}$$

where $y \in \mathbb{R}^d$ and $g : \mathbb{R}^d \to \mathbb{R}^d$ represents the gradient approximation function. Under the hypothesis of computing $m_k = d$ function evaluations, one can estimate $g$ by solving the interpolation linear system of equations

$$\sigma Q_U g = \Phi_U, \tag{5.14}$$

where

$$Q_U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_d \end{bmatrix}$$

and

$$\Phi_U = \begin{bmatrix} \phi(x)-\phi(x+\sigma u_1) \\ \phi(x)-\phi(x+\sigma u_2) \\ \vdots \\ \phi(x)-\phi(x+\sigma u_d) \end{bmatrix},$$

respectively being the matrix which rows are the $d$ directions $u_i$ and the vector with entries $\phi(x) - \phi(x+\sigma u_i)$.

In summary, the main objectives of this line of research are:

(i) to extend the standard results of stochastic optimization to the case where the gradients can not be computed explicitely, but need be approximated by means of function evaluations only, with particular interest in the Finite Differences vs Gaussian Smoothing cases;

(ii) to gain a thorough understanding of the optimization landscape addressed by derivative free optimization methods applied to policy optimization for reinforcement learning;

(iii) to compare the performance of standard derivative-free approaches applied to stochastic optimization with state-of-the-art algorithms designed by the RL community;

(iv) to numerically investigate the conjecture that in zeroth-order stochastic optimization one can design an efficient algorithm employing less than $d$ function evaluations at each iteration.

## 5.2 Preliminary analysis

The stochastic optimization problem object of the analysis is problem 5.3, which can be approximated by a finite sum-structured problem

$$\min_{w\in\mathbb{R}^d} F(w), \tag{5.15}$$

where

$$F(w) = E_\zeta[f^0(w,\zeta)] \sim \frac{1}{N}\sum_{i=1}^{N} f_i^0(w) := \frac{1}{N}\sum_{i=1}^{N} \phi(w,\zeta_i).$$

Here, $\zeta$ is a random variable representing stochasticity due to the reinforcement learning environment simulation and $\zeta_i$ represents the $i$-th realization of $\zeta$. Notice that problem 5.15 is a finite sum approximation of problem 5.3 in the case the expected value of the objective function is approximated, throughout the solution algorithm, by the average over $N$ replicates of the simulation.

To simplify the subsequent analysis, the function $f(w,\zeta)$ will identify a sample approximation of the true function $F$. Sometimes, the stochastic function $f(w,\zeta)$ will be referred to as a *noisy function evaluation*.

The generic iteration of a (approximate) gradient-based method is

$$w_{k+1} = w_k - \alpha_k g_k, \tag{5.16}$$

where $\alpha_k$ is a positive steplength that may vary or be fixed during the optimization process, and $g_k$ is an approximation to the gradient of $F$.

The analysis will consider a standard zeroth-order stochastic gradient descent algorithm 16, where $g_k$ is a gradient approximation computed, e.g., by means of Gaussian Smoothing or Finite Differences. More details on the two methods will be described in the following subsections.

**Algorithm 16:** Zeroth-order Stochastic Gradient Method

**Data:** $w_0 \in \mathbb{R}^d$, $\nu > 0$, positive sequences $\{\alpha_k\}_{k=1}^K$
**for** $k = 1,\dots,K$ **do**
  Compute $g_k$, based on noisy function evaluations only;
  Compute $w_{k+1} = w_k - \alpha_k g_k$;
**end**

The assumption below will be useful in the theoretical analysis.

**Assumption 11.** *There exists $\sigma > 0$ such that for any $w \in \mathbb{R}^d$ it holds*

$$E\left[\|\nabla f(w,\zeta) - \nabla F(w)\|^2\right] \le \sigma^2.$$

In subsections 5.2.1 and 5.2.2, the properties of Gaussian Smoothing and Finite Differences, respectively, will be shown. The main reason why a thorough analysis of these methods is needed is that, in the stochastic optimization case, they introduce a bias in the gradient approximation. Indeed, throughout the following theoretical analysis, the hypothesis that the direction employed by the DFO method is an unbiased estimator of the true gradient *will never hold*. To relate this fact to the analysis in the stochastic setting of chapter 3, employing a direction that is a biased estimate of the gradient direction implies that there exist a neighborhood of the solution (stationary point) of problem 1.3 where the expected value of the direction can be an ascent direction. Thus, convergence to a solution (stationary point) of the problem can not be ensured, even with a diminishing stepsize, as will be clearer in the next two sections.

### 5.2.1 Gaussian smoothing

If the direction $g_k$ is computed through a (forward) Gaussian Smoothing approximation, then

$$g^{GS}(w_k;\zeta,u) = \frac{1}{m_k}\sum_{j=1}^{m_k}\frac{f(w_k+\nu u^j;\zeta)-f(w_k;\zeta)}{\nu}u^j, \tag{5.17}$$

where $u_k^j \sim N(0,I_d)$ for all $j,k$ and $m_k$ is the budget of function evaluations at iteration $k$. Notice that the random variables $\zeta$ and $u$ are assumed to be independent. The subsequent analysis will assume, without loss of generality, $m_k = 1$, i.e.

$$g^{GS}(w_k;\zeta,u) = \frac{f(w_k+\nu u;\zeta)-f(w_k;\zeta)}{\nu}u. \tag{5.18}$$

Observe that, by estimating $\nabla F(w)$ by means of a Gaussian Smoothing method like in (5.17), there are two sources of error:

(i) the unavoidable error in trying to approximate an expected value, i.e. $\nabla F(w) = E[\nabla f^0(w,\zeta)]$, by its sample approximation $\nabla f(w,\zeta)$;

(ii) the error given by the GS approximation, which only uses function values computed in random directions to estimate the gradient.

In the analysis, the following quantities may appear:

- the smoothed function and the smoothed gradient, i.e. the expected value, with respect to the random direction $u$, of the gradient approximation (5.18)

$$f^\nu(w;\zeta) = E_u[f(w+\nu u;\zeta)], \qquad \nabla f^\nu(w;\zeta) = E_u[g^{GS}(w_k;\zeta,u)]; \tag{5.19}$$

- the expected value, with respect to the random variable $\zeta$, of the gradient approximation (5.18)

$$G(w;u) := \frac{F(w+\nu u)-F(w)}{\nu}u = E_\zeta[g^{GS}(w_k;\zeta,u)]; \tag{5.20}$$

- the expected value, with respect to both the random variable $\zeta$ and the random direction $u$, of the gradient approximation (5.18)

$$\nabla F^\nu(w) = E_{\zeta,u}[g^{GS}(w_k;\zeta,u)]. \tag{5.21}$$

Recalling the above definitions, the following theorem 32, reported from [76], shows some important properties for the smoothed function and its derivative, with respect to the true function.

**Theorem 32.** *Let $u \sim N(0, I_d)$. Then*

$$E_u[\|u\|^p] \leq (d+p)^{\frac{p}{2}} \qquad \forall\, p \geq 2.$$

*Furthermore, if $\nabla f(w;\zeta)$ is $L_w$-Lipschitz continuous for any realization of $\zeta$:*

*(i) $f^\nu$ is $L_\nu$-Lipschitz continuous with $L_\nu \leq L_w$;*

*(ii) for all $w \in \mathbb{R}^d$, and for any realization $\overline{\zeta}$ of $\zeta$,*

$$|f^\nu(w;\overline{\zeta}) - f(w;\overline{\zeta})| \leq \tau_\nu, \tag{5.22}$$

$$\|\nabla f^\nu(w;\overline{\zeta}) - \nabla f(w;\overline{\zeta})\| \leq \eta_\nu; \tag{5.23}$$

*(iii) for all $w \in \mathbb{R}^d$, and for any realization $\overline{\zeta}$ of $\zeta$,*

$$\frac{1}{\nu^2} E_u\left[(f(w+\nu u;\overline{\zeta}) - f(w;\overline{\zeta}))^2 \|u\|^2\right] \leq \frac{\nu^2}{2} L_w^2 (d+6)^3 + 2(d+4)\|\nabla f(w;\overline{\zeta})\|^2, \tag{5.24}$$

*where $\tau_\nu = \frac{\nu^2}{2} L_w d > 0$ and $\eta_\nu = \frac{\nu}{2} L_w (d+3)^{\frac{3}{2}} > 0$.*

Theorem 32 bounds the bias of the smoothed function and gradient with respect to the true ones, in particular

$$\tau_\nu = \frac{\nu^2}{2} L_w d \quad \rightarrow \quad \text{Bound on } |f^\nu(w;\overline{\zeta}) - f(w;\overline{\zeta})|, \tag{5.25}$$

$$\eta_\nu = \frac{\nu}{2} L_w (d+3)^{\frac{3}{2}} \quad \rightarrow \quad \text{Bound on } \|\nabla f^\nu(w;\overline{\zeta}) - \nabla f(w;\overline{\zeta})\|, \tag{5.26}$$

Based on the above results, the following lemma from [7] bounds the variance of the Gaussian Smoothing gradient estimate with respect to the true gradient.

**Lemma 7.** *Let $g^{GS}(w_k;\zeta,u)$ be computed as in* (5.17) *and let assumption 11, 1, 2 hold. Then*

$$E\left[\|g^{GS}(w_k;\zeta,u) - \nabla F^\nu(w)\|^2\right] \leq \frac{2(d+5)(B^2+\sigma^2)}{m_k} + \frac{\nu^2}{2m_k} L_w^2 (d+3)^3, \tag{5.27}$$

*and*

$$E\left[\|g^{GS}(w_k;\zeta,u) - \nabla F(w)\|^2\right] \leq C_k^\nu, \tag{5.28}$$

*where*

$$C_k^\nu = \frac{4(d+5)(B^2+\sigma^2)}{m_k} + \frac{3\nu^2}{2} L_w^2 (d+3)^3. \tag{5.29}$$

Such variance is bounded by the sum of two terms, one depending on the bound on the gradient approximation variance in assumption 11, the other depending on the differencing interval $\nu$. Notice that computing the function $f$ by a sample estimate with a sample of size $|S| > 1$ would modify the constant in (5.29) in the following, simple way:

$$C_k^\nu = \frac{4(d+5)(B^2+\frac{\sigma^2}{|S|})}{m_k} + \frac{3\nu^2}{2} L^2 (d+3)^3. \tag{5.30}$$

### 5.2.2 Finite differences

Another, simple method to compute $g_k$ in algorithm 15 is (forward) Finite Differences, where the gradient estimator is

$$g^{FD}(w_k;\zeta) = \left[\frac{f(w_k+\nu e_i;\zeta)-f(w_k,\zeta)}{\nu}\right]_{i=1}^{d}, \tag{5.31}$$

where $e_i \in \mathbb{R}^d$ represents the $i$-th coordinate direction.

Lemma 8 shows that, under assumption 11 and 1, the gradient approximation variance of (5.31) is bounded above by a constant term.

**Lemma 8.** *Let $g^{FD}(w;\zeta)$ be computed as in* (5.31) *and let assumptions 1 and 11 hold. It follows that, for any realization $\overline{\zeta}$ of $\zeta$,*

$$E[\|g^{FD}(w;\overline{\zeta})-\nabla F(w)\|^2] \leq 2d\left(\frac{L_w\nu}{2}+\frac{2\varepsilon_m}{\nu}\right)^2+2\sigma^2. \tag{5.32}$$

*Proof.*

$$\begin{aligned}
E[\|g^{FD}(w;\overline{\zeta})-\nabla F(w)\|^2] &= E[\|g^{FD}(w;\overline{\zeta})-\nabla f(w;\overline{\zeta})+\nabla f(w;\overline{\zeta})-\nabla F(w)\|^2] \\
&\leq 2\|g^{FD}(w;\overline{\zeta})-\nabla f(w;\overline{\zeta})\|^2+2E[\|\nabla f(w;\overline{\zeta})-\nabla F(w)\|^2] \\
&\leq 2\sigma^2+2\|g^{FD}(w;\overline{\zeta})-\nabla f(w;\overline{\zeta})\|^2.
\end{aligned}$$

The quantity $\|g^{FD}(w;\overline{\zeta})-\nabla f(w;\overline{\zeta})\|^2$ can be bounded above, i.e.

$$\begin{aligned}
\|g^{FD}(w;\overline{\zeta})-\nabla f(w;\overline{\zeta})\|^2 &= \sum_{i=1}^{d}\left(\frac{f(w+\nu e_i;\overline{\zeta})-f(w,\overline{\zeta})}{\nu}-\left[\nabla f(w;\overline{\zeta})\right]_i\right)^2 \\
&\leq \sum_{i=1}^{d}\left(\frac{L_w\nu}{2}+\frac{2\varepsilon_m}{\nu}\right)^2 \\
&= d\left(\frac{L\nu}{2}+\frac{2\varepsilon_m}{\nu}\right)^2,
\end{aligned}$$

where $\varepsilon_m$ represents the machine precision. The above two relations combined give (5.32). □

Observe that the optimal choice for $\nu > 0$ is

$$\nu = \sqrt{\frac{2\varepsilon_m}{L_w}}, \tag{5.33}$$

since the following holds

$$\begin{aligned}
E[\|g^{FD}(w;\overline{\zeta})-\nabla F(w)\|] &\leq E[\|\nabla g^{FD}(w;\overline{\zeta})-\nabla f(w;\overline{\zeta})\|]+\sigma \\
&\leq \sqrt{d}\left(\frac{L_w\nu}{2}+\frac{2\varepsilon_m}{\nu}\right)+\sigma,
\end{aligned}$$

and, by minimizing with respect to $\nu$, one gets (5.33).

## 5.3 Convergence analysis of a stochastic DFO method

Now that a bound on the gradient approximation variance has been computed for both Gaussian Smoothing and Finite Differences, the main convergence analysis will rely on the assumption that a generic constant bound $C > 0$ on the gradient approximation variance exists, namely $\exists\, C > 0$ such that

$$E[\|g_k - \nabla F(w_k)\|^2] \leq C. \tag{5.34}$$

Observe that such constant $C$ is defined by (5.32) in the finite differences case, by (5.29) in the Gaussian smoothing approach. Furthermore, recalling the analysis in chapter 3, remark that the above gradient approximation variance is not controlled by the stepsize $\alpha$. This will impact the convergence properties, since letting the stepsize go to zero will not help in reducing the gradient approximation error.

Initially, it will be assumed that the function $F$ is estimated via sample approximation with a fixed sample size $|S|$ throughout the iterations. Then, the dynamic batching case will be analysed.

### 5.3.1 Fixed sample size

Theorem 33 makes it explicit that function descent can not be ensured at each iteration and depends on the gradient approximation variance.

**Theorem 33.** *Let $\{w_k\}$ be the iterates generated by* (5.16) *and let $0 < \alpha_k < \frac{1}{L_w}$. Let assumptions 11 and 1 hold. Then*

$$E[F(w_{k+1})] \leq F(w_k) - \frac{\alpha_k}{2}\|\nabla F(w_k)\|^2 + \frac{\alpha_k}{2}C. \tag{5.35}$$

*Proof.* Following the lines of [8, Theorem 4.1], by assumption 1,

$$E[F(w_{k+1})] \leq F(w_k) - \alpha_k \nabla F(w_k)^T g_k + \frac{L_w \alpha_k^2}{2}\|g_k\|^2.$$

Recalling assumption 11, for all $k$ such that $\alpha_k < \frac{1}{L_w}$ it holds that

$$\begin{aligned}
E[F(w_{k+1})] &\leq F(w_k) - \alpha_k \nabla F(w_k)^T g_k + \frac{L_w \alpha_k^2}{2}\|g_k\|^2 \\
&= F(w_k) - \alpha_k \nabla F(w_k)^T (\nabla F(w_k) + e(w_k)) + \frac{L_w \alpha_k^2}{2}\|\nabla F(w_k) + e(w_k)\|^2 \\
&= F(w_k) - \alpha_k \left(1 - \frac{\alpha_k L_w}{2}\right)\|\nabla F(w_k)\|^2 - \alpha_k(1 - \alpha_k L_w)\nabla F(w_k)^T e(w_k) + \frac{L_w \alpha_k^2}{2}\|e(w_k)\|^2 \\
&\leq F(w_k) - \alpha_k \left(1 - \frac{\alpha_k L_w}{2}\right)\|\nabla F(w_k)\|^2 + \alpha_k(1 - \alpha_k L_w)\|\nabla F(w_k)\|\|e(w_k)\| + \frac{L_w \alpha_k^2}{2}\|e(w_k)\|^2 \\
&\leq F(w_k) - \frac{\alpha_k}{2}\|\nabla F(w_k)\|^2 + \frac{\alpha_k}{2}\|e(w_k)\|^2
\end{aligned}$$

where $e(w_k) = g_k - \nabla F(w_k)$ and it was used the fact that $(\frac{1}{\sqrt{2}}\|F(w_k)\| - \frac{1}{\sqrt{2}}\|e(w_k)\|)^2 \geq 0$. Taking the total expectation and recalling (5.34), the proof of (5.35) follows. □

In the following, convergence analysis for algorithm 16 in the nonconvex and strongly convex cases is presented. All the analysis will consider a setting where the stepsize is fixed, i.e. $\alpha_k = \alpha > 0$ for all $k$. The analysis for a diminishing stepsize is not reported, since there is no theoretical gain in employing a diminishing stepsize: indeed, being the gradient approximation a biased estimator of the true gradient, convergence to a neighborhood of the solution is the best that can be achieved.

**Theorem 34** (nonconvex objectives - fixed stepsize)**.** *Let $\{w_k\}$ be the iterates generated by* (5.16) *and let assumptions 11, 1, 2 hold. Let $F$ be bounded from below and let $F_{inf}$ be its lower bound. Let the stepsize be*

$$0 < \alpha < \frac{1}{L_w}.$$

*Then*

$$\lim_{K\to\infty} E\left[\frac{1}{K}\sum_{k=1}^{K} \|\nabla F(w_k)\|^2\right] \leq \frac{2}{K\alpha}(F(w_1) - F_{inf}) + \frac{\alpha}{2}C. \tag{5.36}$$

*Proof.* From (5.35),

$$E[F(w_{k+1})] - F(w_k) \leq -\frac{\alpha}{2}E[\|\nabla F(w_k)\|^2] + \frac{\alpha}{2}C,$$

and by summing over $k = 1, \ldots, K$ one gets

$$F_{inf} - F(w_1) \leq E[F(w_{K+1})] - F(w_1) \leq -\frac{\alpha}{2}E\left[\sum_{k=1}^{K} \|\nabla F(w_k)\|^2\right] + K\frac{\alpha}{2}C,$$

which yields

$$E\left[\sum_{k=1}^{K} \|\nabla F(w_k)\|^2\right] \leq \frac{2}{\alpha}(F(w_1) - F_{inf}) + K\frac{\alpha}{2}C.$$

Taking the limit for $K \to \infty$ and dividing by $K$, (5.36) follows. □

**Theorem 35** ($\mu$-strongly convex objectives - fixed stepsize)**.** *Let $\{w_k\}$ be the iterates generated by* (5.16) *and let assumptions 11, 1, 2 and 3 hold. Let the stepsize be*

$$0 < \alpha < \frac{1}{L_w}.$$

*Then, after K iterations,*

$$E[F(w_{K+1})] - F(w^\star) \leq \rho^K [F(w_1) - F(w^\star)] + \frac{C}{2\mu}, \tag{5.37}$$

*where $\rho = 1 - \mu\alpha$. Therefore, if $K \to \infty$,*

$$\lim_{K\to\infty} E[F(w_K)] - F(w^\star) \leq \frac{C}{2\mu}. \tag{5.38}$$

*Proof.* From assumption 3 and (5.35), since $\alpha < \frac{1}{L_w}$, it holds for any $k = 1, \dots, K$

$$\begin{aligned} E[F(w_{k+1})] - F(w^\star) &\leq F(w_k) - F(w^\star) - \frac{\alpha}{2}\|\nabla F(w_k)\|^2 + \frac{\alpha}{2}C \\ &\leq F(w_k) - F(w^\star) - \mu\alpha(F(w_k) - F(w^\star)) + \frac{\alpha}{2}C \\ &= (1 - \mu\alpha)(F(w_k) - F(w^\star)) + \frac{\alpha}{2}C \\ &= \rho(F(w_k) - F(w^\star)) + \frac{\alpha}{2}C \\ &\leq \rho(F(w_k) - F(w^\star)) + \frac{\alpha}{2}C, \end{aligned}$$

where $\rho = 1 - \mu\alpha < 1$. The above relation yields

$$E[F(w_{K+1})] - F(w^\star) \leq \rho^K(F(w_1) - F(w^\star)) + \frac{\alpha C}{2(1-\rho)},$$

which implies (5.37). Since $\rho < 1$ , by taking the limit for $K \to \infty$, (5.38) follows. □

Theorem 35 explicitely shows the dependence of the neighborhood reached by algorithm 16 on the gradient approximation variance bound $C$.

The above results are very similar to the ones in the analysis of SG methods of chapter 3. Observe, nevertheless, that in this case, where the direction $g_k$ is not an unbiased estimate of the true gradient, even employing a diminishing stepsize would not yield convergence to the solution of problem 1.3.

### 5.3.2 Dynamic batching

Recalling (5.32) and (5.28), when a sample approximation based on a sample of size $|S|$ is employed, the bound on the gradient approximation variance $E[\|g_k - \nabla F(w)\|^2]$ depends on $\sigma^2$ in the following way, respectively in the GS and FD cases,

$$\frac{4(d+5)(B^2 + \frac{\sigma^2}{|S|})}{m_k} + \frac{3\nu^2}{2}L_w^2(d+3)^3,$$

$$2d\left(\frac{L_w\nu}{2} + \frac{2\varepsilon_m}{\nu}\right)^2 + 2\frac{\sigma^2}{|S|}.$$

Therefore, a basic approach to dynamic batching could be to let the sample size $|S| \to \infty$, in order to let the term $\frac{\sigma^2}{|S_k|} \to 0$. To preserve linear convergence to a neighborhood of the solution, in the $\mu$-strongly convex case, one can employ an updating scheme for the sample size satisfying $|S| \sim O\left(\gamma^k\right)$, for a $\gamma > 1$ [44]. Nevertheless, in a setting where an unbiased estimate of the gradient is not available, there is no reason why the sample size should grow to infinity, since the variance of the error $E[\|e_k\|^2]$ will never go to zero. Therefore, for example in the finite differences case, it seems reasonable to only require

$$|S| \to O\left(\frac{\sigma^2}{d}\left(\frac{L_w\nu}{2} + \frac{2\varepsilon_m}{\nu}\right)^{-2}\right).$$

A more advanced approach is the *norm test* [22], defined to decide, at each iteration, whether the size of the sample size should be increased or not:

$$\|g_k - \nabla F(w_k)\| \leq \theta\|g_k\|, \tag{5.39}$$

where $\theta \in (0,1)$. The test checks whether the gradient approximation is close enough to the true gradient, although the latter is not available in real cases and therefore an approximation was proposed. Nevertheless, the test in (5.39) is not appropriate in the zeroth-order setting, since $g_k$ is a biased estimator of $\nabla F(w_k)$ and (5.39) would eventually never hold in a neighborhood of the solution. Therefore, by taking the finite differences example and recalling that in the FD case the biased estimate of the gradient is

$$\nabla F^{FD}(w_k) = E_\zeta[g^{FD}(w_k,\zeta)],$$

the following modification to the *norm test* can be proposed

$$\|g_k - \nabla F^{FD}(w_k)\| \leq \theta \|g_k\|, \tag{5.40}$$

where $\theta \in \left(0, \sqrt{2}-1\right)$. This modification is based on property (5.32), which yields

$$\|g_k - \nabla F(w_k)\| \leq \|g_k - \nabla F^{FD}(w_k)\| + \|\nabla F^{FD}(w_k) - \nabla F(w_k)\| \sim O\left(\frac{\sigma}{\sqrt{|S|}}\right) + C,$$

and from which it is clear that employing a dynamic batching scheme will impact the term $O\left(\frac{\sigma}{\sqrt{|S|}}\right)$, but the constant term $C$ will never be driven to zero.

Lemma 9 is useful for the following convergence analysis.

**Lemma 9.** *Assume* (5.40) *holds. Then*

$$\|g_k\|^2 \leq \frac{2\theta^2}{(1-\theta)^2}\|\nabla F(w_k)\|^2 + \frac{2}{(1-\theta)^2}C. \tag{5.41}$$

*Proof.* Equation (5.40) immediately yields

$$\|g_k\| \leq \frac{1}{1-\theta}\|\nabla F^{FD}(w_k)\|.$$

Therefore

$$\begin{aligned}
\|e_k\| &\leq \|g_k - \nabla F^{FD}(w_k)\| + \|\nabla F^{FD}(w_k) - \nabla F(w_k)\| \\
&\leq \theta\|g_k\| + \sqrt{C} \\
&\leq \frac{\theta}{1-\theta}\|\nabla F^{FD}(w_k)\| + \sqrt{C} \\
&\leq \frac{\theta}{1-\theta}(\sqrt{C} + \|\nabla F(w_k)\|) + \sqrt{C} \\
&= \frac{\theta}{1-\theta}\|\nabla F(w_k)\| + \left(1 + \frac{\theta}{1-\theta}\right)\sqrt{C}.
\end{aligned}$$

Taking the square of the above,

$$\begin{aligned}
\|e_k\|^2 &\leq 2\frac{\theta^2}{(1-\theta)^2}\|\nabla F(w_k)\|^2 + 2\left(1 + \frac{\theta}{1-\theta}\right)^2 C \\
&= 2\frac{\theta^2}{(1-\theta)^2}\|\nabla F(w_k)\|^2 + \frac{2}{(1-\theta)^2}C.
\end{aligned}$$

□

Theorem 36 shows the convergence properties of a dynamic batching scheme based on a norm test applied to the zeroth-order SGD setting with strongly convex objectives.

**Theorem 36** ($\mu$-strongly convex objectives - norm test)**.** *Let Assumptions 1 and 3 hold. Assume that the test* (5.40) *holds at any iteration $k = 1, \dots, K$ with $\theta \in (0, \sqrt{2}-1)$ and that*

$$\alpha_k = \alpha < \frac{1}{L_w} \quad \forall\, k. \tag{5.42}$$

*Then*

$$\lim_{k\to\infty} F(w_k) - F(w^\star) \leq \frac{C}{\mu(1-2\theta-\theta^2)}. \tag{5.43}$$

*Proof.* By assumption 1 and recalling lemma 9,

$$\begin{aligned} F(w_{k+1}) &\leq F(w_k) - \frac{\alpha}{2}\|\nabla F(w_k)\|^2 + \frac{\alpha}{2}\|e_k\|^2 \\ &\leq F(w_k) - \frac{\alpha}{2}\|\nabla F(w_k)\|^2 + \alpha\left[\frac{\theta^2}{(1-\theta)^2}\|\nabla F(w_k)\|^2 + \frac{1}{(1-\theta)^2}C\right] \\ &= F(w_k) - \frac{\alpha}{2}\left[1 - 2\frac{\theta^2}{(1-\theta)^2}\right]\|\nabla F(w_k)\|^2 + \alpha\frac{C}{(1-\theta)^2}. \end{aligned}$$

Therefore, by Assumption 3,

$$\begin{aligned} F(w_{k+1}) - F(w^\star) &\leq F(w_k) - F(w^\star) - \mu\alpha\left(1 - 2\frac{\theta^2}{(1-\theta)^2}\right)[F(w_k) - F(w^\star)] + \alpha\frac{C}{(1-\theta)^2} \\ &= \left[1 - \mu\alpha\left(1 - 2\frac{\theta^2}{(1-\theta)^2}\right)\right][F(w_k) - F(w^\star)] + \alpha\frac{C}{(1-\theta)^2} \\ &= \rho\,[F(w_k) - F(w^\star)] + \alpha\frac{C}{(1-\theta)^2}, \end{aligned}$$

where $\rho = 1 - \mu\alpha\left(1 - 2\frac{\theta^2}{(1-\theta)^2}\right) < 1$. By taking the limit for $k \to \infty$, it follows

$$\lim_{k\to\infty} F(w_k) - F(w^\star) \leq \frac{\alpha\frac{C}{(1-\theta)^2}}{1-\rho},$$

and by simplifying the above one gets (5.43). □

## 5.4 Preliminary experiments

One of the goals of these preliminary experiments was to compare Gaussian Smoothing (GS) approaches, much used in the RL community, to more standard gradient approximation techniques, like Finite Differences (FD) and Random Coordinate Descent (RCD). Indeed, given a fixed budget of function evaluations that can be computed at each iteration, one would expect GS and RCD to be comparable, since the two methods both compute function evaluations in random directions to compute a gradient approximation. The final aim is to get a better understanding of the optimization landscape and, if GS is confirmed to be the best performing gradient approximation method, to relate its performance to the structure of the optimization problem.

The RL environments tested in the following preliminary experiments are taken from the OpenAI gym library [21]. In particular, the continuous control tasks from MuJoCo [109], in which both the state and action spaces are real-valued, were used. Table 5.1 summarizes the characteristics of the 5 reinforcement learning tasks on which the DFO algorithms were applied.

The policy was parameterized by a neural network, with two hidden layers and a number of neurons for each layer that was tuned based on the environment. Since the action space is constrained to be in $[-1,1]^{d_A}$, tanh nonlinearities were employed in the neural networks.

| **Environment** | **State dim** | **Action dim** | **Num neurons per layer** | **Num of parameters** $d$ |
|---|---|---|---|---|
| Swimmer | 8 | 2 | 20 | 642 |
| HalfCheetah | 17 | 6 | 60 | 5106 |
| Walker2d | 17 | 6 | 60 | 5106 |
| Hopper | 11 | 3 | 40 | 723 |
| Reacher | 11 | 2 | 20 | 702 |

**Table 5.1.** MuJoCo continuous control environments

To make the experiments replicable, the seeds to be used throughout the optimization procedure were always fixed in advance, by generating a list of random integers, based on an initial seed, that were then used as seeds for each step of the optimization procedure. In particular, the seeds were used for (i) initializing the initial state for each episode run (one at each iteration) and (ii) determining the realization of the actions random variable. Doing so, the optimization procedure reduces to a deterministic simulation-optimization framework, where the randomness is controlled by the seed.

At each iteration, the gradient estimate was computed:

- for Full (Forward) Finite Differences, by computing $d$ function evaluations along the coordinate directions and then differencing;
- for Random Coordinate Descent, by computing 1 function evaluation in a random coordinate direction and then differencing only along such coordinate direction, namely computing only a gradient component and setting all the others to 0;
- for Gaussian Smoothing, by computing 1 function evaluation in a random, normally distributed direction and then differencing.

The 'true' objective function (necessary for plots), which is an expected value and therefore needs be approximated by sample averaging, was computed as the average over $M = 100$ replicates, i.e.

$$F(w_k) \approx \frac{1}{M} \sum_{i=1}^{M} \phi(w_k, \zeta_i),$$

where the index $i$ in the random variable $\zeta_i$ represents different seeding.

One of the crucial parameters to be tuned is the differencing interval $\nu > 0$, which is present in both the gradient approximation methods considered in this section, namely Gaussian Smoothing and Finite Differences. Indeed, although there exists a theoretical optimal value for $\nu$, this is not applicable in this setting, since the function evaluations are affected by noise which can not easily be bounded. Furthermore, being the function potentially noncontinuous, a too small value of $\nu$ may lead to numerical instability and extremely high variance in the gradient estimate (the continuity properties

of the objective function will be further discussed in the next section). Therefore, the parameter $\nu$ was tuned by a grid search, which led to the choice of $\nu = 0.1$.

Another design choice is the standard deviation of the actions $\sigma_A$. This was chosen to diminish to zero with a rule

$$\sigma_A = \frac{1}{1+\tau k},$$

where $k$ is the iteration counter and $\tau > 0$ controls the speed of decrease, in order to allow for more exploration in the early iterations of the algorithm. The parameter $\tau$, which controls the speed of decay of the actions standard deviation $\sigma_A$, was chosen through a grid search, together with the fixed stepsize $\alpha > 0$. The two parameters were tuned for each environment and gradient approximation method separately, specifically $\alpha \in [10^{-8}, 10^{0}]$ and $\tau \in [10^{-3}, 10^{0}]$. As reported in tables 5.2-5.4, the chosen stepsize was always very small, which is expected due to the high variance in the gradient estimate. This will also be further developed in the next section.

| **Environment** | $\alpha$ | $\tau$ |
|---|---|---|
| Swimmer | $10^{-4}$ | 1 |
| HalfCheetah | $10^{-6}$ | 0.1 |
| Walker2d | $10^{-6}$ | 1 |
| Hopper | $10^{-4}$ | 0.1 |
| Reacher | $10^{-6}$ | 0.01 |

**Table 5.2.** Gaussian Smoothing: values chosen for the stepsize $\alpha$ and the action std decay parameter $\tau$

| **Environment** | $\alpha$ | $\tau$ |
|---|---|---|
| Swimmer | $10^{-4}$ | 1 |
| HalfCheetah | $10^{-4}$ | 0.1 |
| Walker2d | $10^{-4}$ | 1 |
| Hopper | $10^{-6}$ | 0.1 |
| Reacher | $10^{-4}$ | 1 |

**Table 5.3.** Random Coordinate Descent: values chosen for the stepsize $\alpha$ and the action std decay parameter $\tau$

| **Environment** | $\alpha$ | $\tau$ |
|---|---|---|
| Swimmer | $10^{-4}$ | 1 |
| HalfCheetah | $10^{-4}$ | 1 |
| Walker2d | $10^{-4}$ | 1 |
| Hopper | $10^{-6}$ | 1 |
| Reacher | $10^{-4}$ | 1 |

**Table 5.4.** Full Finite Differences: values chosen for the stepsize $\alpha$ and the action std decay parameter $\tau$

Figures 5.2-5.6 seem to suggest that GS is the better gradient approximation method for these continuous control RL tasks. The open question, which will be further discussed in next section, is

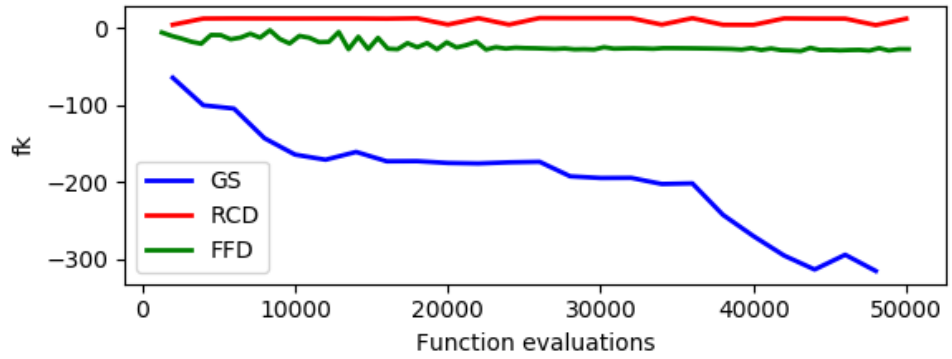


**Figure 5.2.** Swimmer-v2 environment, function value (fk) vs function evaluations for Gaussian Smoothing (GS), Random Coordinate Descent (RCD) and Full Finite Differences (FFD)

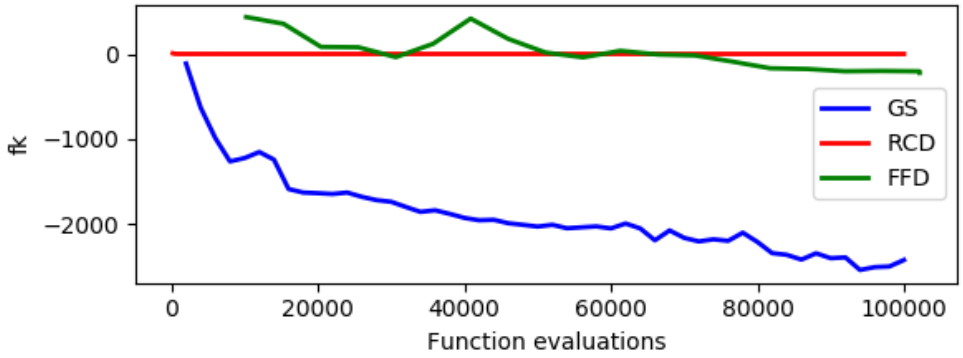


**Figure 5.3.** HalfCheetah-v2 environment, function value (fk) vs function evaluations for Gaussian Smoothing (GS), Random Coordinate Descent (RCD) and Full Finite Differences (FFD)

why this happens, given that GS computes gradient approximations by function evaluations in random directions, just like RCD.

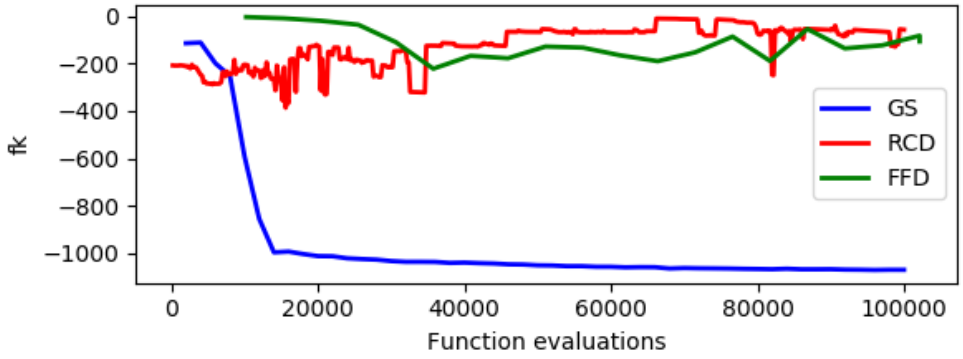


**Figure 5.4.** Walker2d-v2 environment, function value (fk) vs function evaluations for Gaussian Smoothing (GS), Random Coordinate Descent (RCD) and Full Finite Differences (FFD)

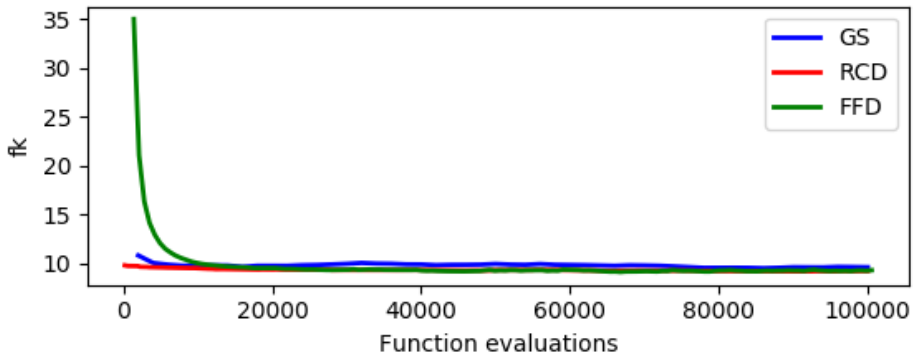


**Figure 5.5.** Reacher-v2 environment, function value (fk) vs function evaluations for Gaussian Smoothing (GS), Random Coordinate Descent (RCD) and Full Finite Differences (FFD)

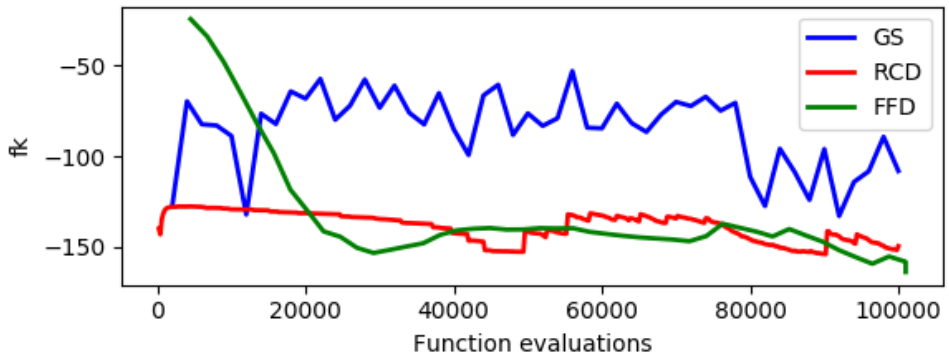


**Figure 5.6.** Hopper-v2 environment, function value (fk) vs function evaluations for Gaussian Smoothing (GS), Random Coordinate Descent (RCD) and Full Finite Differences (FFD)

## 5.5 Discussion and future directions

In this section, a discussion of the results shown above is presented, with some considerations on the potential reasons behind the results. In particular, the focus will be on the characterization of the optimization landscape and on potential approaches to improve the DFO methods applied to such a landscape.

### 5.5.1 The optimization landscape

In this sub-section, some plots of the reward function along the search directions employed at different stages of the optimization process are presented for the policy optimization of the 'Swimmer' environment, one of the continuous control tasks introduced in the previous section.

These plots are intended to be indicative of the behavior of the function to be maximized during the optimization. Indeed, the aim is to support the conjecture that such function is not only highly nonlinear, nonconvex and nonsmooth, but also discontinuous, questioning the convergence properties of virtually any optimization algorithm employed so far in the RL literature.

In particular, the plots show the objective function behavior along search directions, which are

1. coordinate directions in Random Coordinate Descent algorithm;
2. random Gaussian directions in Gaussian Smoothing algorithm.

These were plotted at different stages of the optimization process, i.e. when far from a good solution and when close to a good solution. Furthermore, the objective function was plotted on a single episode (namely, a single noisy function evaluation, named from now on *sampled function*) and as the average over 100 episodes (namely, a sample approximation of the true function, named from now on *true function*).

Figure 5.7 shows the comparison of the sampled and true functions along coordinate and random Gaussian directions, in a point close to a solution of the Swimmer environment (i.e., the Swimmer environment is considered solved with a reward of 360 - recall that the aim of the DFO algorithm is to minimize the negative reward).

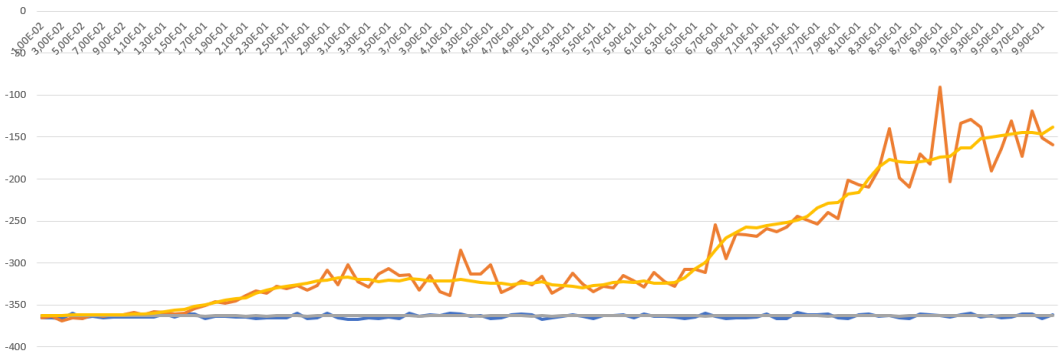


**Figure 5.7.** Swimmer environment, close to solution. In yellow, the sampled function over a random Gaussian direction, in orange the full function over the same random Gaussian direction. In blue, the sampled function over a random coordinate direction, in grey the full function over the same random coordinate direction.

Clearly, the function is almost flat in the random coordinate direction, while this is not the case in the random gaussian direction. Furthermore, it is already possible to notice in figure 5.7 that

although the sampled function is nonsmooth and noncontinuous, the true function can be continuous and smooth. This is in line with the results in [59], where it is proven that, when approximating the expected value of a stochastic function by sample approximation, every sample approximation may be noncontinuous, but the true function may be continuous and smooth.

In figure 5.8, instead, the comparison between the sampled and true functions along coordinate and random Gaussian directions is shown when far from a solution of the Swimmer environment (i.e., the Swimmer environment is considered solved with a reward of 360 - recall that the aim of the DFO algorithm is to minimize the negative reward).

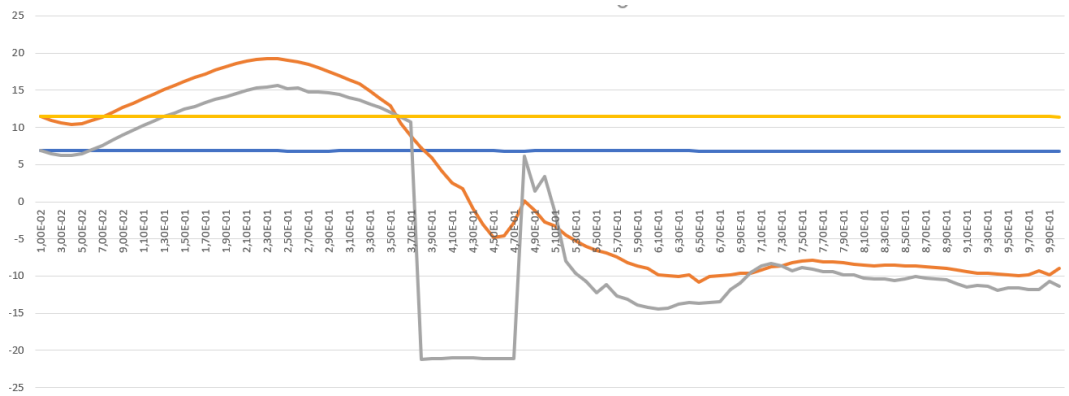


**Figure 5.8.** Swimmer environment, far from solution. In grey, the sampled function over a random Gaussian direction, in orange the full function over the same random Gaussian direction. In blue, the sampled function over a random coordinate direction, in yellow the full function over the same random coordinate direction.

Here, the 'flatness' along a random coordinate direction is even more evident. Indeed, by 'zooming' in and plotting only the sampled and full functions over the random coordinate direction, see figures (5.9) and (5.10), the noncontinuity of the sampled function is even clearer.

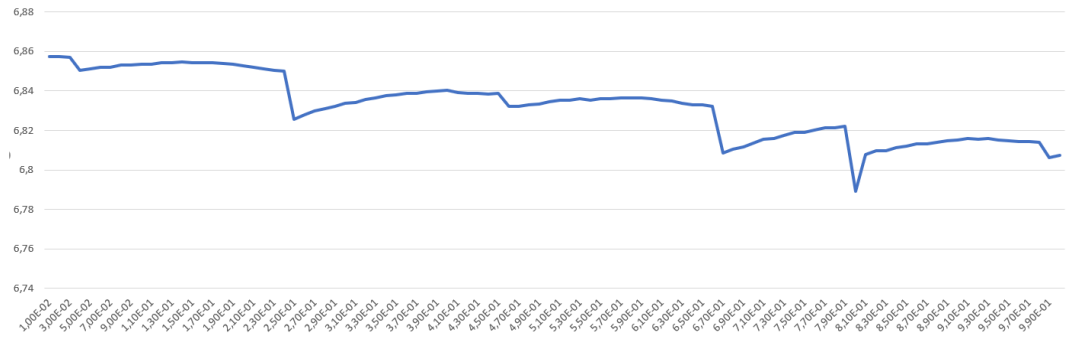


**Figure 5.9.** Swimmer environment, far from solution. In blue, the sampled function over a random coordinate direction.

Therefore, it is not surprising that random coordinate descent methods perform so bad on these tasks, since it seems that the function is often flat (and with 'jumps') in coordinate directions. This is in line with the conjecture that, due to over-parameterization induced by the use of deep neural networks, it is often useless to compute derivatives in many coordinate directions. As a final remark, these experiments were replicated several times in several phases of the optimization, on more than

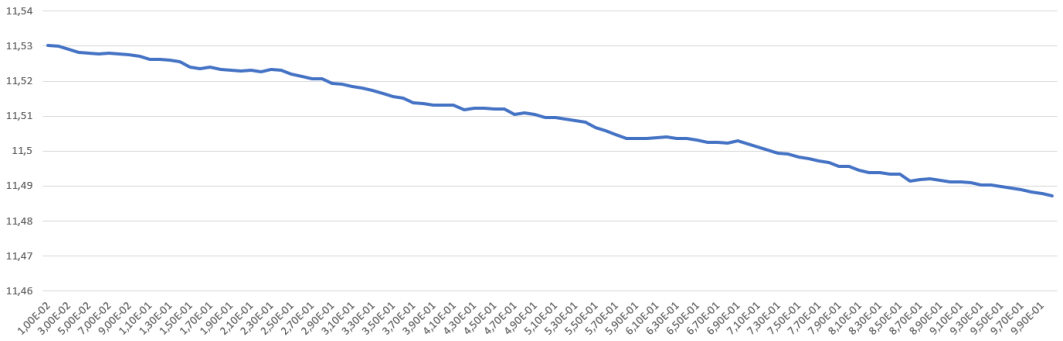


**Figure 5.10.** Swimmer environment, far from solution. In blue the full function over the same random coordinate direction.

one RL environment. For brevity, only a subset of the experiments performed on the Swimmer dataset are reported here.

### 5.5.2 Future directions

Concluding, the main open questions that remain unresolved are:

(i) how can a DFO method approximate the gradient, when the sampled function can be noncontinuous? Are these methods the right ones to employ in this setting?

(ii) How can a DFO method leverage the fact that the objective function is often flat in many coordinate directions? Do GS directions, which span the whole space $\mathbb{R}^d$, overcome this issue? Are there better ways of doing so?

In order to deal with the two open questions introduced above, the future work will focus on two main issues:

1. how to produce gradient estimates, in a setting where the sampled functions are highly affected by noise;

2. how to extend classic linesearch-based DFO methods, in order to leverage random, GS-like search directions in a convergent framework. Furthermore, including a controlled nonmonotonicity to get convergence of the iterates, without hindering the good numerical properties of GS-based methods.

# Chapter 6

# Distributed algorithms for linearly constrained convex problems

This is a joint work with Simone Sagratella, published on the Journal of Global Optimization [31].
The aim of this project was to develop a simple, distributable and parallel algorithm for convex problems with linear coupling constraints (i.e. constraints over all the variables), with application to the dual formulation of SVM's training problem.

The popularity of distributed and parallel algorithms to solve optimization problems has risen consistently in recent years. Indeed, their importance has been even more stressesd by the exponential growth of the amount of data available. While the reason for employing parallelism comes from computational time constraints, the possibility of distributing the data can bring further benefits. In fact, block decomposition and partitioning of the data can be required by, e.g., privacy concerns [23, 33, 47, 93, 94, 95, 112].

Nevertheless, in some applications such data decomposition may be impossible due to the formulation of the optimization problem, e.g., in support vector machines training dual formulation, where the variables are coupled by a single equality constraint, which makes it difficult to distribute the data [66, 69, 71, 72, 81].

A typical solution in optimization, when dealing with complex constraints, is to include those in the objective function, like in the well-known augmented Lagrangian method [16, 36, 37, 68]. Unfortunately, all of these methods usually require the solution of a subproblem, at least to a given accuracy, at each iteration.

In the following, a modified augmented Lagrangian method to solve convex problems with linear coupling constraints is proposed.

## 6.1 Introduction and motivation

Consider a generalization of problem 1.13,

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{s.t.} \quad & h(x) = Ax - b = 0 \\ & x \in X, \end{aligned} \tag{6.1}$$

where $f \in C^{1,1}$ is convex with Lipschitz continuous gradients, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $X \subseteq \mathbb{R}^n$ is convex, compact and encompasses a structure that is separable in $N$ blocks, namely

$$X \triangleq \prod_{v=1}^{N} X_v,$$

where $X_v \subseteq \mathbb{R}^{n_v}$, with $n = n_1 + \cdots + n_N$. For the sake of notational simplicity the feasible set of Problem (6.6) will be denoted by $S$. We observe that the set $S$ is in general not separable due to the presence of the constraints $h$ that may tie together variables of different blocks. For this reason we refer to $h$ as the coupling constraints. Notice that the case with linear inequality constraints can be included in the above framework by simply adding slack variables.

A solution to problem 6.1 can be found by many (synchronous) distributed algorithms. A distributed algorithm allows to split the computation at each iteration into $N$ independent blocks, usually distributed to several parallel processes, which then communicate with each other thanks to a communication phase. One further characteristic one may ask to a distributed algorithm is the possibility

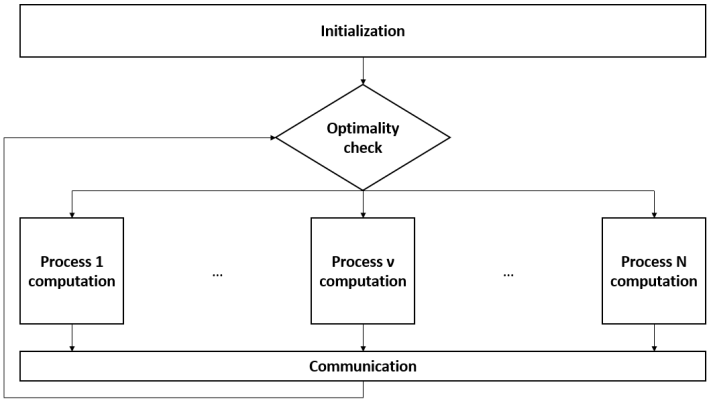


**Figure 6.1.** General distributed algorithmic scheme

of partitioning the data into $N$ blocks, where each data block is only known by the respective process. This property may be desirable, or even necessary, when the data comes from several sources and can not be stored in a unique place, for example when there exist privacy issues or the dimension of the data is huge.

One way of solving problem 6.1 is the classical gradient projection algorithm [14]:

$$x^{k+1} = \mathscr{P}_S\left[x^k - \alpha_k \nabla f(x^k)\right], \tag{6.2}$$

where $\alpha_k$ is a positive stepsize and $\mathscr{P}_S(z)$ denotes the projection of $z$ over the convex set $S$. The convergence of iteration (6.2) can be proven if the steplength $\alpha_k$ is computed by one of the following [11]:

(i) a linesearch procedure;

(ii) a diminishing rule;

(iii) a fixed, sufficiently small, $\alpha_k = \alpha \in \left(0, \frac{2}{L}\right)$.

Unfortunately this method does not fall into the class of distributed algorithms because the projection on $S$, in general, can not be decomposed with respect to the $N$ blocks, because of the *coupling constraints*.

A solution to this difficulty is the Augmented Lagrangian Method (ALM), where some (or all) of the constraints are moved to the objective function in order to make the optimization easier. The Lagrangian function can be defined as

$$\mathscr{L}_\rho(x,\mu) \triangleq f(x) + \mu^T h(x) + \frac{1}{2}\rho\|h(x)\|^2, \tag{6.3}$$

where only the equality constraints where included, since $X$ represents "simple" constraints, $\mu \in \mathbb{R}^m$ is the vector of Lagrangian multipliers and $\rho > 0$ is the parameter of the penalty term. It is well known that finding a solution to problem 6.1 is equivalent to finding a saddle point $\widetilde{x} \in X$ of (6.3) [14, 20]. In a general augmented Lagrangian method (see e.g. [16] for an efficient version of such method) a solution of Problem (6.6) can be found by using the scheme described in Algorithm 17.

**Algorithm 17:** Basic augmented Lagrangian method

1 **for** $k = 0, 1, \ldots$ **do**
2 $\quad x^{k+1} = \arg\min_{x\in X} \mathscr{L}_{\rho_k}(x, \mu^k)$;
3 $\quad \mu^{k+1} = \mathscr{P}_C\left[\mu^k + \beta_k \nabla_\mu \mathscr{L}_{\rho_k}(x^{k+1}, \mu^k)\right]$;
4 $\quad \rho_{k+1} \geq \rho_k$.
5 **end**

At step 2, the augmented Lagrangian function is minimized with respect to $x \in X$; then, the multipliers $\mu_k$ are updated in the gradient direction with a positive stepsize $\beta_k$, in order to perform an ascent step; finally, at step 4 the penalty parameter $\rho_k$ is (possibly) increased, usually only if a sufficient decrease in the constraints violation is not obtained.

The above general scheme converges under a suitable choice of the sequence of positive stepsizes $\beta_k$, or by updating the penalty parameter $\rho_k$ based on the amount of violation of the coupling constraints $h$. Furthermore, convergence can be guaranteed even if the exact minimization at step 2 is substituted by an inexact minimization: in this case, $x^{k+1}$ must satisfy $\mathscr{L}_{\rho_k}(x^{k+1}, \mu^k) \leq \mathscr{L}_{\rho_k}(x, \mu^k) + \varepsilon_k$, for all $x \in X$, with $\varepsilon_k \downarrow 0$ during the optimization process.

Algorithm 17 can itself be distributed by distributing the minimization at step 2 as in the scheme described in figure 6.1. Indeed, $X$ is separable in $N$ blocks and therefore the projection operation may be distributed, leading to an algorithm that iteratively performs a distributed minimization over $X$, divided in $N$ sub-minimizations. Such algorithm can be viewed as a distributed algorithm [72, 93]. This notwithstanding, the (inexact) solution of an optimization problem at each iteration may be a burden from a computational perspective.

Another primal-dual method that has been much studied in the last years is the Alternating Direction Method of Multipliers (ADMM), presented in algorithm 18. ADMM was developed to solve problems of the form

$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{s.t.} \quad & h(x) = Ax + Bz - c = 0, \end{aligned} \tag{6.4}$$

where both the variables and the objective function can be divided in two separable blocks.

**Algorithm 18:** Alternating Directions Method of Multipliers

1 **for** $k = 0, 1, \ldots$ **do**
2 $\quad x^{k+1} = \arg\min_x \mathscr{L}_{\rho_k}(x, z^k, \mu^k)$;
3 $\quad z^{k+1} = \arg\min_z \mathscr{L}_{\rho_k}(x^{k+1}, z, \mu^k)$;
4 $\quad \mu^{k+1} = \mu^k + \rho_k(Ax^{k+1} + Bz^{k+1} - c)$;
5 $\quad \rho_{k+1} \geq \rho_k$.
6 **end**

At any iteration $k$, ADMM performs:

(i) a minimization of the Lagrangian function over $x$;

(ii) a minimization of the Lagrangian function over $z$;

(iii) a maximization step over the multipliers $\mu$ with stepsize $\rho_k$.

In the framework of problem 6.1, ADMM reduces to the method of multipliers [14], which can be seen as a variant of algorithm 17 with $\beta_k = \rho_k$. This means that the above considerations apply to the ADMM setting, given that an (inexact) minimization with respect to the primal variables $x$ must be performed at each iteration (see e.g. [20, 53]).

A natural question that may arise is if the substitution of the (inexact) minimization with just a gradient projection step would still yield convergence. The answer is no, in general, as explained by the following counterexample.

**Example 2.** *Consider the problem*

$$\begin{aligned} \textit{minimize} \quad & x_1 \\ \textit{s.t.} \quad & h(x) = x_1 - x_2 = 0 \\ & x \in [-1, 1]^2, \end{aligned} \tag{6.5}$$

*which has a unique solution* $x^\star = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$.

*In [16] the authors prove that Algorithm 17 globally converges to a solution of the problem, with* $C = \{0\}$ *and the update*

$$\rho_{k+1} = 2\rho_k \textit{ if } \|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2,\ \rho_{k+1} = \rho_k \textit{ otherwise},$$

*where* $\tau \in (0, 1)$. *Indeed, this setting implies that* $\mu$ *vanishes and the Lagrangian function is*

$$\mathscr{L}_{\rho_k}(x) = x_1 + \frac{\rho_k}{2}\|x_1 - x_2\|^2,$$

*with gradient*

$$\nabla\mathscr{L}_{\rho_k}(x) = \begin{pmatrix} 1 + \rho_k(x_1 - x_2) \\ \rho_k(x_2 - x_1) \end{pmatrix}.$$

*The substitution of step 2 with a gradient projection step would yield the following modified version of the algorithm, for any iterate* $k \geq 0$*:*

*(i)* $\alpha_k = \frac{1}{L+\rho_k\|A\|^2} = \frac{1}{2\rho_k}$;

*(ii)* $x^{k+1} = \mathscr{P}_{[-1,1]^2}\left[x^k - \alpha_k \nabla \mathscr{L}_{\rho_k}(x^k)\right]$;

*(iii)* $\rho_{k+1} = 2\rho_k$ *if* $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$; $\rho_{k+1} = \rho_k$ *otherwise.*

*In particular, by choosing the parameters as* $\tau \in \left(0, \frac{1}{4}\right)$, $\rho_0 = 2$ *and* $x^0 = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$, *it holds, for any k, that condition* $\|h(x^k)\|^2 > \tau\|h(x^{k-1})\|^2$ *is satisfied and:*

$$\rho_k = 2\rho_{k-1} = 2^{k+1},$$

$$\nabla \mathscr{L}_{\rho_k}(x^k) = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$$x^k = \begin{pmatrix} 0 \\ \frac{1}{2^{k+1}} \end{pmatrix},$$

$$h(x^k) = -x_2 = -\frac{1}{2^{k+1}}.$$

*But this implies*

$$\lim_{k\to\infty} x^k = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \neq x^\star,$$

*that is, such modification of algorithm 17 does not converge to the solution of the problem.* □

## 6.2 Distributed algorithms for convex problems with linear coupling constraints

The general formulation of a convex programming problem with linear coupling constraints can be written as

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{s.t.} \quad & h(x) = Ax - b = 0 \\ & x \in X, \end{aligned} \tag{6.6}$$

where $S$ is assumed to be non-empty. Therefore, an optimal solution $x^\star$ exists such that

$$x^\star \in S, \qquad f(x^\star) \leq f(x) \quad \forall\, x \in S.$$

Algorithm 19 employs the well-known projected gradient to solve the augmented Lagrangian reformulation of Problem (6.6). It is characterized by two main parameters:

(i) $\widehat{\mu} > 0$, used to define the compact set for the multipliers $\mu^k$;

(ii) $\widehat{\rho} > 0$, namely an upper bound for the penalty parameter $\rho_k$.

Both parameters will play a key role in the convergence of the algorithm, as will be shown in the following sections.

The following are the main two properties of algorithm 19:

**Algorithm 19:** Gradient projection augmented Lagrangian method

**Data:** $\widehat{\rho} > 0$, $\widehat{\mu} > 0$, $x^0 \in X$, $\mu^0 \in [-\widehat{\mu}, \widehat{\mu}]^m$, $\rho_0 \in (0, \widehat{\rho})$, $\gamma > 0$, $\delta > 0$, $\tau \in (0,1)$, $\widehat{k} = 0$

1 **for** $k = 0, 1, \ldots$ **do**

2 $\quad \alpha_k = \frac{1}{\overline{L} + \rho_k \|A\|^2 + \gamma(k - \widehat{k})}$

3 $\quad x^{k+1} = \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\rho_k}(x^k, \mu^k)]$

4 $\quad$ **if** $\rho_k < \widehat{\rho}$ **then**

5 $\quad\quad \mu^{k+1} \in [-\widehat{\mu}, \widehat{\mu}]^m$

6 $\quad\quad \widehat{k} = \widehat{k} + 1$

7 $\quad$ **end**

8 $\quad$ **else**

9 $\quad\quad \mu_i^{k+1} = \begin{cases} -\widehat{\mu}, & i : h_i(x^{k+1}) < 0 \\ \widehat{\mu}, & otherwise \end{cases}, \quad i = 1, \ldots, m$

10 $\quad$ **end**

11 $\quad \rho_{k+1} \in \begin{cases} [\min\{\rho_k + \delta, \widehat{\rho}\}, \widehat{\rho}], & \|h(x^{k+1})\| > \tau \|h(x^k)\| \\ \{\rho_k\}, & otherwise \end{cases}$

12 **end**

- distribution of the computation is easy and, if $f$ is quadratic, the data can be distributed;
- each iteration only requires the computation of a gradient projection step over $x$.

Observe that when the steplength $\alpha_k$ is computed at step 2, this is to ensure that, whenever $\widehat{k} = k$ (i.e. $\rho_k < \widehat{\rho}$), it dynamically estimates the quantity $\frac{1}{L^k_{\mathscr{L}}}$, where $L^k_{\mathscr{L}} = \overline{L} + \rho_k \|A\|^2$ (with $\overline{L}$ being the Lipschitz constant of $\nabla f$) is the Lipschitz constant of $\nabla_x \mathscr{L}_{\rho_k}(x^k, \mu^k)$ over $X$. Furthermore, it holds that when $\widehat{k} < k$ (i.e. $\rho_k = \widehat{\rho}$), $\alpha_k \downarrow 0$ and $\alpha_k$ is squared summable, but not summable, namely

$$\sum_{k=0}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=0}^{\infty} \alpha_k = \infty. \tag{6.7}$$

The following sections will show how such stepsize updating rule is key to the convergence of the algorithm.

After the stepsize update at step 2, algorithm 19 computes the new iterate $x^{k+1}$ by a gradient projection step, then performs an update of the multipliers $\mu^k$ (steps 5-8). Notice that if $\rho_k < \widehat{\rho}$ then the choice of the new multiplier vector $\mu^{k+1}$ is free, as long as it is included in the compact set $M_{\widehat{\mu}} \triangleq [-\widehat{\mu}, \widehat{\mu}]^m$, that is, the update may be the same as in algorithm 17. If, instead, $\rho_k = \widehat{\rho}$, then $\mu^{k+1}$ is computed so that the following holds

$$\mu^{k+1} \in \arg\max_{\mu \in M_{\widehat{\mu}}} \mathscr{L}_{\rho_k}(x^{k+1}, \mu).$$

From now on, the set $M_{\widehat{\mu}}$ will be referred to as $M$ when the dependence on $\widehat{\mu}$ is not of interest. The last step of the algorithm is the update of the penalty parameter $\rho_k$, which is increased (up to a threshold $\widehat{\rho} > 0$) if the equality constraint did not achieve a sufficient decrease in the last iteration. As a last comment, observe that algorithm 19 generates a bounded sequence $\{(x^k, \mu^k, \rho_k)\}$.

Recalling example 2 shown in the introduction, in example 3 algorithm 19 is applied to the same problem to outline the differences between the two algorithms.

**Example 3.** *Consider again the problem in Example 2. Assume that the penalty parameter is updated such that* $\rho_{k+1} = 2\rho_k$ *if* $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$, *where* $\tau \in \left(0, \frac{1}{4}\right)$, $\rho_{k+1} = \rho_k$ *otherwise. Algorithm 19 is applied with* $\widehat{\mu} = 0$, $\widehat{\rho} = 2$, $\gamma = \frac{1}{10}$, $\delta = 1$. *The Lagrangian function and its gradient are*

$$\mathscr{L}_{\rho_k}(x) = x_1 + \frac{\rho_k}{2}\|x_1 - x_2\|^2,$$

$$\nabla\mathscr{L}_{\rho_k}(x) = \begin{pmatrix} 1 + \rho_k(x_1 - x_2) \\ \rho_k(x_2 - x_1) \end{pmatrix}.$$

*Algorithm 19 performs the following steps at each iteration:*

*(i)* $\alpha_k = \frac{1}{L + \rho_k\|A\|^2 + \gamma(k-\widehat{k})} = \frac{1}{2\rho_k + \gamma(k-\widehat{k})}$;

*(ii)* $x^{k+1} = \mathscr{P}_{[-1,1]^2}\left[x^k - \alpha_k \nabla\mathscr{L}_{\rho_k}(x^k)\right]$;

*(iii)* $\rho_{k+1} = \min\{2\rho_k, \widehat{\rho}\}$ *if* $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$, $\rho_{k+1} = \rho_k$ *otherwise.*

*By choosing again* $\rho_0 = 2$ *and* $x^0 = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$, *the same starting point of Example 2 and for which the naive modification of Algorithm 17 does not converge, it follows*

$$x^1 = \begin{pmatrix} 0 \\ \frac{1}{4} \end{pmatrix}$$

*and, for any* $k \geq 1$,

$$\rho_k = \widehat{\rho} = 2,\ \widehat{k} = 0 \text{ and } \alpha_k = \frac{10}{40 + k}.$$

*Then, the algorithm outputs the following sequence for* $k = 2, \ldots, 9$:

$$x^k = \begin{pmatrix} -\frac{1}{2}\sum_{i=1}^{k-1}\alpha_i \\ \frac{1}{4} - \frac{1}{2}\sum_{i=1}^{k-1}\alpha_i \end{pmatrix}.$$

*In* $k = 10$, *the lower bound for* $x_1$ *is reached:*

$$x^{10} = \begin{pmatrix} -1 \\ \frac{1}{4} - \frac{1}{2}\sum_{i=1}^{9}\alpha_i \end{pmatrix}.$$

*Finally, for all* $k \geq 11$,

$$x^k = \begin{pmatrix} -1 \\ \left(\frac{1}{4} - \frac{1}{2}\sum_{i=1}^{9}\alpha_i\right)\left(\prod_{i=10}^{k-1}(1-2\alpha_i)\right) - \sum_{i=10}^{k-1} 2\alpha_i \left(\prod_{j=i+1}^{k-1}(1-2\alpha_j)\right) \end{pmatrix}.$$

*Such sequence* $\left\{x^k\right\}$ *converges (from above) to the optimal solution* $x^\star = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$. □

## 6.3 Convergence analysis

In order to give the main convergence result, some preliminary results must be outlined. But first, the nonsmooth value function $\phi_{\widehat{\mu},\widehat{\rho}}$, related to the Lagrangian function (6.3) and to Problem (6.6), must be introduced:

$$\phi_{\widehat{\mu},\widehat{\rho}}(x) \triangleq f(x) + \widehat{\mu}\|h(x)\|_1 + \frac{\widehat{\rho}}{2}\|h(x)\|^2. \tag{6.8}$$

Notice that $\phi$ relies on the same parameters $\widehat{\mu}$ and $\widehat{\rho}$ used in algorithm 19.

Propositions 1 and 2 underline some properties of $\phi_{\widehat{\mu},\widehat{\rho}}$.

**Proposition 1.** *Let $\widehat{\mu},\widehat{\rho}$ be two positive scalars. Then, $\phi_{\widehat{\mu},\widehat{\rho}}$ is convex and locally Lipschitz continuous.*

*Proof.* Since the sum of convex functions is convex, the proof follows by noticing that $f$ is convex by assumption, $\|h(x)\|_1 = \|Ax-b\|_1 = \sum_{i=1}^{m} |A_{i*}x - b_i| = \sum_{i=1}^{m} \max\{A_{i*}x - b_i, -A_{i*}x + b_i\}$ is convex, and $\|h(x)\|^2 = x^T A^T Ax - 2b^T Ax + b^T b$ is also convex. Local Lipschitz continuity follows by recalling that any convex function is locally Lipschitz continuous [28, Proposition 2.2.6]. □

Observe that the linearity of the constraints $h$ is necessary to prove that the nonsmooth function $\phi_{\widehat{\mu},\widehat{\rho}}$ is convex.

**Proposition 2.** *The subdifferential $\partial\phi_{\widehat{\mu},\widehat{\rho}}(x)$ of $\phi$ is non-empty and*

$$\nabla f(x) + \widehat{\mu}\sum_{i=1}^{m}\xi_i + \frac{\widehat{\rho}}{2}\nabla h(x)h(x) \in \partial\phi_{\widehat{\mu},\widehat{\rho}}(x),$$

*with*

$$\xi_i \in \begin{cases} \{A_{i*}^T\}, & \text{if } A_{i*}x - b_i > 0, \\ \{-A_{i*}^T\}, & \text{if } A_{i*}x - b_i < 0, \\ conv\{-A_{i*}^T, A_{i*}^T\}, & \text{if } A_{i*}x - b_i = 0. \end{cases}$$

*Proof.* From [28, Corollary 3], the subdifferential of the sum of convex functions equals the sum of the subdifferentials of the functions. Therefore, it holds

$$\partial\phi_{\widehat{\mu},\widehat{\rho}}(x) = \partial f(x) + \widehat{\mu}\partial\|h(x)\|_1 + \frac{\widehat{\rho}}{2}\partial\|h(x)\|^2,$$

where

$$\partial f(x) = \nabla f(x), \qquad \partial\|h(x)\|^2 = \nabla h(x)h(x) = 2A^T(Ax-b),$$

$$\partial\|h(x)\|_1 = \left(\sum_{i=1}^{m}\partial|A_{i*}x - b_i|\right) = \left(\sum_{i=1}^{m}\partial\max\{A_{i*}x - b_i, -A_{i*}x + b_i\}\right) \ni \sum_{i=1}^{m}\xi_i,$$

with

$$\xi_i \in \begin{cases} \{A_{i*}^T\}, & A_{i*}x - b_i > 0, \\ \{-A_{i*}^T\}, & A_{i*}x - b_i < 0, \\ conv\{-A_{i*}^T, A_{i*}^T\}, & A_{i*}x - b_i = 0, \end{cases}$$

where $conv\{x,y\}$ denotes the convex hull of the vectors $x, y \in \mathbb{R}^n$ (see e.g. [85, Exercise 8.31]). □

Now that $\phi$ has been introduced and its properties shown, proposition 3 states that any solution to the problem

$$\min_{x\in X} \phi_{\widehat{\mu},\widehat{\rho}}(x)$$

is an $\varepsilon$-approximate solution of problem (6.6), where $\varepsilon = \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$.

**Proposition 3.** *Given $\widehat{\mu} > 0$ and $\widehat{\rho} > 0$, let $\bar{x}$ be a minimum for $\phi_{\widehat{\mu},\widehat{\rho}}$ over $X$. Then the following holds:*

1. $f(\bar{x}) \leq f(x^\star)$,
2. $\|h(\bar{x})\| \leq \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$,

*where $x^\star$ is a solution of problem* (6.6)*, $f^{min} = \min_{x\in X} f(x)$ and $f^{max} = \max_{x\in X} f(x)$.*

*Proof.* To prove the first inequality, it suffices to write

$$f(\bar{x}) \leq f(\bar{x}) + \widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(x^\star) + \widehat{\mu}\|h(x^\star)\|_1 + \frac{\widehat{\rho}}{2}\|h(x^\star)\|^2 = f(x^\star),$$

since $\widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \geq 0$, $x^\star \in X$ and $h(x^\star) = 0$.

To prove the second assertion of this proposition, let $\widetilde{x} \in X$ be such that $h(\widetilde{x}) = 0$, i.e. $\widetilde{x}$ is a feasible point of problem (6.6). Then it holds that

$$f^{min} \leq f(\bar{x}) \quad \text{and} \quad f(\widetilde{x}) + \widehat{\mu}\|h(\widetilde{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\widetilde{x})\|^2 = f(\widetilde{x}) \leq f^{max}.$$

The two inequalities above yield

$$f^{min} + \widehat{\mu}\|h(\bar{x})\|_1 \leq f(\bar{x}) + \widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(\widetilde{x}) + \widehat{\mu}\|h(\widetilde{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\widetilde{x})\|^2 \leq f^{max},$$

which implies

$$\|h(\bar{x})\| \leq \|h(\bar{x})\|_1 \leq \frac{f^{max} - f^{min}}{\widehat{\mu}}.$$

On the other hand,

$$f^{min} + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(\bar{x}) + \widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(\widetilde{x}) + \widehat{\mu}\|h(\widetilde{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\widetilde{x})\|^2 = f(\widetilde{x}) \leq f^{max},$$

from which

$$\|h(\bar{x})\|^2 \leq \frac{f^{max} - f^{min}}{\widehat{\rho}},$$

which completes the proof. □

It easily follows that a minimal point of $\phi_{\widehat{\mu},\widehat{\rho}}$ that is feasible for problem (6.6) is a solution of problem (6.6), as shown by the following corollary.

**Corollary 1.** *Let the same setting of proposition 3 applies. Then, if $\|h(\bar{x})\| = 0$, $\bar{x}$ is an optimal solution of problem* (6.6).

*Proof.* From proposition 3, it holds that

$$f(\bar{x}) \leq f(x^*),$$

where $x^*$ is a solution of problem (6.6). By the definition of minimal point, it must hold

$$f(\bar{x}) = f(x^*),$$

that is, since $h(\bar{x}) = 0$ by hypothesis, $\bar{x}$ is also a solution. □

One may question whether the viceversa of proposition 3 holds or not. The following counterexample proves that this is not true in general.

**Example 4.** *Consider the problem*

$$\min_{x \in [0,2]:\ x=1} x^2,$$

*which trivial solution is $x^\star = 1$. By definition, this is an $\varepsilon$-approximate solution for any $\varepsilon > 0$.*

*Fixing $\widehat{\mu} = 1$ and $\widehat{\rho} = 0$, the corresponding $\phi_{\widehat{\mu},\widehat{\rho}}$ is*

$$\phi_{1,0}(x) = x^2 + |x-1|,$$

*which unique solution over $[0,2]$ is $\bar{x} = \frac{1}{2}$, that is different from $x^\star$.*

*A final remark is that, as $\widehat{\mu}$ grows to infinity, the minimizer of $\phi_{\widehat{\mu},\widehat{\rho}}$ goes to the minimizer of $|x-1|$, which is the solution of the original problem.* □

Proposition 3 shows that a minimizer of $\phi_{\widehat{\mu},\widehat{\rho}}$ is an $\varepsilon$-approximate solution of Problem (6.6). Specifically, the bigger the parameters $\widehat{\mu}$ and $\widehat{\rho}$, the smaller the value of $\varepsilon$, the better the approximate solution. The next proposition shows that if one of the parameters goes to infinity, then $\varepsilon$ goes to zero.

**Proposition 4.** *Let $\{\bar{x}^k\}$ be a sequence of minimizers of $\phi_{\widehat{\mu}^k,\widehat{\rho}^k}(x)$ over $X$. If either $\lim_{k\to\infty} \widehat{\mu}^k = \infty$ or $\lim_{k\to\infty} \widehat{\rho}^k = \infty$, then any accumulation point $\bar{x}$ of the sequence $\{\bar{x}^k\}$ is a solution of problem* (6.6).

*Proof.* Assume, without loss of generality, that the first limit applies. By proposition 3, it holds for any $k$ that

$$\bar{x}^k \in X, \quad f(\bar{x}^k) \leq f(x^*), \quad \|h(\bar{x}^k)\| \leq \frac{f^{max} - f^{min}}{\widehat{\mu}^k}.$$

By the continuity of $\|h(\bar{x}^k)\|$,

$$\|h(\bar{x})\| = \lim_{k\to\infty} \|h(\bar{x}^k)\| \leq \lim_{k\to\infty} \frac{f^{max} - f^{min}}{\widehat{\mu}^k} = 0$$

and therefore $h(\bar{x}) = 0$. Similarly, by recalling that $x^\star$ is a solution of problem (6.6), $f(\bar{x}) \leq f(x^\star)$. □

In theorem 37 algorithm 19 is proven to converge in the worst case to an $\varepsilon$-approximate solution of Problem (6.6), i.e. a point $\bar{x}$ such that

$$\bar{x} \in X,\ \|h(\bar{x})\| \leq \varepsilon, \qquad f(\bar{x}) \leq f(x) \quad \forall\, x \in S.$$

On the other hand, if $\widehat{\rho}$ is large enough the algorithm provably converges to a solution $x^\star$ of problem (6.6), that is a 0-approximate solution. Observe that $f(\bar{x}) \leq f(x^\star)$, namely the value of the objective function at $\bar{x}$ is a lower bound of the optimal value. Furthermore, it can be shown that the value of $\varepsilon$ can be controlled by suitably choosing the parameters $\widehat{\mu}$ and $\widehat{\rho}$.

**Theorem 37.** *Let $\{(x^k, \mu^k, \rho_k)\}$ be the sequence generated by algorithm 19 and $(\bar{x}, \overline{\mu}, \overline{\rho})$ be any of its limit points. Then there are two cases:*

1. *if $\overline{\rho} < \widehat{\rho}$, then $\bar{x}$ is a solution of problem* (6.6)*;*
2. *otherwise, $\overline{\rho} = \widehat{\rho}$ and $\bar{x}$ is an $\varepsilon$-approximate solution of problem* (6.6) *with* $\varepsilon = \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$.

*Proof.* The proof starts with showing that if $\overline{\rho} < \widehat{\rho}$ then $\bar{x}$ is a solution of problem (6.6).
Recalling step 10 of algorithm 19, if $\overline{\rho} < \widehat{\rho}$ then a $\overline{k} \geq 0$ exists such that for all $k \geq \overline{k}$

$$\|h(x^{k+1})\| \leq \tau \|h(x^k)\| < \|h(x^k)\|, \tag{6.9}$$

by recalling that $\tau \in (0,1)$. This yields, together with the positiveness of the sequence $\{\|h(x^k)\|\}$, that

$$\lim_{k\to\infty} \|h(x^k)\| = \overline{h} \geq 0.$$

Taking the limit for $k \to \infty$ in (6.9), one gets

$$\overline{h} \leq \tau \overline{h},$$

which, since $\tau \in (0,1)$, is true only if $\overline{h} = 0$. Therefore, by the continuity of $h$, it holds that

$$h(\bar{x}) = 0. \tag{6.10}$$

Moreover, recalling a well-known property of the projection operator, it holds for all $z \in \mathbb{R}^n$

$$(y - \mathscr{P}_X[z])^T (z - \mathscr{P}_X[z]) \leq 0 \quad \forall y \in X.$$

Therefore

$$\left(x^k - \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)]\right)^T \left(x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k) - \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)]\right) \leq 0,$$

and by expanding the products

$$\left\|x^k - \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)]\right\|^2 + \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)^T \left(\mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)] - x^k\right) \leq 0,$$

from which, by recalling that $x^{k+1} = \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)]$, it follows

$$\nabla_x \mathscr{L}_{\overline{\rho}}(x^k, \mu^k)^T \left(x^{k+1} - x^k\right) \leq -\frac{1}{\alpha_k} \left\|x^{k+1} - x^k\right\|^2. \tag{6.11}$$

The descent lemma [14, Lemma 2.1] and (6.11) therefore yield

$$\begin{aligned} \mathscr{L}_{\overline{\rho}}(x^{k+1}, \mu^k) &\leq \mathscr{L}_{\overline{\rho}}(x^k, \mu^k) + \nabla_x \mathscr{L}_{\rho_k}(x^k, \mu^k)^T (x^{k+1} - x^k) + \frac{L^k_{\mathscr{L}}}{2} \|x^{k+1} - x^k\|^2 \leq \\ &\leq -\left(\frac{1}{\alpha_k} - \frac{L^k_{\mathscr{L}}}{2}\right) \|x^{k+1} - x^k\|^2 \end{aligned}$$

and consequently

$$f(x^{k+1}) - f(x^k) + \mu^{k^T} (h(x^{k+1}) - h(x^k)) + \frac{\overline{\rho}}{2} (\|h(x^{k+1})\|^2 - \|h(x^k)\|^2) \leq -\frac{\overline{L} + \overline{\rho}\|A\|^2}{2} \|x^{k+1} - x^k\|^2.$$

By taking the limit for $k \to \infty$ of the above inequality,

$$\lim_{k\to\infty}(f(x^{k+1})-f(x^k))+\left(\lim_{k\to\infty}\mu^k\right)^T(\overline{h}-\overline{h})+\frac{\overline{\rho}}{2}(\overline{h}^2-\overline{h}^2)\leq-\frac{\overline{L}+\overline{\rho}\|A\|^2}{2}\lim_{k\to\infty}\|x^{k+1}-x^k\|^2,$$

from which

$$\lim_{k\to\infty}\left(f(x^{k+1})-f(x^k)\right)\leq 0.$$

This yields

$$\lim_{k\to\infty}f(x^k)=f(\overline{x})=\bar{f}, \tag{6.12}$$

because $f$ is continuous and bounded from below over $X$. It follows that

$$\lim_{k\to\infty}\|x^{k+1}-x^k\|^2=0.$$

Focusing on step 3 of the algorithm, by the limit above and since $\alpha_k=\overline{\alpha}=\frac{1}{\overline{L}+\overline{\rho}\|A\|^2}$ for all $k\geq\overline{k}$, this yields

$$\nabla_x\mathscr{L}_{\overline{\rho}}(\overline{x},\overline{\mu})^T(x-\overline{x})\geq 0 \qquad \forall\, x\ \in\ X. \tag{6.13}$$

Suppose by contradiction that $\bar{f}$ is greater than the optimal value of Problem (6.6). Therefore a point $\widetilde{x}\in X$ exists such that $h(\widetilde{x})=0$ and $f(\widetilde{x})<\overline{f}=f(\overline{x})$. The following can be written

$$0\leq\left(\nabla f(\overline{x})+A^T\overline{\mu}\right)^T(\widetilde{x}-\overline{x})=\nabla f(\overline{x})^T(\widetilde{x}-\overline{x})\leq f(\widetilde{x})-f(\overline{x})<0,$$

where the first inequality comes from (6.13), the equality is a consequence of the fact that $h(\widetilde{x})=0$ and (6.10), the second inequality is due to the convexity of $f$ and the last inequality comes from the hypotheses that $\overline{x}$ is not optimal and $\widetilde{x}$ is optimal. But this is impossible. Finally, $\overline{x}$ is a solution of problem (6.6) and this completes the proof of the first part.

To prove the second assertion, i.e. if $\overline{\rho}=\widehat{\rho}$ then $\overline{x}$ is an $\varepsilon$-approximate solution of Problem (6.6) with $\varepsilon=\min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}},\sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$, the first step is to prove that the updates at steps 3 and 8 of Algorithm 19 are eventually equivalent to employ a gradient projection method to minimize the nonsmooth function $\phi_{\widehat{\mu},\widehat{\rho}}(x)$, defined in (6.8), over $X$. Proposition 2 showed that the subdifferential of $\phi$ satisfies for all $x\ \in\ X$

$$\nabla f(x)+\widehat{\mu}\sum_{i=1}^m\xi_i+\frac{\widehat{\rho}}{2}\nabla h(x)h(x)\in\partial\phi_{\widehat{\mu},\widehat{\rho}}(x),$$

with

$$\xi_i\ \in\ \begin{cases}\{A_{i*}^T\}, & \text{if } A_{i*}x-b_i>0,\\ \{-A_{i*}^T\}, & \text{if } A_{i*}x-b_i<0,\\ conv\{-A_{i*}^T,A_{i*}^T\}, & \text{if } A_{i*}x-b_i=0.\end{cases}$$

The gradient of $\mathscr{L}_{\widehat{\rho}}(x^k,\mu^k)$ with respect to $x$ is

$$\nabla\mathscr{L}_{\widehat{\rho}}(x^k,\mu^k)=\nabla f(x^k)+A^T\mu^k+\frac{\widehat{\rho}}{2}\nabla h(x)h(x),$$

where, recalling the update rule of $\mu^k$ at step 8 of Algorithm 19, $A^T\mu^k$ can be rewritten as

$$A^T\mu^k=\sum_{i=1}^m\mu_i^kA_{i*}=\widehat{\mu}\left(\sum_{i:h_i(x^k)\geq 0}A_{i*}^T+\sum_{i:h_i(x^k)<0}-A_{i*}^T\right)=\widehat{\mu}\sum_{i=1}^m\psi_i,$$

where

$$\psi_i = \begin{cases} A_{i*}^T, & \text{if } A_{i*}x - b_i \geq 0, \\ -A_{i*}^T, & \text{if } A_{i*}x - b_i < 0, \end{cases}$$

which yields

$$\nabla_x \mathscr{L}_{\widehat{\rho}}(x^k, \mu^k) \subseteq \partial_x \phi_{\widehat{\mu},\widehat{\rho}}(x^k) \qquad \forall\, k \geq \widehat{k}.$$

This proves that step 3 of algorithm 19 is eventually equivalent to employ a gradient projection method to minimize the nonsmooth function $\phi_{\widehat{\mu},\widehat{\rho}}(x)$, which is a convex and Lipschitz-continuous function by proposition 1. This implies that (see e.g. [13, Theorem 3.2.6]) $\bar{x}$ is the optimal solution of the problem

$$\min_{x \in X} \phi_{\widehat{\mu},\widehat{\rho}}(x).$$

The proof of the second assertion, and therefore of the theorem, follows by proposition 3. □

The following facts must be remarked:

(i) the updating rule of the stepsize $\alpha_k$ defined in step 2 of algorithm 19 can be modified, as long as it satisfies (6.7) when $\rho_k = \widehat{\rho}$, namely it must be squared summable, but not summable;

(ii) as long as $\rho_k < \widehat{\rho}$, the updating rule of the multipliers at step 5 does not play any role and therefore any bounded $\mu_k$ is acceptable; this can be useful in the first phase of algorithm 19 where different updating rules for $\mu_k$ can be developed;

(iii) algorithm 19, with $\widehat{\mu} = 0$, is a modified version of a sequential penalty algorithm and therefore the above theoretical analysis can be directly applied to such framework.

## 6.4 Distributed implementation

In the following, a possible distributed implementation of algorithm 19 will be introduced. In particular, the quadratic case will be considered, i.e. where the objective function can be written as

$$f(x) = \frac{1}{2} x^T Q x + c^T x.$$

The aim is to show how algorithm 19 can easily distribute the computation and, in the quadratic case, the data to the available processes. Indeed, each process $\nu = 1, \ldots, N$ is expected to

- update only its assigned block of variables $x_{(\nu)} \in \mathbb{R}^{n_\nu}$,
- employ only its assigned block of data, namely the columns $Q_{*(\nu)} \in \mathbb{R}^{n \times n_\nu}$ of the matrix $Q \in \mathbb{R}^{n \times n}$, $c_{(\nu)}$, $X_\nu$ and $A_{*(\nu)}$,
- exchange only vectors (never matrices) with the other processes and converge to a ($\varepsilon$-approximate) solution of problem (6.6).

Observe that, given the quadratic assumption, the objective function $f$ can be rewritten as

$$\nabla f(x) = \left(\nabla f(x)_{(\nu)}\right)_{\nu=1}^{N}.$$

Algorithm 20 meets all the above expectations.

**Algorithm 20:** Gradient projection augmented Lagrangian method - distributed implementation

**Data:** $\widehat{\rho} > 0$, $\widehat{\mu} > 0$, $x^0 \in X$, $\mu^0 \in [-\widehat{\mu}, \widehat{\mu}]^m$, $\rho_0 \in (0, \widehat{\rho})$, $\gamma > 0$, $\delta > 0$, $\tau \in (0,1)$, $\widehat{k} = 0$

1 **for** $k = 0, 1, \ldots$ **do**

2 $\quad \alpha_k = \frac{1}{\overline{L} + \rho_k \|A\|^2 + \gamma(k - \widehat{k})}$

3 $\quad h(x^k) = \sum_{\xi=1}^{N} h^{\xi,k} - b$

4 $\quad$ (where all the $h^{\xi,k} = A_{*(\xi)}\, x^k_{(\xi)}$ come from the communication phase at step 12)

5 $\quad \mu_i^{k+1} = \begin{cases} -\widehat{\mu}, & i \,:\, h_i(x^{k+1}) < 0 \\ \widehat{\mu}, & otherwise \end{cases}, \quad i = 1, \ldots, m$

6 $\quad \rho_k = \begin{cases} \min\{\rho_{k-1} + \delta,\ \widehat{\rho}\}, & \|h(x^k)\| > \tau \|h(x^{k-1})\| \\ \rho_k, & otherwise \end{cases}$

7 $\quad \nabla \mathscr{L}_{\rho_k}(x^k, \mu^k)_{(\nu)} = \sum_{\xi=1}^{N} a^{\xi,k}_{(\nu)} + c_{(\nu)} + A^T_{*(\nu)} \mu^k + \rho_k A^T_{*(\nu)} h(x^k)$

8 $\quad$ (where all the $a^{\xi,k} = Q_{*(\xi)}\, x^k_{(\xi)}$ come from the communication phase at step 11)

9 $\quad x^{k+1}_{(\nu)} = \mathscr{P}_{X_\nu}\left[ x^k_{(\nu)} - \alpha_k \nabla \mathscr{L}_{\rho_k}(x^k, \mu^k)_{(\nu)} \right]$

10 $\quad \widehat{k} = \begin{cases} k+1, & \rho_k < \widehat{\rho} \\ \widehat{k}, & otherwise \end{cases}$

11 $\quad a^{\nu,k+1} = Q_{*(\nu)}\, x^{k+1}_{(\nu)}$ and broadcast

12 $\quad h^{\nu,k+1} = A_{*(\nu)}\, x^{k+1}_{(\nu)}$ and broadcast

13 **end**

Observe that algorithm 20 exchanges only vectors between the processes, in particular $a^{\nu,k+1}$ and $h^{\nu,k+1}$ at steps 11 and 12, which is not a heavy computational overhead if the number of processes $N$ is not too big. Then, every process $\nu$ works only on its block of variables $x_{(\nu)}$ and needs only to access its data block $Q_{*(\nu)}$, $A_{*(\nu)}$, $c_{(\nu)}$, $X_\nu$ and $b$. Data partitioning can be very beneficial from a hardware point of view, and sometimes is requested by the application, like e.g. support vector machines.

At step 2, each process $\nu$ computes the stepsize $\alpha_k > 0$ such that

(i) it is squared summable but not summable for all $k$ such that $\rho_k = \widehat{\rho}$;

(ii) it satisfies $\alpha_k = \frac{1}{L^k_{\mathscr{L}}}$ for all $k :\ \rho_k < \widehat{\rho}$.

At step 5 $\mu^k$ is updated in order for $\mu^{k+1}$ to be the maximum of $\mathscr{L}_{\rho_k}(x^k, \mu)$ over $M_{\widehat{\mu}}$ with respect to $\mu$. At step 6 the penalty parameter is increased if a sufficient descent in the violation of the coupling constraints defined by $h$ is not achieved. Finally, at steps 7-9 every process $\nu$ updates its block of variables $x_{(\nu)}$ by a gradient projection step over $X_\nu$. Recall that such update requires only the computation of a gradient, and not the (approximate) solution of any subproblem over $X_\nu$, which instead is required in algorithm 17.

## 6.5 Numerical experiments

Some preliminary tests employing the proposed distributed implementation are reported in this section.

The quadratic case is considered, namely

$$\min_{x\in[-10,10]^n : Ax=0} x^t Qx + c^T x,$$

where $n = 1000$, $m = 100$, $Q$ is a positive definite matrix and $\|Q\| = 1$ (i.e. $\overline{L} = 1$), $c$ is the vector of all ones, $\|A\| = 1$, $b = 0$, $X = [-10,10]^n$.

Algorithm 20 is applied with $\gamma = 1$, $\delta = 0.5$, $\tau = 0.9$, $x^0 = 0$, $\mu^0 = 0$, $\rho_0 = 1$.

To experiment the impact of the main two parameters $\widehat{\mu}$ and $\widehat{\rho}$, these were set to two different values each. This resulted in four experiments, as reported in table 6.1.

| **Experiment** | $\widehat{\mu}$ | $\widehat{\rho}$ |
|---|---|---|
| exp1 | 1e-1 | 1e3 |
| exp2 | 1e1 | 1e3 |
| exp3 | 1e-1 | 1e4 |
| exp4 | 1e1 | 1e4 |

**Table 6.1.** Experimental setup for algorithm 20

The tests were run on a PC Windows with CPU Intel Core i7-8650U - 4 cores (base frequency 1.9 GHz, turboboost up to 4.2 GHz) and RAM 16 GB, Python 3.6.3, and MPI 3.0.

Figures 6.2-6.5 report, for all the 4 experiments: in solid line the objective relative error (i.e. $\max\{1e\text{-}4, (f(x^k) - f^*)/f^*\}$, where $f^*$ is the optimal value of the problem), and in dashed line the coupling constraints violation (i.e. $\|h(x^k)\|/\sqrt{n}$), versus iterations (the first 120k iterations are reported).

In all the experiments the algorithm returns an $\varepsilon$-approximate solution of the problem, i.e. a point $\bar{x} \in X$ such that $f(\bar{x}) \leq f(x^*)$ and $\|h(\bar{x})\| \leq \varepsilon$. As theoretically observed in theorem 37, figures 6.2-6.5 confirm that the larger $\widehat{\mu}$ and $\widehat{\rho}$, the smaller $\varepsilon$.

Furthermore, it is clear by the results that algorithm 20 has a slower convergence rate when $\rho_k \geq \widehat{\rho}$. Indeed, exp1 and exp2 present a slower convergence with respect to exp3 and exp4, and this seems to depend on the fact that $\rho_k$ reaches its upper bound $\widehat{\rho}$ earlier in the former two (around iteration 2k), later in the latter two (around iteration 20k). This suggests that reasonable guidelines for the parameters $\widehat{\rho}$, $\delta$, and $\tau$ should be a big enough $\widehat{\rho}$, a small $\delta$ and a close-to-1 $\tau$, in order to grand a large amount of iterations in phase 1 (i.e. when $\rho_k < \widehat{\rho}$). The choice of $\widehat{\mu}$ is even more critical: in fact, although theorem 37 states that large values of $\widehat{\mu}$ yield smaller values of the approximation $\varepsilon$, if such value is too big then the joint convergence of the objective function and the constraints violation may be negatively affected. Indeed, the performance deterioration can be seen in exp3 and exp4, where around iteration 20k (i.e. when switching from phase 1 to phase 2) the dashed line presents a discontinuity.

The algorithm was tested with $N = 1$, 2, and 4 parallel processes, with every process updating $1000/N$ variables. The speedup in terms of CPU time was almost ideal, namely the execution of 200k iterations took around 700 seconds with $N = 1$, around 360 seconds with $N = 2$, and around 190 seconds with $N = 4$, for all the experiments.

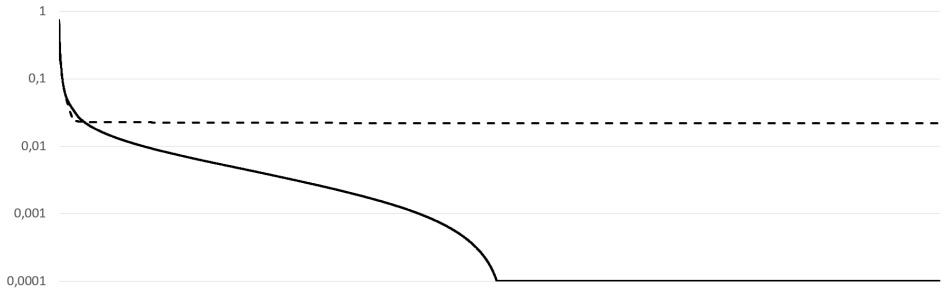


**Figure 6.2.** Exp1


**Figure 6.3.** Exp2

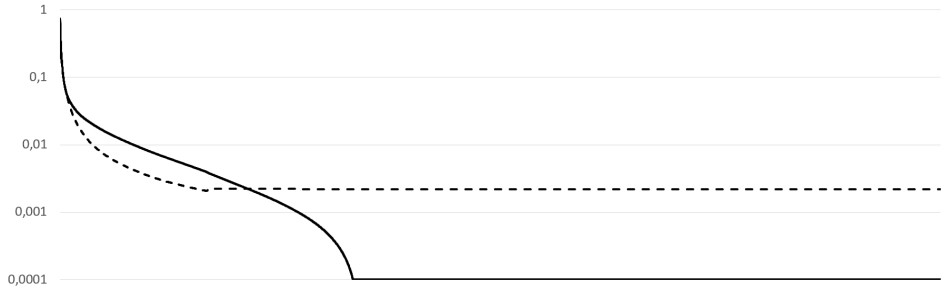


**Figure 6.4.** Exp3


**Figure 6.5.** Exp4

# Chapter 7

# DNN and SVM to detect postural diseases

This project was conducted in collaboration with prof. Laura Palagi and the Department of Physical Medicine and Rehabilitation (PMR), which was responsible for the acquisition of data (i.e. the patient selection, the postural evaluation, the scoliosis/healthy diagnosis and the acquisition by the rasterstereography). The aim of the project was to develop automatic procedures to detect adolescent idiopathic scoliosis (AIS) based on data extracted by a rasterstereographic machine and fed to machine learning models. The results of this work are published in the technical report [30] and will be submitted to journal.

The classifiers employed in the project were implemented in Python, by using ScikitLearn [41] library for Machine Learning, which employs many (and variations) of the SG/ IG methods from chapter 3. Indeed, to the best of author's knowledge, such methods are the core of any library for Machine Learning.

## 7.1 Introduction

Adolescent idiopathic scoliosis (AIS) is a three-dimensional deformity of the spine, which is characterized by deformation of spinal curvatures on the sagittal, frontal and transverse plane. The diagnosis of AIS is made by X-rays, that allow to detect vertebral rotation and to compute Cobb angle, needed for AIS classification. X-rays, however, carry health risk from repetitive exposure to ionizing radiation [38] and cannot aid physician to detect postural changes associated to AIS. Postural assessment with the study of the "static" standing posture represents a relevant issue in routinely practice of physicians involved in the management of back diseases, especially those involving children and adolescents, in whom a particular attention should be payed that the developing body is growing up correctly. Nowadays physicians usually perform postural evaluation on the basis of a clinical examination, mainly supported by their own experience, aimed to detect deformities in bending, by means of the Adams test, as well asymmetries between the two sides of the body. The main difficulty, here, is to define which postural parameters need to be considered for diagnosis, and the border line between normal and pathologic features.

Recently, rasterstereography has been proposed as an objective method for instrumented three-dimensional (3D) back shape analysis and reconstruction of spinal curvatures and deformities without radiation exposure [77, 39]. Rasterstereography is based on stereophotogrammetric surface measuring of the back, and it provides more than one-hundred different quantitative parameters concerning 3D subject's posture with a single exam. The main problem with the application of rasterstereography to clinical practice, as well as, for example, to its use on AIS screening, is represented by the lack

of a codified system to analyze and to interpret the whole amount of parameters derived from any single acquisition. No ranges of normality still exist for many of rasterstereography parameters, often resulting in a subjective interpretation of objective data.
Indeed, the analysis of data either from a single patient or from a population of subjects is one of the most critical issues in modern medicine. Despite technological advances, too much data could be difficult to understand and could slow down the diagnostic and therapeutic approach, potentially causing unpleasant consequences for patients and operators.

Data Mining (DM) and more specifically Machine Learning (ML) techniques have obtained much interest in medicine field to obtain relevant information from different medical data sets. The use of these techniques in medical areas are changing the way to approach to the patients, because they could simplify and make clinical processes faster [1, 78]. The hypothesis is that DM techniques could be useful in the field of postural analysis, due to the above mentioned difficulties in finding patterns of normality, and that they could be applied to parameters objectively derived from rasterstereography. Particularly, the aim is to determine specific datasets of a limited number of features able to assist physicians in distinguishing between AIS subjects and healthy ones, only on the basis of rasterstereographic measurements.
For these reasons, the objective of the present study has been to apply unsupervised and supervised ML techniques to automatically distinguish AIS from healthy subjects, using a subset of rasterstereography parameters that can be identified as those that bring most of the information.

The first step of the project was therefore to adopt an unsupervised strategy (i.e. clustering) to check how good is the information tied to the only features without driving the classification by the known labels of the subjects. In a second phase, the same data were fed to supervised models to see how supervision would improve results. Unsupervised learning or clustering consists in detecting if samples can be split into groups, i.e. the clusters, which possess some similarities in a defined metric. In this project, the number of clusters is inherently defined by the disease, indeed scoliotic and healthy subjects form the two groups, and the interest is in determining if the two groups are identifiable by an unsupervised model, or if supervision is needed. The most common algorithm to perform clustering is K-means [57], where the number of groups $K$ one wants to identify must be given in input. For supervised learning, two standard models were selected, namely support vector machines (SVM) [25, 81] and deep networks (DN) [48]. Notice that SVM has been already used for detecting postural diseases tied to scoliosis in [1] with a different settings of data. DN, instead, have been mainly used in the medical field for image recognition to identify damaged vertebrae in the spine (see e.g. [26, 45]). Besides obtaining a good classifier to detect healthy vs scoliotic subjects, the aim of this study lies also in the comparison between SVM and DN on this task. Feature selection techniques were also used, indeed different feature selection methods were combined in a unified strategy. The features selected were then validated by the physicians and are not trivially obtainable by medical observations.

The rest of the chapter is organized as follows: in section 7.2 the acquisition of data and the first cleaning based on physician observations are described. Furthermore, the features extraction procedure, classification models and performance measures considered are introduced. In section 7.3 the results and presented and, finally, in section 7.4 some comments and concluding remarks are reported.

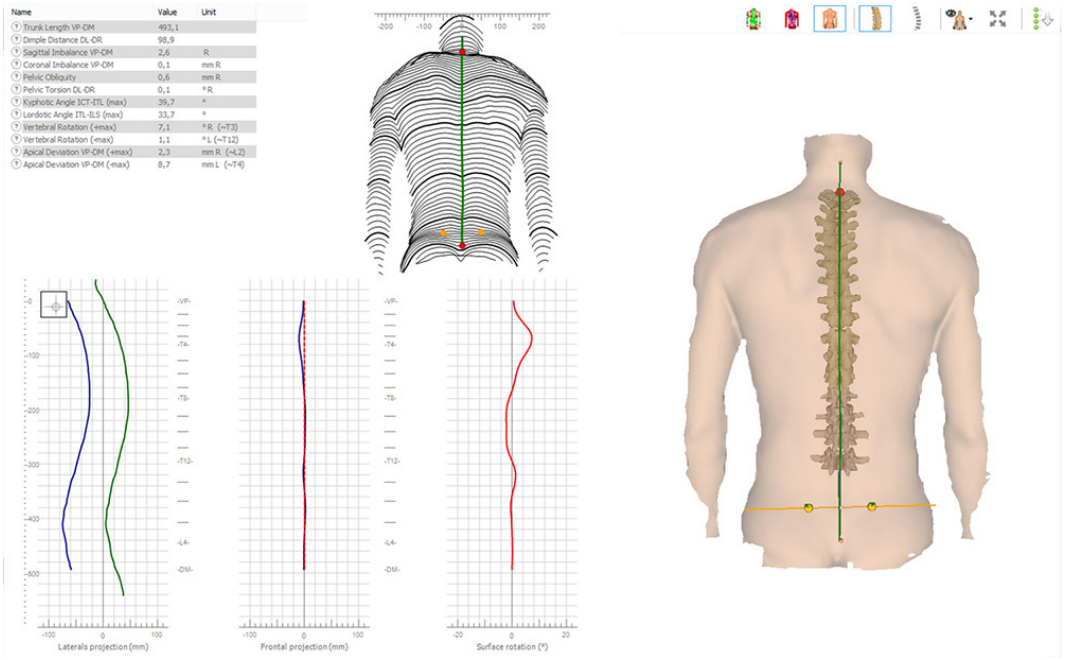


**Figure 7.1.** Formetric's output representation (from https://diers.eu)

## 7.2 Data analysis and preprocessing

### 7.2.1 Rasterstereography acquisition of data

The acquisition of data was performed through rasterstereography by the Formetric™4D system, reported in figure 7.1. Briefly, parallel light lines are projected onto the back surface of undressed patients. The three-dimensional back shape leads to a deformation of the parallel light lines, which can be detected by a camera positioned at a different angle from the projector (triangulation system). Using a standardized mathematical analysis, the following specific landmarks are automatically determined by assigning concave and convex areas to the curved light pattern:

(i) the spinous process of 7th cervical vertebra (Vertebra Prominens – VP);

(ii) the spinous process of 12th thoracic vertebra (Th12);

(iii) the midpoint between the lumbar dimples;

(iv) the cervical-thoracic inflexion point (ICT);

(v) the thoracic-lumbar inflexion point (ITL);

(vi) and the lumbar-sacral inflection point (ILS).

The patient is asked to stand still in an upright posture at a fixed distance from the camera for 6 seconds, during which a total number of 12 scans are performed. The mean value of the 12 measures is reported as output. Based on these landmarks, a three-dimensional model of the whole spine, the sagittal profile, and shape parameters describing this profile are generated. The accuracy of such measures and Formetric™functioning can be found in [79, 61]. Derived parameters from automatic

landmarks are, among the others, thoracic kyphosis angle, lumbar lordosis angle, fléche lumbaire, fléche cervicale and kyphotic apex as described by Stagnara.

For each patient, the total number of rasterstereography features calculated by Formetric™is 40. The features are all numerical and the full list is reported in Table 7.1 together with the units of measure.

| Feature | Unit of Measure | Feature | Unit of Measure |
|---|---|---|---|
| Trunk length_VP-DM | mm | Fléche lombaire_(Stagnara) | mm |
| Trunk length_VP-SP | mm | Kyphosis angle_ICT-ITL | degree |
| Trunk length_VP-SP | % | Kyphosis angle_VP-ITL | degree |
| Dimple distance-DR | mm | Kyphosis angle_VP-T12 | degree |
| Dimple distance_DL-DR | % | Lordotic angle_ITL-ILS_(max) | degree |
| Trunk inclination_VP-DM | degree | Lordotic angle_ITL-DM | degree |
| Trunk inclination_VP-DM | mm | Lordotic angle_T12-DM | degree |
| Lateral_flexion_VP-DM | degree | Pelvic inclination | degree |
| Lateral_flexion_VP-DM | mm | Surface rotation_(rms) | degree |
| Pelvic obliquity_DL-DR | degree | Surface rotation_(max) | degree |
| Pelvic obliquity_DL-DR | mm | Surface rotation_(+max) | degree |
| Pelvic torsion_DL-DR | degree | Surface rotation_(-max) | degree |
| Pelvic inclination_(dimple) | degree | Surface rotation_(width) | degree |
| Pelvis rotation | degree | Pelvic torsion | degree |
| Inflexion point_ICT | mm | Lateral deviation_VPDM_(rms) | mm |
| Kypothic apex_KA_(VPDM) | mm | Lateral deviation_VPDM_(max) | mm |
| Inflexion point_ITL | mm | Lateral deviation_VPDM_(+max) | mm |
| Lordotic apex_LA_(VPDM) | mm | Lateral deviation_VPDM_(-max) | mm |
| Inflexion point_ILS | mm | Lateral deviation_(width) | mm |
| Fléche cervicale_(Stagnara) | mm | Pain_index_(Dr_Weiss)_rel | number |

**Table 7.1.** The full list of Formetric™features

### 7.2.2 Data preprocessing

The rasterstereographic collection of data was conducted for clinical purposes in the Department of PMR of Sapienza during the period January $1^{st}$, 2010 – December $31^{st}$, 2016. Each sample composing the initial database represents the rasterstereography record of one subject, selected according to the following inclusion criteria: (i) male or female and (ii) age between 14 and 30. The following subjects were excluded: (i) subject with a clinical history of congenital/acquired pathologic condition of vertebrae (e.g. Scheuermann's disease, spondylolysis, spondylolisthesis); (ii) subjects with a history of vertebral fractures and/or vertebral surgery; (iii) subjects with a diagnosis of disc protrusion/hernia at any spinal level; (iv) subjects with a diagnosis of scoliosis secondary to neurologic, rheumatologic and/or congenital conditions; (v) subjects with a diagnosis of AIS with Cobb angle measured on X-rays $> 45$ degrees; (vi) subjects with a diagnosis of any neurologic and/or rheumatologic conditions. Once analyzed inclusion and exclusion criteria of patients screened for eligibility, a total of 298 subjects were enrolled. For each patient, Formetric™ returns more than one measure each representing a sample in the dataset. In particular patients enrolled with diagnosis of scoliosis were 272 ($\sim 90\%$ of total) for a total of 1111 Formetric™ samples. Healthy patients were 26 for a total of 194 Formetric™ samples. The number of samples of healthy/scoliotic is strongly imbalanced and this is a well-known cause of bias in the learning process [63]. To overcome drawbacks due to imbalance the Formetric™ measures of the scoliotic patients (that were about 4 for each subject) were averaged, obtaining 272

AIS averaged samples.

| | |
|---|---|
| Acquisition date | 2010 - 2016 |
| Number of distinct patients | 298 |
| Healthy/scoliosis ratio of patients | 0.1 |
| Number of samples after balancing | 466 |
| Number of healthy samples after balancing | 194 |
| Number of AIS samples after balancing | 272 |
| Healthy/scoliosis ratio in the target set | 0.7 |

**Table 7.2.** Summary of statistics on the dataset

Finally, the target set was obtained by merging samples for the two population of AIS (averaged) and healthy (not averaged) patients, so that a dataset of $m = 466$ samples was obtained, each sample represented by the 40 Formetric™ features and a label in $\{-1,1\}$ that corresponds to healthy/scioliotic status. The main statistics of the target set are summarized in table 7.2.
As mentioned in the introduction, before undergoing the learning phase, data must go through a cleaning and feature selection phase which usually reduces both the number of samples and the number of features which can be redundant with respect to the learning aim. Indeed learning machines performance are influenced both by the number of samples and the number of features of the target set used for training (see e.g. the surveys [24, 80]).

Before the automatic feature selection phase, which is described in section 7.2.3, data were briefly analyzed together with the physicians, for cleaning and scaling purpose. Indeed, the physicians recognized that some of the features obtained by the Formetric™ contained duplicate information, in the sense that they correspond to measures of the same quantity, expressed in different units (e.g. mm or degrees). Hence those duplicate measures were eliminated and the final number of distinct features were equal to 33. The eliminated features are reported in table 7.3.

After this basic feature reduction, a first run of classification using the tools described in section 7.2 was performed. Since the results obtained by unsupervised and supervised classification presented inconsistencies a deeper analysis led to the finding that there were features directly tied to trunk length, therefore related to the age of the patient. Those features had a dominant role in classification, thus resulting in poor generalization. Such features, which were eliminated, are reported in table 7.4. Thus the number of total features in the target set reduced further to 27.

Among the 27 remaining features, some still were somewhat indirectly dependent on trunk length: those features, reported in table 7.5, were normalized by dividing each value by the trunk length in mm, thus obtaining an adimensional value. In this way the target set was not biased by the age of the patients.

At the end of this process the target set was made up of $m = 467$ samples (referring to 299 patients), each characterized by $n = 27$ input features $x^i \in \mathbb{R}^{27}$, which are reported in table 7.6, and one output label $y^i \in \{-1,1\}$ which identifies healthy/scoliotic samples. The target set will be denoted by

$$\mathscr{T} = \{(x^i, y^i) \in \mathbb{R}^n \times \{-1,1\}, i = 1\ldots,m\}$$

### 7.2.3 Feature selection procedure

Before entering a feature selection phase using standard tools in machine learning, basic statistical analysis of the target set was applied to check if it is possible to identify a high degree of correlation

| Feature | Unit of Measure | Eliminated |
|---|---|---|
| Trunk inclination_VP-DM | degree | Y |
| Trunk inclination_VP-DM | mm | N |
| Lateral_flexion_VP-DM | degree | Y |
| Lateral_flexion_VP-DM | mm | N |
| Pelvic obliquity_DL-DR | degree | Y |
| Pelvic obliquity_DL-DR | mm | N |
| Kyphosis angle_ICT-ITL_(max) | degree | N |
| Kyphosis angle_VP-ITL | degree | Y |
| Kyphosis angle_VP-T12 | degree | Y |
| Lordotic angle_ITL-ILS_(max) | degree | N |
| Lordotic angle_ITL-DM | degree | Y |

**Table 7.3.** Duplicated features eliminated with physicians' support: 'Y'=eliminated, 'N'=mantained

| Feature | Unit of Measure |
|---|---|
| Trunk length_VP-DM | mm |
| Trunk length_VP-SP | mm |
| Trunk length_VP-SP | % |
| Dimple distance_DL-DR | mm |
| Dimple distance_DL-DR | % |

**Table 7.4.** Eliminated features since highly dependent on trunk length

| Feature | Unit of Measure |
|---|---|
| Inflexion point_ICT | mm |
| Kypothic apex_KA_(VPDM) | mm |
| Inflexion point_ITL | mm |
| Lordotic apex_LA_(VPDM) | mm |
| Inflexion point_ILS | mm |

**Table 7.5.** Features dependent on trunk length normalized by trunk length_VP-DM in mm

| Feature | Name | Unit of Measure |
|---|---|---|
| $x_1^i$ | Trunk inclination_VP-DM | mm |
| $x_2^i$ | Lateral_flexion_VP-DM | mm |
| $x_3^i$ | Pelvic obliquity_DL-DR | mm |
| $x_4^i$ | Pelvic torsion_DL-DR | degree |
| $x_5^i$ | Pelvic inclination_(dimple) | degree |
| $x_6^i$ | Pelvis rotation | degree |
| $x_7^i$ | Inflexion point_ICT/trunk length_VP-DM | adim |
| $x_8^i$ | Kypothic apex_KA_(VPDM)/trunk length_VP-DM | adim |
| $x_9^i$ | Inflexion point_ITL/trunk length_VP-DM | adim |
| $x_{10}^i$ | Lordotic apex_LA_(VPDM)/trunk length_VP-DM | adim |
| $x_{11}^i$ | Inflexion point_ILS/trunk length_VP-DM | adim |
| $x_{12}^i$ | Fléche cervicale_(Stagnara) | mm |
| $x_{13}^i$ | Fléche lombaire_(Stagnara) | mm |
| $x_{14}^i$ | Kyphosis angle_ICT-ITL | degree |
| $x_{15}^i$ | Lordotic angle_ITL-ILS_(max) | degree |
| $x_{16}^i$ | Pelvic inclination | degree |
| $x_{17}^i$ | Surface rotation_(rms) | degree |
| $x_{18}^i$ | Surface rotation_(max) | degree |
| $x_{19}^i$ | Surface rotation_(+max) | degree |
| $x_{20}^i$ | Surface rotation_(-max) | degree |
| $x_{21}^i$ | Surface rotation_(width) | degree |
| $x_{22}^i$ | Pelvic torsion | degree |
| $x_{23}^i$ | Lateral deviation_VPDM_(rms) | mm |
| $x_{24}^i$ | Lateral deviation_VPDM_(max) | mm |
| $x_{25}^i$ | Lateral deviation_VPDM_(+max) | mm |
| $x_{26}^i$ | Lateral deviation_VPDM_(-max) | mm |
| $x_{27}^i$ | Lateral deviation_(width) | mm |

**Table 7.6.** List of features constituting the row $x^i \in \mathbb{R}^{27}$ of the clean data set

either among pairs of input features or among input features and the output class. In particular, a Pearson test was performed on the target set $\mathscr{T}$. None of the features present a strong correlation with the output being the max score 0.59. However, some pairs of features are highly correlated with each other, i.e. they present a Pearson score greater than 0.8 (e.g $x_5$ and $x_{16}$ which represent different procedures to measure the pelvic inclination). The full Pearson matrix is reported in table 7.8, where variables with Pearson score greater than 0.8 are boldface in a green box. The identified relationships will be used in connection with the feature selection procedure in the next section 7.2.3. In table 7.7 a summary of max/min indices in the Pearson matrix is also reported.

| | features vs features | features vs output |
|---|---|---|
| Max abs correlation | 0.96 | 0.59 |
| Min abs correlation | 0.00 | 0.04 |
| Avg abs correlation | 0.19 | 0.26 |

**Table 7.7.** Summary of Pearson coefficients (absolute values)

| | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ | $x_{14}$ | $x_{15}$ | $x_{16}$ | $x_{17}$ | $x_{18}$ | $x_{19}$ | $x_{20}$ | $x_{21}$ | $x_{22}$ | $x_{23}$ | $x_{24}$ | $x_{25}$ | $x_{26}$ | $x_{27}$ | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | -0.10 | 0.10 | -0.02 | 0.18 | -0.03 | -0.26 | -0.41 | -0.06 | 0.06 | -0.03 | 0.33 | **-0.88** | -0.36 | -0.22 | 0.18 | 0.10 | -0.11 | -0.04 | -0.12 | 0.05 | -0.17 | 0.19 | 0.08 | 0.12 | -0.06 | 0.17 | 0.26 |
| $x_2$ | | -0.20 | -0.33 | -0.01 | 0.13 | -0.04 | 0.10 | 0.09 | 0.06 | 0.09 | -0.06 | 0.09 | -0.05 | -0.02 | -0.02 | -0.04 | 0.21 | 0.18 | 0.14 | 0.06 | 0.36 | -0.02 | -0.09 | -0.08 | -0.05 | -0.03 | 0.05 |
| $x_3$ | | | -0.14 | 0.05 | -0.04 | -0.06 | 0.05 | 0.04 | 0.05 | 0.04 | -0.04 | -0.09 | -0.07 | 0.02 | 0.05 | -0.01 | 0.01 | 0.06 | -0.03 | 0.07 | 0.13 | 0.09 | -0.00 | 0.03 | -0.11 | 0.10 | -0.04 |
| $x_4$ | | | | 0.02 | -0.16 | 0.02 | 0.00 | 0.01 | -0.01 | -0.04 | -0.06 | 0.00 | 0.00 | 0.05 | 0.03 | 0.18 | -0.14 | -0.06 | -0.16 | 0.07 | -0.13 | 0.09 | 0.09 | 0.10 | -0.02 | 0.10 | 0.11 |
| $x_5$ | | | | | -0.09 | -0.09 | -0.20 | -0.31 | -0.38 | -0.44 | -0.50 | -0.32 | -0.26 | 0.72 | **0.85** | 0.42 | -0.07 | 0.14 | -0.27 | 0.34 | -0.04 | 0.24 | 0.14 | 0.18 | -0.00 | 0.14 | 0.59 |
| $x_6$ | | | | | | 0.06 | 0.08 | 0.11 | 0.11 | 0.12 | 0.03 | 0.03 | 0.00 | -0.11 | -0.10 | -0.07 | 0.27 | 0.26 | 0.18 | 0.10 | 0.24 | -0.01 | 0.05 | 0.01 | 0.03 | -0.01 | -0.11 |
| $x_7$ | | | | | | | 0.31 | 0.19 | 0.18 | 0.15 | 0.07 | 0.02 | 0.11 | -0.02 | -0.03 | 0.04 | 0.24 | 0.25 | 0.09 | 0.16 | 0.15 | 0.08 | -0.02 | 0.08 | -0.14 | 0.15 | -0.21 |
| $x_8$ | | | | | | | | 0.74 | 0.73 | 0.70 | -0.21 | 0.35 | -0.09 | 0.03 | -0.10 | -0.12 | 0.13 | 0.06 | 0.10 | -0.03 | 0.10 | -0.04 | 0.09 | 0.03 | 0.11 | -0.07 | -0.38 |
| $x_9$ | | | | | | | | | **0.90** | **0.86** | 0.07 | 0.09 | -0.15 | -0.17 | -0.20 | -0.23 | 0.11 | -0.03 | 0.17 | -0.15 | -0.01 | -0.11 | 0.04 | -0.05 | 0.12 | -0.14 | -0.39 |
| $x_{10}$ | | | | | | | | | | **0.96** | 0.14 | -0.01 | -0.20 | -0.27 | -0.23 | -0.25 | 0.11 | -0.04 | 0.19 | -0.18 | -0.03 | -0.09 | 0.07 | -0.02 | 0.13 | -0.11 | -0.45 |
| $x_{11}$ | | | | | | | | | | | 0.19 | 0.11 | -0.14 | -0.29 | -0.30 | -0.34 | 0.14 | -0.05 | 0.26 | -0.25 | -0.03 | -0.17 | 0.03 | -0.10 | 0.16 | -0.20 | -0.47 |
| $x_{12}$ | | | | | | | | | | | | -0.25 | 0.14 | -0.65 | -0.51 | -0.35 | -0.06 | -0.23 | 0.16 | -0.34 | -0.10 | -0.21 | -0.08 | -0.16 | 0.03 | -0.13 | -0.19 |
| $x_{13}$ | | | | | | | | | | | | | 0.44 | 0.12 | -0.31 | -0.32 | 0.05 | -0.13 | 0.23 | -0.30 | 0.13 | -0.34 | -0.12 | -0.24 | 0.15 | -0.32 | -0.39 |
| $x_{14}$ | | | | | | | | | | | | | | 0.09 | -0.21 | -0.30 | 0.00 | -0.15 | 0.20 | -0.29 | -0.05 | -0.40 | -0.17 | -0.34 | 0.13 | -0.38 | -0.14 |
| $x_{15}$ | | | | | | | | | | | | | | | 0.79 | 0.23 | -0.02 | 0.09 | -0.11 | 0.16 | 0.01 | -0.01 | 0.01 | -0.04 | 0.07 | -0.11 | 0.37 |
| $x_{16}$ | | | | | | | | | | | | | | | | 0.33 | -0.04 | 0.13 | -0.18 | 0.26 | -0.00 | 0.16 | 0.12 | 0.11 | 0.05 | 0.05 | 0.48 |
| $x_{17}$ | | | | | | | | | | | | | | | | | -0.13 | 0.32 | -0.65 | 0.79 | -0.03 | 0.59 | 0.27 | 0.52 | -0.17 | 0.54 | 0.46 |
| $x_{18}$ | | | | | | | | | | | | | | | | | | 0.83 | 0.70 | 0.23 | 0.48 | -0.09 | -0.25 | -0.20 | -0.20 | -0.06 | -0.08 |
| $x_{19}$ | | | | | | | | | | | | | | | | | | | 0.31 | 0.69 | 0.49 | 0.24 | -0.07 | 0.12 | -0.29 | 0.27 | 0.18 |
| $x_{20}$ | | | | | | | | | | | | | | | | | | | | -0.48 | 0.32 | -0.49 | -0.39 | -0.55 | -0.01 | -0.47 | -0.31 |
| $x_{21}$ | | | | | | | | | | | | | | | | | | | | | 0.21 | 0.60 | 0.23 | 0.53 | -0.26 | 0.61 | 0.41 |
| $x_{22}$ | | | | | | | | | | | | | | | | | | | | | | 0.07 | -0.07 | 0.02 | -0.16 | 0.15 | -0.06 |
| $x_{23}$ | | | | | | | | | | | | | | | | | | | | | | | 0.30 | 0.72 | -0.34 | **0.84** | 0.26 |
| $x_{24}$ | | | | | | | | | | | | | | | | | | | | | | | | **0.80** | 0.67 | 0.22 | 0.08 |
| $x_{25}$ | | | | | | | | | | | | | | | | | | | | | | | | | 0.21 | 0.69 | 0.16 |
| $x_{26}$ | | | | | | | | | | | | | | | | | | | | | | | | | | -0.52 | -0.10 |
| $x_{27}$ | | | | | | | | | | | | | | | | | | | | | | | | | | | 0.20 |

**Table 7.8.** Pearson Correlation matrix among features: in a green box the most correlated ones

A critical aspect for the success of any learning procedure stays in the reduction, if needed, of the number of input features. Indeed it may happen that some features are redundant and/or add noisy information, so that eliminating them will help the learning task. Furthermore, the feature selection can give insights on which features are the ones that hold the most significant information and hence can give doctors indications about the key measures of Formetric 4D. To this aim, a feature selection phase was performed before entering the true learning phase. In the literature different methods for feature selection have been proposed. Four different algorithms, described below, were employed in this work, with the aim of defining a ranking of the features by assigning "votes" based on the number of algorithms that selected it. Then, this ranking was used to reduce the dimension of the training set in the experiments, to test whether the most selected features actually include the most significant patterns. To this aim, the *minimal features set* can be defined as the set of those features chosen by at least 3 of the 4 algorithms.

As tools for feature selection the following techniques, both supervised and unsupervised, were

employed:

1. L2-regularized SVM [25]

$$\min_{w\in\mathbb{R}^n} \|w\|_2^2 + C\sum_{i=1}^{m} \max\{0, 1 - y_i(w^T x^i + b)\};$$

2. L1-regularized SVM [118]

$$\min_{w\in\mathbb{R}^n} \|w\|_1 + C\sum_{i=1}^{m} \max\{0, 1 - y_i(w^T x^i + b)\};$$

3. Mutual information (MI), which is a non-negative value that measures the dependency between two random variables. The function relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances as described, e.g., in [60, 86];

4. Analysis Of Variance (ANOVA) [55].

It must be emphasized that the selection of the features depends on the data at hand. In order to take care of this aspect, the selection was replicated $M$ times, selecting 70% of the available data randomly in each replicate. For each technique, the features selected more than 80% of the times over the $M$ replicates were chosen.

The feature selection procedure can then be described as:

- Given the target set in a table $m-\text{by}-(n+1) = 466 \times 28$;
- Repeat $M$ times
  - Randomly extract 70% of the target set;
  - Apply the four different feature selection algorithms;
- For each algorithm, rank the features according to the number of times selected over the $M$ runs;
- For each algorithm, choose the features selected more than 80% of the times;
- Construct the *minimal features set* by choosing the features selected by at least 3.

At the end of the feature selection, a *reduced target set* was identified, namely obtained from the full target set by considering only the minimal set of features. The results on the reduced target set were then compared to the ones on the full target set.

The feature selection procedure led to the results summarized in table 7.9. In particular, for each feature selection technique, the features selected at least in the 80% of the $M$ random runs are indicated by an "x". Furthermore, the count of how many algorithms selected each feature is also reported.

The selected features are the ones that are selected by at least 3 out of the 4 algorithms. These are the first 14 rows in table 7.9, which constitutes the minimal feature set:

$$\{x_4, x_5, x_7, x_9, x_{10}, x_{11}, x_{13}, x_{15}, x_{16}, x_{17}, x_{19}, x_{20}, x_{25}, x_{26}\}.$$

In order to check the effectiveness of the selection procedure with respect to standard statistical tools, a selection by the analysis of Pearson's correlation matrix was also performed. The results are reported

| Variables | name | L2 SVM | L1 SVM | MI | ANOVA | Final score |
|---|---|---|---|---|---|---|
| $x_4$ | Pelvic torsion_DL-DR | x | x | x | x | 4 |
| $x_5$ | Pelvic inclination_(dimple) | x | x | x | x | 4 |
| $x_7$ | Inflexion point_ICT /trunk length_VP-DM | x | x | x | x | 4 |
| $x_9$ | Inflexion point_ITL /trunk length_VP-DM | x | x | x | x | 4 |
| $x_{10}$ | Lordotic apex_LA_(VPDM) /trunk length_VP-DM | x | x | x | x | 4 |
| $x_{13}$ | Fléche lombaire_(Stagnara) | x | x | x | x | 4 |
| $x_{17}$ | Surface rotation_(rms) | x | x | x | x | 4 |
| $x_{25}$ | Lateral deviation_VPDM_(+max) | x | x | x | x | 4 |
| $x_{11}$ | Inflexion point_ILS /trunk length_VP-DM | | x | x | x | 3 |
| $x_{15}$ | Lordotic angle_ITL-ILS_(max) | | x | x | x | 3 |
| $x_{16}$ | Pelvic inclination | x | | x | x | 3 |
| $x_{19}$ | Surface rotation_(+max) | x | x | | x | 3 |
| $x_{20}$ | Surface rotation_(-max) | x | | x | x | 3 |
| $x_{26}$ | Lateral deviation_VPDM_(-max) | x | x | x | | 3 |
| $x_1$ | Trunk inclination_VP-DM | | x | | x | 2 |
| $x_2$ | Lateral_flexion_VP-DM | x | | x | | 2 |
| $x_6$ | Pelvis rotation | x | | | x | 2 |
| $x_8$ | Kypothic apex_KA_(VPDM) /trunk length_VP-DM | x | | x | | 2 |
| $x_{12}$ | Fléche cervicale_(Stagnara) | | | x | x | 2 |
| $x_{14}$ | Kyphosis angle_ICT-ITL | | | x | x | 2 |
| $x_{18}$ | Surface rotation_(max) | x | x | | | 2 |
| $x_{21}$ | Surface rotation_(width) | x | | | x | 2 |
| $x_{24}$ | Lateral deviation_VPDM_(max) | | | x | x | 2 |
| $x_{27}$ | Lateral deviation_(width) | x | x | | | 2 |
| $x_3$ | Pelvic obliquity_DL-DR | | | x | | 1 |
| $x_{23}$ | Lateral deviation_VPDM_(rms) | | | | x | 1 |
| $x_{22}$ | Pelvic torsion | | | | | 0 |

**Table 7.9.** Features ranking

in table 7.8 and the selected features in this case are the ones having a Pearson correlation with the output $\geq 0.2$, namely

$$\{x_1, x_5, x_7, x_8, x_9, x_{10}, x_{11}, x_{13}, x_{15}, x_{16}, x_{17}, x_{20}, x_{21}, x_{23}, x_{27}\}.$$

One interesting analysis is to add the selection based on Pearson's correlation matrix to the voting procedure shown in table 7.9. Doing so, and choosing the features that obtained at least 4 "votes", the following feature set is obtained:

$$\{x_4, x_5, x_7, x_9, x_{10}, x_{11}, x_{13}, x_{15}, x_{16}, x_{17}, x_{20}\},$$

which will be denoted as *Pearson minimal set*. Note that the Pearson minimal set is not the intersection of the Pearson's feature set and the minimal features set. Indeed the feature $x_4$ (Pelvic torsion_DL-DR) has a low value of the Pearson coefficient (0.11) and it would not be selected by solely the Pearson selection rule but it is instead selected by all the other four algorithms.

After the features selection has been performed, four training sets with the same number of samples are available, i.e. the full set of 27 features and the three reduced introduced above. The learning process, both unsupervised and supervised, was applied to these four datasets, namely the full dataset, the minimal set, the Pearson set and the Pearson minimal set, to verify how these selections impact the performance. Nonetheless, it turned out that among the three, the best performance was on the

minimal set. Hence, in the results only the comparison between the full set and the minimal set is detailed.

The results reported in sections 7.3.4 and 7.3.5 show that such a cut in dimensionality does not strongly affect performance of the ML, which is measured by the KPIs introduced above. This suggests that features in table 7.6 have a key role in classifying scoliosis.

## 7.3 Results and discussion

### 7.3.1 Classification models

In this section the three main classes of ML classification methods used to analyze data in the target set are described. In particular, as mentioned in the introduction, both unsupervised clustering and supervised classification are of interest.

Unsupervised learning does not use any a priori information on the labels of the input data, with the aim of grouping 'similar' samples in clusters, on the basis of the input features only. The known label $y^i$ are used in the 'a posteriori' analysis to evaluate the performance, as explained in section 7.3.2. The aim, from a clinical point of view, is to check whether the features describing each patient contain enough 'good' information to allow a natural division into two clusters. Similarity is usually measured by a metric distance between samples both intra-group and extra-group with the aim of maximizing the distance between samples in different clusters, while minimizing the distance between samples belonging to the same cluster. The clustering method employed is a variant of the basic K-Means [56] algorithm called K-Means++ [6], where the number of clusters $K$ is set to $K = 2$ (healthy vs scoliosis). As metric distance the standard Euclidean norm was used. The distance between $x^1$ and $x^2 \in \mathbb{R}^n$ is therefore defined as

$$d(x^1, x^2) = \|x^1 - x^2\| = \sqrt{\sum_{i=1}^{n} (x_i^1 - x_i^2)^2}.$$

The comparison of the results obtained by unsupervised learning versus those obtained by a supervised procedure using the label to drive the learning procedure can help in checking the existence of biasing features. Actually this happened in the first stage of this study and led to the normalization of some features, in order to avoid clusters that are implicitly based on the age of the patient. The results are reported in section 7.3 and they highlight that the data can be clustered sufficiently well.

On the other hand in supervised classification the task is to learn from 'labelled' examples $(x^i, y^i)$ $i = 1, \ldots, m$ as given in the target set $\mathscr{T}$. Support Vector Machines (SVM) (see e.g. [111] and the survey [81]) and Deep Neural Networks (DN) [48] were used as supervised classification models. The aim, beyond the classification performance, is to compare the performance among these two tools. Support Vector Machines (SVMs) are supervised binary classifiers in the class of kernel methods that learn the possibly nonlinear border between data belonging to different classes. Linear SVM classify points using hyperplanes defined by $(w, b) \in \mathbb{R}^n \times \mathbb{R}$ and obtained by solving the following (Primal) optimization problem:

$$\min_{w \in \mathbb{R}^n} \|w\|_2^2 + C \sum_{i=1}^{m} \max\{0, 1 - y^i (w^T x^i + b)\} \tag{7.1}$$

where $C$ plays the role of a penalty parameter on the misclassified points to be set in the tuning phase of the model. However, usually nonlinear SVM, which define a nonlinear decision function to predict

the class of a new input $x$, are used. Such nonlinear decision function can be written as

$$class(x) = sign\left(\sum_{i=1}^{m} \alpha_i k(x^i, x) + \beta\right),$$

where $k(\cdot,\cdot)$ is a kernel function which represents a measure of similarity, i.e. a scalar product among data points in a transformed nonlinear space. The use of kernels allows to define nonlinear separation surface avoiding the explicit nonlinear transformation in a higher dimensional space [91], thus avoiding the so called curse of dimensionality. Indeed the parameters $\alpha_i \in \mathbb{R}$, $i = 1,\ldots,m$ and $\beta \in \mathbb{R}$ appearing in the nonlinear decision function are obtained as the solution of the dual formulation of the SVM training problem

$$\begin{aligned} \min_\alpha \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y^i y^j k(x^i, x^j)\alpha_i\alpha_j - \sum_{j=1}^{m}\alpha_j \\ & \sum_{i=1}^{m} y^j \alpha_j = 0 \\ & 0 \le \alpha_j \le C \qquad j = 1,\ldots,m \end{aligned}$$

The kernel function may depend on hyper-parameters too and suitable values of those were found during the tuning process, using a $k$-fold cross validation procedure. DN are multilayer feedforward neural networks organized into layers numbered from $\ell = 0$ (input layer) to $\ell = L$, the last layer corresponding to the output layer, as reported in picture 7.2. Assuming a linear output unit and hidden

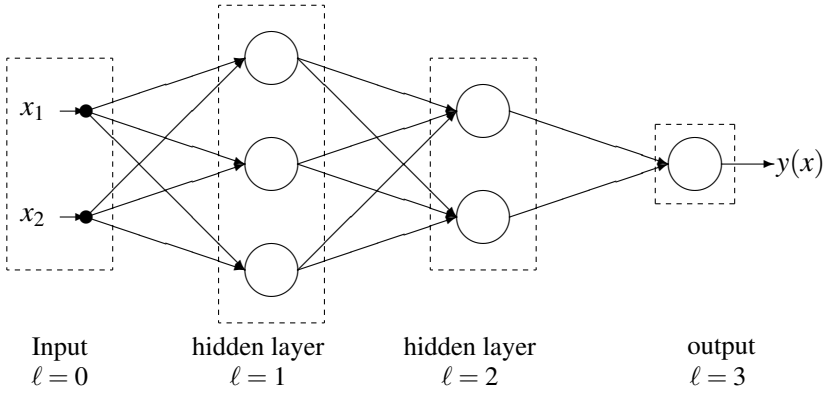


**Figure 7.2.** DN with two inputs ($n = 2$), two hidden layers ($L = 3$) and a single output

units with activation function given by $g(\cdot)$ (assumed to be the same for all the layers) the expression of the output is

$$\tilde{y}(w;x) = W^L g\left(W^{L-1}; g(\ldots; g(W^1; x)))\ldots\right) \tag{7.2}$$

where $W^\ell$ with $\ell = 1,\ldots,L$ is the vector of weights from layer $\ell - 1$ to layer $\ell$ and its size depends on the number of units in each layer. In particular, being $N^\ell$ the number of units in layer $\ell$, $W^\ell \in \mathbb{R}^{N^{\ell-1}\cdot N^\ell}$. The parameters $W^l$, $l = 1,\ldots,L$ are obtained by minimizing the regularized empirical error

$$R = \sum_{i=1}^{m} (y_i - \tilde{y}(w;x^i))^2 + C\|w\|^2$$

where $C$ is a hyper-parameter. In the numerical experiments, no regularization was used, i.e. $C = 0$. Since the problem is a binary classification one, a sigmoid activation function was applied to the output $\tilde{y}(w;x)$ of the network, so that the final output is $\tanh(\tilde{y}(w;x))$, returning a value in $\{-1,1\}$.

### 7.3.2 Performance measures

The ultimate task of a ML model is to give good performance on new unseen samples with unknown label. This task is called generalization and it may be in contrast with the perfect learning of the training data that leads to the so-called over-fitting phenomenon (see e.g. [80] and references therein). Hence, both in supervised and in unsupervised learning, two main phases must be distinguished: the training phase, where the machine is trained using the samples of the target set, and a prediction phase, where the trained machine is used to predict the label (namely the belonging class or cluster) of future unseen data samples. In order to check the performance of a learner without being biased by the learning process itself, usually the training phase is repeated by inserting some randomness in the process.

In particular, for unsupervised learning, the whole target set was used as training set, the K-means++ procedure [6] was repeated $M$ times, due to the randomly seeded initial conditions, and the results were averaged. In particular, to check the correctness of the clusters $\mathscr{C}_i$, $i = 1,2$ obtained at the end of each of the $M$ runs of the training phase, the known labels $y^i$ were used. The correctness of the clusters was measured by means of the so called *purity*, which is a simple and transparent accuracy measure. In such performance measure, each cluster is assigned to the class which most frequently appears as label for the points in the cluster, and then the accuracy is measured by counting the average number of correctly assigned samples. Formally it can be written as:

$$ACC = \frac{1}{2m} \max \left\{ \sum_{i=1}^{m} |y^i + \text{class}_i|, \sum_{i=1}^{m} |y^i - \text{class}_i| \right\},$$

with

$$\text{class}_i = \begin{cases} +1, & x^i \in \mathscr{C}_1, \\ -1, & x^i \in \mathscr{C}_2. \end{cases}$$

In supervised learning, the target set is usually split into two parts: a training set, used only in the learning phase to train the machine, and a test set, used only in a post-learning analysis to quantify the generalization performance. In this way the performance indicators of the learning machine are computed on patients never shown to the learning process. In particular, at each run the available data were split randomly in training and test set, on which the Key Performance Indicators (KPIs) were computed. The average of these KPIs over the $M$ runs represents an estimation of the generalization performance.

The KPIs to measure the quality of a classification machine used in this work were:

- a $2 \times 2$ confusion matrix, where each element represents the (averaged) percentage of True Negatives (TN), True Positives (TP), False positives (FP) and False Negatives (FN), as shown below

| **Real / Predicted** | **Scoliotic** | **Healthy** |
|---|---|---|
| **Scoliotic** | TP | FN |
| **Healthy** | FP | TN |

- classification accuracy (i.e. percentage of correct classified patients)

$$\text{ACC} = \frac{TP + TN}{TP + FN + TN + FP}$$

- balanced classification accuracy (BACC)

$$\text{BACC} = \frac{1}{2}\left[\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right]$$

The above KPIs are reported as an average on the test set when supervised learning is used. The perfect learning corresponds to a 100% accuracy and it consists in having the sum over the diagonal of the confusion matrix being 100. This situation, whenever it appears, is in general untrustworthy, being often a signal of overfitting.

The training procedure, both in the case of unsupervised and supervised learning, is summarized below:

- Given the target set $\mathscr{T}$ given by $m-\text{by}-(n+1) = 466 \times 28$ samples;
- Repeat $M$ times
  1. Extract randomly the training and the test sets as a percentage of the $m$ samples (patients):
     - for supervised learning respectively 70% and 30%
     - for unsupervised learning respectively 100% and 0% ;
  2. Train a classifier using the training set;
  3. Compute the KPIs of the obtained classifier;
- Average the KPIs over the $M$ runs.

### 7.3.3 Classifiers tuning and training

In both the two models used for supervised classification, DNN and SVM, some hyper-parameters must be chosen in order to get the best performance of the machines.

The SVM classifiers has two type of parameters: the regularization parameter $C$, which controls misclassified points, and the kernel parameters. Both linear and Gaussian kernels were tested. Linear kernel $k(x^i,x) = x^{iT}x$ does not depend on hyper-parameters, while Gaussian kernel

$$k(x^i,x) = e^{-\gamma\|x^i-x\|^2},$$

which corresponds to a transformation of the data points $x^i$ into an infinite dimensional space, presents the hyper-parameter $\gamma$, called the width of the kernel.

Regarding DN, the number of layers $L$, neurons per layer $N^\ell$, $\ell = 1,\dots L$ and the activation function $g(z)$ needs be chosen. Both the ReLU (Rectified Linear Unit), $g(z) = \max\{0,z\}$, and the sigmoid, $g(z) = \frac{e^z}{e^z+1}$, functions were tested from the library ScikitLearn [41]. The tuning procedure was therefore applied for different values of $L$ and $N^\ell$, $\ell = 1,\dots L$. The hyper-parameters to be chosen for each classifier are summarized in table 7.10.

The selection of the correct hyper-parameters $C,\gamma$ has been done by using a $k$-fold cross validation procedure with a grid search. The average of these errors over the $k$ runs represents an estimation of the generalization performance and the setting of hyper-parameters that gave the best average validation error was selected. The process can be summarized as:

- Given the target set $\mathscr{T}$;
- Define a grid $\mathscr{H}$ of hyper-parameters values;

| Model | Hyper-parameters | |
|---|---|---|
| **SVM** | $C$ | $\gamma$ |
| **DNN** | $N^{\ell}$ | $L$ |

**Table 7.10.** Hyper-parameters for SVM/DN classifiers

| Full / Selected features | Healthy | Scoliotic |
|---|---|---|
| **Healthy** | 25.7 / 32.4 | 16.1 / 9.2 |
| **Scoliotic** | 22.2 / 18.7 | 36.0 / 39.8 |

**Table 7.11.** Confusion matrix of the unsupervised classifier: the first number refers to the full target set whilst the second number refers to the minimal set.

- For each $h \in \mathscr{H}$, select the $h$-th hyperparameter pair;
  - Repeat $M$ times
    1. Extract randomly the training and the validation sets;
    2. Train a classifier using the training set;
    3. Compute the KPIs on the validation set;
  - Average the KPIs over the $M$ runs;
- Select the $h \in \mathscr{H}$ that gives the best average KPIs.

### 7.3.4 Performance of unsupervised classifiers

First, an unsupervised learning process was applied to the two target sets defined in section 7.2.3, from which the labels are 'hidden' to the machine. The unsupervised learner was then applied with the task of grouping the patients with the aim of 'identifying' the two original classes.

The results in terms of accuracy and confusion matrix are reported in table 7.11. The accuracy reached using the full set of 27 features and only the 14 selected features of the minimal set 61.7% and 72.2%, respectively. It is interesting to see how the accuracy increased with the features reduction. Indeed this was foreseeable since lower dimension makes the task easier for a learning machine that does not exploit the labels to cluster the samples in groups. The confusion matrices of the unsupervised learning machine obtained on the two target sets are shown in table 7.11. In both the experiments, false positives and false negatives are well balanced, showing once more the robustness of the target data, namely that the rastereographic information allows to clearly separate the two clusters, i.e. subjects with scoliosis vs healthy ones.

It is worth to mention that the clustering procedure was performed as a first step at the very beginning of the project, in order for the results to point out any biases in the data that induced wrong clusters. Such information derived from the first clustering results were then used to clean up the data, as explained in the previous sections.

### 7.3.5 Performance of supervised classifiers

The number of replicates $M$ was set to 100 in the training procedure. The DNN classifiers were first trained on the full data set using different architectures with the aim of exploring the role of wideness

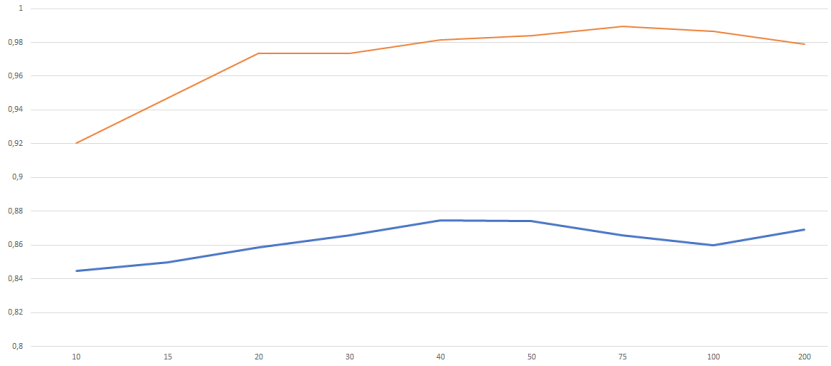


**Figure 7.3.** Test (blue) and Training (red) accuracy for increasing number of neurons in the unique hidden layer

| Data set | ACC | BACC |
|---|---|---|
| **Full set** | 87.5% | 87.4% |
| **Minimal set** | 83.7 % | 83.4% |

**Table 7.12.** Accuracy (ACC) and Balanced Accuracy (BACC) of a shallow NN with $N^1 = 40$.

and deepness. Both sigmoid and ReLu activation functions were tested, but the ReLu performed significantly worst. Hence, the results are reported only for the sigmoid.

In particular, a shallow network (i.e. one hidden layer) was trained with increasing number of neurons $N$ in order to understand the role of wideness. The number of neurons $N$ was increased from 5 to 200 and the results in terms of average test (blue) and training accuracy (red) are reported in figure 7.3. The averaged classification accuracy is almost everywhere higher than 80% being $40 \leq N \leq 50$ the best range of neurons. Observe that the training accuracy reaches almost 100% for $N \geq 50$. In table (7.12), the accuracy and balanced accuracy of the best configuration, which corresponds to $N^1 = 40$, are reported.

A test increasing the deepness from $L = 2$ to $L = 10$ was also performed, with a fixed number of neurons per layer of $N^\ell = 20$ for all $\ell = 1, \ldots, L$. The results are reported in figure 7.4.

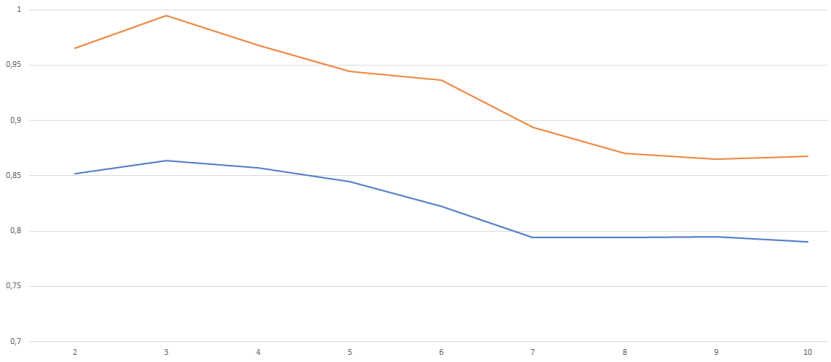


**Figure 7.4.** Test (blue) and Training (red) accuracy for increasing number of layers with $N^\ell = 20$ for all $\ell = 1, \ldots, L$

The best results correspond to $L = 3$ (two hidden layers). The picture shows that increasing the deepness of the network beyond such number does not produce any better results. This may also be

| | ACC | BACC |
|---|---|---|
| **Full set** | 86.3% | 86.6% |
| **Minimal set** | 85.5 % | 85.5% |

**Table 7.13.** Accuracy (ACC) and Balanced Accuracy (BACC) of Deep Network with $L = 3$.

| Full / minimal set | Healthy | Scoliotic |
|---|---|---|
| **Healthy** | 16.82/16.25 | 2.59/3.55 |
| **Scoliotic** | 3.81/4.11 | 23.78/23.09 |

**Table 7.14.** Confusion matrix of shallow network with $N = 40$ with all the features and the 14 in the minimal set respectively

due to the well known difficulties in training such deep networks using (stochastic) gradient based methods, like Adam. The accuracy and balance accuracy are reported in table (7.13).

The behavior on the target set with only the 14 features selected as described in section 7.2.3 is also analyzed.

For a comparison, the average confusion matrix obtained in the two cases, full vs reduced target set, are reported in tables 7.14 and 7.15.
From the above results, it seems that the performance decreases with the reduction of the features only by $2-3\%$. This seems to suggest that the 14 features selected bring the most significant information. Indeed the physicians analyzed them and the comments are reported at the end of the section.

The nonlinear SVM classifier employed was the one implemented in LIBSVM library, with a Gaussian kernel function with spread $\gamma$. The best values of the parameters $C$ and $\gamma$ were set with the tuning procedure as described in section 7.3.3. The procedure has been replicated both for the full dataset and the minimal features set. The resulting values of $C$ and $\gamma$ are reported in table 7.16.
The classification accuracy and balanced accuracy reached by the SVM classifier both for the full and the minimal dataset are reported in table 7.17.

The confusion matrix of the SVM classifier for the two sets, i.e. full dataset and minimal features set, is also reported in table 7.18. In both cases, the false positives and false negatives are well balanced, namely the percentage of healthy patients classified as scoliotic is almost equal to the percentage of scoliotic patients classified as healthy.

Observe that a reduction of the input dimensionality by almost 50% brought only a slight deterioration of the accuracy, thus confirming the potential strong role played by the selected features, in line with the results of the DNN classifier. This result could help physicians in the diagnostic process, when it is usually needed to look at multiple different variables and potentially invasive tests (i.e. X-ray) to detect a scoliosis.

| Full / minimal set | Healthy | Scoliotic |
|---|---|---|
| **Healthy** | 35.77/ 33.62 | 6.15/ 7.43 |
| **Scoliotic** | 8.32/ 8.64 | 49.77/ 50.32 |

**Table 7.15.** Confusion matrix of deep network with $L = 3$ with all the features and the 14 in the minimal set respectively

| | Full | Minimal |
|---|---|---|
| $C$ | 10 | 10 |
| $\gamma$ | $10^{-3}$ | $10^{-2}$ |

**Table 7.16.** Parameters of SVM defined by the tuning procedure

| | ACC | BACC |
|---|---|---|
| **Full set** | 84.9% | 84.7% |
| **Minimal set** | 82.2 % | 81.5% |

**Table 7.17.** Accuracy (ACC) and Balanced Accuracy (BACC) of SVM.

## 7.4 Clinical comments and conclusion

### 7.4.1 Clinical comments

In this section, the 14 features (i.e. rasterstereography parameters) identified by means of the the feature selection process are analyzes, with the aim of verifying their clinical role. Among the identified parameters reported as the first 14 in table 7.9 there are measures on the three planes, i.e. lateral, sagittal and frontal plane. Five of the selected parameters, identified by the features $x_{17}, x_{19}, x_{20}, x_{25}, x_{26}$, are commonly related to the evaluation and diagnosis of scoliosis, in fact lateral deviation and vertebral rotation are well known clinical signs of the disease. Indeed the scoliosis is defined as a lateral curvature of more than 10 degrees as measured by the Cobb Technique on standing anterior posterior radiograph of the spine. Scoliosis is a complex three-dimensional spinal deformity, that forms a complex curve that leads to deformities not only in the coronal plane but in all three planes, which is caused by the self-rotating movement of the spine. An important feature of idiopathic scoliosis deformity is the vertebral axial rotation which accompanies the vertebral lateral deviation. Mechanical interactions within the spine have been implicated in causing vertebral rotation with lateral deviation. This rotation is thought to be significant for initiation and progression of scoliosis. The magnitude of vertebral axial rotation correlates with the lateral deviation of vertebrae from the spinal axis, and the rotation is maximal near the curve apex [29, 103, 114, 115].

Nine parameters, identified by th $x_4, x_5, x_7, x_9, x_{10}, x_{11}, x_{13}, x_{15}, x_{16}$, are instead related to the sagittal plane and these results may seem unexpected because the scoliosis is predominantly characterized by alterations on the frontal and the transversal plane. However recent clinical papers seem to suggest a role of these parameters. Sullivan et al [104] underlined the importance of sagittal plane and the need of a global assessment in the evaluation of scoliosis. They find a strong correlation between scoliosis severity and loss of 3D kyphosis. Increasing severity of coronal plane curvature is associated with a progressive loss of thoracic kyphosis. Moreover previous study have suggested that thoracic

| Full / Minimal set | Healthy | Scoliotic |
|---|---|---|
| **Healthy** | 34.0 / 31.9 | 7.0 / 9.6 |
| **Scoliotic** | 8.1 / 8.3 | 50.9 / 50.3 |

**Table 7.18.** Results of the supervised SVM classifier on the full and minimal features set

hypokyphosis is a primary event in the development of Idiopatic Scolisis (IS) [102]. However, because of historic restrictions of planar imaging of this multidimensional deformity, there is little information regarding the correlation of thoracic kyphosis with increasing severity of idiopathic scoliosis. Modern, low-radiation-exposure 3D imaging systems have now made routine clinical 3D imaging feasible. These imaging modalities offer the possibility to study the components of the scoliotic deformity in the planes of origin for each vertebra, free of the distortions on 2D images [77]. Sullivan et al [104] found a strong linear correlation between the magnitude of the main thoracic coronal curve and loss of 3D thoracic kyphosis. Three of these sagittal parameters, $x_4, x_5, x_{16}$, are related to sagittal alignment of the pelvis, but this was expected since an influence of the sagittal parameters of the column in the identification of patients is known. Legayeet et al [62] demonstrated the key importance of the anatomical parameter of pelvic incidence in the regulation of the sagittal curves and this is maintained when the scoliosis desease occurs. Moreover Fei Han et al [43] underlined that patients with degenerative scoliosis (DS) may have a higher pelvic incidence, which may impact the pathogenesis of DS. In fact an unbalancing of incidence pelvic should cause scoliosis if the degeneration speed of the two sides differs [65]. The limits of traditional 2D imaging have restricted the evaluation of the adolescent idiopathic scoliosis effects on the sagittal plane, in fact it is impossible to perform simultaneous evaluation of the frontal and sagittal profiles of the spine. Moreover when sagittal evaluation is performed, the patient is asked to put arms forward which is a non-natural position. Since RX evaluation exposes the patients to ionizing radiations, it can not be made frequently in clinical follow up frequency. The rasterstereography provides a three-dimensional reconstruction of the spine curvature and the patient can assume a natural position. Moreover it provides a dynamic evaluation: indeed, in six seconds, twelve frames are recorded and the average results are obtained. So it is particularly suited to the sagittal plane parameters of the spine without any ionizing radiations exposure and any risk for the patient [116]. Summarizing, the features selected by the procedure seems to have some clinical usefulness thus validating the overall learning procedure based on DNN or SVM.

### 7.4.2 Conclusion

In this study, both supervised and unsupervised learning were proven to give high accuracy results in classifying AIS patient versus healthy ones, using rasterstereography data. As expected, the accuracy is higher in the supervised case than in the unsupervised one. Indeed, the supervised algorithms used, i.e. DNN and SVM, performed quite well, with an accuracy over 80%. However, the use of a clustering procedure also allowed to group patients in well-separated clusters, which showed strong intra-group similarity.

The above evidences confirm that data mining can represent a new approach to identify patients with AIS from healthy ones. Moreover, the results confirm that a subset of rasterstereography parameters can be used in the screening of AIS patients, although X-ray imaging cannot be replaced by this method. Indeed, rasterstereography can be used to perform a scoliosis screening in order to improve the selection of patients that need to undergo X-ray examination. Finally, thanks to the fact that Formetric is easily transportable, it can be used to propose scholar screening for pre-adolescent pupils.

# List of Figures

# List of Tables

# Bibliography


[1] ADANKON, M. M., DANSEREAU, J., LABELLE, H., AND CHERIET, F. Non invasive classification system of scoliosis curve types using least-squares support vector machines. *Artificial Intelligence in Medicine*, **56** (2012), 99.

[2] AL BORNO, M., DE LASA, M., AND HERTZMANN, A. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics*, **19** (2012), 1405.

[3] ALLEN-ZHU, Z. AND HAZAN, E. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, pp. 699–707 (2016).

[4] ALLEN-ZHU, Z., YUAN, Y., AND SRIDHARAN, K. Exploiting the structure: Stochastic gradient methods using raw clusters. In *Advances in Neural Information Processing Systems*, pp. 1642–1650 (2016).

[5] AMARAN, S., SAHINIDIS, N. V., SHARDA, B., AND BURY, S. J. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, **240** (2016), 351.

[6] ARTHUR, D. AND VASSILVITSKII, S. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics (2007).

[7] BALASUBRAMANIAN, K. AND GHADIMI, S. Zeroth-order nonconvex stochastic optimization: Handling constraints, high-dimensionality and saddle-points. -, (2018).

[8] BERAHAS, A. S., BYRD, R. H., AND NOCEDAL, J. Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM Journal on Optimization*, **29** (2019), 965.

[9] BERAHAS, A. S., CAO, L., CHOROMANSKI, K., AND SCHEINBERG, K. Linear interpolation gives better gradients than gaussian smoothing in derivative-free optimization. *arXiv preprint arXiv:1905.13043*, (2019).

[10] BERAHAS, A. S., CAO, L., CHOROMANSKI, K., AND SCHEINBERG, K. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *arXiv preprint arXiv:1905.01332*, (2019).

[11] BERTSEKAS, D. P. *Nonlinear programming*. Athena scientific Belmont (1999).

[12] BERTSEKAS, D. P. Incremental proximal methods for large scale convex optimization. *Mathematical programming*, **129** (2011), 163.

[13] BERTSEKAS, D. P. AND SCIENTIFIC, A. *Convex optimization algorithms*.

[14] BERTSEKAS, D. P. AND TSITSIKLIS, J. N. *Parallel and distributed computation: numerical methods*, vol. 23. Prentice hall Englewood Cliffs, NJ (1989).

[15] BERTSEKAS, D. P. AND TSITSIKLIS, J. N. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, **10** (2000), 627.

[16] BIRGIN, E. G. AND MARTINEZ, J. M. *Practical augmented Lagrangian methods for constrained optimization*, vol. 10. SIAM (2014).

[17] BLATT, D., HERO, A. O., AND GAUCHMAN, H. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, **18** (2007), 29.

[18] BOLLAPRAGADA, R., BYRD, R., AND NOCEDAL, J. Adaptive sampling strategies for stochastic optimization. *arXiv preprint arXiv:1710.11258*, (2017).

[19] BOTTOU, L., CURTIS, F. E., AND NOCEDAL, J. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, (2016).

[20] BOYD, S., PARIKH, N., CHU, E., PELEATO, B., ECKSTEIN, J., ET AL. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, **3** (2011), 1.

[21] BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., AND ZAREMBA, W. Openai gym (2016). `arXiv:arXiv:1606.01540`.

[22] BYRD, R. H., CHIN, G. M., NOCEDAL, J., AND WU, Y. Sample size selection in optimization methods for machine learning. *Mathematical programming*, **134** (2012), 127.

[23] CANNELLI, L., FACCHINEI, F., AND SCUTARI, G. Multi-agent asynchronous nonconvex large-scale optimization. In *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 1–5. IEEE (2017).

[24] CHANDRASHEKAR, G. AND SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering*, **40** (2014), 16.

[25] CHANG, C.-C. AND LIN, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, **2** (2011), 27.

[26] CHEN, H., SHEN, C., QIN, J., NI, D., SHI, L., CHENG, J. C. Y., AND HENG, P.-A. Automatic localization and identification of vertebrae in spine ct via a joint learning model with deep neural networks. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (edited by N. Navab, J. Hornegger, W. M. Wells, and A. Frangi), pp. 515–522. Springer International Publishing, Cham (2015). ISBN 978-3-319-24553-9.

[27] CHOROMANSKI, K., ROWLAND, M., SINDHWANI, V., TURNER, R. E., AND WELLER, A. Structured evolution with compact architectures for scalable policy optimization. *arXiv preprint arXiv:1804.02395*, (2018).

[28] CLARKE, F. H. *Optimization and nonsmooth analysis*, vol. 5. Siam (1990).

[29] COBB, J. Outline for the study of scoliosis. *Instr Course Lect AAOS*, **5** (1948), 261.

[30] COLOMBO, T., MANGONE, M., BERNETTI, A., PAOLONI, M., SANTILLI, V., AND PALAGI, L. Supervised and unsupervised learning to classify scoliosis and healthy subjects based on non-invasive rasterstereography analysis. DIAG Technical Reports 2019-08, Department of Computer, Control and Management Engineering, Universita' degli Studi di Roma "La Sapienza" (2019).

[31] COLOMBO, T. AND SAGRATELLA, S. Distributed algorithms for convex problems with linear coupling constraints. *Journal of Global Optimization*, (2019), 1.

[32] CRAMER, H. *Mathematical Methods of Statistics*. Princeton University Press (1946).

[33] DANESHMAND, A., SUN, Y., SCUTARI, G., FACCHINEI, F., AND SADLER, B. M. Decentralized dictionary learning over time-varying digraphs. *arXiv preprint arXiv:1808.05933*, (2018).

[34] DEFAZIO, A., BACH, F., AND LACOSTE-JULIEN, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pp. 1646–1654 (2014).

[35] DEFAZIO, A., DOMKE, J., ET AL. Finito: A faster, permutable incremental gradient method for big data problems. In *International Conference on Machine Learning*, pp. 1125–1133 (2014).

[36] DI PILLO, G. AND LUCIDI, S. On exact augmented lagrangian functions in nonlinear programming. In *Nonlinear Optimization and Applications*, pp. 85–100. Springer (1996).

[37] DI PILLO, G. AND LUCIDI, S. An augmented lagrangian function with improved exactness properties. *SIAM Journal on Optimization*, **12** (2002), 376.

[38] DOODY, M. M., LONSTEIN, J. E., STOVALL, M., HACKER, D. G., LUCKYANOV, N., LAND, C. E., ET AL. Breast cancer mortality after diagnostic radiography: findings from the us scoliosis cohort study. *Spine*, **25** (2000), 2052.

[39] DRERUP, B. AND HIERHOLZER, E. Back shape measurement using video rasterstereography and three-dimensional reconstruction of spinal shape. *Clinical Biomechanics*, **9** (1994), 28.

[40] DUAN, Y., CHEN, X., HOUTHOOFT, R., SCHULMAN, J., AND ABBEEL, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338 (2016).

[41] ET AL., P. Scikit-learn: Machine learning in python. *JMLR*, **12** (2011), 2825.

[42] FAZEL, M., GE, R., KAKADE, S. M., AND MESBAHI, M. Global convergence of policy gradient methods for the linear quadratic regulator. *arXiv preprint arXiv:1801.05039*, (2018).

[43] FEI, H., LI, W.-S., SUN, Z.-R., JIANG, S., AND CHEN, Z.-Q. Effect of patient position on the lordosis and scoliosis of patients with degenerative lumbar scoliosis. *Medicine*, **96** (2017).

[44] FRIEDLANDER, M. P. AND SCHMIDT, M. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, **34** (2012), A1380.

[45] GAONKAR, B., HOVDA, D., MARTIN, N., AND MACYSZYN, L. Deep learning in the small sample size setting: cascaded feed forward neural networks for medical image segmentation (2016). doi:10.1117/12.2216555.

[46] GLASS, G. V., PECKHAM, P. D., AND SANDERS, J. R. Consequences of failure to meet assumptions underlying the fixed effects analyses of variance and covariance. *Review of educational research*, **42** (1972), 237.

[47] GONDZIO, J. AND GROTHEY, A. Exploiting structure in parallel implementation of interior point methods for optimization. *Computational Management Science*, **6** (2009), 135.

[48] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep Learning*. MIT Press (2016). http://www.deeplearningbook.org.

[49] GRIPPO, L. AND SCIANDRONE, M. *Metodi di ottimizzazione non vincolata*. Springer Science & Business Media (2011).

[50] GURBUZBALABAN, M., OZDAGLAR, A., AND PARRILO, P. A. On the convergence rate of incremental aggregated gradient algorithms. *SIAM Journal on Optimization*, **27** (2017), 1035.

[51] HAYKIN, S. *Neural networks: a comprehensive foundation*. Prentice Hall PTR (1994).

[52] HOFMANN, T., LUCCHI, A., LACOSTE-JULIEN, S., AND MCWILLIAMS, B. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pp. 2305–2313 (2015).

[53] HONG, M. AND LUO, Z.-Q. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, **162** (2017), 165.

[54] HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, **4** (1991), 251.

[55] IVERSEN, G. R., WILDT, A. R., NORPOTH, H., AND NORPOTH, H. P. *Analysis of variance*. 1. Sage (1987).

[56] JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, **31** (2010), 651 . Award winning papers from the 19th International Conference on Pattern Recognition (ICPR). doi:https://doi.org/10.1016/j.patrec.2009.09.011.

[57] JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, **31** (1999), 264.

[58] JOHNSON, R. AND ZHANG, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323 (2013).

[59] KIM, S., PASUPATHY, R., AND HENDERSON, S. G. A guide to sample average approximation. In *Handbook of simulation optimization*, pp. 207–243. Springer (2015).

[60] KRASKOV, A., STÖGBAUER, H., AND GRASSBERGER, P. Estimating mutual information. *Physical review E*, **69** (2004), 066138.

[61] KUMAR, V., COLE, A., BREAKWELL, L., AND MICHAEL, A. L. R. Comparison of the diers formetric 4d scanner and plain radiographs in terms of accuracy in idiopathic scoliosis patients. *Global Spine Journal*, **6** (2016), s.

[62] LEGAYE, J., DUVAL-BEAUPERE, G., HECQUET, J., AND MARTY, C. Pelvic incidence: a fundamental pelvic parameter for three-dimensional regulation of spinal sagittal curves. *European Spine Journal*, **7** (1998), 99.

[63] LEMAÎTRE, G., NOGUEIRA, F., AND ARIDAS, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, **18** (2017), 1.

[64] LEVINE, S., FINN, C., DARRELL, T., AND ABBEEL, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, **17** (2016), 1334.

[65] LI, W.-S., SUN, Z.-R., AND CHEN, Z.-Q. Radiographic analysis of sagittal spino-pelvic alignment in asymptomatic chinese adults. *Chinese Journal of Orthopaedics*, (2013), 447.

[66] LIN, C.-J., LUCIDI, S., PALAGI, L., RISI, A., AND SCIANDRONE, M. Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds. *Journal of Optimization Theory and Applications*, **141** (2009), 107.

[67] LOSHCHILOV, I. AND HUTTER, F. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, (2015).

[68] LUCIDI, S. New results on a class of exact augmented lagrangians. *Journal of Optimization Theory and Applications*, **58** (1988), 259.

[69] LUCIDI, S., PALAGI, L., RISI, A., AND SCIANDRONE, M. A convergent decomposition algorithm for support vector machines. *Computational Optimization and Applications*, **38** (2007), 217.

[70] MANIA, H., GUY, A., AND RECHT, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, (2018).

[71] MANNO, A., PALAGI, L., AND SAGRATELLA, S. Parallel decomposition methods for linearly constrained problems subject to simple bound with application to the svms training. *Computational Optimization and Applications*, pp. 1–31.

[72] MANNO, A., SAGRATELLA, S., AND LIVI, L. A convergent and fully distributable svms training algorithm. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pp. 3076–3080. IEEE (2016).

[73] MNIH, V., ET AL. Human-level control through deep reinforcement learning. *Nature*, **518** (2015), 529.

[74] MOKHTARI, A., GURBUZBALABAN, M., AND RIBEIRO, A. Surpassing gradient descent provably: A cyclic incremental method with linear convergence rate. *SIAM Journal on Optimization*, **28** (2018), 1420.

[75] NESTEROV, Y. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, **3** (1998), 5.

[76] NESTEROV, Y. AND SPOKOINY, V. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, (2017).

[77] NEWTON, P. O., FUJIMORI, T., DOAN, J., REIGHARD, F. G., BASTROM, T. P., AND MISAGHI, A. Defining the "three-dimensional sagittal plane" in thoracic adolescent idiopathic scoliosis. *JBJS*, **97** (2015), 1694.

[78] NGAN, P. S., WONG, M. L., LAM, W., LEUNG, K. S., AND CHENG, J. C. Medical data mining using evolutionary computation. *Artificial Intelligence in Medicine*, **16** (1999), 73.

[79] PADULO, J. AND ARDIGÒ, L. P. Formetric 4d rasterstereography. *BioMed research international*, **2014** (2014).

[80] PALAGI, L. Global optimization issues in deep network regression: an overview. *Journal of Global Optimization*, (2018), 1.

[81] PICCIALLI, V. AND SCIANDRONE, M. Nonlinear optimization and support vector machines. *4OR*, (2018).

[82] RAJESWARAN, A., KUMAR, V., GUPTA, A., VEZZANI, G., SCHULMAN, J., TODOROV, E., AND LEVINE, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, (2017).

[83] REDDI, S. J., HEFNY, A., SRA, S., POCZOS, B., AND SMOLA, A. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pp. 314–323 (2016).

[84] ROBBINS, H. AND MONRO, S. A stochastic approximation method. *Ann. Math. Statist.*, **22** (1951), 400. Available from: `https://doi.org/10.1214/aoms/1177729586`, `doi:10.1214/aoms/1177729586`.

[85] ROCKAFELLAR, R. T. AND WETS, R. J.-B. *Variational analysis*, vol. 317. Springer Science & Business Media (2009).

[86] ROSS, B. C. Mutual information between discrete and continuous data sets. *PloS one*, **9** (2014), e87357.

[87] SALIMANS, T., HO, J., CHEN, X., SIDOR, S., AND SUTSKEVER, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, (2017).

[88] SCHEFFE, H. *The analysis of variance*, vol. 72. John Wiley & Sons (1999).

[89] SCHMIDT, M., LE ROUX, N., AND BACH, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, **162** (2017), 83.

[90] SCHMINDER, E., ZIEGLER, M., DANAY, E., BEYER, L., AND BÜHNER, M. Is it really robust? reinvestigating the robustness of anova against violations of the normal distribution. *European Research Journal of Methods for the Behavioral and Social Sciences*, **6** (2010), 147.

[91] SCHOLKOPF, B. AND SMOLA, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press (2001).

[92] SCHULMAN, J., LEVINE, S., ABBEEL, P., JORDAN, M., AND MORITZ, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897 (2015).

[93] SCUTARI, G., FACCHINEI, F., AND LAMPARIELLO, L. Parallel and distributed methods for constrained nonconvex optimization—part i: Theory. *IEEE Transactions on Signal Processing*, **65**, 1929.

[94] SCUTARI, G., FACCHINEI, F., LAMPARIELLO, L., SARDELLITTI, S., AND SONG, P. Parallel and distributed methods for constrained nonconvex optimization-part ii: applications in communications and machine learning. *IEEE Transactions on Signal Processing*, **65**, 1945.

[95] SCUTARI, G., FACCHINEI, F., LAMPARIELLO, L., AND SONG, P. Parallel and distributed methods for nonconvex optimization. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 840–844. IEEE (2014).

[96] SILVER, D., LEVER, G., HEESS, N., DEGRIS, T., WIERSTRA, D., AND RIEDMILLER, M. Deterministic policy gradient algorithms (2014).

[97] SILVER, D., ET AL. Mastering the game of go with deep neural networks and tree search. *nature*, **529** (2016), 484.

[98] SOLODOV, M. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, **11** (1998), 23. Available from: `https://doi.org/10.1023/A:1018366000512`, `doi:10.1023/A:1018366000512`.

[99] SPALL, J. C. ET AL. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, **37** (1992), 332.

[100] STEIN, C. ET AL. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*. The Regents of the University of California (1972).

[101] STEIN, C. M. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, (1981), 1135.

[102] STIRLING, A. J., HOWEL, D., MILLNER, P. A., SADIQ, S., SHARPLES, D., AND DICKSON, R. A. Late-onset idiopathic scoliosis in children six to fourteen years old.: A cross-sectional prevalence study. *JBJS*, **78** (1996), 1330.

[103] STOKES, I. A. AND GARDNER-MORSE, M. Analysis of the interaction between vertebral lateral deviation and axial rotation in scoliosis. *Journal of biomechanics*, **24** (1991), 753.

[104] SULLIVAN, T. B., REIGHARD, F. G., OSBORN, E. J., PARVARESH, K. C., AND NEWTON, P. O. Thoracic idiopathic scoliosis severity is highly correlated with 3d measures of thoracic kyphosis. *JBJS*, **99** (2017), e54.

[105] SUTTON, R. S. AND BARTO, A. G. *Reinforcement Learning: An Introduction*. MIT Press (2017).

[106] SUTTON, R. S., BARTO, A. G., ET AL. *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge (1998).

[107] SUTTON, R. S., MCALLESTER, D. A., SINGH, S. P., AND MANSOUR, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063 (2000).

[108] TASSA, Y., EREZ, T., AND TODOROV, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. IEEE (2012).

[109] TODOROV, E., EREZ, T., AND TASSA, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE (2012).

[110] TSENG, P. AND YUN, S. Incrementally updated gradient methods for constrained and regularized optimization. *Journal of Optimization Theory and Applications*, **160** (2014), 832. Available from: `https://doi.org/10.1007/s10957-013-0409-2`, `doi:10.1007/s10957-013-0409-2`.

[111] VAPNIK, V. N. *Statistical Learning Theory*. Wiley (1998).

[112] WOODSEND, K. AND GONDZIO, J. Hybrid mpi/openmp parallel linear support vector machine training. *Journal of Machine Learning Research*, **10** (2009), 1937.

[113] XIAO, L. AND ZHANG, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, **24** (2014), 2057.

[114] YAMAN, O. AND DALBAYRAK, S. Idiopathic scoliosis. *Turkish neurosurgery*, **24** (2014), 646.

[115] YANG, M., ZHAO, Y., YIN, X., CHEN, Z., YANG, C., LI, L., AND LI, M. Prevalence, risk factors, and characteristics of the "adding-on" phenomenon in idiopathic scoliosis after correction surgery: A systematic review and meta-analysis. *Spine*, **43**, 780.

[116] ZAINA, F., DONZELLI, S., LUSINI, M., AND NEGRINI, S. How to measure kyphosis in everyday clinical practice: a reliability study on different methods. *Studies in health technology and informatics*, **176** (2012), 264.

[117] ZHI-QUAN, L. AND PAUL, T. Analysis of an approximate gradient projection method with applications to the backpropagation algorithm. *Optimization Methods and Software*, **4** (1994), 85. Available from: `https://doi.org/10.1080/10556789408805580`, `doi:10.1080/10556789408805580`.

[118] ZHU, J., ROSSET, S., TIBSHIRANI, R., AND HASTIE, T. J. 1-norm support vector machines. In *Advances in neural information processing systems*, pp. 49–56 (2004).