Technische Universität München
TUM School of Computation, Information and Technology

Topics in Stochastic Optimization: Learning with Implicit and Adaptive Steps

Fabian Schaipp

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:		Prof. Dr. Mathias Drton
Prüfende der Dissertation:	1.	Prof. Dr. Michael Ulbrich
	2.	Prof. Dr. Suvrit Sra
	3.	Prof. Dr. Dirk Lorenz

Die Dissertation wurde am 09.02.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 20.06.2024 angenommen.

Abstract

We investigate stochastic optimization methods with the main application of training machine learning models. One approach involves a practical stochastic proximal point method with variance reduction, where the subproblem is solved via semismooth Newton. We also consider algorithms that use the stochastic Polyak step size. Here, we propose a proximal version for regularized problems. Using a model-based viewpoint of momentum, we derive Polyak-type adaptive learning rates for momentum methods.

Acknowledgements

First and foremost, I would like to thank my advisor Prof. Michael Ulbrich for his continuously strong support throughout the PhD. His valuable comments and suggestions deepened my mathematical understanding, and helped me to improve my work in terms of mathematical rigor and simplicity. I would like to thank Prof. Michael Ulbrich for providing me the opportunity to attend several workshops and conferences and to visit researchers abroad. I thank him for his guidance that enabled me to quickly develop research ideas independently.

I am very grateful to Prof. Christian L. Müller for always having an open ear for questions and research ideas. Christian has been a great mentor throughout my PhD. Through him I had the opportunity to get a broad view of scientific research, a more eclectic perspective on the field of optimization, and also to start new collaborations.

I want to thank Robert M. Gower for his tremendous support over the last two years of my PhD. Conducting research with Robert has been a privilege and an exciting journey. I thank Robert for hosting me twice at the Flatiron Institute in New York City. The working environment and the conversations I had with my research colleagues at the Flatiron Institute have led to many new ideas and advancements for my own work.

I am very grateful to the Simons Foundation for providing financial support for both visits in New York. The computations in this thesis were, in part, run at the Leibniz-Rechenzentrum Munich and at facilities supported by the Scientific Computing Core at the Flatiron Institute. I want to thank both institutions for providing the access to computational resources.

I want to thank all of my co-authors for their suggestions, feedback and inspiration during the work on the contents of this thesis – your support is highly appreciated.

I would like to thank all of my colleagues in Munich, who fostered a welcoming working atmosphere from the first day and always provided help or suggestions when needed.

Finally, I owe many thanks to my parents and to my sister for their strong support and patience throughout the years. I want to thank Lili for being a wonderful companion ever since.

Contents

List of Publications and Preprints

A. Milzarek, F. Schaipp, and M. Ulbrich, *A Semismooth Newton Stochastic Proximal Point Algorithm with Variance Reduction*, SIAM Journal on Optimization, 34 (2024), pp. 1157–1185, https://epubs.siam.org/doi/10.1137/22M1488181.

F. Schaipp, R. M. Gower, and M. Ulbrich, *A Stochastic Proximal Polyak Step Size*, Transactions on Machine Learning Research, (2023), https://openreview.net/forum?id=jWr41htaB3.

F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower, *MoMo: Momentum Models for Adaptive Learning Rates*, 41st International Conference on Machine Learning, 2024, https://proceedings.mlr.press/v235/schaipp24a.html.

The following publications and preprints have been completed during the time of the dissertation, but their contents do not form an essential part of the thesis.

F. Schaipp, O. Vlasovets, and C. L. Müller, *GGLasso - a Python package for General Graphical Lasso computation*, Journal of Open Source Software, 6 (2021), p. 3865, https://joss.theoj.org/papers/10.21105/joss.03865.

F. Schaipp, *Decay no more*, in ICLR Blogposts 2023, https://iclr-blogposts.github.io/2023/blog/2023/adamw/.

G. Garrigos, R. M. Gower, F. Schaipp, *Function Value Learning: Adaptive Learning Rates Based on the Polyak Stepsize and Function Splitting in ERM*, 2023, arXiv:2307.14528 [cs.LG].

F. Schaipp, U. Şimşekli, R. M. Gower, *Robust gradient estimation in the presence of heavy-tailed noise*, in NeurIPS 2023 Workshop Heavy Tails in Machine Learning, 2023, https://openreview.net/forum?id=C6PiH9Fkjd.

Notation

Sets	
$\mathbb{R}_+, \mathbb{R}_{++}$	Set of non-negative and positive real numbers
$\overline{\mathbb{R}}$	Set of real extended numbers, $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$
$\mathbb{S}^n$	Set of symmetric $n \times n$ matrices
$\mathbb{S}^n_+, \mathbb{S}^n_{++}$	Set of symmetric, positive (semi)definite $n \times n$ matrices
$\operatorname{int}(C),\ \operatorname{ri}(C)$	Interior and relative interior of a set C
$[n]$	Set of integers $\{1, \dots, n\}$

Linear Algebra	
$\langle \cdot, \cdot \rangle$	Euclidean inner product
$\| \cdot \|$	Euclidean norm, i.e. $\|x\| = \sqrt{\langle x, x \rangle}$
$\langle \cdot, \cdot \rangle_M$	Inner product induced by $M \in \mathbb{S}^n_{++}$, i.e. $\langle x, y \rangle_M = \langle x, My \rangle$
$\| \cdot \|_M$	Norm induced by $M \in \mathbb{S}^n_{++}$, i.e. $\|x\|_M = \sqrt{\langle x, x \rangle_M}$
$A^\top$	Transposed of matrix A
$\lambda_{\min}(A),\ \lambda_{\max}(A)$	Smallest and largest eigenvalue of $A \in \mathbb{S}^n$
$\operatorname{tr}(A)$	Trace of square matrix A
$\operatorname{Diag}(v)$	Diagonal matrix with the entries of $v \in \mathbb{R}^n$ on the diagonal
$\operatorname{Diag}((v_i)_{i=1,\dots,n})$	— " —
$\operatorname{diag}(A)$	The diagonal of a square matrix A
$\mathbf{1}_n$	Vector of ones of length n
$\mathbf{Id}_n$	Identity matrix in $\mathbb{R}^n$; we omit the index n when the context is clear
$\mathbf{e}_i$	i-th unit vector of the standard Euclidean basis

For $A \in \mathbb{S}^n_+$ with eigen-decomposition $A = U\Lambda U^\top$, its square root is given by $A^{1/2} = U\Lambda^{1/2}U^\top$, where $\Lambda^{1/2}$ is obtained by taking the square root of all diagonal elements of Λ.

In the following, let $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $\Phi : \mathbb{R}^n \to \mathbb{R}^m$ be functions.

Functions and Operators	
$\operatorname{dom}(\varphi)$	Domain of φ, i.e. all points x such that $\varphi(x) < +\infty$
$\mathbb{1}_{(\cdot)}$	Boolean indicator function, $\mathbb{1}_{\text{True}} = 1$ and $\mathbb{1}_{\text{False}} = 0$
$\boldsymbol{\delta}_C$	Indicator function of set C, defined as $\boldsymbol{\delta}_C = 0$ if $x \in C$ and $\boldsymbol{\delta}_C = +\infty$ else
$\Phi \in \mathcal{C}^1$	The function Φ is continuously differentiable
$\operatorname{Proj}_C(x)$	Euclidean projection of x onto a non-empty, closed, convex set C
$(\cdot)_+$	Positive part, defined by $(x)_+ := \max\{x, 0\}$
$\odot$	Elementwise multiplication

We write $\tilde{\mathcal{O}}$ when dropping logarithmic terms in the $\mathcal{O}$-notation, e.g. $\mathcal{O}(\frac{\ln(1+K)}{K}) = \tilde{\mathcal{O}}(\frac{1}{K})$.

Introduction

What do you want from theory alone?

Cillian Murphy in *Oppenheimer*

Over the past decades, hand in hand with the availability of more data and increasing computing power, machine learning has become a discipline that affects many parts of modern society. While machine learning models have continuously become more proficient in the classical domains of language modeling and computer vision [18, 32], there has been tremendous progress as well in many scientific applications over the years: for example, machine learning has been proven to be useful for weather forecasting and climate modeling [40, 65, 78], protein folding [72] or cosmology [24].

Training of machine learning models is inherently an optimization problem: the goal is to find the set of parameters for a given architecture that performs best on a certain task. This is typically achieved by minimizing a predescribed loss function over the training data. This problem can be formalized as

$$\min_{x \in \mathbb{R}^n} \mathbb{E}_{s \sim P}[f(x; s)], \tag{1}$$

where x are learnable parameters, s is a random vector distributed according to the training-data distribution P, and $f(x; s)$ is the loss function. The core contribution of optimization research within machine learning is (i) to design algorithms that can solve the above training task in a reliable and efficient manner and (ii) to prove convergence guarantees for these algorithms. In many applications, P is not entirely known or it is too expensive to compute the full expectation in the objective of (1), or gradients thereof. Therefore, stochastic methods are employed, having access to only one sample of s per iteration [16, 104, 145]. Undoubtedly, the most comprehensively studied method in this domain is stochastic gradient descent (`SGD`) [15, 122], while `Adam` [74, 89] is nowadays the most widely used method in practice.

Over the last years, research in optimization within this area faced a dilemma: The dominating trend in machine learning applications has been to *scale up* the model and train with *more data* in order to obtain improvements [63, 149]. This in consequence leads to a higher number of learnable parameters (dimension n) as well as to the loss (and its gradient) being more expensive to compute (size of sample space associated with P).

While training algorithms had to be improved in order to deal with the continuously growing problem scale, at the same time, theoretical understanding of the loss landscape and optimization dynamics was lagging behind. Even state-of-the art theoretical results for `SGD` or `Adam` often

require assumptions that are not practical, hard to verify (for example, upper bounds on the step size or noise assumptions) or that are not met in modern model architectures (for example, convexity of the objective function). For non-convex problems the additional caveat comes into play that convergence rates generally apply to *stationarity* instead of *minimality*. To make a very concrete example, the convergence of gradient flow to a global minimum for a network with one hidden layer, the arguably simplest non-convex architecture, has not been proven in full generality, even though there has been recent progress [20, 69]. As a consequence, convergence results often have to be read in a comparative manner, and the results in this thesis will make no exception.

In the practical world, `SGD` and `Adam` are established as default methods for training, despite the proposal of many other algorithmic schemes for which it has been claimed that they improve these baselines. The performance of these optimization algorithms depends largely on the tuning of their hyperparameters – for example, the learning rate or momentum coefficient – and their optimal values can be different for each model and dataset [25, 137]. This tuning procedure is extremely cumbersome for machine learning practitioners, as well as computationally expensive given that even for modern research problem scales, one single training run can easily take several hours on a GPU. In the recent years, inspired by the insights of extensive benchmarking studies [25, 137], more focus has been put on addressing these issues by making existing algorithms more robust and easier to use [28, 67, 107]. Continuing this research direction, a recurring theme throughout this thesis will be to design and analyze optimization methods that need less tuning.

Training Machine Learning Models with Stochastic Optimization. Most of the thesis deals with problems of the form (1), or more generally, the regularized problem

$$\min_{x \in \mathbb{R}^n} f(x) + \varphi(x), \quad f(x) := \mathbb{E}_{s \sim P}[f(x; s)], \tag{2}$$

where φ is a convex, possibly non-smooth regularization function. Classical statistical learning problems often employed the ℓ_1- or ℓ_2-norm for φ, in order to obtain (group) sparse solutions [58, 143]. However, the standard (explicit) regularization function of modern deep learning is the squared ℓ_2-norm, also called *weight decay* [77, 161].

Studying and developing stochastic optimization methods for problems of the form (2) is at the core of this thesis. In Chapter 2 we outline classical results for proximal stochastic gradient descent which can be considered the most profoundly studied method for this problem class. We also present recent algorithmic advancements, such as variance-reduced methods like `SAGA` or `SVRG` [27, 56, 68, 71, 157], and adaptive methods like `Adam` or `AdaGrad` [36, 74, 89, 94]. The chapter ends with the concept of model-based stochastic optimization [6, 26, 37], which comprises many known schemes and lays the groundwork for novel methods we develop in Chapters 4 and 5.

Stochastic Proximal Point with Variance Reduction. The proximal point method [124, 125] can be considered an implicit version of the proximal gradient method. Only very recently, the *stochastic* proximal point method has been analyzed [6, 12, 17, 26, 114, 152], however mostly as a theoretical method in absence of an efficient way to compute the implicit step. In Chapter 3, we aim to make stochastic proximal point a *practical and implementable* method. This is achieved by deriving a dual formulation of the implicit update, and solving it via an efficient, globalized semismooth Newton method. This technique is related to the algorithmic development of highly efficient (deterministic) methods in [84, 164, 167]. Our setup consists of a regularized, finite-sum problem formulation where each summand is the composition of a linear mapping and a nonlinear loss function. Our framework further allows to incorporate variance reduction into stochastic

proximal point. We establish convex and non-convex convergence results that match the rates of `SVRG`, taking into account the inexactness of our update. An extension of our approach for problems with the composition of two non-linear functions is provided as well. The main content of Chapter 3 is based on the article [98].

Stochastic Proximal Polyak Step Size. Boris Polyak proposed a step size rule for problems where the optimal value is known [117]. This idea witnessed a surprising revival for training machine learning models with `SGD`, where the optimal value can often be guessed due to overparametrization and the use of non-negative loss functions [59, 88, 91, 109]. For regularized problems of form (2), this poses the question how to handle the additional term $\varphi(x)$. In Chapter 4 we propose a proximal version of the (stochastic) Polyak step size (`ProxSPS`). We establish convergence theory for convex and non-convex *regularized* problems that extend the existing theory for stochastic Polyak step sizes. The main focus lies on problems with squared ℓ_2-regularization, where `ProxSPS` has a closed-form update. Our experiments show that `ProxSPS` is favourable in comparison to previous ways of handling the regularization term, and competitive in comparison to `SGD` or `Adam` while being less sensitive to hyperparameter choices. Chapter 4 is based on the article [132].

Adaptive Learning Rates for Momentum Methods. Connecting to the previous paragraph, an important open question is how to combine momentum and adaptive step sizes like the one of Polyak [116, 117]. This is mainly motivated by the fact that momentum methods in general are superior in practice for deep learning [148]. In Chapter 5 we propose an answer to this question, and derive adaptive learning rates for `SGD` with momentum, which we call `MoMo`. The key insight is to re-interpret momentum as an averaging of linearizations of the loss around past iterates. This naturally connects momentum to model-based stochastic optimization, allowing us to combine it with the model-based viewpoint of the Polyak step size, which we have already used in Chapter 4. Even more so, incorporating a preconditioning matrix in the model-based update, we obtain Polyak-type learning rates for *any* momentum method, most importantly `Adam`, within one single framework. `MoMo` strongly improves the stability with respect to the learning-rate value, and often leads to more accurate models when compared to the baseline methods at fixed tuning budget. From various experiments across different deep learning tasks, models, and datasets, we conclude that `MoMo` reduces the tuning for `SGD` with momentum or `Adam`. Chapter 5 is based on the article [134]. An implementation of the method is provided in [133].

Structure of this thesis. The first chapter presents the mathematical tools and concepts that will be used throughout. In Chapter 2, we set the stage by describing the central problem setup and summarizing classical methods and results of stochastic optimization. We further provide a short review of machine learning landmarks in Section 2.1, situating the context for optimization problems we are interested in.

The third, fourth and fifth chapter constitute the main contributions of this thesis. Chapter 3 presents a practical stochastic proximal point method for finite-sum problems. Chapter 4 and Chapter 5 are closely related to one another, both dealing with stochastic Polyak step sizes and showing how they can be extended to regularized problems and momentum.

All of these three chapters share a common basic structure: each chapter contains a detailed *Introduction* to the specific topic, a section with *Background and Contributions*, as well as a short section *Conclusions and Open Questions* in the end. In each chapter, we defer technical proofs and supplementary material to the end.

Chapter 1

Background and Preliminaries

In this chapter, we introduce formal definitions and basic properties of the mathematical concepts that we will need throughout the thesis. There exist several standard textbooks which contain a complete presentation of the selected topics. For convex analysis, we refer to the book by Rockafellar [123], by Beck [9], by Bauschke and Combettes [8], by Hiriart-Urruty and Lemaréchal [61], and by Nesterov [106]. More advanced topics, particularly on the concept of subdifferentials, can be found in the book by Rockafellar and Wets [126]. Regarding the Clarke subdifferential and semismoothness, we point to the book by Clarke [22], by Ulbrich [153], and the books of Facchinei and Pang [42, 43].

We also refer to the table of notation given on page xi.

1.1 Clarke Subdifferential

We say that a function F is locally Lipschitz near x if F is Lipschitz continuous in a neighborhood of x. Let $U \subset \mathbb{R}^n$ be nonempty, open and let $F : U \to \mathbb{R}^m$. Denote by $\Omega_F \subset U$ the set of points where F is differentiable. Rademacher's Theorem states that if F is locally Lipschitz, then the set $U \setminus \Omega_F$ has Lebesgue measure zero and hence F is differentiable almost everywhere. If $x \in \Omega_F$, we denote by $DF(x)$ the (Frèchet) derivative of F at x. This leads us to the following definition:

Definition 1.1. *Let $U \subset \mathbb{R}^n$ be nonempty, open and let $F : U \to \mathbb{R}^m$ be locally Lipschitz near x. Then, for $x \in U$ we define*

$$\partial_B F(x) := \{M \in \mathbb{R}^{m \times n} \mid \exists (x_k)_{k \in \mathbb{N}} \subset \Omega_F,\ x_k \to x,\ DF(x_k) \to M,\ k \to \infty\}.$$

The Clarke subdifferential of F at $x \in U$ is given by $\partial F(x) := \operatorname{conv}\big(\partial_B F(x)\big)$.

The set $\partial_B F$ is called *Bouligand subdifferential*; the *Clarke subdifferential* has been introduced in the seminal work [22]. We state a few useful properties: from the definition we see that if F is continuously differentiable in a neighborhood of x then $\partial F(x) = \{DF(x)\}$. Further, if F_1 and F_2 are locally Lipschitz near x and regular in the sense of [22, Def. 2.3.4], then $\partial(F_1 + F_2)(x) = \partial F_1(x) + \partial F_2(x)$ [22, Cor. 3, Thm. 2.5.1]. In particular, convex functions and continuously differentiable functions are regular [22, Prop. 2.3.6, Cor. on p. 32].

1.2 Convexity and Fenchel Conjugate

An extended real-valued function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called convex if for all $x, y \in \operatorname{dom}(\varphi) \subset \mathbb{R}^n$ and all $\lambda \in [0,1]$ it holds $\varphi(\lambda x + (1-\lambda)y) \leq \lambda\varphi(x) + (1-\lambda)\varphi(y)$. The domain $\operatorname{dom}(\varphi)$ of a convex function φ is a convex set [8, Prop. 8.2, Prop. 8.4]. The function φ is called ρ-weakly convex for $\rho \geq 0$ if $\varphi + \frac{\rho}{2}\|\cdot\|^2$ is convex. The function φ is called μ-strongly convex for $\mu \geq 0$ if $\varphi - \frac{\mu}{2}\|\cdot\|^2$ is convex.

Let φ be proper and convex. For $x \in \operatorname{dom}(\varphi)$, the vector v is a subgradient (cf. [9, Def. 3.1]) of φ at x if and only if

$$\varphi(y) \geq \varphi(x) + \langle v, y - x\rangle \quad \text{for all } y \in \mathbb{R}^n.$$

The set of all subgradients is called (convex) subdifferential of φ at x and denoted by $\partial\varphi(x)$. If $x \notin \operatorname{dom}(\varphi)$, we define $\partial\varphi(x) = \emptyset$. The sum rule of subdifferential calculus states: if $\varphi_1, \ldots, \varphi_m$ are proper, convex functions, and if $\cap_{i=1}^m \operatorname{ri}(\operatorname{dom}(\varphi_i)) \neq \emptyset$, then for any $x \in \mathbb{R}^n$ it holds $\partial\Big(\sum_{i=1}^m \varphi_i\Big)(x) = \sum_{i=1}^m \partial\varphi_i(x)$ [123, Thm. 23.8].

For a convex, real-valued function $\varphi : \mathbb{R}^n \to \mathbb{R}$ its convex subdifferential is equal to its Clarke differential up to transposition [22, Thm. 2.5.1, Thm. 2.2.7].

Next, we will introduce a more general notion of a subdifferential for non-convex functions. Consider a function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$. If $\varphi(x) = +\infty$ we define $\partial\varphi(x) = \emptyset$. If $\varphi(x)$ is finite, we define the subdifferential of φ at x as the set of all vectors $v \in \mathbb{R}^n$ satisfying

$$\varphi(y) \geq \varphi(x) + \langle v, y - x\rangle + o(\|y - x\|) \quad \text{as } y \to x.$$

The above is known as the *regular subdifferential* [126, Def. 8.3]. We will summarize several properties: first, if x is a local minimum of φ, then by definition it holds $0 \in \partial\varphi(x)$. If φ is differentiable in x, then $\partial\varphi(x) = \{\nabla\varphi(x)\}$ [126, Ex. 8.8 (a)]. If $\varphi = \varphi_1 + \varphi_2$, $x \in \operatorname{dom}(\varphi_1)$, and φ_2 is continuously differentiable in a neighborhood of x, then due to [126, Ex. 8.8 (c)] we have

$$\partial\varphi(x) = \partial\varphi_1(x) + \nabla\varphi_2(x). \tag{1.1}$$

Moreover, if φ is convex, then the regular subdifferential coincides with the convex subdifferential [126, Prop. 8.12].

Now, let $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ be ρ-weakly convex and hence $\hat{\varphi} := \varphi + \frac{\rho}{2}\|\cdot\|^2$ is convex. Then for any $x \in \operatorname{dom}(\varphi)$, the subdifferential of φ can be characterized by

$$\partial\varphi(x) = \{v \in \mathbb{R}^n \mid v = w - \rho x, \quad w \in \partial\hat{\varphi}(x)\}. \tag{1.2}$$

Here, $\partial\hat{\varphi}$ is the convex subdifferential. The above result follows from the fact that $\partial\big(-\frac{\rho}{2}\|x\|^2\big) = \{-\rho x\}$ and (1.1). We also refer to [26, Lem. 2.1].

As weakly convex, real-valued functions are a sum of a convex and a continuously differentiable function, they are regular functions (cf. [22, Def. 2.3.4, Prop. 2.3.6]). Due to the sum rule for the Clarke subdifferential for regular functions, we obtain that the Clarke subdifferential coincides with the characterization in (1.2).

As shown, the regular and Clarke (and convex) subdifferential all coincide for real-valued, weakly convex (convex) functions; hence, we will always denote the subdifferential of φ by $\partial\varphi$ and it will be clear from the context which definition can be applied.

For a function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ its (Fenchel) conjugate function is defined as

$$\varphi^*(z) = \sup_{y \in \mathbb{R}^n} \langle z, y \rangle - \varphi(y). \tag{1.3}$$

The following result on the gradient of the Fenchel conjugate of a strongly convex function essentially follows from Danskin's theorem.

Proposition 1.2. *Let $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper and closed. If g is μ-strongly convex, then its conjugate g^* is closed, proper, convex, and Fréchet differentiable and its gradient is given by $\nabla g^*(x) = \arg\max_{z \in \mathbb{R}^n} \langle x, z \rangle - g(z)$. In addition, $\nabla g^* : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous with Lipschitz constant μ^{-1}.*

Proof. The first part is a consequence of [9, Cor. 4.21] and [8, Prop. 17.36]. The remaining properties follow from [8, Prop. 13.11, Thm. 13.32, and Thm. 18.15]. □

1.3 Proximal Operator and Moreau Envelope

Definition 1.3. *Let $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper, closed, and ρ-weakly convex. Suppose that $M \in \mathbb{S}_{++}$ satisfies $\rho < \lambda_{\min}(M)$. Then, the proximal operator is defined as*

$$\operatorname{prox}_{\varphi}^{M}(x) := \arg\min_{z \in \mathbb{R}^n} \varphi(z) + \frac{1}{2}\|z - x\|_M^2. \tag{1.4}$$

Moreover, we write $\operatorname{prox}_{\varphi}(x) := \operatorname{prox}_{\varphi}^{\mathbf{Id}}(x)$ if $\rho < 1$.

We make a few remarks on the above definition. The proximal operator has been studied in many standard textbooks on convex analysis and optimization such as [8, 9, 126]. It is often introduced for convex functions and the standard Euclidean norm (e.g. [9, Def. 6.1]). An overview of the proximal operator for convex functions but arbitrary norms is given in [97, Sec. 3]. The condition $\rho < \lambda_{\min}(M)$ is simply due to the fact that the objective of (1.4) needs to be strongly convex in order for $\operatorname{prox}_{\varphi}^{M}(x)$ to be well-defined. In particular, $\operatorname{prox}_{\varphi}(x)$ is well-defined for any closed, proper, and convex function φ. It is easy to see that by rescaling we obtain $\operatorname{prox}_{\alpha\varphi}(x) = \operatorname{prox}_{\frac{1}{\alpha}\varphi}^{\mathbf{Id}}(x)$. Thus, if φ is ρ-weakly convex, then $\operatorname{prox}_{\alpha\varphi}(x)$ is well-defined for any $\alpha < \rho^{-1}$.

Assume in the following that $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is proper, closed, and ρ-weakly convex and $M \in \mathbb{S}_{++}$ satisfies $\rho < \lambda_{\min}(M)$. The objective of problem (1.4) is strongly convex. Using the necessary and sufficient first-order optimality condition [9, Thm. 3.63] together with (1.2), we can characterize the proximal operator by

$$y = \operatorname{prox}_{\varphi}^{M}(x) \iff 0 \in \partial\varphi(y) + M(y - x). \tag{1.5}$$

If we have additionally that $\varphi = \varphi_1 + \varphi_2$ where φ_1 is ρ-weakly convex and $\varphi_2 \in \mathcal{C}^1$, then it holds

$$\begin{aligned} y = \operatorname{prox}_{\varphi}^{M}(x) &\iff 0 \in \partial\varphi_1(y) + \nabla\varphi_2(y) + M(y - x) \\ &\iff 0 \in \partial\varphi_1(y) + M[y - (x - M^{-1}\nabla\varphi_2(y))] \\ &\overset{(1.5)}{\iff} y = \operatorname{prox}_{\varphi_1}^{M}(x - M^{-1}\nabla\varphi_2(y)). \end{aligned} \tag{1.6}$$

In particular, if $M = \frac{1}{\alpha}\mathbf{Id}$ with $\alpha < \rho^{-1}$ we obtain the implicit characterization

$$y = \operatorname{prox}_{\alpha\varphi}(x) \iff y = \operatorname{prox}_{\alpha\varphi_1}(x - \alpha\nabla\varphi_2(y)). \tag{1.7}$$

If φ is closed, proper, convex, the proximal operator is firmly nonexpansive [9, Thm. 6.42], i.e.,

$$\|\operatorname{prox}_\varphi(x) - \operatorname{prox}_\varphi(y)\|^2 \leq \langle x - y, \operatorname{prox}_\varphi(x) - \operatorname{prox}_\varphi(y)\rangle \quad \forall\, x, y \in \mathbb{R}^n. \tag{1.8}$$

In particular, $\operatorname{prox}_\varphi$ is then Lipschitz continuous with constant 1.

A concept closely related to the proximal operator is the Moreau envelope [100].

Definition 1.4. *Let $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper, closed, and convex and let $\alpha > 0$. Then, the Moreau envelope is defined as*

$$\operatorname{env}_\varphi^\alpha(x) := \min_{z\in\mathbb{R}^n} \varphi(z) + \frac{1}{2\alpha}\|z - x\|^2. \tag{1.9}$$

We will further use the notation $\operatorname{env}_\varphi(x) := \operatorname{env}_\varphi^1(x)$ and remark that $\operatorname{env}_{\alpha\varphi} = \alpha \cdot \operatorname{env}_\varphi^\alpha$. The Moreau envelope is continuously differentiable and its derivative is given by [9, Thm. 6.60], i.e.

$$\nabla\operatorname{env}_\varphi^\alpha(x) = \frac{1}{\alpha}(x - \operatorname{prox}_{\alpha\varphi}(x)). \tag{1.10}$$

Due to (1.5) it holds

$$\hat{x} = \operatorname{prox}_{\alpha\varphi}(x) \Longrightarrow \exists v \in \partial\varphi(\hat{x}) : \quad \|v\| = \alpha^{-1}\|\hat{x} - x\| = \|\nabla\operatorname{env}_\varphi^\alpha(x)\|,$$

which explains why the gradient of the Moreau envelope can be interpreted as a measure of stationarity at $\hat{x}$ [26]. Definition 1.4 and its properties above can be extended to φ being ρ-weakly convex as long as $\alpha \in (0, \rho^{-1})$, since $\varphi + \frac{\rho}{2}\|\cdot\|^2$ is convex (see [26, Lem. 2.2] and Lemma 1.8 for details).

1.4 Smoothness and Semismoothness

A proper convex function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called *essentially differentiable*, if φ is differentiable on $\operatorname{int}(\operatorname{dom}(\varphi)) \neq \emptyset$, and $\lim_{j\to\infty} \|\nabla\varphi(x_j)\| = +\infty$ for any sequence (x_j) converging to a boundary point of $\operatorname{int}(\operatorname{dom}(\varphi))$ [49, p. 2]. If φ is closed, proper, convex and essentially differentiable, then we have $\partial\varphi(x) = \emptyset$ if $x \notin \operatorname{int}(\operatorname{dom}(\varphi))$ [123, Thm. 26.1].

A function $f \in \mathcal{C}^1$ is called *L-Lipschitz smooth* (or *L-smooth*) if its gradient is Lipschitz continuous with constant L, i.e. it holds

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \text{for all } x, y \in \mathbb{R}^n.$$

We will sometimes use the common abbreviation *L-smoothness*. A fundamental property of L-smooth functions is the *descent lemma* which states as follows.

Lemma 1.5 (Lem. 5.7 in [9])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be L-Lipschitz smooth over a convex set $U \subset \mathbb{R}^n$. Then, for any $x, y \in U$ it holds*

$$f(y) \leq f(x) + \langle\nabla f(x), y - x\rangle + \frac{L}{2}\|x - y\|^2. \tag{1.11}$$

The gradient norm of L-smooth functions can be bounded as follows.

Lemma 1.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be L-smooth over $\mathbb{R}^n$ and bounded from below, i.e. $\inf f > -\infty$. Then, it holds*

$$f(x) - \inf f \geq \frac{1}{2L}\|\nabla f(x)\|^2 \quad \text{for all } x \in \mathbb{R}^n.$$

Proof. Using (1.11), we obtain that for all $x, y \in \mathbb{R}^n$ it holds

$$\inf f \leq f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|x - y\|^2.$$

The right-hand side is minimized at $y = x - \frac{1}{L}\nabla f(x)$. Plugging in yields

$$\inf f \leq f(x) - \langle \nabla f(x), \tfrac{1}{L}\nabla f(x)\rangle + \tfrac{L}{2}\|\tfrac{1}{L}\nabla f(x)\|^2 = f(x) - \tfrac{1}{2L}\|\nabla f(x)\|^2.$$

□

The notion of semismooth mappings dates back to Mifflin [96]. Its relevance and broad applicability stems from the fact that it allows to generalize Newton's method to nonsmooth operators: for solving the equations $F(x) = 0$, Newton's method converges locally q-superlinearly if F is semismooth [119, 153].

Definition 1.7. *Let $F : V \to \mathbb{R}^m$ be locally Lipschitz on a nonempty, open set $V \subset \mathbb{R}^n$. F is called* semismooth *at $x \in V$ (with respect to ∂F) if F is directionally differentiable at x and if it holds*

$$\sup_{M \in \partial F(x+s)} \|F(x+s) - F(x) - Ms\| = o(\|s\|) \quad \text{as } s \to 0. \tag{1.12}$$

*If F is semsimooth at every $x \in V$, then F is called semismooth (on V). Further, for $0 < \nu \leq 1$, F is called ν-*order semismooth *(*strongly semismooth *for $\nu = 1$) at $x \in V$, if F is ν-order B-differentiable at x (cf. [153, Def. 2.6 (c)]) and we have*

$$\sup_{M \in \partial F(x+s)} \|F(x+s) - F(x) - Ms\| = \mathcal{O}(\|s\|^{1+\nu}) \quad \text{as } s \to 0. \tag{1.13}$$

Next, we summarize a few useful results on semismoothness; we refer to the book by Ulbrich [153, Chapter 2] and references therein for more details. If F is a piecewise $\mathcal{C}^1(\mathcal{C}^2)$-function it is semismooth (strongly semismooth) [153, Prop. 2.26]. The classes of semimsooth and ν-order semismooth functions are closed under composition [153, Prop. 2.9, Prop. 2.16].

In Definition 1.7 we did not specify the subdifferential, but the natural choice is the Clarke subdifferential (see also [153]). However, the Clarke subdifferential does not admit a chain rule (with equality) in general. If $F : \mathbb{R}^p \to \mathbb{R}^n$, $G : \mathbb{R}^n \to \mathbb{R}^p$ are semismooth at $G(x)$ and x respectively, then for the composition $(F \circ G)(x)$ condition (1.12) holds for the surrogate subdifferential $\{VW | V \in \partial F(G(x)),\ W \in \partial G(x)\}$ [43, Thm. 7.5.17].

1.5 Supplementary Results

Lemma 1.8. *Let $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper, closed, and ρ-weakly convex function. Then, for $\alpha < \rho^{-1}$ the Moreau envelope $\mathrm{env}_\varphi^\alpha$ given by (1.9) is well-defined and it holds*

$$\nabla \mathrm{env}_\varphi^\alpha(x) = \frac{1}{\alpha}(x - \mathrm{prox}_{\alpha\varphi}(x)).$$

Proof. Clearly, for $\alpha < \rho^{-1}$ the objective function in (1.9) is strongly convex and hence $\mathrm{env}_\varphi^\alpha$ is well-defined. Denote $\hat{\varphi} := \varphi + \frac{\rho}{2}\|\cdot\|^2$ and hence $\hat{\varphi}$ is convex. Define $\mu := \frac{1}{1-\rho\alpha} > 0$. For $x, y \in \mathbb{R}^n$ we have

$$\begin{aligned}\|y - x\|^2 &= \|y - \mu x\|^2 + (\mu-1)^2\|x\|^2 + 2\langle y - \mu x, (\mu-1)x\rangle \\ &= \|y - \mu x\|^2 + 2(\mu-1)\langle x, y\rangle + (1-\mu^2)\|x\|^2.\end{aligned}$$

Further, it holds

$$\begin{aligned}\varphi(y) + \frac{1}{2\alpha}\|y - x\|^2 &= \varphi(y) + \frac{\rho}{2}\|y\|^2 + \frac{1-\rho\alpha}{2\alpha}\|y - x\|^2 - \rho\langle x, y\rangle + \frac{\rho}{2}\|x\|^2 \\ &= \hat{\varphi}(y) + \frac{1}{2\alpha\mu}\|y - \mu x\|^2 \underbrace{+\frac{1-\mu^2}{2\alpha\mu}\|x\|^2 + \frac{\rho}{2}\|x\|^2}_{\text{cst.}(y)},\end{aligned}$$

where we used that $\frac{2(\mu-1)}{2\alpha\mu} = \frac{\rho}{\mu(1-\rho\alpha)} = \rho$. Simple calculations show $\frac{1-\mu^2}{2\alpha\mu} + \frac{\rho}{2} = -\frac{\rho\mu}{2}$. Minimizing both sides in y, we get that $\mathrm{prox}_{\alpha\varphi}(x) = \mathrm{prox}_{\alpha\mu\hat{\varphi}}(\mu x)$ and from the definition of the Moreau envelope

$$\mathrm{env}_\varphi^\alpha(x) = \mathrm{env}_{\hat{\varphi}}^{\mu\alpha}(\mu x) - \frac{\rho\mu}{2}\|x\|^2.$$

Differentianting both sides in x and applying [9, Thm. 6.60] on $\mathrm{env}_{\hat{\varphi}}^{\mu\alpha}(\mu x)$, we get

$$\nabla\mathrm{env}_\varphi^\alpha(x) = \mu\frac{1}{\mu\alpha}\big[\mu x - \mathrm{prox}_{\mu\alpha\hat{\varphi}}(\mu x)\big] - \rho\mu x = \frac{1}{\alpha}\big[\mu(1-\alpha\rho)x - \mathrm{prox}_{\alpha\varphi}(x)\big].$$

As $\mu(1-\alpha\rho) = 1$, the claim follows. □

Chapter 2

Classical Results of Stochastic Optimization for Machine Learning

This thesis deals with optimization problem where the objective function consists of an expected value. Stochastic optimization encompasses algorithms that, in each iteration, can not access the expectation-valued objective function or its derivatives. Instead, the methods we consider can access only random samples of the objective (or its gradient); as a consequence the iterates of the algorithm become random as well. The most famously known method in this framework is stochastic gradient descent (`SGD`) which is often attributed to Robbins and Monro [122] and Kiefer and Wolfowitz [73].

Organization of the chapter. We start by embedding stochastic optimization into the context of training machine learning models, which is the main application we are interested in for this thesis. In the main part of this chapter, we first establish the problem setup of stochastic optimization, and then explain the notion of stochastic oracles and relate this to empirical risk minimization problems. Subsequently, a summary of convergence results for the famous (proximal) stochastic gradient method is provided. This serves as baseline for more recent optimization techniques such as variance reduction, adaptive methods and model-based optimization. The latter will be the central building block and guiding principle for much of the ideas, algorithm designs and proof techniques that are presented in subsequent chapters.

Most (if not all) contents of this chapter are widely established and known results in the optimization community. The reader who is familiar with this literature can comfortably jump ahead to subsequent chapters which contain the main contributions of this thesis.

2.1 Optimization in the Context of Machine Learning

Suppose that we are given a dataset $(y_i)_{i=1,\dots,N}$ where $y_i = (y_i^{\text{in}}, y_i^{\text{out}})$ is a pair of input-output data. Typically, the input y_i^{in} can be an image, a sentence of words, or a vector of known numerical features, while y_i^{out} is the target or label, for example an image class, the translation of a sentence, or a numerical quantity of interest that we want to predict.

Given a single data pair $y = (y^{\text{in}}, y^{\text{out}})$, the goal of *supervised learning* is to train a (parametrized) model $\texttt{forward}_x(y^{\text{in}})$ which performs well at predicting y^{out}. Here, we denote with $x \in \mathbb{R}^n$ the

vector of all (learnable) parameters of `forward`, thus we are free to choose the values of x.[1]

If there are no targets y^{out} given, an alternative task is to learn only from the input data, for example to identify structures in an image (e.g. auto-encoder architectures) or to predict the next word of a sentence (e.g. large language models such as `GPT` [18]). This is often referred to as *unsupervised* or *self-supervised* learning [81, 160].

Machine learning is connected to optimization by the fact that for training the model we have to find the best choice of parameters x in order to perform well at the respective learning task. For example, choose a loss function `loss` that compares the model output $\texttt{forward}_x(y^{\text{in}})$ and the target y^{out}. From this, we naturally obtain an optimization problem, namely minimizing $\texttt{loss}(\texttt{forward}_x(y^{\text{in}}), y^{\text{out}})$, averaged over the training dataset, with respect to the learnable parameters x. In a self-supervised setting, where no targets y^{out} are given, the objective is typically a function of the form $\texttt{loss}(\texttt{forward}_x(y^{\text{in}}), y^{\text{in}})$ instead.[2]

In classical statistical learning, much focus was put on *(generalized) linear models* [58]. As the name suggests, the principal idea is to restrain the model to be a linear function of the input: we have $\texttt{forward}_x(y^{\text{in}}) = \langle x, y^{\text{in}} \rangle$ (wlog. an intercept can be modeled by adding a feature of ones to y^{in}). However, it is clear that such an approach can not capture nonlinear effects; *generalized linear models* address this by using pre-computed features, derived from y^{in}, as input. This technique is closely related to kernel methods [57, 138]. Generalized linear models are often combined with regularization. For example, it can be desirable to obtain a sparse x, meaning that the final learned parameter vector contains many zeros; sparsity is useful in high-dimensional learning problems and/or if the practitioner seeks for interpretability. Using the ℓ_1-norm as regularization function, we obtain the famously known *Lasso* for sparse regression [150]. We refer to textbooks on statistical learning [57, 58] for a more detailed introduction.

Generalized linear models have several short-comings: first, if we want to capture nonlinear effects, it is necessary to manually engineer features from the raw input data which often requires domain expertise. One paradigm of modern deep learning is to *learn* these features instead of pre-selecting them [81]. Second, for certain types of input data additional structural information can be leveraged; for example, stacking all pixel values of an image would ignore pixel position, as small patches of pixels typically contain (local) information of the image.

In the last decades, the increasing availability of data and computational resources led to an enormous progress in machine learning. Arguably this would not have been possible without fundamental contributions in numerical methods, optimization and development of open-source software frameworks such as `Pytorch` [113] or `Tensorflow` [1]. We refer the reader to [81] for a detailed review of the deep learning (r)evolution. Below we give a short list of influential developments regarding model architectures. We want to stress that this list is by far not complete – it mainly serves to give historical context for the architectures which will be used later in our numerical experiments.

Multi-Layer Perceptron (`MLP`). At least since the 1960's, scientists developed learning techniques based on the functional principles of the human brain [3, 66, 130]. `MLP` architectures consists of several layers, where each layer is a linear transformation followed by a (nonlinear)

[1] The standard notation in machine learning literature would be to use w or θ for learnable parameters and to use $(x_i, y_i)_{i=1,\dots N}$ for input and output data. However, as the topics of this thesis are mainly situated in the field of optimization, we use the standard notation where x is the optimization variable.

[2] From an optimization perspective, the setting of supervised and self-/unsupervised learning is very similar and hence we will focus on the supervised setting in our examples.

activation function: in particular, we multiply a weight matrix (which is part of x) by the output of the previous layer (plus a bias term). Then, we apply a nonlinear activation function elementwise on the result of this linear transformation. Commonly used activation functions are $\texttt{Sigmoid}(z) = \frac{1}{1+\exp(-z)}$ and $\texttt{ReLU}(z) = (z)_+$ or variants thereof.

Convolutional Neural Networks (CNN). If the input data is an image, it has proven successful to leverage the structure of images for designing the model architecture. Convolutions can be understood as inner products of small patches (e.g. five by five pixels) with (learnable) kernels of the same size. This technique is able to detect local patterns (e.g. edges) which is then aggregated globally (e.g. by pooling) [81]. From the pioneering work of Fukushima [45] in 1980 to today, architectures have become simultaneously deeper (by stacking several convolutional layers) and more powerful, with `LeNet` [80], `DanNet` [21], `AlexNet` [76], and `VGG` [144] along the way. Later, `ResNet` models added residual connections to avoid training instabilities [60] and remained the gold standard until the rise of the vision transformer (see below).

Transformer. Language and text data is sequential by nature. Thus, the type of architecture for language tasks (e.g. translation) is distinct from the ones used for image data. Most modern language architectures are based on the transformer [154]. The main distinction of transformer architectures to previous approaches consists in its design centered around the attention mechanism. In short, the model learns for each text element (called *token*) which parts of the remaining sentence are important, for example, in order to predict the next word. We refer to [154] for a detailed description of the architecture as well as for historical background. Ever since this breakthrough has been achieved in language modeling, the attention mechanism was subsequently used in other domains, such as the vision transformer for image classification [32] or for diffusion models used in image-generation [127].

2.2 Problem Setup

Having described how optimization problems arise naturally in machine learning tasks, we can now formulate a problem setting. Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be given by

$$f(x) := \int_{\mathcal{S}} f(x; s) dP(s) = \mathbb{E}[f(x; S)], \tag{2.1}$$

where $\mathcal{S}$ is a sample space (or sample set) and $P(s)$ is the associated probability measure. We assume that $f(x; s)$ is measurable with respect to $P(s)$ for fixed $x \in \mathbb{R}^n$. We assume that for each $s \in \mathcal{S}$ the function $f(\cdot; s) : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz. If not specified otherwise, we denote by $\partial f(x; s)$ the Clarke subdifferential of $f(\cdot; s)$ at $x \in \mathbb{R}^n$.

Throughout this thesis, we will consider problems of the form

$$\min_{x \in \mathbb{R}^n} \psi(x), \quad \psi(x) := f(x) + \varphi(x), \tag{P}$$

where $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is a proper, closed, and convex function. We will refer to f as the *loss function* and to φ as the *regularization function*. For problems of form (P) without regularization, i.e. if $\varphi = 0$, the most famously known method is *stochastic gradient descent* (`SGD`).[3] It dates back to

[3] The name for this method has been established even though it is not a descent method [16].

the seminal works of Robbins and Monro [122] and Kiefer and Wolfowitz [73] in the 1950's and is given by

$$x^{k+1} = x^k - \alpha_k g_k, \quad g_k \in \partial f(x^k; S_k), \tag{SGD}$$

where $S_k \in \mathcal{S}$ and $\alpha_k > 0$ is a step size.

In the context of training machine learning models, x denotes the set of learnable parameters. The distribution of input data is given via $\mathcal{S}$. The loss function $f(x; s)$ for a single sample (typically a mini-batch) is our formal notion of the function $\texttt{loss}(\texttt{forward}_x(y^{\text{in}}), y^{\text{out}})$ from the previous section.

Notation. We denote elements of the sample space $\mathcal{S}$ by lowercase s; the capital letter S denotes a (possibly multivariate) random variable mapping to $\mathcal{S}$; its push-forward measure is given by $P(s)$. Realizations of S will be typically denoted by S_0, S_1, et cetera. We will write $\mathbb{E}$ whenever the associated distribution is P. Moreover, for any random variable $X(s)$ we denote $\mathbb{E}[X(S)] := \int_{\mathcal{S}} X(s) dP(s)$, i.e. the distribution of X is given implicitly by S.

Optimality and stationarity criteria. If ψ is non-convex, it is hard to prove convergence to a global minimum, but we might desire theoretical guarantees that an optimization method converges to a stationary point of ψ, i.e. a point such that zero is contained in the (regular) subdifferential $\partial\psi$. Thus, algorithms are often analyzed in terms of the *gradient mapping* [48], a quantity which we define below.

Proposition 2.1. *Let $f \in \mathcal{C}^1$ and φ be closed, convex. For $\alpha > 0$, define the gradient mapping*

$$\mathcal{G}_\alpha(x) := \frac{1}{\alpha}(x - \operatorname{prox}_{\alpha\varphi}(x - \alpha\nabla f(x))).$$

Then, $0 \in \partial\psi(x)$ if and only if $\mathcal{G}_\alpha(x) = 0$ for some $\alpha > 0$. Moreover, if f is L-smooth, then $\mathcal{G}_\alpha$ is $\frac{2+\alpha L}{\alpha}$-Lipschitz continuous.

Proof. For $\alpha > 0$ it holds $0 \in \partial\psi(x) \iff 0 \in \alpha(\nabla f(x) + \partial\varphi(x))$. Moreover, we have $\mathcal{G}_\alpha(x) = 0 \iff x = \operatorname{prox}_{\alpha\varphi}(x - \alpha\nabla f(x)) \iff 0 \in \alpha\nabla f(x) + \alpha\partial\varphi(x)$ from the optimality conditions of the proximal operator. For the second statement, using non-expansiveness of the proximal operator, for any x, y we have

$$\begin{aligned}\|\mathcal{G}_\alpha(x) - \mathcal{G}_\alpha(y)\| &= \frac{1}{\alpha}\|x - y + \operatorname{prox}_{\alpha\varphi}(x - \alpha\nabla f(x)) - \operatorname{prox}_{\alpha\varphi}(y - \alpha\nabla f(y))\| \\ &\leq \frac{1}{\alpha}(\|x - y\| + \|x - y + \alpha(\nabla f(y) - \nabla f(x))\|) \leq \frac{2 + \alpha L}{\alpha}\|x - y\|.\end{aligned}$$

□

For some methods, it might be more convenient to work with the gradient of the Moreau envelope as a measure of stationarity (see e.g. [26, Sec. 2.2]). It can be related to the gradient mapping by the following result.

Lemma 2.2 (Thm. 4.5 in [35])**.** *Let f be L-smooth and φ be closed, convex. For any $\alpha > 0$ and $x \in \mathbb{R}^n$, it holds*

$$\tfrac{1}{(1+\alpha L)(1+\sqrt{\alpha L})}\|\nabla \operatorname{env}_\psi^{\frac{\alpha}{1+\alpha L}}(x)\| \leq \|\mathcal{G}_\alpha(x)\| \leq \tfrac{1+2\alpha L}{1+\alpha L}\left(\sqrt{\tfrac{\alpha L}{1+\alpha L}} + 1\right)\|\nabla \operatorname{env}_\psi^{\frac{\alpha}{1+\alpha L}}(x)\|.$$

In particular, for $\alpha = 1/L$, it holds

$$\tfrac{1}{4}\|\nabla \operatorname{env}_\psi^{1/(2L)}(x)\| \leq \|\mathcal{G}_{1/L}(x)\| \leq \tfrac{3}{2}(1 + \tfrac{1}{\sqrt{2}})\|\nabla \operatorname{env}_\psi^{1/(2L)}(x)\| \quad \forall x \in \mathbb{R}^n.$$

2.3 Stochastic Oracles and Empirical Risk Minimization

Stochastic optimization is of interest when the computation of (parts of) the objective or its derivatives are computationally infeasible or expensive. For problems of form (P), this will typically be the case for f if the sample space $\mathcal{S}$ is large. Hence, we consider methods where in each iteration only a stochastic approximation of f or its derivatives is used. We call such a mechanism a *stochastic oracle*. In the setting presented in (2.1) we will assume that in the k-th iteration we can access $f(\cdot; S_k)$ and $g_k \in \partial f(x^k; S_k)$ where $S_k \in \mathcal{S}$ is drawn at random. Under suitable assumptions (cf. Lemma 2.3) we have that g_k is conditionally unbiased, i.e. it holds

$$\mathbb{E}[g_k | \mathcal{F}_k] \in \partial f(x^k), \tag{2.2}$$

where $\mathcal{F}_k$ is a suitable σ-algebra such that x^k is $\mathcal{F}_k$-measurable (cf. Assumption 1). We remark that the concept of stochastic oracles has been introduced in more general form [16, 47, 48, 103]. However, these works typically assume the unbiasedness property (2.2) in their analysis. Biased oracles are not presented in this thesis; we refer to [2] and references therein for an overview.

Lemma 2.3. *Let $f(\cdot; s)$ be ρ_s-weakly convex for each $s \in \mathcal{S}$. Let $f(x) = \mathbb{E}[f(x; S)]$ and assume that $\rho := \mathbb{E}[\rho_S] < \infty$. Then,*

i) f is ρ-weakly convex, and

ii) it holds $\partial f(x) = \mathbb{E}[\partial f(x; S)]$.[4] In particular, for every measurable selection $g(x; s) \in \partial f(x; s)$ we have $\mathbb{E}[g(x; S)] \in \partial f(x)$.

Proof. The first statement follows from applying [11, Lem. 2.1] to the convex function $f(\cdot; s) + \frac{\rho_s}{2}\|\cdot\|^2$. The second statement follows from [11, Prop. 2.2] applied to $f(\cdot; s) + \frac{\rho_s}{2}\|\cdot\|^2$ and (1.2). Alternatively, one could proof this using [22, Thm. 2.7.2] and the fact that weakly convex functions are regular. □

With machine learning applications in mind, why is it efficient to compute only approximations of the loss function or its gradient? In learning tasks, the overarching goal is to train a model that finds patterns in a given dataset; this contains the implicit assumption that such a pattern exists and hence many training samples should be similar. Stochastic methods can make use of this redundancy within individual loss functions $f(x; s)$ and decrease the per-iteration complexity.[5]

To conclude this section, we state assumptions for stochastic oracles that will be needed for the convergence theory later on.

Assumption 1. *It is possible to generate a sequence of i.i.d. realizations $S_0, S_1, \ldots \in \mathcal{S}$. We denote by $\mathcal{F}_k$ the filtration that is generated by $\{S_j \mid j = 0, \ldots, k-1\}$.*

Further, we state an assumption on global boundedness of the second moment (or the variance respectively[6]) of the stochastic (sub)gradient.

[4] See Eq. 13 in [11] for a formal definition of the expectation over a set.

[5] In fact, this is exactly how Yann Le Cun argues in his lecture around minute 19:15 in https://youtu.be/d9vdh3b787Y.

[6] More precisely, the quantity in (O2) is the trace of the covariance matrix of the vector-valued random variable $g(x; S)$.

Assumption 2. *Consider the setting from* (2.1).

(O1) There exists $\sigma > 0$ *such that* $\mathbb{E}\|g(x;S)\|^2 \leq \sigma^2$ *for every measurable selection* $g(x;s) \in \partial f(x;s)$ *and all* $x \in \mathbb{R}^n$.

(O2) There exists $\sigma > 0$ *such that for all* $x \in \mathbb{R}^n$ *and for every measurable selection* $g(x;s) \in \partial f(x;s)$ *it holds* $\mathbb{E}\|g(x;S) - \mathbb{E}[g(x;S)]\|^2 \leq \sigma^2$.

Due to $\mathbb{E}\|X - \mathbb{E}[X]\|^2 \leq \mathbb{E}\|X\|^2$, (O2) is (strictly) weaker than (O1). Moreover, if $f(\cdot;s)$ and f are continuously differentiable then $\mathbb{E}[\nabla f(x;S)] = \nabla f(x)$ in the setting of (2.1) and we can read (O2) as

$$\mathbb{E}\|\nabla f(x;S) - \nabla f(x)\|^2 \leq \sigma^2 \quad \text{for all } x \in \mathbb{R}^n.$$

For the rest of this chapter, we will work under Assumption 1. In contrast, we will state explicitly whenever we assume (O1) or (O2).

Empirical Risk Minimization. In practice, we typically are given a finite dataset of cardinality $N \in \mathbb{N}$ and a loss function of the form

$$f_i : \mathbb{R}^n \to \mathbb{R}, \quad f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x). \tag{2.3}$$

This is referred to as a *finite-sum loss* or *empirical risk*. In order to obtain (2.3) as a special case of (2.1), we can formalize it as:

$$\mathcal{S} = \{s_1, \ldots, s_N\},\ P(s_i) = \frac{1}{N},\ f(x;s_i) = f_i(x) \quad i = 1, \ldots, N. \tag{ER}$$

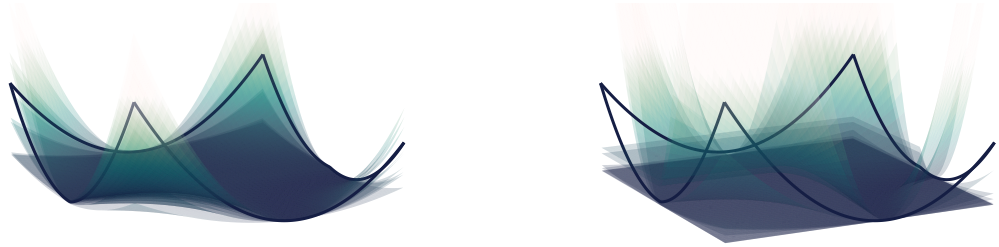
Clearly, for the above choice we have $\mathbb{E}[f(x;S)] = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$ where $S \sim P$. However, in the (ER) setting, sampling $S_k \in \mathcal{S}$ means that we can access only a single element of the dataset per iteration. In practice, it might be favorable to use a small fraction of the dataset instead. This is called *mini-batching* and can be formalized as follows: for $b \in [N]$, let

$$\begin{aligned} &\mathcal{S} = \{s = (i_1, \ldots, i_b) \mid i_j \in [N] \text{ with or without replacement}\},\ P(s) = \frac{1}{|\mathcal{S}|}\ \forall s \in \mathcal{S}, \\ &f(x;s) = \frac{1}{b}\sum_{j=1}^{b} f_{i_j}(x). \end{aligned} \tag{MB-ER}$$

In other words, we choose uniformly from the set of b-tuples of elements of $[N]$, where we either allow duplicate indices or not (*with or without replacement*). Thus, we either have $|\mathcal{S}| = N^b$ (with replacement) or $|\mathcal{S}| = \binom{N}{b}$ (without replacement). In both cases, it is easy to show that for (MB-ER) it again holds $\mathbb{E}[f(x;S)] = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$. Fig. 2.1 illustrates the principle of noise in stochastic optimization: we plot the actual loss $f(x)$ (dark blue border) and the sampled loss $f(x;s)$ (transparent surfaces). In this example, the generated dataset contains many similar samples and hence $f(x;s)$ captures roughly the landscape of $f(x)$, with the noise increasing as the batch size decreases.

2.4 Proximal Stochastic Gradient Descent

The goal of this section is to analyze *proximal stochastic gradient descent* (ProxSGD), stated in Algorithm 1, for solving problems of form (P).



(a) Batch size $b = 10$ (b) Batch size $b = 1$

Figure 2.1: Surface plot of loss function $f(x) = \frac{1}{2}\|y^{\text{out}} - x_2(x_1 \cdot y^{\text{in}})_+\|^2$ for randomly generated input data $y^{\text{in}} \in \mathbb{R}^N$, $N = 1000$, and targets $y^{\text{out}} = 1.5(0.5y^{\text{in}})_+ + 0.1 \cdot \varepsilon \in \mathbb{R}^N$ with $\varepsilon \sim \mathcal{N}(0, 1)$. In the spirit of the applications we have in mind, the loss resembles a simplified two-layer neural network. The dark blue border represents $f(x)$ while each transparent surface represents a sampled mini-batch loss $f(x; s)$ (cf. (`MB-ER`)).

Algorithm 1 `ProxSGD`

Require: Starting point $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$.

1: **for** $k = 0, \ldots, K-1$ **do**
2: Sample S_k and compute $g_k \in \partial f(x^k; S_k)$.
3: Update
$$x^{k+1} = \text{prox}_{\alpha_k \varphi}(x^k - \alpha_k g_k).$$
4: **end for**
5: **return** x^K

Some remarks on the above: we stated the algorithm such that it is also applicable for non-smooth loss functions $f(\cdot; s)$. However, the theoretical results we present later focus on the differentiable setting. Further, applying the definition of the proximal operator and adding the constant $f(x^k; S_k)$, we can rewrite the `ProxSGD` update as follows:

$$\begin{aligned} x^{k+1} &= \arg\min_{y \in \mathbb{R}^n} \varphi(y) + \frac{1}{2\alpha_k}\|y - (x^k - \alpha_k g_k)\|^2 \\ &= \arg\min_{y \in \mathbb{R}^n} \varphi(y) + f(x^k; S_k) + \langle g_k, y - x^k \rangle + \frac{1}{2\alpha_k}\|y - x^k\|^2. \end{aligned} \tag{2.4}$$

Thus, `ProxSGD` takes a proximal step of the mapping $y \mapsto f(x^k; S_k) + \langle g_k, y - x^k \rangle + \varphi(y)$ with step size α_k. We will come back to this fact in Section 2.7 in the context of model-based stochastic proximal point.

2.4.1 Convergence Results

We briefly summarize some convergence results of `ProxSGD`. A compact, but more illustrative overview of the following is Table 2.1.
Davis and Drusvyatskiy analyze `ProxSGD` for f being weakly convex and thus including the case of f being L-smooth [26]. Their analysis is based on $\nabla \text{env}_\psi^{1/(2L)}$ as stationarity criterion and the result stated in Table 2.1 is obtained using Lemma 2.2. For their result, the step sizes are

Assum. on f; ψ	Other	α_k	Result	Reference
L-smooth	(O2)	$\min\{\frac{1}{2L}, \mathcal{O}(K^{-1/2})\}$	$\mathbb{E}\|\mathcal{G}_{1/L}(\hat{x})\|^2 = \mathcal{O}(\frac{1}{\sqrt{K}})$	[26, Cor. 3.6]
		$\frac{1}{2L}$	$\mathbb{E}\|\mathcal{G}_{1/(2L)}(\hat{x})\|^2 = \mathcal{O}(\frac{1}{K}) + \text{cst.}(\sigma^2)$	[48, Cor. 3]
L-smooth, convex	(O2)	$\mathcal{O}(1/k^\beta)$, $\beta > 1/2$	$\mathbb{E}[\psi(\hat{x}) - \psi(x^\star)] = \mathcal{O}(\frac{1}{K^{1-\beta}})$	[48, Thm. 2]
L-smooth; str. convex	(O2)	$\mathcal{O}(1/k)$	$\mathbb{E}\|x^K - x^\star\|^2 = \mathcal{O}(\frac{1}{K})$	[129, Thm. 3.5]

Table 2.1: An overview of convergence results for `ProxSGD`. Here, we denote by $x^\star$ a solution to (P) and with $\hat{x}$ a point chosen randomly from the set of iterates $\{x^0, \dots, x^K\}$ (the distribution can be found in the respective references).

chosen constant, but depending on the maximum number of iterations K.[7] It can be interpreted to choose, a priori, a total number of iterations K in order to achieve a desired accuracy. On the other hand, Ghadimi et al. [48] use a constant step size but their result contains a constant term in σ^2. Therefore, exact convergence[8] can only be guaranteed if the variance of the stochastic oracle would be zero.

The convex case with diminishing step sizes is also covered in [48]. The established rate in terms of objective convergence is almost optimal (the optimal rate is $1/\sqrt{K}$ according to [103]). A similar result is given in [129, Thm. 3.4].
For the strongly convex case, i.e. assuming f to be convex and ψ to be strongly convex, the rate wrt. squared distance to the solution is still sublinear, but of order $1/K$ [129, Thm. 3.5]. Moreover, in this setting, and for step sizes $\alpha_k = \mathcal{O}(1/k)$, the iterates converge almost surely to the optimizer [129, Thm. 3.6]. Zhao and Zhang prove a rate of $\mathcal{O}(\frac{\ln(K)}{K})$ in [166, Thm. 1] for proximal steps with respect to general Bregman distances (however, their setting requires additional assumptions on φ).

For the special case $\varphi = 0$, the literature on `SGD` is exuberant and hence we focused on results for the regularized setting above. Having $\varphi = 0$ does not fundamentally improve the results or convergence rates, but often simplifies the proofs. We refer the interested reader to [46] for an overview of results and proof techniques. For this thesis, we confine ourselves to present one classical convergence result of `SGD` for the unregularized setting: if f is L-smooth and μ-strongly convex and under the assumption of (O2), Bottou, Curtis, and Nocedal [16, Thm. 4.6] showed that[9]

$$\mathbb{E}[f(x^K) - \min f] \leq (1 - \mu\alpha)^K \big(f(x^0) - \inf f\big) + \frac{\alpha L \sigma^2}{2\mu}, \tag{2.5}$$

for a constant step size $\alpha_k = \alpha = \mathcal{O}(\frac{1}{L})$. Hence, `SGD` converges linearly into a neighborhood of the solution whose size can be controlled by reducing the step size α.

[7] We will call results where the step size α_k does not depend on the maximum number of iterations *anytime results*. If this condition is not met, we call it a *non-anytime result*. Typically, non-anytime results can improve the rates up to a logarithmic factor.
[8] We will call results where convergence is shown up to a constant *non-exact convergence*.
[9] Their result is actually more general and also covers biased gradient oracles.

2.5 Variance Reduction

In ProxSGD, we introduce variance in each iteration by using g_k instead of $\nabla f(x^k)$. Thus, for constant step sizes $\alpha_k = \alpha$ the algorithm naturally can converge only to a neighbourhood of the minimizer (cf. (2.5)). As g_k is multiplied with α_k in each step, we can compensate for this with diminishing step sizes. For example, step sizes are often assumed to satisfy the *Robbins-Monro condition* (cf. [122] and [16, Thm. 4.7]) given by

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

The above is fulfilled for the choice $\alpha_k = \mathcal{O}(1/k^\beta),\ \beta > 1/2$. However, this has the drawback of taking progressively smaller steps and slow convergence as a consequence. Recent developments have tried to overcome this issue by variance reduction (VR). Conceptually, this comprises methods for which the variance of the gradient estimator diminishes when approaching the solution. A thorough review of these methods is given in [53]. The respective article also explains how SGD with momentum or increasing batch size can be interpreted as variance-reduced methods. In this section we first present a conceptual method for the purpose of illustrating the mechanism of variance reduction, and then introduce in detail two popular and practical VR schemes.

2.5.1 An Illustrative Variance-Reduced Method

In this section, let the loss be given in finite-sum structure (2.3), consider (ER) and let $\varphi = 0$. Assume that all f_i are differentiable and let $x^\star \in \arg\min_{x\in\mathbb{R}^n} f(x)$. For this setting, the method $\text{SGD}_\star$ proposed in [50] is given by

$$x^{k+1} = x^k - \alpha\big(\nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^\star)\big), \tag{$\text{SGD}_\star$}$$

where $\alpha > 0$ and $i_k \in [N]$ is drawn uniformly at random in each iteration. As we do not know $\nabla f_i(x^\star)$ in general, the above method is clearly not practicable. However, it nicely illustrates the principal idea of VR methods. The following considerations are taken from [53, Sec. E]. If all f_i are L_i-Lipschitz-smooth and $L_{\max} := \max_{i=1,\dots,N} L_i$, the second moment of the update direction (and hence its variance) satisfies $\mathbb{E}\|\nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^\star)\|^2 \leq L_{\max}^2 \mathbb{E}\|x^k - x^\star\|^2 \to 0$ if $\mathbb{E}\|x^k - x^\star\|^2 \to 0$. This reasoning motivates why $\text{SGD}_\star$ can be considered a VR-method.

Moreover, if f is strongly convex, then the iterates of $\text{SGD}_\star$ converge linearly to the solution for a *constant* step size $\alpha \leq \frac{1}{L_{\max}}$ [53, Thm. 1]. This improves the results we presented for ProxSGD: the convergence rate is linear instead of sublinear and constant step sizes are allowed (or alternatively, it is an exact convergence result in contrast to (2.5)). In the next section, we show that such improved convergence results can indeed be obtained for *practical* VR-methods.

2.5.2 SVRG and SAGA

Supposedly the first VR method is SAG which came out in 2013 [136], followed by SVRG [71] and SDCA [140] in 2013 as well and SAGA (SAG "amélioré") in 2014 [27, 53]. In this section, we will focus on SVRG and SAGA and summarize their theoretical contributions. Both methods, by construction, only work for finite-sum loss functions. Hence, for this section let us assume that we aim to solve problem (P) with the loss given as in (2.3), i.e. $f(x) = \frac{1}{N}\sum_{i=1}^N f_i(x)$. We assume that each $f_i : \mathbb{R}^n \to \mathbb{R}$ is differentiable.

Algorithm 2 `SVRG`

Require: Starting point $\tilde{x}^0 \in \mathbb{R}^n$, step size $\alpha > 0$, batch size $b \in [N]$, $m \in \mathbb{N}$.

for $s = 0, 1, \ldots, \mathsf{S} - 1$ **do**
  Set $x^{s,0} = \tilde{x} = \tilde{x}^s$
  for $k = 0, \ldots, m - 1$ **do**
    Draw a b-tuple $S_k = \{i_1, \ldots, i_b\}$ of elements of $\{1, \ldots, N\}$ and compute $v^k = \left(\frac{1}{b}\sum_{l=1}^{b} \nabla f_{i_l}(x^{s,k}) - \nabla f_{i_l}(\tilde{x})\right) + \nabla f(\tilde{x})$.
    Update
$$x^{s,k+1} = \mathrm{prox}_{\alpha\varphi}(x^{s,k} - \alpha v^k).$$
  end for
  Option I: Set $\tilde{x}^{s+1} = \frac{1}{m}\sum_{j=1}^{m} x^{s,j}$.
  Option II: Set $\tilde{x}^{s+1} = x^{s,m}$.
end for
return $\tilde{x}^{\mathsf{S}}$

Algorithm 3 `SAGA`

Require: Starting point $x^0 \in \mathbb{R}^n$, step size $\alpha > 0$, batch size $b \in [N]$.

Set $\phi_i^0 = x^0$ for $i \in [N]$ and $g^0 = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\phi_i^0)$.
for $k = 0, 1 \ldots, K - 1$ **do**
  Draw b-tuple $S_k = \{i_1, \ldots, i_b\}$ of elements of $\{1, \ldots, N\}$ and compute $v^k = \left(\frac{1}{b}\sum_{l=1}^{b} \nabla f_{i_l}(x^k) - \nabla f_{i_l}(\phi_{i_l}^k)\right) + g^k$.
  Set $x^{k+1} = \mathrm{prox}_{\alpha\varphi}(x^k - \alpha v^k)$.
  Update $\phi_i^{k+1} = x^k$ for $i \in S_k$ and $\phi_i^{k+1} = \phi_i^k$ else.
  Set $g^{k+1} = g^k - \frac{1}{N}\sum_{l=1}^{b}\left(\nabla f_{i_l}(\phi_{i_l}^k) - \nabla f_{i_l}(\phi_{i_l}^{k+1})\right)$.
end for
return x^K

The core idea of both `SVRG` and `SAGA` is the following: consider the setting (`MB-ER`) and fix $x \in \mathbb{R}^n$. Instead of approximating $\nabla f(x)$ with the mini-batch gradient $\nabla f(x; s)$, for any random variable $Z(s)$ we can use instead

$$v(s) := \nabla f(x; s) - Z(s) + \mathbb{E}[Z(S)].$$

Clearly, in the setting (`MB-ER`) we have $\mathbb{E}[v(S)] = \mathbb{E}[\nabla f(x; S)] = \nabla f(x)$. In iteration k, setting $x = x^k$ and $S = S_k$, we can apply the same reasoning for the conditional expectation (conditioned on the history $\mathcal{F}_k$) as long as $Z(S_k)$ is $\mathcal{F}_k$-measurable. In this case, we have the (random) gradient estimator $v(S_k) = \nabla f(x^k; S_k) - Z(S_k) + \mathbb{E}[Z(S_k)|\mathcal{F}_k]$. Both `SAGA` and `SVRG` use this trick and only differ in their choice of Z. We present the full algorithms in Algorithms 2 and 3.

Let us make some bibliographic remarks on Algorithm 2 and Algorithm 3:

- `SVRG` has been initially proposed in [71] for $b = 1$ and $\varphi = 0$ and was extended to the convex, composite case later in [157]. Some articles refer to Algorithm 2 and Algorithm 3 as `ProxSVRG` and `ProxSAGA`. However, we always mean the proximal versions when we write `SVRG`/`SAGA`.
- For both `SVRG` and `SAGA`, a mini-batch version for the non-convex case is analyzed in [68]. Both algorithms are typically used with constant step size and batch size, but clearly are a priori not restricted to this.

- In the original paper [27], `SAGA` was proposed for single-batch $b = 1$. In practice, the $\phi_i^k \in \mathbb{R}^n$ in `SAGA` are never actually stored. It is sufficient to keep track of a gradient table of size $N \times n$ containing $\nabla f_i(\phi_i)$ for $i = 1, \dots, N$. In many practical situations this can even further be reduced to only store a vector of length N. We refer to [27] for further considerations on implementation.

Next, we present two convergence results for `SVRG` that have been established in the literature: linear convergence for the smooth, strongly convex setting (Theorem 2.4) and a sublinear rate of $\mathcal{O}(1/(m\mathsf{S}))$ in the non-convex, smooth setting (Theorem 2.5). Notably, both results hold for *constant step size*. Similar results can be found for `SAGA` in [27] and [68, Thm. 4].

For the following results, we assume that, in each iteration of Algorithm 2, each element of the tuple S_k is drawn from $[N]$ with uniform distribution and with replacement.

Theorem 2.4 (Thm. 3.1 in [157])**.** *Let φ be proper, closed, and convex and let each f_i be convex and L_i-smooth.*[10] *Furher, assume that $\psi = f + \varphi$ is μ-strongly convex and let $x^\star = \arg\min \psi$. Assume that $\tilde{x}^\mathsf{S}$ is generated by Algorithm 2 with Option I, step size α, batch size $b = 1$ and inner loop length m. Let $L_{\max} := \max_{i=1,\dots,N} L_i$ and assume that $0 < \alpha < \frac{1}{4L_{\max}}$ and m is sufficiently large such that*

$$\rho := \frac{1}{\mu\alpha(1 - 4L_{\max}\alpha)m} + \frac{4L_{\max}\alpha(m+1)}{(1 - 4L_{\max}\alpha)m} < 1,$$

then it holds

$$\mathbb{E}[\psi(\tilde{x}^\mathsf{S})] - \psi(x^\star) \leq \rho^\mathsf{S}\big(\psi(\tilde{x}^0) - \psi(x^\star)\big).$$

Theorem 2.5 (Thm. 2 in [68])**.** *Let φ be closed, and convex with closed domain and let each f_i be L_i-smooth. Let $L_{\max} := \max_{i=1,\dots,N} L_i$ and $x^\star \in \arg\min \psi$. Assume that we run Algorithm 2 with Option II, step size $\alpha = \frac{1}{3L_{\max}}$, batch size $b = N^{2/3}$ (assumed to be integer) and inner loop length $m = \lfloor N^{1/3} \rfloor$. Let $x_\sim^\mathsf{S}$ be chosen uniformly from the iterates $\{\{x^{s,k}\}_{k=0}^{m-1}\}_{s=0}^{\mathsf{S}-1}$ of Algorithm 2. Then, it holds*

$$\mathbb{E}\|\mathcal{G}_\alpha(x_\sim^\mathsf{S})\|^2 \leq \frac{18L_{\max}(\psi(\tilde{x}^0) - \psi(x^\star))}{m\mathsf{S}}.$$

2.5.3 Interpolation

To conclude this section, we want to connect variance reduction to an analysis of `SGD` with interpolation. Consider problem (`P`) with $\varphi = 0$. We say that *perfect interpolation* holds at $x^\star \in \arg\min_{x \in \mathbb{R}^n} f(x)$ if $x^\star$ is also a minimizer of $f(\cdot; s)$ for all $s \in \mathcal{S}$. It is useful to quantify interpolation by the following constants:

$$\sigma_\star^2 := \mathbb{E}\|\nabla f(x^\star; S)\|^2, \qquad \sigma_I^2 := f(x^\star) - \mathbb{E}[\inf_z f(z; S)].$$

Clearly, $\sigma_I^2 \geq 0$ and perfect interpolation holds if and only if $\sigma_I^2 = 0$. Further, if all $f(\cdot; s)$ are L_s-smooth and lower bounded, applying Lemma 1.6 to $f(\cdot; s)$ and applying expectation yields $\sigma_\star^2 \leq 2(\sup_{s \in \mathcal{S}} L_s)\sigma_I^2$ (cf. [46, Lem. 4.18]). It has been shown empirically that highly overparameterized machine learning models still generalize well even if the model (approximately) interpolates the training set, i.e. the loss is minimized at every single training sample [52, 91]. This led to a

[10] The paper [157] does not list the assumption that f_i is convex, but it is actually used in the beginning of the proof of [157, Lem. 3.4].

resurgence of interest in theoretical analysis of SGD in the interpolation regime. We state a classical result by Needell et al. [102, Thm. 2.1]: if all $f(\cdot; s)$ are L_s-smooth, f is μ-strongly convex, then SGD with constant step size $\alpha < \frac{1}{\sup_{s\in\mathcal{S}} L_s}$ satisfies

$$\mathbb{E}\|x^K - x^\star\|^2 \leq \big(1 - 2\alpha\mu(1 - \alpha \sup_{s\in\mathcal{S}} L_s)\big)^K \|x^0 - x^\star\|^2 + \frac{\alpha\sigma_\star^2}{\mu(1 - \alpha \sup_{s\in\mathcal{S}} L_s)}.$$

In particular, we obtain a linear rate for constant step size if $\sigma_\star^2 = 0$ which improves the convergence result (2.5) of SGD presented above. We point out that the work by Needell et al. above was preceded by a similar result by Bach and Moulines [101, Thm. 1] which had a suboptimal dependence on the condition number. A subsequent work relaxes the condition on the step size from $\sup_{s\in\mathcal{S}} L_s$ to an *expected smoothness* constant in the (ER) setting [52, Ass. 2.1, Thm. 3.1].

For perfect interpolation we have $\sigma_\star^2 = 0$ which is equivalent to $\nabla f_i(x^\star) = 0$ for all $i \in [N]$ in the (ER) setting. Thus, perfect interpolation implies that running SGD is the same as running $\text{SGD}_\star$. Combining this perspective with the results in Section 2.5.1, it is not surprising that SGD achieves linear convergence in the setting of perfect interpolation, as in this case SGD is the same as $\text{SGD}_\star$ and hence is effectively a variance-reduced method.

2.6 Adaptive Methods

In this section, we discuss a different modification of ProxSGD, namely the family of *adaptive methods*, and focus on its two most prominent members AdaGrad [36, 94] and Adam [74]. The term *adaptivity* is certainly ambiguous, but in this section it describes methods that adapt the step size to each coordinate based on the training trajectory. We will later also use the term *preconditioning* for this type of adaptivity. AdaGrad has been very influential for machine learning since the early 2010's; Adam is presumably the most used optimization algorithm in modern deep learning. Many closely related adaptive methods exist but will not be discussed here in detail. For convergence results of Adam and AdaGrad, we refer the interested reader to [29] and references therein. It should also be pointed out that Adam does not necessarily converge for simple convex problems [120].

We present AdaGrad for regularized problems of form (P) in Algorithm 4 and remark that to the best of our knowledge there is no clear proximal version of AdaGrad – the algorithm presented here is taken from [99, sec. 4]. However, depending on φ it might be hard to compute $\text{prox}_\varphi^{\Lambda_k}$, and other versions are possible. Further, ε is typically chosen small in order to avoid division by zero, e.g. $\varepsilon = 10^{-10}$ is the Pytorch default. The operations $\odot$ and $\sqrt{\cdot}$ in line 4 are done elementwise.

The main difference between ProxSGD and AdaGrad is the norm of the proximal step. This becomes clear when rewriting the AdaGrad update as

$$\begin{aligned} x^{k+1} &= \text{prox}_\varphi^{\Lambda_k}(x^k - \Lambda_k^{-1} g_k) \\ &= \arg\min_{y\in\mathbb{R}^n} \varphi(y) + \tfrac{1}{2}\|y - (x^k - \Lambda_k^{-1} g_k)\|_{\Lambda_k}^2 \\ &= \arg\min_{y\in\mathbb{R}^n} \varphi(y) + f(x^k; S_k) + \langle g_k, y - x^k\rangle + \tfrac{1}{2}\|y - x^k\|_{\Lambda_k}^2. \end{aligned}$$

Comparing the above to (2.4), we observe that in fact the only difference is the norm $\|\cdot\|_{\Lambda_k}^2$ instead of $\frac{1}{\alpha_k}\|\cdot\|^2$. With Λ_k being diagonal, it can be interpreted as the inverse of a coordinatewise

Algorithm 4 `AdaGrad`

Require: Starting point $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$, parameter $\varepsilon > 0$.

1: Initialize $v_{-1} = 0$.
2: **for** $k = 0, \dots, K-1$ **do**
3: Sample S_k and compute $g_k \in \partial f(x^k; S_k)$.
4: Set $v_k = v_{k-1} + g_k \odot g_k$ and $\Lambda_k = \alpha_k^{-1} \operatorname{Diag}(\varepsilon \mathbf{1}_n + \sqrt{v_k})$.
5: Update

$$x^{k+1} = \operatorname{prox}_{\varphi}^{\Lambda_k}(x^k - \Lambda_k^{-1} g_k).$$

6: **end for**
7: **return** x^K

Algorithm 5 `Adam`

Require: Starting point $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$, parameter $\varepsilon > 0$, $\beta_1, \beta_2 \in [0, 1)$.

1: Initialize $m_{-1} = v_{-1} = 0$.
2: **for** $k = 0, \dots, K-1$ **do**
3: Sample S_k and compute $g_k \in \partial f(x^k; S_k)$.
4: Set $m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k$ and $v_k = \beta_2 v_{k-1} + (1-\beta_2)(g_k \odot g_k)$.
5: Set $\hat{m}_k = \frac{m_k}{1-\beta_1^{k+1}}$ and $\hat{v}_k = \frac{v_k}{1-\beta_2^{k+1}}$.
6: Set $\Lambda_k = \alpha_k^{-1} \operatorname{Diag}(\varepsilon \mathbf{1}_n + \sqrt{\hat{v}_k})$.
7: Update

$$x^{k+1} = x^k - \Lambda_k^{-1} m_k.$$

8: **end for**
9: **return** x^K

step size, or similarly as a preconditioning matrix. One key observation is that `AdaGrad` uses larger step sizes for coordinates where v_k is small, i.e. where the past gradients have been small in absolute value.

For `Adam`, the main difference is that it introduces (heavy-ball) momentum for the update direction as well as for v_k. We remark that `Adam` in most literature (in particular in the original article [74]) and also in this thesis is stated for unregularized problems, i.e. $\varphi = 0$. Given that its main field of application is deep learning, squared ℓ_2-regularization (weight decay) is often sufficient, and proximal versions for this specific regularization have been proposed (see [89, 169] and Chapter 4 for more details).

One property of `AdaGrad` and `Adam` which we like to emphasize is that – in contrast to `ProxSGD` – they are approximately *scale-free*: this means that, if $\varepsilon = 0$, the step size α_k does not need to be rescaled if the objective function f would be scaled by a positive factor. The reason for this is that scaling f affects g_k and $\sqrt{v_k}$ by the same factor and hence cancels out in $\Lambda_k^{-1} g_k$ if $\varepsilon = 0$. For a detailled discussion we refer the reader to [169]. In practice, this is an advantage as the scale of the objective function is typically unknown. For each dataset/model/... configuration, the step size of `ProxSGD` needs to be re-tuned and the optimal value can lie in different intervals. For `AdaGrad` or `Adam` this is not the case and their default step size (e.g. $\alpha_k = 0.001$ for `Adam` in `Pytorch`) often yields good results.

Algorithm 6 Model-based Stochastic Proximal Point

Require: Starting point $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$.

1: **for** $k = 0, \dots, K-1$ **do**

2: Sample S_k and update

$$x^{k+1} = \arg\min_{y \in \mathbb{R}^n} f_{x^k}(y; S_k) + \varphi(y) + \frac{1}{2\alpha_k}\|y - x^k\|^2.$$

3: **end for**

4: **return** x^K

2.7 Model-based Stochastic Optimization

In this section we will present a more general approach to stochastic optimization, namely model-based stochastic proximal point. In particular, we will see that this framework comprises an entire family of methods including `ProxSGD`.

The theoretical foundations for this framework have been established by Duchi and Ruan [37], Asi and Duchi [5, 6], and Davis and Drusvyatskiy [26]. The main difference between the latter two concurrent works is that [6] considers convex models and loss functions and set-constraint regularization functions $\varphi = \boldsymbol{\delta}_{\mathcal{X}}$ with $\mathcal{X} \subset \mathbb{R}^n$ closed, convex; the setting of [26] uses weakly convex models and allows for more general regularization functions.

Fix $x \in \mathbb{R}^n$ and $s \in \mathcal{S}$. The core idea of model-based stochastic optimization is to construct an approximation of the loss $f(\cdot; s)$ around x, the *model*, which is denoted by $f_x(\cdot; s)$. For a step size $\alpha > 0$, we compute the update as

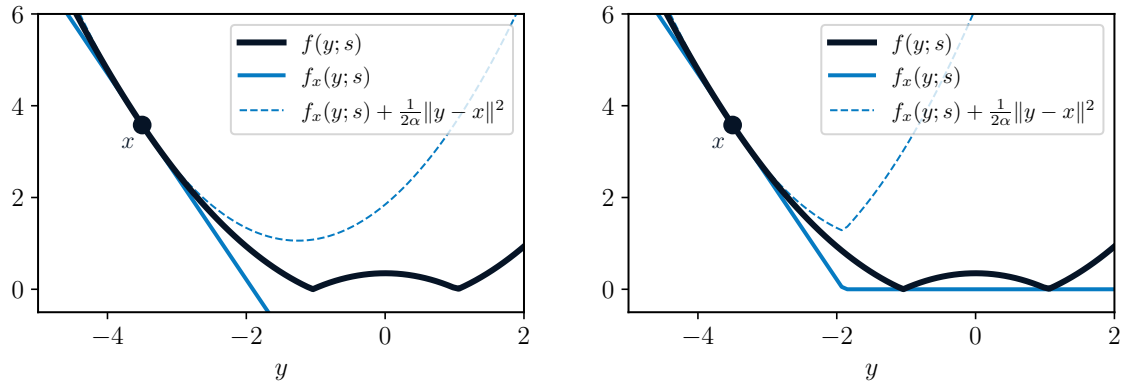
$$x^+ = \arg\min_{y \in \mathbb{R}^n} f_x(y; s) + \varphi(y) + \frac{1}{2\alpha}\|y - x\|^2. \tag{2.6}$$

Doing the above procedure iteratively, with step sizes $\alpha_k > 0$ and samples $S_k \in \mathcal{S}$ results in Algorithm 6. Clearly, this algorithm is so far only conceptual as we haven't specified the model $f_x(\cdot; s)$ yet. We will give three examples for a model; all of them have been discussed in [6].

Full model. If we choose $f_x(y; s) = f(y; s)$, we obtain stochastic proximal point. If $f(\cdot; s) \in \mathcal{C}^1$ and using (1.7) for $f(\cdot; s) + \varphi$, update (2.6) can be written as the implicit equation $x^+ = \operatorname{prox}_{\alpha\varphi}(x - \alpha\nabla f(x^+; s))$. We will refer to this method as `SPP` and will discuss it in detail in Chapter 3.

Linear model. Consider the choice $f_x(y; s) = f(x; s) + \langle g, y - x\rangle$ with $g \in \partial f(x; s)$. As shown in (2.4), update (2.6) results in $x^+ = \operatorname{prox}_{\alpha\varphi}(x - \alpha g)$. Hence, Algorithm 6 is equal to `ProxSGD`.

Truncated model. In many applications, the quantity $\inf_z f(z; s)$ is known or can be estimated; for example, many commonly used loss functions are non-negative. Hence, we can truncate the linearization at a lower bound $C(s) \leq \inf_z f(z; s)$ and set $f_x(y; s) = \max\{f(x; s) + \langle g, y - x\rangle, C(s)\}$ with $g \in \partial f(x; s)$. If $g \neq 0$ and $\varphi = 0$, update (2.6) can then be computed as



(a) Linear model

(b) Truncated model

Figure 2.2: Illustration of stochastic loss function (dark blue), model (solid light blue), and objective function (dashed light blue) of (2.6), setting $\varphi = 0, \alpha = 1$. All curves are plotted as function of y with the dot marking $y = x$. Left: Linear model $f_x(y;s) = f(x;s) + \langle g, y - x\rangle$ with $g \in \partial f(x;s)$. Right: Truncated model $f_x(y;s) = \max\{f(x;s) + \langle g, y - x\rangle, C(s)$ with $C(s) = 0$.

$x^+ = x - \min\{\alpha, \frac{f(x;s)-C(s)}{\|g\|^2}\}g$. This has been shown in [6] for $C(s) = \inf_z f(z;s)$ and we will give a formal proof later (cf. Section 4.3).

The difference between linear and truncated model is illustrated in Fig. 2.2. Clearly, the truncated model yields a tighter approximation of the original function $f(y;s)$; the truncated model is practical especially when a tight estimate for the lower bound $C(s)$ is available. This insight will be fundamental for the contents of Chapter 4 and Chapter 5 and methods developed therein.

Next, we present a selection of the theoretical results from [6, 26]. For this, let us make the technical assumption that $f_x(y;s)$ is measurable with respect to $P(s)$ for any $x, y \in \mathbb{R}^n$.

2.7.1 Almost Sure Convergence for SPP

In this subsection, let $\mathcal{X} \subset \mathbb{R}^n$ be closed, convex and let $\varphi = \delta_{\mathcal{X}}$. Further, suppose that $f(\cdot;s)$ is convex on $\mathcal{X}$ for all $s \in \mathcal{S}$. We state the properties of a stochastic model presented in [6]:

Assumption 3 (cf. [6], (C.i)-(C.iv))**.**

(C1) *We have* $f_x(x;s) = f(x;s)$ *and* $f_x(y;s) \leq f(y;s)$ *for all* $x, y \in \mathcal{X}$.

(C2) *The mapping* $y \mapsto f_x(y,s)$ *is convex for all* $x \in \mathcal{X}$, $s \in \mathcal{S}$.

(C3) *It holds* $f_x(y;s) \geq \inf_{z\in\mathcal{X}} f(z;s)$ *for all* $x, y \in \mathcal{X}$, $s \in \mathcal{S}$.

(C4) *There exists* $\varepsilon > 0$ *and* $H_s \in \mathbb{R}_+$ *with* $\mathbb{E}[H_S] < \infty$ *such that for all* $x \in \mathcal{X}$, *the point* x^+ *given by* (2.6) *and the model* $f_x(\cdot;s)$ *satisfy*

$$f(x^+;s) \leq f_x(x^+;s) + \frac{1-\varepsilon}{2\alpha}\|x^+ - x\|^2 + H_s\alpha.$$

It is easy to see that the full model $f_x(y;s) = f(y;s)$ satisfies all of the properties above, in particular (C4) holds with $\varepsilon = 1$ and $H_s = 0$.

The next theorem combines two statements on boundedness [6, Thm. 1] and almost sure convergence [6, Prop. 1] of stochastic model-based methods, in particular SPP.

Theorem 2.6 (cf. Thm. 1 and Prop. 1 in [6]). *Suppose that (C1), (C2) and (C4) of Assumption 3 hold. Let $(x^k)_{k\in\mathbb{N}}$ be the sequence generated by Algorithm 6 with step sizes that satisfy $\sum_{k=0}^{\infty}\alpha_k = \infty$, $\sum_{k=0}^{\infty}\alpha_k^2 < \infty$. Under mild assumptions on the gradient noise (cf. [6, Ass. A1 and A2]) the sequence (x^k) is bounded with probability one and there exists $x^\star \in \arg\min_{x\in\mathcal{X}} f(x)$ such that $\|x^k - x^\star\| \to 0$ almost surely.*

2.7.2 The Weakly Convex Case

We now summarize the main results of [26] which are established in the weakly convex setting. The following assumption describes *stochastic one-sided models* as introduced in [26].

Assumption 4 (cf. [26], Assum. B). *Let $\tau, \eta \geq 0$.*

(B1) There exists an open convex set $U \subset \mathbb{R}^n$ containing $\operatorname{dom}(\varphi)$ and we have $\mathbb{E}[f_x(x;S)] = f(x)$ and

$$\mathbb{E}[f_x(y;S) - f(y)] \leq \frac{\tau}{2}\|y-x\|^2 \quad \text{for all } x, y \in U.$$

(B2) The mapping $y \mapsto f_x(y;s) + \varphi(y)$ is η-weakly convex for all $x \in U$ and all $s \in \mathcal{S}$.

(B3) There exists $G_s \geq 0$ for all $s \in \mathcal{S}$ and $\mathsf{G} \in \mathbb{R}$ such that $\mathbb{E}[G_S^2] \leq \mathsf{G}^2$ and

$$f_x(x;s) - f_x(y;s) \leq G_s\|x-y\| \quad \forall x, y \in U, \quad s \in \mathcal{S}.$$

Remark 1. *We did not list Assumption 1 which is part of [26, Assum. B] but we have already assumed this throughout the chapter.*

The properties of Assumption 4 imply that $\psi = f + \varphi$ is $(\tau+\eta)$-weakly convex and that f is G-Lipschitz continuous [26, Lem. 4.1]. Clearly, (B1) and (B2) are similar but more general than (C1) and (C2).

The main non-convex convergence result for Algorithm 6 is the following:

Theorem 2.7 (Thm. 4.3 in [26]). *Let Assumption 4 hold and assume that $\min\psi > -\infty$. Set $\bar\rho = 2(\tau+\eta)$ and let $\{x^k\}$ be generated by Algorithm 6 with $\alpha_k = \left(\bar\rho + \sqrt{\frac{2\bar\rho\mathsf{G}^2K}{\Delta}}\right)^{-1}$ for $\Delta \geq \operatorname{env}_\psi^{1/\bar\rho}(x^0) - \min\psi$. Then, for $x_\sim^K$ drawn uniformly at random from $\{x^0, \ldots, x^{K-1}\}$ it holds*

$$\mathbb{E}\|\nabla \operatorname{env}_\psi^{1/\bar\rho}(x_\sim^K)\|^2 \leq \frac{4\bar\rho\Delta}{K} + 8\mathsf{G}\sqrt{\frac{2\bar\rho\Delta}{K}}.$$

Chapter 3

A Semismooth Newton Stochastic Proximal Point Algorithm With Variance Reduction

The chapter is mainly based on the article

[98] A. Milzarek, F. Schaipp, and M. Ulbrich, *A Semismooth Newton Stochastic Proximal Point Algorithm with Variance Reduction*, SIAM Journal on Optimization, 34 (2024), pp. 1157–1185, https://epubs.siam.org/doi/10.1137/22M1488181.

A preprint version of the article is available at https://arxiv.org/abs/2204.00406. The initial idea for this project was proposed by Andre Milzarek and Michael Ulbrich. The proof technique for the weakly convex case was initially developped by Andre Milzarek. For the strongly convex case, a first proof, based on the technique in [157], was later substituted by a proof technique more coherent with proximal point at the proposal of Andre Milzarek.

The contents of Sections 3.9 and 3.10 are not part of the article [98]. They have been developed independently for this thesis outlining possible extensions and providing supplementary material.

3.1 Introduction

The proximal point algorithm has been established by Martinet [92,93] and mainly Rockafellar in his seminal work [124,125]. In each iteration, the proximal point algorithm for minimizing a closed, convex function $\psi : \mathbb{R}^n \to \overline{\mathbb{R}}$, computes the next iterate as

$$x^{k+1} = \operatorname*{arg\,min}_{x\in\mathbb{R}^n} \psi(x) + \frac{1}{2\alpha_k}\|x - x^k\|^2, \tag{3.1}$$

where α_k is a sequence of positive step sizes.

Rockafellar established convergence to a minimizer of ψ even if the subproblem is solved inexactly, that is if it holds

$$\|x^{k+1} - z^{k+1}\| \leq \varepsilon_k,\ \|x^{k+1} - z^{k+1}\| \leq \delta_k\|x^{k+1} - x^k\|,\quad \sum_{k=0}^{\infty}\varepsilon_k < \infty,\ \sum_{k=0}^{\infty}\delta_k < \infty,$$

where $z^{k+1} := \arg\min_{x\in\mathbb{R}^n} \psi(x) + \frac{1}{2\alpha_k}\|x - x^k\|^2$ is the exact minimizer. For details, we refer to [124, Thm. 1] and [125, Thm. 1, Thm. 4]. In fact, [125] covers a very general setting where ψ is defined on a possibly infinite-dimensional Hilbert space. Proximal point is connected to several other fundamental concepts within optimization, such as augmented Lagrangian [124] or Douglas-Rachford splitting [39]. Its convergence theory is widely established, for example due to the work of Güler [54].

At first glance, update (3.1) looks equally difficult to compute than solving the original problem $\min_{x\in\mathbb{R}^n} \psi(x)$. As a consequence, the proximal point algorithm is often considered a conceptual method only [33]. However, in past decades, many efficient practical versions of the proximal point method have been proposed, for example for semidefinite programming [167], Lasso problems [84], or Graphical Lasso [165]. We remark that the connection between the augmented Lagrangian method and proximal point applied to the dual is fundamentally important to many of these works. In particular for Lasso problems, where the number of data points is much smaller compared to the number of features, the `SSNAL` method presented in [84] excels because each subproblem is solved in a dual space where the dimension is the *number of data points*.

Even though the groundwork of proximal point was established at least since the 1970's, stochastic versions have been explored only recently [6,12,17,26,114,152]. These papers investigate the theoretical convergence of stochastic proximal point (`SPP`) in different convex and non-convex settings. However, they do not answer the question how to obtain a practical version of `SPP`. In the experiments of the above references, the update of `SPP` is computed only for restricted settings, for example with a batch size of one and no regularization. Proposing a general framework for efficiently computing the `SPP` update will be one of the core contributions of this chapter.

As we will see, the derivation of our method is not particularly related to `SSNAL`, however one consistent feature is that we will formulate the subproblem, solved in each iteration, in a dual space. As a consequence of our stochastic setup, the dimensionality of this dual space will be equal to the *batch size*. By construction, this makes the proposed method well suited for problems where the number of feature is very high, because only a minor part of the computations will be performed in the feature space.

Problem setup. In this work, we consider optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \ \psi(x) := f(x) + \varphi(x), \quad f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(A_i x), \tag{3.2}$$

where $A_i \in \mathbb{R}^{m_i\times n}$ is given and the functions $f_i : \mathbb{R}^{m_i} \to \mathbb{R}$, $m_i \in \mathbb{N}$, $i \in \{1,\dots,N\}$, are supposed to be continuously differentiable. The mapping $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is assumed to be convex, proper, and lower semicontinuous.

This is the same finite-sum structure as in (`ER`), just that here we explicitly introduce the linear mappings A_i. Even though the method we will present is not restricted to this application, the context we have in mind are classical statistical learning problems for generalized linear models with regularization, for example sparse (logistic) regression. In this case, A_i is a matrix of one row ($m_i = 1$) and f_i is a scalar function. For more details, see the experiments in Section 3.7.2 below; we also refer to Section 3.10 where the approach is extended beyond linear models.

3.2 Background and Contributions

Our main contributions and the core challenges addressed in this article are as follows:

- We introduce `SNSPP`, a semismooth Newton stochastic proximal point method with variance reduction for the composite problem (3.2). We prove linear convergence in the strongly convex case and a sublinear rate is established in the weakly convex case. Our results hold for constant step sizes and match the rates of `SVRG` [68, 157].
- Semismooth Newton-based proximal point algorithms have been shown to be highly efficient in deterministic problems [84, 158, 164, 167]. Our proposed algorithmic strategy benefits from the fast local convergence properties of the semismooth Newton method; it further allows to reduce the computational complexity of the occurring subproblems which can be directly controlled through the batch size. Further, our technique for approximately solving the `SPP` subproblem can be immediately used for standard `SPP` without variance reduction (or for other `VR` schemes).
- We present a unified analysis that takes into account the inexactness of each stochastic proximal step. This closely ties together theory and practice, thereby allowing for broader applicability of the `SPP` method and `SNSPP` in particular.
- Numerical experiments suggest that `SNSPP` performs on-par or better in comparison to state-of-the-art algorithms `SAGA`, `SVRG` and `AdaGrad`, and is more robust to step-size selection.

3.3 The Stochastic Proximal Point Method

3.3.1 Preliminaries and Assumptions

As we will later allow f_i to be weakly-convex, we will need to "convexify" the proximal point update appropriately. From the derivation of (1.5), we have that for $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ weakly convex and $M \in \mathbb{S}_+$, as long as the mapping $g(\cdot) + \frac{1}{2}\|\cdot - x\|_M^2$ is convex for any $x \in \mathbb{R}^n$, the proximal operator $\operatorname{prox}_g^{\mathbf{Id}+M}$ is well-defined (cf. also [126, Def. 8.45, Prop. 8.46]) and can be characterized by

$$p = \operatorname{prox}_g^{\mathbf{Id}+M}(x) \quad \iff \quad p \in x - M(p - x) - \partial g(p). \tag{3.3}$$

For problem (3.2), we introduce the proximal gradient mapping as a measure of stationarity, i.e., for $\alpha > 0$, we define

$$F_{\mathrm{nat}}^{\alpha} : \mathbb{R}^n \to \mathbb{R}^n, \quad F_{\mathrm{nat}}^{\alpha}(x) := x - \operatorname{prox}_{\alpha\varphi}(x - \alpha\nabla f(x))$$

and $F_{\mathrm{nat}}(x) := F_{\mathrm{nat}}^{1}(x)$ (cf. [48, 105] and note that $F_{\mathrm{nat}}^{\alpha}$ is similar to the gradient mapping $\mathcal{G}_\alpha$ in Proposition 2.1). If f is L-smooth, then the function F_{nat} is Lipschitz continuous with constant $2 + L$.

We now specify the basic assumptions under which we construct and study our stochastic proximal point method. Throughout this chapter, we assume that the functions $f_i : \mathbb{R}^{m_i} \to \mathbb{R}$, $i \in [N]$, are continuously differentiable and $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is a proper, closed, and convex mapping. Further conditions on f and φ are stated below.

Assumption 5. *Let f be defined as in* (3.2). *We assume:*

(A1) *The functions* $f_i : \mathbb{R}^{m_i} \to \mathbb{R}$ *are* L_i*-smooth and* γ_i*-weakly convex for all* i*.*

(A2) *The objective function* ψ *is bounded from below by* $\psi^\star := \inf_x \psi(x)$*.*

(A3) *The mapping* $x \mapsto \operatorname{prox}_{\alpha\varphi}(x)$ *is semismooth for all* $\alpha > 0$ *and all* $x \in \mathbb{R}^n$*.*

We note that L_i-smoothness already ensures weak convexity of f_i, $i \in [N]$ (but with a potentially different constant). Let us also set $\hat{f}_i(z) := f_i(z) + (\gamma_i/2)\|z\|^2$. We work with the following assumptions for the conjugates $\hat{f}_i^*$:

Assumption 6.

(A4) *The functions* $\hat{f}_i^*$ *are essentially differentiable with locally Lipschitz continuous gradients on the sets* $\mathcal{D}_i := \operatorname{int}(\operatorname{dom}(\hat{f}_i^*)) \neq \emptyset$*,* $i \in [N]$*.*

(A5) *The mappings* $\nabla \hat{f}_i^*$ *are semismooth on* $\mathcal{D}_i$ *for all* i*.*

By [8, Thm. 18.15], condition (A1) guarantees that the mappings $\hat{f}_i^*$ are $1/(L_i + \gamma_i)$-strongly convex on $\operatorname{dom}(\hat{f}_i^*)$. This, together with (A4), ensures that each $\hat{f}_i^*$ has uniformly positive definite second derivatives, i.e., there exists $\mu_* \geq \min_{i\in[N]} 1/(L_i + \gamma_i) > 0$ such that for all $i \in [N]$

$$\langle h, M_i(z)h\rangle \geq \mu_*\|h\|^2 \quad \forall\, h \in \mathbb{R}^{m_i}, \quad \forall\, M_i(z) \in \partial(\nabla \hat{f}_i^*)(z), \quad \forall\, z \in \mathcal{D}_i, \tag{3.4}$$

see [62, Ex. 2.2]. Moreover, (A4) implies that each $\hat{f}_i$ is essentially locally strongly convex [49, Cor. 4.3]. Next, we state two stronger versions of (A3) and (A5):

Assumption 7.

($\tilde{\text{A}}$3) *For every* $\alpha > 0$ *the proximal operator* $x \mapsto \operatorname{prox}_{\alpha\varphi}(x)$ *is* ν*-order semismooth on* $\mathbb{R}^n$ *with* $0 < \nu \leq 1$*.*

($\tilde{\text{A}}$5) *The mappings* $z \mapsto \nabla \hat{f}_i^*(z)$ *are* ν*-order semismooth on* $\mathcal{D}_i$ *with* $0 < \nu \leq 1$ *for all* i*.*

In the stochastic optimization literature, convexity and/or L-smoothness are standard assumptions for the component functions f_i (see [6, 27, 48, 68, 157]). In contrast to other recent works on stochastic proximal point methods, we neither assume convexity of ψ (as in, e.g., [6]) nor Lipschitz continuity of f (as in, e.g., [26]).
The additional condition (A4) and the semismoothness properties (A3) and (A5) (or ($\tilde{\text{A}}$3) and ($\tilde{\text{A}}$5), respectively) hold for many classical loss functions and regularizers. In fact, assumption (A3) is satisfied for (group) sparse regularizations based on ℓ_1- or ℓ_2-norms or low rank terms such as the nuclear norm. More generally, semismoothness of the proximal operator can be guaranteed when φ is semialgebraic or tame. We refer to [13, 97] for a detailed discussion of this observation. Strong semismoothness of $\operatorname{prox}_{\alpha\varphi}$ can be ensured whenever $\operatorname{prox}_{\alpha\varphi}$ is a piecewise $\mathcal{C}^2$-function (see [153, Prop. 2.26]). For instance, if $\varphi(x) = \lambda\|x\|_1$ is an ℓ_1-regularization with $\lambda > 0$, then the associated proximal operator is the well-known soft-thresholding operator which is piecewise affine. If every mapping $\nabla \hat{f}_i^*$ is piecewise $\mathcal{C}^1$, then assumption (A5) holds and ($\tilde{\text{A}}$5) is satisfied with $\nu = 1$ if all $\nabla \hat{f}_i^*$, $i \in [N]$, are piecewise $\mathcal{C}^2$.

3.3.2 Algorithmic Framework

We now motivate and develop our algorithmic approach in detail.

Stochastic Proximal Point Steps. Our core idea is to perform stochastic proximal point updates that mimic the classical proximal point iterations, [92,93,124], for minimizing the potentially nonconvex and nonsmooth objective function ψ in (3.2):

$$x^{k+1} = \operatorname{prox}_{\alpha_k \psi}(x^k),$$

where $\alpha_k > 0$ is a suitable step size. While f is possibly nonconvex, we can conclude from (A1) that $x \mapsto f_i(A_i x) + \frac{\gamma_i}{2}\|A_i(x - z)\|^2$ is a convex mapping for every $z \in \mathbb{R}^n$. Hence, setting $M_N := \frac{1}{N}\sum_{i=1}^N \gamma_i A_i^\top A_i$, the step

$$\begin{aligned} x^{k+1} &= \operatorname{prox}_{\alpha_k \psi}^{\mathbf{Id} + \alpha_k M_N}(x^k) \\ &= \arg\min_x \ \psi(x) + \frac{1}{2N}\sum\nolimits_{i=1}^N \gamma_i \|A_i(x - x^k)\|^2 + \frac{1}{2\alpha_k}\|x - x^k\|^2 \end{aligned}$$

is well-defined. Utilizing (3.3), we have $p = x^{k+1}$ if and only if $p \in [x^k - \alpha_k \nabla f(p) - \alpha_k M_N(p - x^k)] - \alpha_k \partial\varphi(p)$ and, hence, the proximal point update can be equivalently rewritten as the following implicit proximal gradient-type step

$$x^{k+1} = \operatorname{prox}_{\alpha_k \varphi}(x^k - \alpha_k \nabla f(x^{k+1}) - \alpha_k M_N(x^{k+1} - x^k)). \tag{3.5}$$

This implicit iteration forms the conceptual basis of our method. However, as our aim is to solve the finite-sum problem (3.2) in a stochastic fashion, we will use stochastic oracles to approximate the full gradient ∇f in each iteration [47, 48, 99]. In our case, the function f corresponds to an empirical expectation and thus, sampling a random subset of summands $f_i(A_i \cdot)$ can be understood as a possible stochastic oracle for f and ∇f. Specifically, for some given tuple $S \subseteq [N]$, we can consider the following stochastic variants of f, ∇f, and ψ:

$$f_S(x) := \frac{1}{|S|}\sum\nolimits_{i \in S} f_i(A_i x), \quad \nabla f_S(x) := \frac{1}{|S|}\sum\nolimits_{i \in S} A_i^\top \nabla f_i(A_i x),$$

and $\psi_S(x) := f_S(x) + \varphi(x)$. Let $S_k \subseteq [N]$ be the tuple drawn randomly at iteration k. The stochastic counterpart of the update (3.5) is then obtained by replacing the gradient ∇f with the estimator ∇f_{S_k} and M_N with $M_{S_k} := |S_k|^{-1}\sum_{i \in S_k} \gamma_i A_i^\top A_i$. This yields

$$x^{k+1} = \operatorname{prox}_{\alpha_k \varphi}(x^k - \alpha_k \nabla f_{S_k}(x^{k+1}) - \alpha_k M_{S_k}(x^{k+1} - x^k)). \tag{3.6}$$

This step can also be interpreted as a stochastic proximal point iteration

$$x^{k+1} = \operatorname{prox}_{\alpha_k \psi_{S_k}}^{\mathbf{Id} + \alpha_k M_{S_k}}(x^k)$$

for the sampled objective function ψ_{S_k}. Consequently, the update (3.6) can be seen as a combination of the stochastic model-based proximal point frameworks derived in [6,26] and of variable metric proximal point techniques [14, 112].

Incorporating Variance Reduction. Variance reduction has proven to be a powerful tool in order to accelerate stochastic optimization algorithms (cf. Section 2.5 and [27,56,68,157]). Similar to [68], we consider `SRVG`-type stochastic oracles that additionally incorporate the following gradient information in each iteration

$$v^k := \nabla f(\tilde{x}) - \nabla f_{S_k}(\tilde{x}), \tag{3.7}$$

where $\tilde{x} \in \mathbb{R}^n$ is a reference point that is generated in an outer loop. This leads to stochastic proximal point-type updates of the form

$$x^{k+1} = \operatorname{prox}_{\alpha_k \varphi}(x^k - \alpha_k[\nabla f_{S_k}(x^{k+1}) + v^k] - \alpha_k M_{S_k}(x^{k+1} - x^k)). \tag{3.8}$$

Our subsequent analysis and discussion focuses on this general formulation.

An Implementable Strategy for the Implicit Update. We now introduce an alternative equation-based characterization of the implicit update (3.8). Let us set $b_k := |S_k|$ and let $(\kappa_k(1), \dots, \kappa_k(b_k))$ enumerate the elements of the tuple S_k. We will often abbreviate κ_k by κ. We now define $\xi^{k+1} = (\xi_1^{k+1}, \dots, \xi_{b_k}^{k+1})$ by

$$\xi_i^{k+1} := \nabla \hat{f}_{\kappa(i)}(A_{\kappa(i)} x^{k+1}) = \nabla f_{\kappa(i)}(A_{\kappa(i)} x^{k+1}) + \gamma_{\kappa(i)} A_{\kappa(i)} x^{k+1} \quad i \in [b_k]. \tag{3.9}$$

Under assumption (A1), [9, Thm. 4.20] yields

$$\xi_i^{k+1} = \nabla \hat{f}_{\kappa(i)}(A_{\kappa(i)} x^{k+1}) \quad \iff \quad \nabla \hat{f}^*_{\kappa(i)}(\xi_i^{k+1}) = A_{\kappa(i)} x^{k+1}. \tag{3.10}$$

Thus, setting $\hat{v}^k := \alpha_k(v^k - M_{S_k} x^k)$, the step (3.8) is equivalent to the system

$$\left[\begin{array}{l} x^{k+1} = \operatorname{prox}_{\alpha_k \varphi}\left(x^k - \frac{\alpha_k}{b_k} \sum_{i=1}^{b_k} A_{\kappa(i)}^\top \xi_i^{k+1} - \hat{v}^k \right), \\ \nabla \hat{f}^*_{\kappa(i)}(\xi_i^{k+1}) = A_{\kappa(i)} \operatorname{prox}_{\alpha_k \varphi}\left(x^k - \frac{\alpha_k}{b_k} \sum_{i=1}^{b_k} A_{\kappa(i)}^\top \xi_i^{k+1} - \hat{v}^k \right) \quad \forall\, i \in [b_k]. \end{array} \right. \tag{3.11}$$

Similar to [84, 167], we use a semismooth Newton method to solve the system of nonsmooth equations defining ξ^{k+1} in an efficient way. Importantly, the dimension of this system and of ξ^{k+1} is controlled by the batch size b_k which is an advantage if $b_k \ll n$. We allow approximate solutions of the system (3.11) which results in inexact proximal steps. This potential inexactness is an important component of our algorithmic design and convergence analysis that has not been considered in other stochastic proximal point methods [6, 26]. The full method is shown in Algorithm 7. The semismooth Newton method for (3.11) is specified and discussed in the next section. In this article, we primarily focus on the variance-reduced update (3.8), yet the technique and results presented in Section 3.4 also hold true for the general update (3.6).

Sampling Assumptions. We now formally specify the notion of admissible stochastic oracles for our problem.

Definition 3.1. *Let $S \sim P$ be a b-tuple of elements of $[N]$, where $b \in [N]$ is fixed. Let $\kappa(i) \in [N]$ denote the random number in the i-th position of S. We call $S \sim P$ an admissible sampling procedure if, for all $z_i \in \mathbb{R}^\ell$, $i \in [N]$, $\ell \in \mathbb{N}$, it holds that $\mathbb{E}_P[z_S] = \frac{1}{N} \sum_{i=1}^N z_i$ where $z_S := \frac{1}{b} \sum_{i=1}^b z_{\kappa(i)}$.*

If $S \sim P$ is an admissible sampling procedure, then we have $\mathbb{E}_P[f_S(x)] = f(x)$ and $\mathbb{E}_P[\nabla f_S(x)] = \nabla f(x)$ for all $x \in \mathbb{R}^n$. In the simplest case, we can choose S by drawing b elements from $[N]$ under a uniform distribution (cf. [157] for a similar setting). This is an admissible sampling procedure in the sense of Definition 3.1, regardless of whether we draw with or without replacement (cf. [87, §2.8]).

Remark 2. *Note that x^k, α_k, etc., serve as abbreviations when the value of s is clear. The notation $x^{s,k}$, α_k^s, etc., can be used to highlight the full (s,k)-dependence.*

3.4 A Semismooth Newton Method for Solving the Subproblem

In the following, we assume that we are given a b-tuple $S = (\kappa(1), \dots, \kappa(b))$ of elements of $[N]$, a step size $\alpha > 0$, and vectors $x, v \in \mathbb{R}^n$. Let $m_S := \sum_{i=1}^b m_{\kappa(i)}$ denote the dimension of the subproblem and let $\mathcal{D} = \prod_{i \in S} \mathcal{D}_i \subseteq \mathbb{R}^{m_S}$. Now, the second line in (3.11) corresponds to a system of nonlinear equations which can be reformulated as

$$\mathcal{V}(\xi) = 0, \tag{3.12}$$

Algorithm 7 SNSPP

Require: $\tilde{x}^0 \in \mathbb{R}^n$, $m, \mathsf{S} \in \mathbb{N}$, and, for $s = 0, \dots, \mathsf{S}$, $k = 0, \dots, m-1$, step sizes $\alpha_k^s > 0$, batch sizes $b_k^s \in \mathbb{N}$, and tolerances $\varepsilon_k^s \geq 0$.

1: **for** $s = 0, 1, 2, \dots, \mathsf{S}$ **do**
2: Set $x^0 := x^{s,0} := \tilde{x} := \tilde{x}^s$, and, for $0 \leq k < m$, $\alpha_k := \alpha_k^s$, $b_k := b_k^s$, and $\varepsilon_k := \varepsilon_k^s$.
3: **for** $k = 0, 1, 2, \dots, m-1$ **do**
4: **(Sampling)** Sample $S_k = S_k^s$ with $|S_k| = b_k$ and set
$$v^k := v^{s,k} := \nabla f(\tilde{x}) - \nabla f_{S_k}(\tilde{x}), \quad \hat{v}^k := \hat{v}^{s,k} := \alpha_k(v^k - M_{S_k} x^k).$$
5: **(Solve subproblem)** Compute $\xi^{k+1} = \xi^{s,k+1}$ by invoking Algorithm 8 with input $x^k, \alpha_k, S_k, -\hat{v}^k$ and ε_k.
6: **(Update)** Set $x^{k+1} := x^{s,k+1} := \operatorname{prox}_{\alpha_k \varphi}\left(x^k - \frac{\alpha_k}{b_k} \sum_{i=1}^{b_k} A_{\kappa(i)}^\top \xi_i^{k+1} - \hat{v}^k\right)$.
7: **end for**
8: Option I: Set $\tilde{x}^{s+1} := x^m$.
9: Option II: Set $\tilde{x}^{s+1} := \frac{1}{m} \sum_{k=1}^m x^k$.
10: **end for**
11: **return** $\tilde{x}^{\mathsf{S}+1}$; x_π drawn uniformly from $\{x^{s,k}\}_{0 \leq k < m}^{0 \leq s \leq \mathsf{S}}$.

where $\mathcal{V} : \mathcal{D} \to \mathbb{R}^{m_S}$, $\mathcal{V}(\xi) := (\mathcal{V}_1(\xi)^\top, \dots, \mathcal{V}_b(\xi)^\top)^\top$, and $\xi := (\xi_1^\top, \dots, \xi_b^\top)^\top$. Setting $\mathcal{A}_S := \frac{1}{b}(A_{\kappa(1)}^\top, \dots, A_{\kappa(b)}^\top)^\top \in \mathbb{R}^{m_S \times n}$, each $\mathcal{V}_i$ is defined via

$$\mathcal{V}_i : \mathcal{D} \to \mathbb{R}^{m_{\kappa(i)}}, \quad \mathcal{V}_i(\xi) = \nabla \hat{f}_{\kappa(i)}^*(\xi_i) - A_{\kappa(i)} \operatorname{prox}_{\alpha\varphi}\left(x - \alpha \mathcal{A}_S^\top \xi + v\right). \tag{3.13}$$

The Newton step of this system is given by

$$\mathcal{W}(\xi) d = -\mathcal{V}(\xi), \tag{3.14}$$

where $\mathcal{W}(\xi) \in \hat{\partial}\mathcal{V}(\xi)$ is an element of the surrogate differential $\hat{\partial}\mathcal{V}$ defined via

$$\hat{\partial}\mathcal{V}(\xi) := \Big\{ \operatorname{Diag}\left(H_i(\xi_i)_{i=1,\dots,b}\right) + \alpha b \mathcal{A}_S U(\xi) \mathcal{A}_S^\top \;\Big|\; U(\xi) \in \partial \operatorname{prox}_{\alpha\varphi}\big(x - \alpha \mathcal{A}_S^\top \xi + v\big),\ H_i(\xi_i) \in \partial(\nabla \hat{f}_{\kappa(i)}^*)(\xi_i)\ \forall\ i \in [b] \Big\}.$$

We first present several basic properties of the operators and functions involved in the Newton step (3.14). The nonexpansiveness of the proximal operator (1.8) and [70, Prop. 2.3] imply the next result (see also [97, Lem. 3.3.5]).

Proposition 3.2. *Let $\alpha > 0$ and $x \in \mathbb{R}^n$ be given. Each element $U \in \partial \operatorname{prox}_{\alpha\varphi}(x)$ is a symmetric and positive semidefinite $n \times n$ matrix.*

Proposition 3.3. *Suppose that the conditions* (A3), (A4), *and* (A5) *are satisfied. Let S be a b-tuple of elements of $[N]$, and let $\alpha > 0$ and $x, v \in \mathbb{R}^n$ be given. Then, the function $\mathcal{V}$ is semismooth on $\mathcal{D}$ w.r.t. $\hat{\partial}\mathcal{V}$. If* (Ã3) *and* (Ã5) *hold instead of* (A3) *and* (A5), *$\mathcal{V}$ is ν-order semismooth w.r.t. $\hat{\partial}\mathcal{V}$.*

Proof. The first claim follows using the chain rule for semismooth functions [43, Thm. 7.5.17]. If we assume (Ã3) and (Ã5) instead, the claim follows from [153, Prop. 3.8] and [141, Prop. 3.6]. □

In the following, we show that the function $\mathcal{V}$ can be interpreted as a gradient mapping. Thus, finding a root of $\mathcal{V}$ is equivalent to finding a stationary point.

Proposition 3.4. *Let the assumptions* (A1) *and* (A4) *hold. Let* $\alpha > 0$ *and* $x, d \in \mathbb{R}^n$ *be given and let* S *be a* b*-tuple of elements of* $[N]$*. For* $\xi \in \mathbb{R}^{m_S}$*, we define*

$$\mathcal{U}(\xi) := \sum\nolimits_{i=1}^{b} \hat{f}^*_{\kappa(i)}(\xi_i) + \frac{b}{2\alpha}\|z(\xi)\|^2 - \frac{b}{\alpha}\mathrm{env}_{\alpha\varphi}(z(\xi)), \quad z(\xi) := x - \alpha\mathcal{A}_S^\top \xi + v.$$

Then, $\mathcal{U}$ *is* μ_**-strongly convex on the set* $\mathcal{E} := \prod_{i=1}^{b} \mathrm{dom}(\hat{f}^*_{\kappa(i)})$ *and we have* $\nabla\mathcal{U}(\xi) = \mathcal{V}(\xi)$ *for all* $\xi \in \mathcal{D}$ *where* $\mathcal{V}$ *is defined in* (3.13)*.*

Proof. For every $\xi \in \mathcal{D}$ and $i \in [b]$, we have $\frac{\partial z}{\partial \xi_i}(\xi) = -\frac{\alpha}{b} A_{\kappa(i)}^\top$ and

$$\begin{aligned}\nabla_{\xi_i}\mathcal{U}(\xi) &= \nabla \hat{f}^*_{\kappa(i)}(\xi_i) + \frac{b}{\alpha}\frac{\partial z}{\partial \xi_i}(\xi)^\top \Big(z(\xi) - \big(z(\xi) - \mathrm{prox}_{\alpha\varphi}(z(\xi))\big)\Big) \\ &= \nabla \hat{f}^*_{\kappa(i)}(\xi_i) - A_{\kappa(i)}\mathrm{prox}_{\alpha\varphi}(z(\xi)) = \mathcal{V}_i(\xi),\end{aligned}$$

where we used (1.10) and $\mathrm{env}_{\alpha\varphi} = \alpha\mathrm{env}^{\alpha}_{\varphi}$. For the first statement, note that (A1) implies strong convexity of $\hat{f}_i^*$ on $\mathrm{dom}(\hat{f}_i^*)$ for $i = 1, \dots, N$. Applying Moreau's identity, [9, Thm. 6.67],

$$\frac{1}{2}\|z\|^2 - \mathrm{env}_{\alpha\varphi}(z) = \alpha^2 \mathrm{env}_{\alpha^{-1}\varphi^*}(z/\alpha), \tag{3.15}$$

we can use the fact that the Moreau envelope of a proper, closed, and convex function is convex [9, Thm. 6.55]. Hence, the mapping $\xi \mapsto \frac{b}{2\alpha}\|z(\xi)\|^2 - \frac{b}{\alpha}\mathrm{env}_{\alpha\varphi}(z(\xi))$ is convex as $\xi \mapsto z(\xi)$ is affine. Altogether, $\mathcal{U}$ is μ_*-strongly convex on $\mathcal{E}$ (cf. (3.4)). □

In Algorithm 8, we formulate a globalized semismooth Newton method for solving the nonsmooth system (3.12). Specifically, the result in Proposition 3.4 enables us to measure descent properties of a semismooth Newton step using $\mathcal{U}$ and to apply Armijo line search-based globalization techniques. Based on the results on SC^1 minimization (cf. [41,153,167]), we obtain the following convergence result.

Theorem 3.5. *Let the assumptions* (A1)–(A5) *be satisfied and let the sequence* $\{\xi^j\}$ *be generated by Algorithm 8. Then,* $\{\xi^j\}$ *converges q-superlinearly to the unique solution* $\hat{\xi} \in \mathcal{D}$ *of* (3.12)*, i.e.,* $\|\xi^{j+1} - \hat{\xi}\| = o(\|\xi^j - \hat{\xi}\|)$ *as* $j \to \infty$*. Moreover, under* (Ã3) *and* (Ã5)*, we obtain*

$$\|\xi^{j+1} - \hat{\xi}\| = \mathcal{O}(\|\xi^j - \hat{\xi}\|^{1+\min\{\tau,\nu\}}) \quad \textit{for all } j \textit{ sufficiently large.}$$

Proof. By construction, we have $\{\xi^j\} \subset \mathcal{D}$ and the set $\mathcal{D}$ is open. Proposition 3.4 and (A4) imply that $\mathcal{U}$ is strongly convex (on $\mathcal{E}$) and essentially differentiable. Hence, $\mathcal{U}$ has a unique minimizer $\hat{\xi} \in \mathcal{D}$ which is also the unique solution to (3.12). For every $\xi \in \mathcal{D}$, the matrices $\mathcal{W}(\xi) \in \hat{\partial}\mathcal{V}(\xi)$ are positive definite by (3.4) and Proposition 3.2. Using standard arguments (see [167, Thm. 3.4] and [84, Thm. 3.6]), it can be shown that the sequence $\{\xi^j\}$ generated by Algorithm 8 converges to $\hat{\xi}$. Under (A1)–(A5), we conclude from equation (67) in the proof of [167, Thm. 3.5] that $\|\xi^j + d^j - \hat{\xi}\| \leq o(\|\xi^j - \hat{\xi}\|)$ holds for all j sufficiently large. If assumptions (Ã3) and (Ã5) are satisfied instead of (A3) and (A5), then we have $\|\xi^j + d^j - \hat{\xi}\| \leq \mathcal{O}(\|\xi^j - \hat{\xi}\|^{1+\min\{\tau,\nu\}})$. Finally, let us show that in a neighborhood of the limit point the unit step size is accepted by the Armijo line search. Setting $\tilde{\mathcal{W}}_j := \mathcal{W} + \eta_j I$ and using $\mathcal{V}(\xi^j) \to 0$, we can infer

$$\|d^j\| = \|\tilde{\mathcal{W}}_j^{-1}(r^j - \mathcal{V}(\xi^j))\| \leq \|\tilde{\mathcal{W}}_j^{-1}\|(\|r^j\| + \|\mathcal{V}(\xi^j)\|) \leq 2\lambda_{\min}(\tilde{\mathcal{W}}_j)^{-1}\|\mathcal{V}(\xi^j)\|,$$

for all j sufficiently large. Thus, we have

$$-\frac{\langle \nabla\mathcal{U}(\xi^j), d^j\rangle}{\|d^j\|^2} \geq \frac{\lambda_{\min}(\tilde{\mathcal{W}}_j)^2}{4}\frac{\langle -\nabla\mathcal{U}(\xi^j), d^j\rangle}{\|\nabla\mathcal{U}(\xi^j)\|^2} \geq \frac{\lambda_{\min}(\tilde{\mathcal{W}}_j)^2}{4\lambda_{\max}(\tilde{\mathcal{W}}_j)},$$

Algorithm 8 Semismooth Newton Method for Solving Eq. (3.12)

Require: $x, v \in \mathbb{R}^n$, $\alpha > 0$, a b-tuple S of elements of $[N]$, and a tolerance ε_{sub}.
Choose an initial point ξ^0 such that $\xi_i^0 \in \mathcal{D}_i$ for all $i = 1, \dots, b$. Choose parameters $\hat{\gamma} \in (0, \frac{1}{2})$, $\eta \in (0,1)$, $\rho \in (0,1)$, $\tau \in (0,1]$, and $\tau_1, \tau_2 \in (0,1)$. Set $j = 0$.

1: **while** $\|\nabla \mathcal{U}(\xi^j)\| > \varepsilon_{\text{sub}}$ **do**

2: **(Newton direction)** Choose $\mathcal{W} \in \hat{\partial}\mathcal{V}(\xi^j)$, set $\eta_j := \tau_1 \min\{\tau_2, \|\mathcal{V}(\xi^j)\|\}$, and approximately solve the linear system

$$(\mathcal{W} + \eta_j I) d^j = -\mathcal{V}(\xi^j)$$

via the conjugate gradient method such that $\|r^j\| \leq \min\{\eta, \|\mathcal{V}(\xi^j)\|^{1+\tau}\}$ with $r^j := (\mathcal{W} + \eta_j I)d^j + \mathcal{V}(\xi^j)$.

3: **(Armijo line search)** Find the smallest non-negative integer ℓ_j such that

$$\mathcal{U}(\xi^j + \rho^{\ell_j} d^j) \leq \mathcal{U}(\xi^j) + \hat{\gamma}\rho^{\ell_j} \langle \nabla \mathcal{U}(\xi^j), d^j \rangle$$

and $\xi_i^j + \rho^{\ell_j} d_i^j \in \mathcal{D}_i$ for all $i = 1, \dots, b$. Set $\beta_j := \rho^{\ell_j}$.

4: **(Update)** Compute the new iterate $\xi^{j+1} = \xi^j + \beta_j d^j$ and set $j \leftarrow j + 1$.

5: **end while**

6: **return** ξ^j

where the second inequality comes from [167, Prop. 3.3]. Due to strong convexity, there exists $\tilde{\rho} > 0$ such that $\frac{\lambda_{\min}(\tilde{\mathcal{W}}_j)^2}{4\lambda_{\max}(\tilde{\mathcal{W}}_j)} \geq \tilde{\rho} > 0$ for all j. Due to [41, Thm. 3.3], $\beta_j = 1$ then fulfills the Armijo condition for j sufficiently large which concludes the proof. □

3.5 Controlling the Inexactness of the Update

In this section, we will discuss the stopping criterion of Algorithm 8. Let $x \in \mathbb{R}^n$, $\alpha > 0$, and a tuple S of elements of $[N]$ be given. By Proposition 3.4, $\mathcal{U}$ is μ_*-strongly convex on $\mathcal{D} \subset \mathcal{E}$. Thus, the gradient $\nabla \mathcal{U}$ is a μ_*-strongly monotone operator on $\mathcal{D}$. Let $\hat{\xi} := \arg\min_\xi \mathcal{U}(\xi) \in \mathcal{D}$ again denote the unique minimizer of $\mathcal{U}$. Then, we have

$$\|\xi - \hat{\xi}\| \leq {\mu_*}^{-1} \|\nabla \mathcal{U}(\xi)\| \quad \forall\, \xi \in \mathcal{D}, \tag{3.16}$$

Hence, the stopping criterion of Algorithm 8 – $\|\nabla \mathcal{U}(\xi^j)\| \leq \varepsilon_{\text{sub}}$ – allows to control the error $\|\xi^j - \hat{\xi}\|$. As we solve each subproblem inexactly, the updates (x^{k+1}, ξ^{k+1}) in Algorithm 7 are not an exact solution to (3.8). It is desirable to control the error $\|x^{k+1} - \hat{x}^{k+1}\|$ in each iteration, where $\hat{x}^{k+1}$ is the exact solution to (3.8). This is addressed in the following result.

Proposition 3.6. *Let us define $\bar{A} := \max_{i \in [N]} \|A_i\|$ and let $x, v \in \mathbb{R}^n$, $\alpha > 0$, and a b-tuple S of elements of $[N]$ be given. Suppose that $\mathcal{U}$, defined in Proposition 3.4, is μ_*-strongly convex on $\mathcal{E}$ and let $\hat{\xi} = \arg\min_\xi \mathcal{U}(\xi) \in \mathcal{D}$ be the unique minimizer of $\mathcal{U}$. Suppose that Algorithm 8 – run with tolerance ε_{sub} – returns ξ. Then, setting*

$$\hat{x}^+ := \operatorname{prox}_{\alpha\varphi}(x - \alpha \mathcal{A}_S^\top \hat{\xi} + v) \quad \text{and} \quad x^+ := \operatorname{prox}_{\alpha\varphi}(x - \alpha \mathcal{A}_S^\top \xi + v),$$

it holds that $\|\xi - \hat{\xi}\| \leq \frac{\varepsilon_{\text{sub}}}{\mu_*}$ and $\|x^+ - \hat{x}^+\| \leq \alpha \|A_S^\top(\xi - \hat{\xi})\| \leq \frac{\alpha \bar{A}}{\mu_* \sqrt{b}} \varepsilon_{\text{sub}}$.

Proof. The bound on $\|\xi - \hat{\xi}\|$ follows directly from (3.16). Utilizing the nonexpansiveness of the proximity operator, we can estimate

$$\begin{aligned}\|x^+ - \hat{x}^+\| &= \|\mathrm{prox}_{\alpha\varphi}\big(x - \alpha\mathcal{A}_S^\top\xi + v\big) - \mathrm{prox}_{\alpha\varphi}\big(x - \alpha\mathcal{A}_S^\top\hat{\xi} + v\big)\| \\ &\leq \alpha\|A_S^\top(\xi - \hat{\xi})\| \leq \frac{\alpha\bar{A}}{\sqrt{b}}\|\xi - \hat{\xi}\| \leq \frac{\alpha\bar{A}}{\mu_*\sqrt{b}}\varepsilon_{\mathrm{sub}}.\end{aligned}$$

□

3.6 Convergence Analysis

We first introduce several important constants. Let (A1) of Assumption 5 be satisfied. We denote the Lipschitz constant of ∇f by L and we set $\bar{L} := \max_i L_i\|A_i\|^2$. For $b \in [N]$, let us define $\bar{L}_b := \max_{S,|S|=b} L_S$ where L_S is the Lipschitz constant of ∇f_S, and $M_S := \frac{1}{b}\sum_{i\in S}\gamma_i A_i^\top A_i$. Then, for any tuple S it holds that

$$L \leq \frac{1}{N}\sum_{i=1}^N L_i\|A_i\|^2 \leq \bar{L}, \qquad \bar{L}_b \leq \bar{L}, \qquad \|M_S\| \leq \max_{i=1,\dots,N}\gamma_i \cdot \bar{A}^2 =: \bar{M}. \tag{3.17}$$

We now formulate the main convergence results for Algorithm 7. For simplicity, we assume that each element of S_k is drawn uniformly from $[N]$ with replacement for all s and k.[1]

3.6.1 Weakly Convex Case

Theorem 3.7. *Let the iterates $\{x^{s,k}\}$ be generated by Algorithm 7 with $\mathsf{S} = \infty$, constant batch sizes $b_k^s = b$, and using Option I. Let Assumption 5 and Assumption 6 be satisfied and assume*

$$\sum_{s=0}^{\infty}\sum_{k=0}^{m-1}\alpha_k^s = \infty, \quad \sum_{s=0}^{\infty}\sum_{k=0}^{m-1}\alpha_k^s(\varepsilon_k^s)^2 < \infty, \quad \alpha_k^s \leq \min\{1, \hat{\alpha}\} \quad \forall\ s,k, \tag{3.18}$$

where $\hat{\alpha} := \bar{\eta}\max\{2L + \bar{M}, [1 + m/\sqrt{2b}]\bar{L} + \max\{L, \bar{M}\}\}^{-1}$ and $\bar{\eta} \in (0,1)$. Then, for all $0 \leq k < m$, $\{\mathbb{E}\|F_{\mathrm{nat}}(x^{s,k})\|\}_{s\in\mathbb{N}_0}$ converges to zero and $\{F_{\mathrm{nat}}(x^{s,k})\}_{s\in\mathbb{N}_0}$ converges to zero almost surely as $s \to \infty$.

Similar to [68, Thm. 1], we obtain the following rate of convergence.

Corollary 3.8. *Let the iterates $\{x^{s,k}\}$ be generated by Algorithm 7 with $\mathsf{S} \in \mathbb{N}$, constant step sizes $\alpha_k^s = \alpha$, constant batch sizes $b_k^s = b$, and using Option I. Let Assumption 5 and Assumption 6 be satisfied. Let $\bar{\eta} \in (0,1)$ be given and assume $\alpha \leq \hat{\alpha}$, where $\hat{\alpha}$ is defined as in Theorem 3.7. Then, it holds that*

$$\mathbb{E}\|F_{\mathrm{nat}}^\alpha(x_\pi)\|^2 \leq \frac{2\alpha[\psi(\tilde{x}^0) - \psi^\star + \alpha\cdot\mathcal{O}(\sum_{s=0}^{\mathsf{S}}\sum_{k=0}^{m-1}(\varepsilon_k^s)^2)]}{(1-\bar{\eta})^3\cdot m(\mathsf{S}+1)}.$$

The proofs are given in Section 3.8.2.

[1] Without replacement, only some of the constants change, see Corollary 3.14 for details.

3.6.2 Strongly Convex Case

In this section, we establish q-linear convergence of Algorithm 7 if ψ is strongly convex. We derive – similar to Thm. 3.1 in [157] – convergence in terms of the objective function if we assume each f_i to be convex. We present an additional result for weakly convex f and strongly convex φ. The proofs are given in Section 3.8.3. In the following, we suppose that in iteration s of the outer and iteration k of the inner loop of Algorithm 7, the tolerances ε_k^s satisfy the bound

$$\varepsilon_k^s \leq \delta_s \|F_{\mathrm{nat}}(\tilde{x}^s)\| \tag{3.19}$$

for all $k \in \{0, \dots, m-1\}$, $s \in \mathbb{N}$, and for some sequence $\mathbb{R}_{++} \ni \delta_s \to 0$. Notice, since $\nabla f(\tilde{x}^s)$ is known, $\|F_{\mathrm{nat}}(\tilde{x}^s)\|$ can be computed without additional costs.

Theorem 3.9. *Let Assumption 5 and Assumption 6 be satisfied and suppose that each function f_i is convex and $\psi = f + \varphi$ is μ-strongly convex with $\mu > 0$. Consider Algorithm 7 with $\mathsf{S} = \infty$ and Option II using constant step sizes $\alpha_k^s = \alpha > 0$, and constant batch sizes $b_k^s = b$. For $\theta \in (0, 1/2)$, let the step size α satisfy*

$$\alpha \leq \left[L + \tfrac{\bar{L}}{b}\left(\tfrac{4}{1-2\theta} + 3\right)\right]^{-1} \tag{3.20}$$

and let condition (3.19) *hold for a given sequence $\{\delta_s\}$. If δ_s is sufficiently small and the inner loop length m sufficiently large, then $\{\psi(\tilde{x}^s)\}$ converges q-linearly in expectation to $\psi^\star$ with rate at least $1-\theta$, i.e., for all s, we have*

$$\mathbb{E}[\psi(\tilde{x}^{s+1}) - \psi^\star] \leq (1-\theta)\mathbb{E}[\psi(\tilde{x}^s) - \psi^\star].$$

More formal and explicit conditions on δ_s and m can be found in the proof of Theorem 3.9 in Section 3.8.3.

Theorem 3.10. *Let Assumption 5 and Assumption 6 be satisfied and let φ and $\psi = f + \varphi$ be μ_φ- and μ-strongly convex, respectively, with*

$$\mu_\varphi I - M_N \succeq \mu I.$$

Consider Algorithm 7 with $\mathsf{S} = \infty$ and Option I, using constant step sizes $\alpha_k^s = \alpha > 0$ and constant batch sizes $b_k^s = b$. Assume that $\alpha \leq [L + \sqrt{2/b} \cdot m\bar{L}]^{-1}$ and let (3.19) *hold for a given sequence $\{\delta_s\}$ satisfying $\delta_s < \min\{\frac{2\alpha\mu}{1+\alpha\mu_\varphi}, \frac{1+\alpha(\mu+\mu_\varphi)}{1+\alpha\mu_\varphi}\}$ for all s. Then, the iterates $\{\tilde{x}^s\}$ converge q-linearly in expectation to the unique solution $x^\star$ of problem* (3.2)*, i.e., as $s \to \infty$, we have*

$$\mathbb{E}\|\tilde{x}^{s+1} - x^\star\|^2 \leq \left[1 - \tfrac{2\alpha\mu}{1+\alpha(\mu_\varphi+\mu)} + \mathcal{O}(\delta_s)\right] \mathbb{E}\|\tilde{x}^s - x^\star\|^2.$$

Theorem 3.9 and Theorem 3.10 are only slightly different in their assumptions and statements. For example, the guarantee is given either in terms of the objective or of the distance to the solution. However, for both results the convergence rate is linear.

3.7 Numerical Experiments

In the following, we investigate the practical performance of Algorithm 7 which we will refer to as `SNSPP`. Specifically, we compare `SNSPP` with three state-of-the-art and benchmark stochastic algorithms – namely `SVRG`, `SAGA`, and `AdaGrad` for all of which we use their proximal versions. For a detailled description and bibliographic notes regarding the benchmark methods, we refer the reader to Section 2.5.2 and Section 2.6 as well as to Algorithms 2 to 4.

3.7.1 General Setting

For all experiments, we set the parameter m in Algorithm 7 to $m = 10$. For Algorithm 8, we use $\hat{\gamma} = 0.4$, $\eta = 10^{-5}$, $\rho = 0.5$, $\tau = 0.9$, $\tau_1 = 0.5$, $\tau_2 = 2 \cdot 10^{-4}$ and terminate if $\|\nabla \mathcal{U}(\xi^j)\| \leq 10^{-3}$. Typically, Algorithm 8 reaches the desired accuracy within less than 10 iterations. We also ran the experiments with the adaptive accuracy $\varepsilon_k^s = \delta_s \|F_{\text{nat}}(\tilde{x}^s)\|$, $\delta_s = 0.1$, introduced in (3.19) but did not observe any significant effects on the results. We precompute an estimate for $\psi^\star$ by running a solver for a large number of iterations: for the experiments in Section 3.7.2, we use `scikit-learn` [115] for this step; for the experiments in Section 3.7.3 we use our own implementation of `SAGA`. We use constant batch and step sizes throughout all experiments. The experiments were performed in `Python 3.8`.[2]

3.7.2 Logistic Regression with ℓ_1-Regularization

Sparse logistic regression is a classical model for binary classification [57, 75]. Let the coefficient matrix $A = (a_1^\top, \dots, a_N^\top)^\top \in \mathbb{R}^{N \times n}$ and the binary labels $b_i \in \{-1, 1\}$, $i = 1, \dots, N$ be given. The associated sparse regression problem can then be formulated as follows:

$$\min_x \ \frac{1}{N} \sum_{i=1}^{N} \ln(1 + \exp(-b_i \langle a_i, x \rangle)) + \lambda \|x\|_1, \quad \lambda > 0.$$

This problem is of the form Eq. (3.2) with $m_i = 1$, $A_i = b_i a_i$, and $f_i(z) = f_{\log}(z) := \ln(1 + \exp(-z))$ for all $i = 1, \dots, N$. The nonsmooth part is given by $\varphi(x) = \lambda \|x\|_1$.

Since $f_{\log}$ is convex, we can set $\gamma_i = 0$ for all i. The conjugate $f_{\log}^*$ of the logistic loss function is given by [75] (see also Section 3.9.1):

$$f_{\log}^*(z) = \begin{cases} -z \ln(-z) + (1+z) \ln(1+z) & -1 < z < 0 \\ +\infty & \text{otherwise.} \end{cases} \tag{3.21}$$

The mapping $f_{\log}^*$ is $\mathcal{C}^\infty$ on $(-1, 0)$ and locally Lipschitz. For all $z \in (-1, 0)$, we have $(f_{\log}^*)'(z) = \ln(1+z) - \ln(-z)$ and $(f_{\log}^*)''(z) = -\frac{1}{z^2+z} \geq 4$. We conclude that $f_{\log}^*$ is essentially differentiable and strongly convex on $(-1, 0)$.

The proximity operator of φ and its Clarke differential are discussed in, e.g., [164]. The proximity operator is the well-known soft-thresholding operator $\text{prox}_{\lambda\|\cdot\|_1}(x) = \text{sign}(x) \odot \max\{|x| - \lambda \mathbf{1}, 0\}$ where all operations are component-wise. We can choose the generalized derivative $D \in \partial \text{prox}_{\lambda\|\cdot\|_1}(x)$ as follows: $D = \text{Diag}(d_i)_{i=1,\dots,n}$ and $d_i = 0$ if $|x_i| \leq \lambda$ and $d_i = 1$ if $|x_i| > \lambda$. Lem. 2.1 in [164] ensures that $\text{prox}_{\lambda\|\cdot\|_1}$ is strongly semismooth. Altogether, Assumption 5 and 6 are satisfied (due to strong semismoothness of $\text{prox}_{\lambda\|\cdot\|_1}$, Assumption 7 holds as well).

Description of Datasets. We use several standard datasets for our experiments, listed in Table 3.1.[3] The dataset `mnist` contains 28×28 pixel pictures of hand-written digits [82]. In order to obtain binary labels, we classify the two sets of digits $\{0, 3, 6, 8, 9\}$ and $\{1, 2, 4, 5, 7\}$.

[2] Code is available at https://github.com/fabian-sp/snspp.

[3] Dataset `sido0` is downloaded from http://www.causality.inf.ethz.ch/challenge.php?page=datasets#cont, `higgs` from https://archive.ics.uci.edu/ml/datasets/HIGGS, and `mnist` from openml.org using the `scikit-learn` API. All other datasets are downloaded from LIBSVM, https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.

Dataset	N	n	$\psi^\star$	λ
`mnist`	56000	784	0.552	0.02
`gisette`	4800	4955	0.476	0.05
`sido0`	10142	4932	0.146	0.01
`covtype`	581012	54	0.642	0.005
`higgs`	$11 \cdot 10^6$	28	0.657	0.005
`madelon.2`	2000	125751	0.568	0.02
`news20`	15996	1355191	0.653	0.001

Table 3.1: Information of the different datasets for sparse logistic regression. The optimal objective function value $\psi^\star$ is rounded to 3 digits.

`gisette` is derived from `mnist` but with additional higher-order and distractor features [55]. The `madelon.2` dataset is obtained as follows: we first scale the original `madelon` dataset (from LIBVSM) having 500 features and 2000 samples obtaining mean-zero and unit-variance features. Then, we apply a polynomial feature expansion of degree two, i.e., we add all pairwise products of features and a constant feature, resulting in $n = 125751$. For `mnist` and `higgs`, we apply standard preprocessing to obtain mean-zero and unit-variance features. The other datasets are already suitably scaled and therefore not preprocessed. For `mnist`, `gisette`, `sido0`, `covtype`, and `news20`, we use 80% of the dataset samples for training and the remaining 20% are used as validation set. Note that `SNSPP` and `SVRG` compute the full gradient for the first time at the starting point $\tilde{x}^0$. In contrast to `SVRG`, the first iterate of `SNSPP` is not deterministic and therefore high variance in the gradient at the starting point could lead to unfavorable performance. In particular, we observe such effect for the `sido0` dataset. We find that this behavior can be easily prevented by running one iteration of `SNSPP` without variance reduction before computing the full gradient. However, for better comparability, for `sido0` we run one epoch of `SAGA` and use the final iterate as starting point for all methods. For all other datasets, we use $\tilde{x}^0 = 0$ as initial point for all algorithms.

Subproblem Complexity. We first illustrate the impact of solving the subproblems, i.e., invoking Algorithm 8, on the overall performance of `SNSPP`. Fig. 3.1 depicts the subproblem complexity (in terms of runtime) and the overall progress of `SNSPP` for different choices of batch sizes using the `news20` dataset. As the batch size b determines the dimension of the subproblem, we see a sharp increase in the runtime for larger choices of b (bottom right). However, a larger batch size also allows to take larger steps and therefore more progress per iteration can be made as demonstrated in the left plot of Fig. 3.1. In our experiments, we typically observe that the subproblems can be solved very efficiently for batch sizes up to the order of few hundreds. For much larger batch sizes, the resulting higher computational costs of the subproblem will start to outweigh the benefits of reducing the variance of ∇f_S.

Stability Analysis. The main focus of our numerical test is put on stability experiments with respect to hyperparameter selection, in particular, the step size. In practical scenarios, it is unlikely that a solver is executed with an intensively tuned step size due to tuning budgets. Hence, it is important to evaluate optimization methods considering the amount of step size tuning needed to reach optimal performance/runtime [137]. A similar comparison of `SPP` and `SGD` was conducted in [6,26], but without variance reduction, on a single batch, and with synthetic

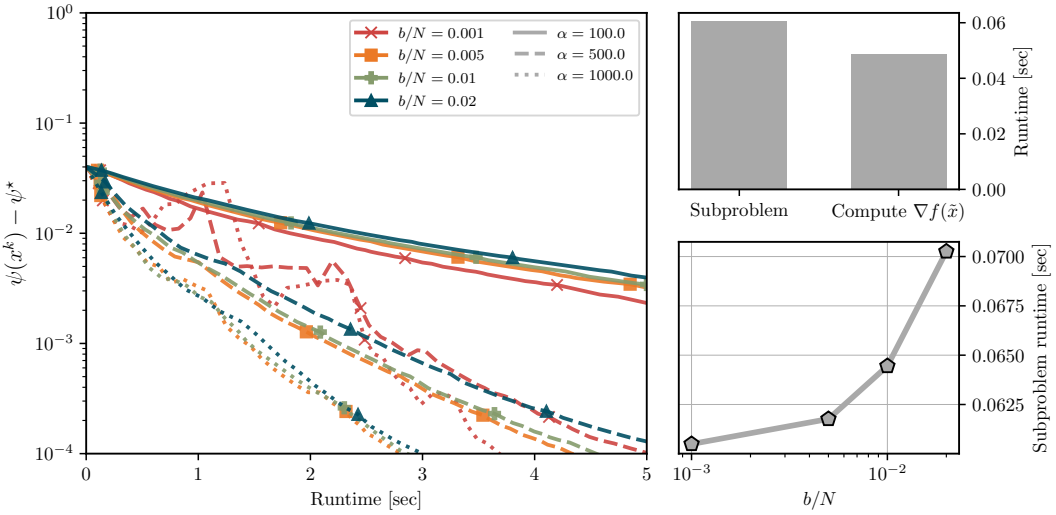


Figure 3.1: Sparse logistic regression for `news20`. Left: Objective function gap for different batch and step sizes. Top Right: average runtime for solving the subproblem once (with Algorithm 8) and for computing $\nabla f(\tilde{x})$. Displayed for $\alpha = 1000, b = 0.005 \cdot N$. Bottom Right: Mean runtime (per iteration index k in `SNSPP`) for solving the subproblem.

data only. We compare `SNSPP` to the other variance-reduced methods `SAGA` and `SVRG`. We solve the ℓ_1-regularized logistic regression problem for several datasets, for a range of step sizes α and different batch sizes b.[4] For `SVRG`, we set the inner loop length to $\lfloor N/b \rfloor$.

The tested algorithms terminate at iteration k, if the criterion

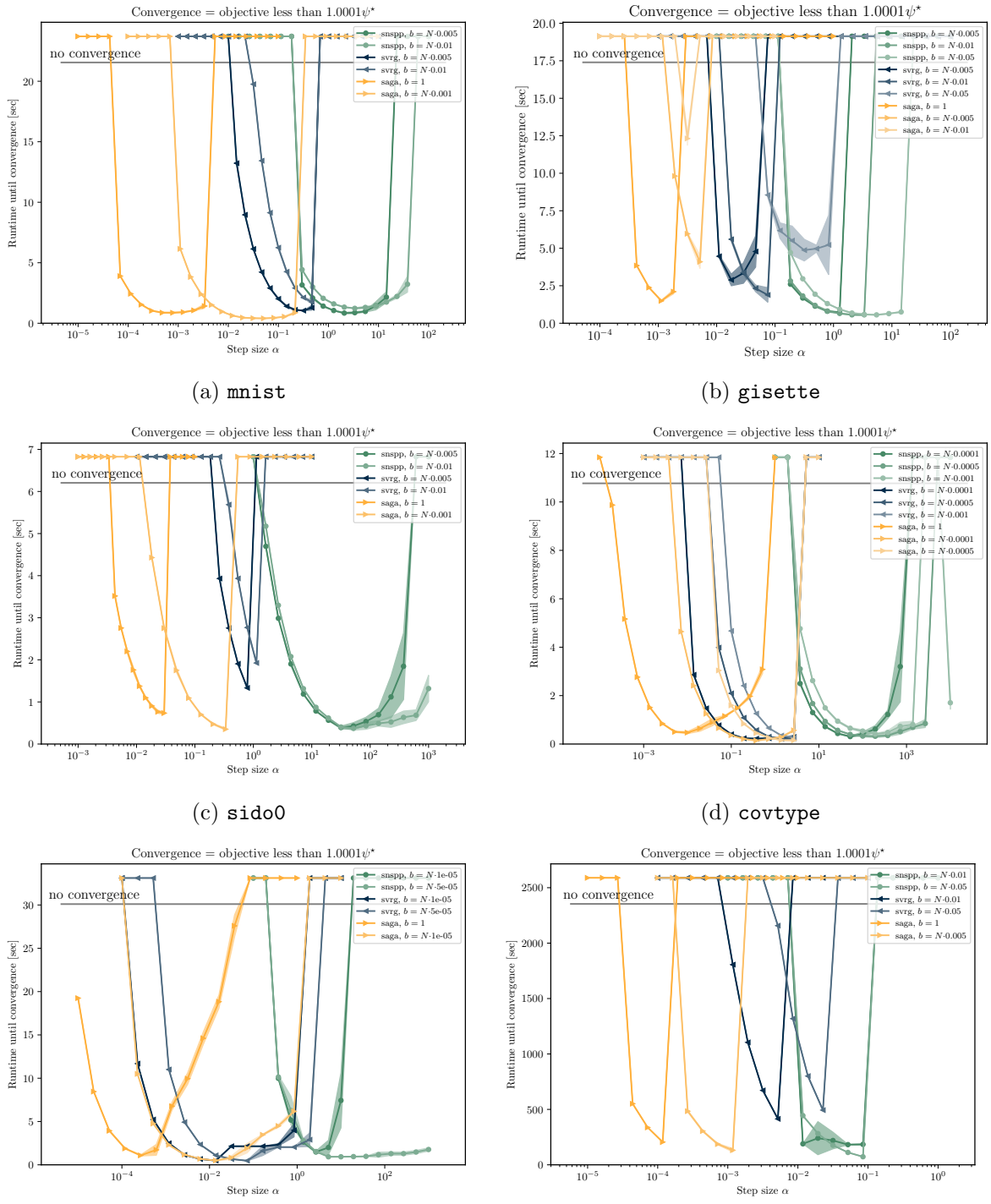
$$\psi(x^k) \leq 1.0001\psi^\star \tag{3.22}$$

is satisfied. The runtime elapsed until fulfilling (3.22), averaged over five independent runs, is plotted in Fig. 3.2. The shaded area depicts the bandwidth of two standard deviations (over the five independent runs). If a method does not satisfy (3.22) within some maximum number of iterations or if it diverges, it is marked as *no convergence*.

Discussion. First, we observe that for all instances, `SNSPP` converges for much larger step sizes than `SAGA` and `SVRG`. The elapsed runtime until convergence of `SNSPP` is robust to step size selection across all datasets. For `mnist`, `covtype` and `higgs`, the robustness of `SNSPP` is comparable to `SAGA` and slightly better than `SVRG`. For `gisette`, `sido0`, and `madelon.2` (all of which are datasets where n is large(r)) the advantage of `SNSPP` is most pronounced: for `madelon.2`, the runtimes of the best parameter settings are: `SNSPP`: 73 sec, `SAGA`: 132 sec, `SVRG`: 413 sec. `SNSPP` further converges in less than 300 seconds for a large range of step sizes (i.e., without extensive tuning) while for `SAGA` and `SVRG` the runtime steeply increases beyond 500 seconds if the step size is chosen too small (see Fig. 3.7 for a convergence plot). Our results underpin the numerical evidence in [6,26] that implicit stochastic proximal point methods tend to be more robust with respect to step size choices than stochastic gradient descent-type approaches.

Speed of Convergence. Based on the stability results depicted in Fig. 3.2, we now illustrate the speed of convergence of `SNSPP` compared to `SAGA`, `SVRG` and `AdaGrad`. For the experiments

[4] We always include results for `SAGA` with $b = 1$ as this setting is widely adopted, for example in `scikit-learn`.



(a) `mnist` (b) `gisette`

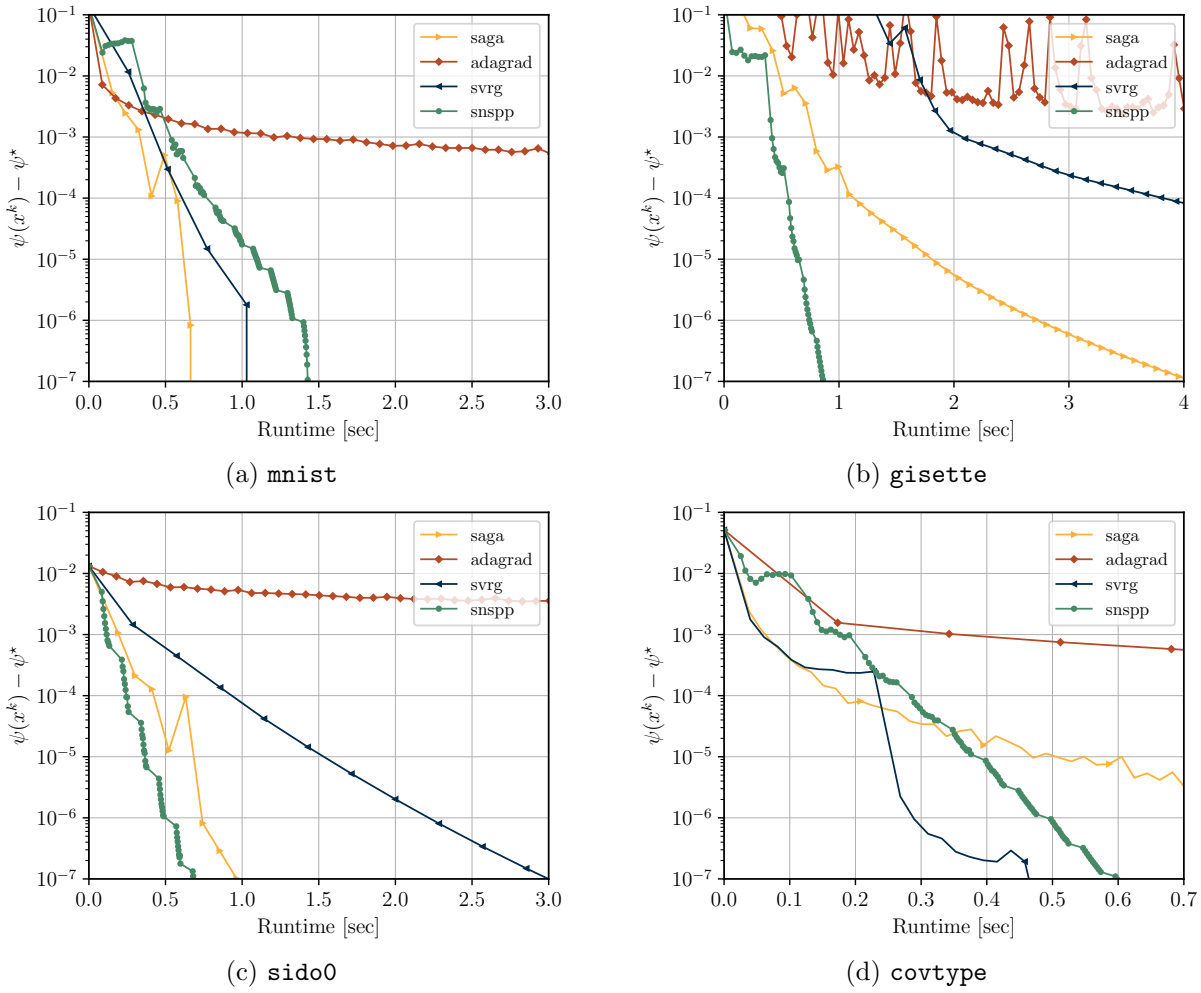
(c) `sido0` (d) `covtype`

(e) `higgs` (f) `madelon.2`

Figure 3.2: Runtime until convergence for different choices of step and batch sizes.

in this section, we choose a manually tuned (constant) batch and step size for all methods in order to allow a fair comparison. Details on the tuning procedure and the specific batch and step sizes values are reported in Table 3.2 in Section 3.8.4.

We plot the objective function value – averaged over 20 independent runs – over the (average)



(a) `mnist` (b) `gisette`

(c) `sido0` (d) `covtype`

Figure 3.3: Objective function convergence for the logistic regression datasets. For `SAGA` and `AdaGrad`, one marker denotes one epoch. For `SVRG` one marker denotes one outer-loop iteration while for `SNSPP` it denotes one (inner-loop) iteration.

cumulative runtime in Fig. 3.3. Due to the incorporated variance reduction, `SNSPP` converges to the optimal value and requires a relatively low number of iterations (compared to the other methods) in order to reach a high accuracy solution. However, in each iteration we need to run Algorithm 8 instead of having a closed-form update. Overall in terms of runtime, for `mnist` and `covtype`, `SNSPP` is slightly slower than `SAGA/SVRG` but still competitive. For `sido0` and `gisette`, `SNSPP` is the fastest method. We also plot the convergence in terms of gradient evaluations, ignoring all other computational costs, in Fig. 3.6.

3.7.3 Sparse Student-t Regression

Next, for a given matrix $A \in \mathbb{R}^{N\times n}$ (with rows a_i) and measurements $b = (b_1, \dots, b_N) \in \mathbb{R}^N$, we consider sparse regression problems of the form

$$\min_x \ \mathcal{L}(Ax - b) + \lambda \|x\|_1. \tag{3.23}$$

where $\lambda > 0$ is a regularization parameter and $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$ is a loss function. In statistical learning, problem (3.23) with the squared loss $\mathcal{L}(r) = \frac{1}{N}\|r\|^2$ is known as the Lasso [150].

Other regularization terms have been proposed in order to model group sparsity or ordered features [143]. While the squared loss is suitable when b is contaminated by Gaussian noise, more heavy-tailed distributions have been studied in the presence of large outliers [64]. For instance, using the Student-t distribution [4, 79] and the respective maximum-likelihood loss, problem (3.23) becomes

$$\min_x \ \frac{1}{N}\sum_{i=1}^{N} \ln\left(1+\hat{\nu}^{-1}(\langle a_i, x\rangle - b_i)^2\right) + \lambda\|x\|_1. \tag{3.24}$$

where $\hat{\nu} > 0$ is the degrees of freedom-parameter of the Student-t distribution. Problem (3.24) is of the form (3.2) with $f_i(z) = \ln(1+\hat{\nu}^{-1}(z-b_i)^2)$.

We now fix $b \in \mathbb{R}$ and consider the scalar function $f_{\text{std}} : \mathbb{R} \to \mathbb{R}$, $f_{\text{std}}(x) := \ln(1+\hat{\nu}^{-1}(x-b)^2)$; we have $f'_{\text{std}}(x) = \frac{2(x-b)}{\hat{\nu}+(b-x)^2}$ and $f''_{\text{std}}(x) = \frac{2(\hat{\nu}-(b-x)^2)}{(\hat{\nu}+(b-x)^2)^2}$. The minimum of f''_{std} is attained at $x \in \{b+\sqrt{3\hat{\nu}}, b-\sqrt{3\hat{\nu}}\}$ and we can conclude $\inf_x f''_{\text{std}}(x) = -\frac{1}{4\hat{\nu}}$. Consequently, f_{std} is $\frac{1}{4\hat{\nu}}$-weakly convex. Next, we compute the convex conjugate of $x \mapsto \hat{f}_{\text{std}}(x) := f_{\text{std}}(x) + \frac{\gamma}{2}x^2$ which is strongly convex for $\gamma > \frac{1}{4\hat{\nu}}$. For fixed $x \in \mathbb{R}$, it holds that

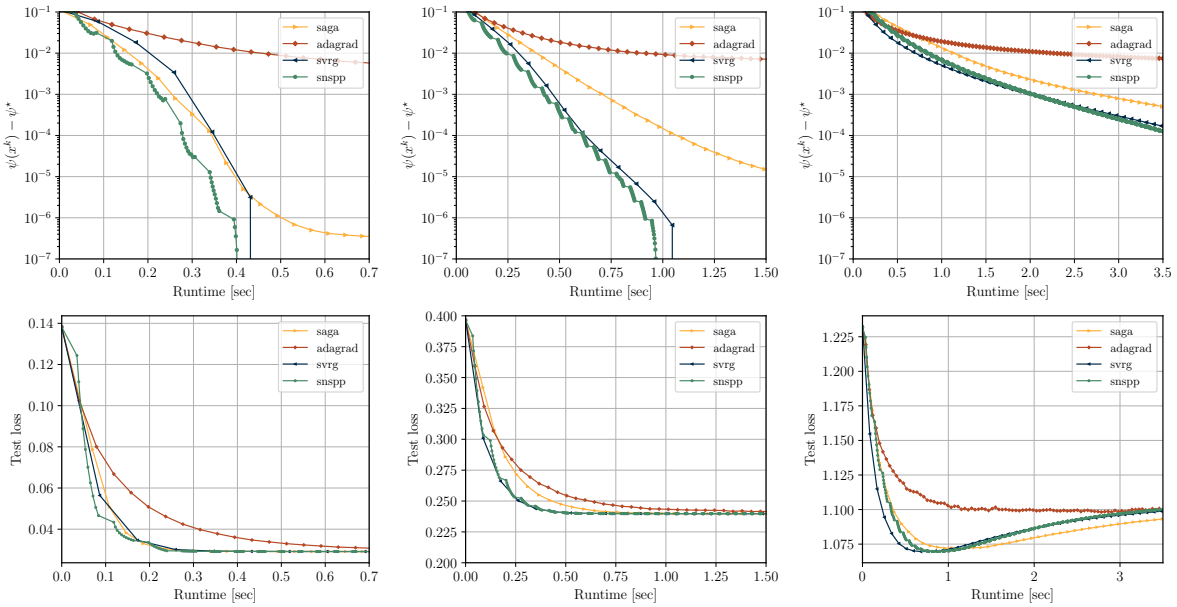
$$\begin{aligned} & z = \arg\sup_y \ xy - \hat{f}_{\text{std}}(y) \iff x - f'_{\text{std}}(z) - \gamma z = 0 \\ \iff & -\gamma z^3 + z^2\,(x+2\gamma b) + z\left(-2bx - 2 - \gamma\hat{\nu} - \gamma b^2\right) + \left(x\hat{\nu} + xb^2 + 2b\right) = 0. \end{aligned} \tag{3.25}$$

Choosing $\gamma > \frac{1}{4\hat{\nu}}$, (3.25) has a unique real solution z^* for any $x, b \in \mathbb{R}$ due to strong convexity. Applying Lemma 3.11 yields

$$\hat{f}^*_{\text{std}}(x) = xz^* - \hat{f}_{\text{std}}(z^*), \quad (\hat{f}^*_{\text{std}})'(x) = z^*, \quad (\hat{f}^*_{\text{std}})''(x) = (\hat{f}''_{\text{std}}(z^*))^{-1}.$$

We solve the cubic polynomial equation in (3.25) using Halley's method [30]. We run two different settings: First, we use a synthetic dataset with $n = 5000$, $N = 4000$, $N_{\text{test}} = 400$, $\lambda = 0.001$, and $\hat{\nu} \in \{0.5, 1, 2\}$. We generate $\hat{x} \in \mathbb{R}^n$ with 20 non-zero entries. To obtain A and b, we first perform a SVD of a $(N+N_{\text{test}}) \times n$-matrix with entries drawn uniformly at random from $[-1, 1]$ and rescale its non-zero singular values to lie in the interval $[1, 15]$. We use $\tilde{A}$ to denote the resulting larger matrix and we compute $\tilde{b}$ via $\tilde{b} = \tilde{A}\hat{x} + 0.1 \cdot \bar{\varepsilon}$ where $\bar{\varepsilon} \in \mathbb{R}^{N+N_{\text{test}}}$ is generated from a Student-t distribution with degrees of freedom $\hat{\nu}$. A and b are then given as the first N rows/entries of $\tilde{A}$ and $\tilde{b}$. The remaining rows/entries are used as a test set. Secondly, we consider problem (3.24) using the feature matrix A from the `sido0` dataset. We generate $\hat{x}$ with 50 non-zero entries and compute $b = A\hat{x} + 0.1 \cdot \bar{\varepsilon}$ where $\bar{\varepsilon} \in \mathbb{R}^N$ is generated from a Student-t distribution with degrees of freedom $\hat{\nu} = 2$. As in the previous test, 20% of the samples are used as test set (applying the same procedure) and we set $\lambda = 0.01$. We follow the same tuning strategy as described in Section 3.7.2. The objective function and test loss are averaged over 20 independent runs.

Discussion. For the synthetic data (Fig. 3.4), we observe that `SNSPP` performs comparably to `SVRG` and `SAGA` in reducing the objective as well as the Student-t likelihood loss on a held-out test set. In order to exclude the possibility that the methods converge to different points with similar objective function values, we verified that the iterates of all methods follow a similar path and that the final iterates stay very close in terms of Euclidean distance. (Only the iterates generated by `AdaGrad` show a more oscillatory behavior which is expected as it does not use variance reduction). Fig. 3.5 shows the results for the regression on `sido0`: here, `SNSPP` performs favorably compared to the other methods – both in terms of objective function and test loss.



(a) $\hat{\nu} = 2$ (b) $\hat{\nu} = 1$ (c) $\hat{\nu} = 0.5$

Figure 3.4: Objective function (top) and test loss (bottom) for Student-t regression with different values of degrees of freedom. Test loss is defined as the value of f_{std} averaged over the samples of the test set.

3.8 Supplementary Material and Missing Proofs

Lemma 3.11. *Let $h : \mathbb{R}^n \to \mathbb{R}$ be a strongly convex, twice continuously differentiable mapping and let $z^*(x)$ denote the (unique) solution to $\max_z \langle x, z\rangle - h(z)$. Then, the convex conjugate $h^* : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^2$ and for all $x \in \mathbb{R}^n$ it holds that*

$$h^*(x) = \langle x, z^*(x)\rangle - h(z^*(x)), \quad \nabla h^*(x) = z^*(x), \quad \nabla^2 h^*(x) = \left[\nabla^2 h(z^*(x))\right]^{-1}.$$

Proof. As h is strongly convex and $\mathcal{C}^2$, $z^*(x)$ is the unique solution to $\nabla h(z) = x$. By [9, Thm. 4.20], we have $\nabla h(y) = x$ if, and only if, $\nabla h^*(x) = y$ for all $x, y \in \mathbb{R}^n$, which implies $\nabla h^*(x) = z^*(x)$. The inverse function theorem yields that $x \mapsto z^*(x)$ is $\mathcal{C}^1$ with Jacobian $Dz^*(x) = [\nabla^2 h(z^*(x))]^{-1}$; as $Dz^*(x) = \nabla^2 h^*(x)$ the statement is proven. □

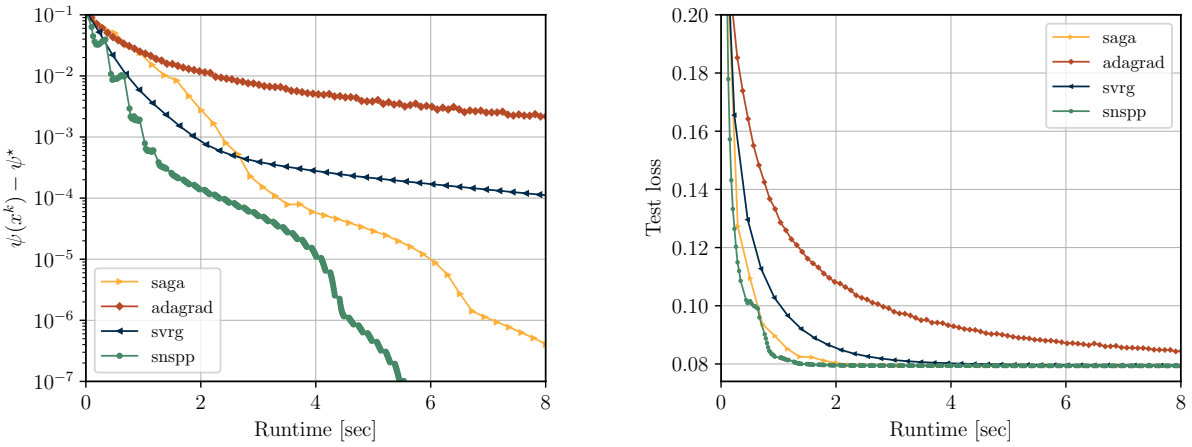
3.8.1 Bounding the Variance

In this section, let $\mathcal{F}$ be a σ-algebra and suppose that x and $\tilde{x}$ are $\mathcal{F}$-measurable random variables in $\mathbb{R}^n$. For $i \in [N]$, let us further define $\zeta_i := A_i^\top(\nabla f_i(A_i x) - \nabla f_i(A_i \tilde{x}))$.

Lemma 3.12. *Suppose that condition* (A1) *is satisfied and let the index i be drawn uniformly from $[N]$ and independently of $\mathcal{F}$. Conditioned on $\mathcal{F}$, we then have $\mathbb{E}\|\zeta_i\|^2 \leq \bar{L}^2\|x - \tilde{x}\|^2$ almost surely. In addition, if every f_i is convex and there exists $x^\star \in \arg\min_x \psi(x)$, then it holds*

$$\mathbb{E}\|\zeta_i\|^2 \leq 4\bar{L}(\psi(x) - \psi(x^\star) + \psi(\tilde{x}) - \psi(x^\star)) \quad \textit{almost surely.}$$

Proof. The first statement follows directly from Lipschitz-smoothness. To prove the second part,



(a) Objective function

(b) Test loss

Figure 3.5: Sparse Student-t regression for `sido0` dataset. Test loss is defined as the value of f_{std} averaged over the samples of the test set.

let us define $\phi_i : x \mapsto f_i(A_i x)$. Lem. 3.4 in [157] (applied to ϕ_i) implies

$$\begin{aligned}\mathbb{E}\|\nabla\phi_i(x) - \nabla\phi_i(\tilde{x})\|^2 &\leq 2\mathbb{E}\|\nabla\phi_i(x) - \nabla\phi_i(x^\star)\|^2 + 2\mathbb{E}\|\nabla\phi_i(\tilde{x}) - \nabla\phi_i(x^\star)\|^2 \\ &\leq 4\bar{L}(\psi(x) - \psi(x^\star) + \psi(\tilde{x}) - \psi(x^\star)),\end{aligned}$$

where we used $\|a+b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. We conclude as $\zeta_i = \nabla\phi_i(x) - \nabla\phi_i(\tilde{x})$. □

Lemma 3.13. *Let S be a b-tuple drawn uniformly from $[N]$ and independently of $\mathcal{F}$. Conditioned on $\mathcal{F}$, the random variable $u := \nabla f_S(x) - \nabla f_S(\tilde{x}) + \nabla f(\tilde{x})$ is an unbiased estimator of $\nabla f(x)$. Moreover, almost surely, if S is drawn*

(i) *with replacement, then $\mathbb{E}\|u - \nabla f(x)\|^2 \leq \frac{1}{b}\mathbb{E}\|\zeta_i\|^2$ for any $i \in S$.*

(ii) *without replacement, then $\mathbb{E}\|u - \nabla f(x)\|^2 \leq (1 - \frac{b}{N})\frac{1}{b(N-1)}\sum_{i=1}^N \mathbb{E}\|\zeta_i\|^2$.*

Proof. Utilizing [87, §2.8], we clearly have $\mathbb{E}[u] = \nabla f(x)$. For part (i), consider $\zeta = \frac{1}{b}\sum_{i\in S}\zeta_i$. It holds $\zeta = u - \nabla f(\tilde{x})$ and, conditioned on $\mathcal{F}$, we obtain

$$\mathbb{E}\|u - \nabla f(x)\|^2 = \frac{1}{b^2}\mathbb{E}\left\|\sum\nolimits_{i\in S}\zeta_i - \mathbb{E}[\zeta_i]\right\|^2$$

since $\mathbb{E}[\zeta_i] = \nabla f(x) - \nabla f(\tilde{x})$ for all $i \in S$. Since the random variables $\{\zeta_i - \mathbb{E}[\zeta_i]\}$ are i.i.d. and have mean zero (conditioned on $\mathcal{F}$), [68, Lem. 7] allows to conclude

$$\mathbb{E}\|u - \nabla f(x)\|^2 = \frac{1}{b^2}\sum\nolimits_{i\in S}\mathbb{E}\|\zeta_i - \mathbb{E}[\zeta_i]\|^2 \leq \frac{1}{b^2}\sum\nolimits_{i\in S}\mathbb{E}\|\zeta_i\|^2. \tag{3.26}$$

This proves the first statement as the random variables ζ_i are identically distributed. The formula in (ii) is shown in [87, §2.8]. □

Combining Lemma 3.12 and Lemma 3.13, we obtain the following result which extends Lem. 3 in [68] and Cor. 3.5 in [157].

Corollary 3.14. *Let* (A1) *hold and let S be a b-tuple drawn uniformly from $[N]$ and independently of $\mathcal{F}$. With u as in Lemma 3.13 and conditioned on $\mathcal{F}$, it holds that*

(i) $\mathbb{E}\|u - \nabla f(x)\|^2 \leq \frac{\bar{L}^2\tau}{b}\|x - \tilde{x}\|^2$;

(ii) *if all* f_i *are convex and* $x^\star \in \arg\min_x \psi(x)$ *exists, then*

$$\mathbb{E}\|u - \nabla f(x)\|^2 \leq \frac{4\bar{L}\tau}{b}(\psi(x) - \psi(x^\star) + \psi(\tilde{x}) - \psi(x^\star));$$

where $\tau = 1$ *if* S *is drawn with replacement and* $\tau = \frac{N-b}{N-1}$ *if* S *is drawn without replacement.*

3.8.2 Proof for the Weakly Convex Case

Proof of Theorem 3.7. Let us fix the index of the outer loop s. Recall the notation $\mathcal{A}_{S_k} = \frac{1}{b}(A_{\kappa_k(1)}^\top, \ldots, A_{\kappa_k(b)}^\top)^\top$ and abbreviate κ_k by κ. Let $(\hat{x}^{k+1}, \hat{\xi}^{k+1})$ denote the pair of exact solutions of the implicit updates Eq. (3.8) and Eq. (3.11). In particular, setting $w^k := \mathcal{A}_{S_k}^\top \xi^{k+1} - M_{S_k} x^k + v^k$ and $\hat{w}^k := \mathcal{A}_{S_k}^\top \hat{\xi}^{k+1} - M_{S_k} x^k + v^k$, we have

$$x^{k+1} = \mathrm{prox}_{\alpha_k\varphi}(x^k - \alpha_k w^k) \quad \text{and} \quad \hat{x}^{k+1} = \mathrm{prox}_{\alpha_k\varphi}(x^k - \alpha_k \hat{w}^k). \tag{3.27}$$

We introduce the deterministic proximal update

$$\bar{x}^{k+1} = \mathrm{prox}_{\alpha_k\psi}^{\mathbf{Id}+\alpha_k M_N}(x^k) = \mathrm{prox}_{\alpha_k\varphi}(x^k - \alpha_k \nabla f(\bar{x}^{k+1}) - \alpha_k M_N(\bar{x}^{k+1} - x^k)). \tag{3.28}$$

Using the Lipschitz smoothness of f, we obtain

$$f(x^{k+1}) \leq f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k \rangle + \frac{L}{2}\|x^{k+1} - x^k\|^2. \tag{3.29}$$

Setting $p = \mathrm{prox}_{\gamma\varphi}(y - \gamma w)$ and applying the optimality condition of the proximity operator (3.3), it holds that

$$\varphi(p) - \varphi(z) \leq -\langle w, p - z \rangle + \frac{1}{2\gamma}\|y - z\|^2 - \frac{1}{2\gamma}\|p - y\|^2 - \frac{1}{2\gamma}\|p - z\|^2 \tag{3.30}$$

for all $y, w \in \mathbb{R}^n$, $z \in \mathrm{dom}(\varphi)$, and $\gamma > 0$, see, e.g., [68, Lem. 1] for comparison. Setting $y = x^k$, $\gamma = \alpha_k$, $w = w^k$, and $z = \bar{x}^{k+1}$, this yields

$$\begin{aligned} \varphi(x^{k+1}) \leq \varphi(\bar{x}^{k+1}) - \langle w^k, x^{k+1} - \bar{x}^{k+1} \rangle + \tfrac{1}{2\alpha_k}\|\bar{x}^{k+1} - x^k\|^2 \\ - \tfrac{1}{2\alpha_k}\|x^{k+1} - x^k\|^2 - \tfrac{1}{2\alpha_k}\|x^{k+1} - \bar{x}^{k+1}\|^2. \end{aligned} \tag{3.31}$$

Moreover, setting $y = x^k$, $\gamma = \alpha_k$, $w = \nabla f(\bar{x}^{k+1}) + M_N(\bar{x}^{k+1} - x^k)$, and $z = x^k$ in (3.30) and recalling (3.28), it follows

$$\varphi(\bar{x}^{k+1}) \leq \varphi(x^k) - \langle \nabla f(\bar{x}^{k+1}) + M_N(\bar{x}^{k+1} - x^k), \bar{x}^{k+1} - x^k \rangle - \tfrac{1}{\alpha_k}\|\bar{x}^{k+1} - x^k\|^2. \tag{3.32}$$

Adding the three inequalities (3.29), (3.31), and (3.32), we obtain

$$\begin{aligned} \psi(x^{k+1}) \leq \psi(x^k) + \langle \nabla f(x^k) - \nabla f(\bar{x}^{k+1}), \bar{x}^{k+1} - x^k \rangle + \langle \nabla f(x^k) - w^k, x^{k+1} - \bar{x}^{k+1} \rangle \\ - \|\bar{x}^{k+1} - x^k\|^2_{M_N + \frac{1}{2\alpha_k}\mathbf{Id}} + \tfrac{1}{2}\left[L - \tfrac{1}{\alpha_k}\right]\|x^{k+1} - x^k\|^2 - \tfrac{1}{2\alpha_k}\|x^{k+1} - \bar{x}^{k+1}\|^2. \end{aligned} \tag{3.33}$$

Let us define $\tilde{v}^k := \nabla f_{S_k}(x^k) - \nabla f_{S_k}(\tilde{x}) + \nabla f(\tilde{x}) = \nabla f_{S_k}(x^k) + v^k$. We decompose the term $\nabla f(x^k) - w^k$ as follows:

$$\begin{aligned}
&\nabla f(x^k) - w^k \\
&= \underbrace{-\mathcal{A}_{S_k}^\top(\xi^{k+1} - \hat{\xi}^{k+1})}_{=:T_1} \underbrace{-M_{S_k}(\hat{x}^{k+1} - x^{k+1}) - (\nabla f_{S_k}(\hat{x}^{k+1}) - \nabla f_{S_k}(x^{k+1}))}_{=:T_2} \\
&\quad \underbrace{-M_{S_k}(x^{k+1} - x^k) - (\nabla f_{S_k}(x^{k+1}) - \nabla f_{S_k}(x^k))}_{=:T_3} + \nabla f(x^k) - \tilde{v}^k,
\end{aligned}$$

where we used $\mathcal{A}_{S_k}^\top \hat{\xi}^{k+1} = \nabla f_{S_k}(\hat{x}^{k+1}) + M_{S_k}\hat{x}^{k+1}$ (cf. (3.9), where ξ^{k+1} and x^{k+1} must be replaced by $\hat{\xi}^{k+1}$ and $\hat{x}^{k+1}$, respectively, since in Algorithm 7 and beyond, ξ^{k+1} and x^{k+1} involve inexactness). Next, from the choice of ε_{sub} and Proposition 3.6, we obtain $\|T_1\|^2 \leq \frac{\bar{A}^2}{\mu_*^2 b}\varepsilon_k^2$. Applying Lipschitz smoothness, Proposition 3.6, and using $\|M_{S_k}\| \leq \bar{M}$ (cf. (3.17)), this yields

$$\|T_2\| \leq (\bar{L}_b + \|M_{S_k}\|)\|\hat{x}^{k+1} - x^{k+1}\| \leq (\bar{L}_b + \bar{M})\sqrt{\bar{A}^2 b^{-1}}\mu_*^{-1}\alpha_k\varepsilon_k$$

for all $k = 0, \ldots, m-1$. Using $M_{S_k} \succeq 0$ and Young's inequality, we have

$$\begin{aligned}
\langle T_3, x^{k+1} - \bar{x}^{k+1}\rangle &= -\langle M_{S_k}(\bar{x}^{k+1} - x^k), x^{k+1} - \bar{x}^{k+1}\rangle - \|x^{k+1} - \bar{x}^{k+1}\|_{M_{S_k}}^2 \\
&\quad - \langle \nabla f_{S_k}(x^{k+1}) - \nabla f_{S_k}(x^k), x^{k+1} - \bar{x}^{k+1}\rangle \\
&\leq \left[\tfrac{\bar{M}}{2\sigma_k^3} + \tfrac{\bar{L}_b}{2\sigma_k^4}\right]\|x^{k+1} - \bar{x}^{k+1}\|^2 + \tfrac{\bar{L}_b\sigma_k^4}{2}\|x^{k+1} - x^k\|^2 + \tfrac{\bar{M}\sigma_k^3}{2}\|\bar{x}^{k+1} - x^k\|^2
\end{aligned}$$

for some $\sigma_k^3, \sigma_k^4 > 0$. Combining these results with (3.33), applying Young's inequality, and defining $\nu_k^1 := L + \frac{1}{2}\bar{M}\sigma_k^3 - \frac{1}{2\alpha_k}$, $\nu_k^2 := \frac{1}{2}[L + \bar{L}_b\sigma_k^4 - \frac{1}{\alpha_k}]$,

$$\nu_k^3 := \tfrac{1}{2}\left[\tfrac{1}{\sigma_k^1} + \tfrac{1}{\sigma_k^2} + \tfrac{\bar{M}}{\sigma_k^3} + \tfrac{\bar{L}_b}{\sigma_k^4} + \tfrac{1}{\sigma_k^5} - \tfrac{1}{\alpha_k}\right],\ \nu_k^4 := \tfrac{1}{2}\left[\sigma_k^1 + \sigma_k^2\alpha_k^2(\bar{L}_b + \bar{M})^2\right]\tfrac{\bar{A}^2}{\mu_*^2 b},$$

we obtain

$$\begin{aligned}
&\psi(x^{k+1}) - \psi(x^k) \\
&\leq \tfrac{\sigma_k^1}{2}\|T_1\|^2 + \tfrac{\sigma_k^2}{2}\|T_2\|^2 + \tfrac{\sigma_k^5}{2}\|\nabla f(x^k) - \tilde{v}^k\|^2 + \left[L + \tfrac{\bar{M}\sigma_k^3}{2} - \tfrac{1}{2\alpha_k}\right]\|\bar{x}^{k+1} - x^k\|^2 \\
&\quad + \tfrac{1}{2}\left[L + \bar{L}_b\sigma_k^4 - \tfrac{1}{\alpha_k}\right]\|x^{k+1} - x^k\|^2 + \nu_k^3\|\bar{x}^{k+1} - x^{k+1}\|^2 \\
&\leq \tfrac{\sigma_k^5}{2}\|\nabla f(x^k) - \tilde{v}^k\|^2 + \nu_k^1\|\bar{x}^{k+1} - x^k\|^2 + \nu_k^2\|x^{k+1} - x^k\|^2 \\
&\quad + \nu_k^3\|\bar{x}^{k+1} - x^{k+1}\|^2 + \nu_k^4\varepsilon_k^2.
\end{aligned}$$

Using Corollary 3.14 (with $x = x^k, \tilde{x} = \tilde{x}^s$), conditioned on all random events occurring until the current iterate $x^k = x^{s,k}$, we have

$$\mathbb{E}\|\nabla f(x^k) - \tilde{v}^k\|^2 \leq \bar{L}^2 b^{-1}\|x^k - \tilde{x}^s\|^2.$$

for all $k \in \{0, \ldots, m-1\}$. Taking expectation in the previous estimate, it holds that

$$\begin{aligned}
\mathbb{E}[\psi(x^{k+1})] &\leq \mathbb{E}[\psi(x^k)] + \tfrac{\bar{L}^2\sigma_k^5}{2b}\mathbb{E}\|x^k - \tilde{x}^s\|^2 + \nu_k^1\mathbb{E}\|\bar{x}^{k+1} - x^k\|^2 \\
&\quad + \nu_k^2\mathbb{E}\|x^{k+1} - x^k\|^2 + \nu_k^3\mathbb{E}\|\bar{x}^{k+1} - x^{k+1}\|^2 + \nu_k^4\varepsilon_k^2.
\end{aligned} \tag{3.34}$$

With the goal of balancing the terms in $\nu_k^1, \dots, \nu_k^4$, we now choose $\sigma_k^1 = m\alpha_k/(1-\bar{\eta})$, $\sigma_k^2 = \sqrt{2b}/[\bar{L}_b\bar{\eta}]$, $\sigma_k^3 = \sigma_k^4 = 1$, and $\sigma_k^5 = \sqrt{2b}/[\bar{L}(m-1)]$. For $k < m$, using $\tilde{x}^s = x^0$ and the Cauchy-Schwarz inequality, we deduce

$$\mathbb{E}\|x^k - \tilde{x}^s\|^2 = \mathbb{E}\left\|\sum\nolimits_{i=0}^{k-1}(x^{i+1}-x^i)\right\|^2 \le k\sum_{i=0}^{k-1}\mathbb{E}\|x^{i+1}-x^i\|^2 \le k\sum_{i=0}^{m-2}\mathbb{E}\|x^{i+1}-x^i\|^2.$$

Summing this estimate for $k = 0, \dots, m-1$ gives

$$\sum_{k=0}^{m-1}\mathbb{E}\|x^k - \tilde{x}^s\|^2 \le \sum_{k=0}^{m-1} k\sum_{i=0}^{m-2}\mathbb{E}\|x^{i+1}-x^i\|^2 \le \tfrac{m(m-1)}{2}\sum_{i=0}^{m-2}\mathbb{E}\|x^{i+1}-x^i\|^2. \tag{3.35}$$

As σ_k^5 is independent of k, summing (3.34) over $k = 0, \dots, m-1$ gives

$$\begin{aligned}\mathbb{E}[\psi(x^m)] \le \mathbb{E}[\psi(x^0)] &+ \sum_{k=0}^{m-1}\left[\tfrac{\bar{L}^2 m(m-1)\sigma_k^5}{4b} + \nu_k^2\right]\mathbb{E}\|x^{k+1}-x^k\|^2 \\ &+ \sum_{k=0}^{m-1}\nu_k^1\mathbb{E}\|\bar{x}^{k+1}-x^k\|^2 + \sum_{k=0}^{m-1}\nu_k^3\mathbb{E}\|x^{k+1}-\bar{x}^{k+1}\|^2 + \sum_{k=0}^{m-1}\nu_k^4\varepsilon_k^2.\end{aligned} \tag{3.36}$$

Let $\bar{\eta} \in (0,1)$ be as stated in Theorem 3.7. Utilizing the specific choices of $\sigma_k^1, \dots, \sigma_k^5$, we obtain $\bar{L}m/\sqrt{2b} + 2\nu_k^2 \le -(1-\bar{\eta})/\alpha_k$ if $\bar{\eta}{\alpha_k}^{-1} \ge [1 + m/\sqrt{2b}]\bar{L} + L$,

$$\nu_k^1 \le -\tfrac{1-\bar{\eta}}{2\alpha_k} \iff \tfrac{\bar{\eta}}{\alpha_k} \ge 2L + \bar{M}, \quad \text{and} \quad \nu_k^3 \le 0 \impliedby \tfrac{1}{\alpha_k} \ge \tfrac{(\bar{M}+\bar{L})m}{m-(1-\bar{\eta})} + \tfrac{\bar{L}m}{\sqrt{2b}}.$$

The second statement follows from plugging in $\sigma_k^1, \dots, \sigma_k^5$, which gives that

$$\nu_k^3 \le 0 \iff \tfrac{1}{\alpha_k} \ge \tfrac{m}{m-1+\bar{\eta}}\big[\bar{L}_b(1+\tfrac{\bar{\eta}}{\sqrt{2b}}) + \bar{M} + \tfrac{\bar{L}(m-1)}{\sqrt{2b}}\big] =: T_4$$

and estimating $T_4 \le \frac{m}{m-1+\bar{\eta}}[\bar{M}+\bar{L}] + \frac{m(m-1+\bar{\eta})}{m-1+\bar{\eta}}\frac{\bar{L}}{\sqrt{2b}}$ due to $\bar{L}_b \le \bar{L}$. Thus, defining

$$\tfrac{1}{\hat{\alpha}} := \tfrac{1}{\bar{\eta}}\max\left\{2L + \bar{M}, \left[1 + \tfrac{m}{\sqrt{2b}}\right]\bar{L} + \max\{\bar{M}, L\}\right\},$$

it holds $\nu_k^4 \le \frac{1}{2}[m/(1-\bar{\eta}) + \sqrt{2b}(1 + \bar{M}/\bar{L}_b)]\frac{\bar{A}^2}{\mu_*^2 b}\alpha_k =: \bar{\nu}\alpha_k$. Further, we have

$$\tfrac{1}{\alpha_k} \ge \tfrac{1}{\hat{\alpha}} \ge \tfrac{1}{\bar{\eta}}\big[\tfrac{m}{\sqrt{2b}}\bar{L} + (\bar{M}+\bar{L})\big] \ge \tfrac{1}{\bar{\eta}}[\bar{M}+\bar{L}] + \tfrac{\bar{L}m}{\sqrt{2b}} \ge \tfrac{(\bar{M}+\bar{L})m}{m-(1-\bar{\eta})} + \tfrac{\bar{L}m}{\sqrt{2b}}.$$

Here, we used $\eta < 1$ and the fact that $m \ge 1$ and thus

$$\tfrac{1}{\bar{\eta}} = 1 + \tfrac{1-\bar{\eta}}{\bar{\eta}} \ge 1 + \tfrac{1-\bar{\eta}}{m-1+\bar{\eta}} = \tfrac{m}{m-(1-\bar{\eta})}.$$

Next, introducing the auxiliary function $y \mapsto \psi_x(y) := \psi(y) + \frac{1}{2}\|y-x\|_{M_N}^2$, we can write $\bar{x}^{k+1} = \operatorname{prox}_{\alpha_k\psi_{x^k}}(x^k)$. Consequently, applying [105, Lem. 2] and [34, Thm. 3.5] with $G = \partial\varphi$, $\Phi = \partial\psi_{x^k}$, $t = \alpha_k$, and $\beta = L + \bar{M}$, it follows:

$$\|\bar{x}^{k+1} - x^k\| \ge (1 - (L+\bar{M})\alpha_k)\|F_{\mathrm{nat}}^{\alpha_k}(x^k)\| \ge (1-\bar{\eta})\alpha_k\|F_{\mathrm{nat}}(x^k)\| \tag{3.37}$$

as long as $\alpha_k \le \min\{\hat{\alpha}, 1\}$. We now use full notations showing the (s,k)-dependence. Let us define $\bar{x}^{s,k+1} := \operatorname{prox}^{\mathbf{Id}+\alpha_k^s M_N}_{\alpha_k^s\psi}(x^{s,k})$,

$$\tau_s^1 := \sum\nolimits_{k=0}^{m-1}(\alpha_k^s)^{-1}\mathbb{E}\|\bar{x}^{s,k+1} - x^{s,k}\|^2, \ \tau_s^2 := \sum\nolimits_{k=0}^{m-1}(\alpha_k^s)^{-1}\mathbb{E}\|x^{s,k+1} - x^{s,k}\|^2.$$

Using $\alpha_k^s \leq \hat{\alpha}$, (3.36) then implies

$$\mathbb{E}[\psi(\tilde{x}^{s+1})] - \mathbb{E}[\psi(\tilde{x}^s)] \leq -\frac{1-\bar{\eta}}{2}(\tau_s^1 + \tau_s^2) + \bar{\nu} \cdot \sum\nolimits_{k=0}^{m-1} \alpha_k^s (\varepsilon_k^s)^2 \tag{3.38}$$

and due to (A2) and $\sum_{s=0}^{\infty}\sum_{k=0}^{m-1} \alpha_k^s (\varepsilon_k^s)^2 < \infty$, we further get

$$\sum\nolimits_{s=0}^{\infty} \tau_s^1 < \infty \quad \text{and} \quad \sum\nolimits_{s=0}^{\infty} \tau_s^2 < \infty. \tag{3.39}$$

Finally, the estimates (3.37) and (3.39) yield

$$\sum_{s=0}^{\infty}\sum_{k=0}^{m-1} \alpha_k^s \mathbb{E}\|F_{\text{nat}}(x^{s,k})\|^2 < \infty \text{ and } \sum_{s=0}^{\infty}\sum_{k=0}^{m-1} \alpha_k^s \|F_{\text{nat}}(x^{s,k})\|^2 < \infty$$

almost surely. Due to $\sum_{s=0}^{\infty}\sum_{k=0}^{m-1} \alpha_k^s = \infty$ and the Borel-Cantelli lemma [38], it holds

$$\liminf_{s\to\infty} \|F_{\text{nat}}(x^{s,k})\|^2 = 0 \quad \text{almost surely for all } \ 0 \leq k < m.$$

The almost sure convergence $F_{\text{nat}}(x^{s,k}) \to 0$ (and $\mathbb{E}\|F_{\text{nat}}(x^{s,k})\| \to 0$) essentially follows from (3.39) and from the Lipschitz continuity of $x \mapsto F_{\text{nat}}(x)$ and can be shown as in [159, Thm. 3.3 and Thm. 4.1]. As this last step is basically identical to the proofs in [159], we omit further details and refer to [159]. □

Proof of Corollary 3.8. We have $\psi(\tilde{x}^{\mathsf{S}+1}) \geq \psi^\star$ for all S. In the previous proof, (3.38) can be obtained directly from (3.36), using only the condition $\alpha_k^s \leq \hat{\alpha}$ on the step sizes (in particular, we do not need to assume $\alpha_k^s \leq \min\{\hat{\alpha}, 1\}$). Summing (3.38) from $s = 0, \ldots, \mathsf{S}$, and setting $\bar{x}_\pi := \operatorname{prox}_{\alpha\psi}^{\mathbf{Id}+\alpha M_N}(x_\pi)$, we conclude

$$\mathbb{E}\|\bar{x}_\pi - x_\pi\|^2 \leq \tfrac{2\alpha}{(1-\bar{\eta})m(\mathsf{S}+1)} \left[\psi(\tilde{x}^0) - \psi^\star + \bar{\nu}\alpha \cdot \sum\nolimits_{s=0}^{\mathsf{S}} \sum\nolimits_{k=0}^{m-1} (\varepsilon_k^s)^2\right].$$

As in (3.37), we can now utilize the bound $\|\bar{x}_\pi - x_\pi\| \geq (1 - (L + \bar{M})\alpha)\|F_{\text{nat}}^\alpha(x_\pi)\| \geq (1 - \bar{\eta})\|F_{\text{nat}}^\alpha(x_\pi)\|$ to express complexity in terms of $\mathbb{E}\|F_{\text{nat}}(x_\pi)\|^2$. □

3.8.3 Proof for the Strongly Convex Case

The proofs of Theorem 3.9 and Theorem 3.10 have several identical steps. We start by proving the latter.

Proof of Theorem 3.10. Fix $s \in \mathbb{N}_0$ and let $k \in \{0, \ldots, m-1\}$ be given. Let again $(\hat{x}^{k+1}, \hat{\xi}^{k+1})$ denote the pair of exact solutions of (3.8) and (3.11). Due to

$$\hat{\xi}_i^{k+1} = \nabla f_{\kappa(i)}(A_{\kappa(i)}\hat{x}^{k+1}) + \gamma_{\kappa(i)} A_{\kappa(i)}\hat{x}^{k+1}, \quad i \in [b], \tag{3.40}$$

(cf. (3.9) replacing again (ξ^{k+1}, x^{k+1}) by $(\hat{\xi}^{k+1}, \hat{x}^{k+1})$) and (3.27), we have

$$\hat{x}^{k+1} = \operatorname{prox}_{\alpha\varphi}(x^k - \alpha[\nabla f_{S_k}(\hat{x}^{k+1}) + v^k] - \alpha M_{S_k}(\hat{x}^{k+1} - x^k)).$$

Furthermore, introducing $\psi_k(x) := \psi_{S_k}(x) + \langle v^k, x - x^k\rangle$, the underlying optimality condition of the proximity operator (3.3) implies

$$\begin{aligned} p = \hat{x}^{k+1} \quad &\iff \quad p \in x^k - \alpha M_{S_k}(p - x^k) - \alpha[\partial\varphi(p) + \nabla f_{S_k}(p) + v^k] \\ &\iff \quad p = \operatorname{prox}_{\alpha\psi_k}^{\mathbf{Id}+\alpha M_{S_k}}(x^k). \end{aligned}$$

Moreover, using (A1), the mapping

$$x \mapsto \hat{F}_k(x) := f_{S_k}(x) + \frac{1}{2}\|x - x^k\|^2_{M_{S_k}} = \frac{1}{b}\sum\nolimits_{i\in S_k} f_i(A_i x) + \frac{\gamma_i}{2}\|A_i(x - x^k)\|^2$$

is convex for all k. Hence, setting $\Psi_k(x) := \hat{F}_k(x) + \varphi(x) + \langle v^k, x - x^k\rangle$, the function $x \mapsto \Psi_k(x) + \frac{1}{2\alpha}\|x - x^k\|^2$ is $(\mu_\varphi + \alpha^{-1})$-strongly convex and due to $\hat{x}^{k+1} = \arg\min_x \Psi_k(x) + \frac{1}{2\alpha}\|x - x^k\|^2$, it follows

$$\begin{aligned}\Psi_k(x) + \tfrac{1}{2\alpha}\|x - x^k\|^2 \geq\ & \Psi_k(\hat{x}^{k+1}) + \tfrac{1}{2\alpha}\|\hat{x}^{k+1} - x^k\|^2 \\ & + \tfrac{1}{2}\left[\mu_\varphi + \tfrac{1}{\alpha}\right]\|x - \hat{x}^{k+1}\|^2\end{aligned} \tag{3.41}$$

for all $x \in \mathrm{dom}(\varphi)$. Next, combining the optimality condition (3.3), the update rule of Algorithm 7, and (3.40), we obtain

$$\begin{aligned}x^{k+1} &\in x^k - \alpha\mathcal{A}_{S_k}^\top \xi^{k+1} - \alpha v^k + \alpha M_{S_k} x_k - \alpha\partial\varphi(x^{k+1}) \\ &= x^k - \alpha\mathcal{A}_{S_k}^\top(\xi^{k+1} - \hat{\xi}^{k+1}) - \alpha[\nabla f_{S_k}(\hat{x}^{k+1}) - \nabla f_{S_k}(x^{k+1})] \\ &\quad - \alpha M_{S_k}(\hat{x}^{k+1} - x^{k+1}) - \alpha[\nabla f_{S_k}(x^{k+1}) + \partial\varphi(x^{k+1}) + v^k + M_{S_k}(x^{k+1} - x^k)].\end{aligned}$$

Setting $h^{k+1} := \mathcal{A}_{S_k}^\top(\xi^{k+1} - \hat{\xi}^{k+1}) + [\nabla f_{S_k}(\hat{x}^{k+1}) - \nabla f_{S_k}(x^{k+1})] + M_{S_k}(\hat{x}^{k+1} - x^{k+1})$, this shows that $-h^{k+1} + (x^k - x^{k+1})/\alpha \in \partial\Psi_k(x^{k+1})$. Thus, due to the strong convexity of Ψ_k and applying $-\langle a, b\rangle = \frac{1}{2}\|a - b\|^2 - \frac{1}{2}\|a\|^2 - \frac{1}{2}\|b\|^2$, it follows

$$\begin{aligned}&\Psi_k(x^{k+1}) - \Psi_k(y) \\ &\quad\leq -\frac{1}{\alpha}\langle x^k - x^{k+1}, y - x^{k+1}\rangle + \langle h^{k+1}, y - x^{k+1}\rangle - \frac{\mu_\varphi}{2}\|y - x^{k+1}\|^2 \\ &\quad= \frac{1}{2\alpha}[\|x^k - y\|^2 - \|x^{k+1} - x^k\|^2] - \frac{1}{2}\left[\mu_\varphi + \frac{1}{\alpha}\right]\|x^{k+1} - y\|^2 + \langle h^{k+1}, y - x^{k+1}\rangle\end{aligned}$$

for all $y \in \mathrm{dom}(\varphi)$. Using this estimate in (3.41) with $y = \hat{x}^{k+1}$ and applying Young's inequality, we have

$$\begin{aligned}&\frac{1}{2\alpha}[(1 + \alpha\mu_\varphi)\|\hat{x}^{k+1} - x\|^2 - \|x^k - x\|^2] \\ &\quad\leq \Psi_k(x) - \Psi_k(x^{k+1}) + \Psi_k(x^{k+1}) - \Psi_k(\hat{x}^{k+1}) - \frac{1}{2\alpha}\|\hat{x}^{k+1} - x^k\|^2 \\ &\quad\leq \Psi_k(x) - \Psi_k(x^{k+1}) - \frac{1}{2\alpha}\|x^{k+1} - x^k\|^2 - \frac{\mu_\varphi}{2}\|\hat{x}^{k+1} - x^{k+1}\|^2 + \frac{\alpha}{2}\|h^{k+1}\|^2.\end{aligned}$$

Next, we expand the first term on the right hand side as follows:

$$\begin{aligned}\Psi_k(x) - \Psi_k(x^{k+1}) &= [\psi(x) - \psi(x^{k+1})] + [f_{S_k}(x) - f(x)] + [\hat{F}_k(x^k) - \hat{F}_k(x^{k+1})] \\ &\quad + \langle v^k, x - x^{k+1}\rangle + [f(x^{k+1}) - f(x^k)] + [f(x^k) - f_{S_k}(x^k)] + \frac{1}{2}\|x - x^k\|^2_{M_{S_k}}.\end{aligned}$$

By the convexity of $\hat{F}_k$, we have $\hat{F}_k(x^k) - \hat{F}_k(x^{k+1}) \leq \langle\nabla\hat{F}_k(x^k), x^k - x^{k+1}\rangle = \langle\nabla f_{S_k}(x^k), x^k - x^{k+1}\rangle$. Combining this with the Lipschitz continuity of ∇f, it further holds that

$$\begin{aligned}&[\hat{F}_k(x^k) - \hat{F}_k(x^{k+1})] + [f(x^{k+1}) - f(x^k)] \\ &\qquad\leq \langle\nabla f(x^k) - \nabla f_{S_k}(x^k), x^{k+1} - x^k\rangle + \frac{L}{2}\|x^{k+1} - x^k\|^2.\end{aligned}$$

In addition, using Young's inequality, we have

$$\|\hat{x}^{k+1} - x\|^2 \geq (1-\rho_1)\|x^{k+1} - x\|^2 + \left[1 - \frac{1}{\rho_1}\right] \|\hat{x}^{k+1} - x^{k+1}\|^2$$

for $\rho_1 \in (0,1)$. Together, applying Young's inequality once more and setting $e_k(x) := f_{S_k}(x) - f(x) + f(x^k) - f_{S_k}(x^k) + \langle v^k, x - x^k\rangle - \frac{1}{2}\|x^k - x\|^2_{M_N - M_{S_k}}$, this yields

$$\begin{aligned}
&(1+\alpha\mu_\varphi)(1-\rho_1)\|x^{k+1} - x\|^2 - \|x^k - x\|^2 \\
&\quad \leq 2\alpha[\psi(x) - \psi(x^{k+1})] + 2\alpha e_k(x) + \alpha\|x - x^k\|^2_{M_N} \\
&\quad + \frac{\alpha}{\rho_2}\|\nabla f(x^k) - \nabla f_{S_k}(x^k) - v^k\|^2 - [1 - (L+\rho_2)\alpha]\,\|x^{k+1} - x^k\|^2 \\
&\quad + \alpha^2\|h^{k+1}\|^2 + (1+\alpha\mu_\varphi)(\rho_1^{-1} - 1)\|\hat{x}^{k+1} - x^{k+1}\|^2
\end{aligned} \tag{3.42}$$

for all $x \in \operatorname{dom}(\varphi)$ and $\rho_2 > 0$, where we used $-\alpha\mu_\varphi - (1+\alpha\mu_\varphi)(1-\rho_1^{-1}) \leq (1+\alpha\mu_\varphi)(\rho_1^{-1} - 1)$. The choice of ε_{sub} and Proposition 3.6 imply $\|\mathcal{A}_{S_k}^\top(\xi^{k+1} - \hat{\xi}^{k+1})\| \leq \frac{\bar{A}}{\mu_*\sqrt{b}}\varepsilon_k$ and $\|\hat{x}^{k+1} - x^{k+1}\| \leq \frac{\alpha\bar{A}}{\mu_*\sqrt{b}}\varepsilon_k$. Moreover, applying Lipschitz smoothness and Proposition 3.6, it holds that

$$\|h^{k+1}\| \leq \tfrac{\bar{A}}{\mu_*\sqrt{b}}\varepsilon_k + (\bar{L}_b + \|M_{S_k}\|)\|\hat{x}^{k+1} - x^{k+1}\| \leq (1 + \alpha[\bar{L} + \bar{M}])\tfrac{\bar{A}}{\mu_*\sqrt{b}}\varepsilon_k.$$

We now choose $x = x^\star$; this yields $\psi(x^\star) - \psi(x^{k+1}) \leq -\frac{\mu}{2}\|x^{k+1} - x^\star\|^2$. Furthermore, Corollary 3.14 (with $x = x^k, \tilde{x} = \tilde{x}^s$) yields $\mathbb{E}\|\nabla f(x^k) - \nabla f_{S_k}(x^k) - v^k\|^2 \leq \bar{L}^2 b^{-1}\|x^k - \tilde{x}^s\|^2$ and we have $\mathbb{E}[e_k(x^\star)] = 0$. In addition, by definition and due to the Lipschitz continuity of F_{nat}, we obtain $\varepsilon_k \leq \delta_s\|F_{\text{nat}}(\tilde{x}^s)\| \leq (2+L)\delta_s\|\tilde{x}^s - x^\star\|$. Using $M_N \preceq (\mu_\varphi - \mu)I$, combining our previous results, and taking expectation, it follows

$$\begin{aligned}
&[1 + \alpha(\mu_\varphi + \mu) - \rho_1(1+\alpha\mu_\varphi)]\mathbb{E}\|x^{k+1} - x^\star\|^2 \\
&\leq [1 + \alpha(\mu_\varphi - \mu)]\mathbb{E}\|x^k - x^\star\|^2 + \frac{\bar{L}^2\alpha}{b\rho_2}\mathbb{E}\|x^k - \tilde{x}^s\|^2 - [1 - (L+\rho_2)\alpha]\mathbb{E}\|x^{k+1} - x^k\|^2 \\
&\quad + \underbrace{\left[(1+\alpha\mu_\varphi)\rho_1^{-1} + (1 + \alpha[\bar{L} + \bar{M}])^2\right]\frac{\bar{A}^2}{\mu_*^2 b}(2+L)^2\alpha^2}_{=:c(\alpha)}\ \delta_s^2\mathbb{E}\|\tilde{x}^s - x^\star\|^2.
\end{aligned}$$

We now suppose that ρ_1 is chosen such that $(1+\alpha\mu_\varphi)\rho_1 \leq 2\alpha\mu$. Then, summing the last estimate for $k = 0, \ldots, m-1$ and invoking (3.35), this implies

$$\begin{aligned}
&[1 + \alpha(\mu_\varphi + \mu) - \rho_1(1+\alpha\mu_\varphi)]\mathbb{E}\|\tilde{x}^{s+1} - x^\star\|^2 \\
&\quad \leq [1 + \alpha(\mu_\varphi - \mu) + c(\alpha)m\delta_s^2]\mathbb{E}\|\tilde{x}^s - x^\star\|^2 \\
&\qquad - \left[1 - \left(L + \rho_2 + \tfrac{\bar{L}^2 m(m-1)}{2b\rho_2}\right)\alpha\right]\sum_{k=0}^{m-1}\mathbb{E}\|x^{k+1} - x^k\|^2.
\end{aligned}$$

Choosing $\rho_2 = \bar{L}\sqrt{m(m-1)}/\sqrt{2b}$, $\alpha \leq (L + \sqrt{2}\bar{L}m/\sqrt{b})^{-1}$, and $\rho_1 = \delta_s$, we obtain

$$\begin{aligned}
\mathbb{E}\|\tilde{x}^{s+1} - x^\star\|^2 &\leq \left[1 - \tfrac{2\alpha\mu}{1+\alpha(\mu_\varphi+\mu)-\rho_1(1+\alpha\mu_\varphi)} + \tfrac{\rho_1(1+\alpha\mu_\varphi)+c(\alpha)m\delta_s^2}{1+\alpha(\mu_\varphi+\mu)-\rho_1(1+\alpha\mu_\varphi)}\right]\mathbb{E}\|\tilde{x}^s - x^\star\|^2 \\
&\leq \left[1 - \tfrac{2\alpha\mu}{1+\alpha(\mu_\varphi+\mu)} + \mathcal{O}(\delta_s)\right]\mathbb{E}\|\tilde{x}^s - x^\star\|^2
\end{aligned}$$

as $s \to \infty$. This proves q-linear convergence of $\{\tilde{x}^s\}$ to $x^\star$ in expectation. □

Proof of Theorem 3.9. Let the iteration index $s \in \mathbb{N}_0$ and $k \in \{0, \dots, m-1\}$ be fixed and let $h^{k+1}, \hat{x}^{k+1}, e_k(x)$ be defined as in the proof of Theorem 3.10. As all f_i are convex we have $M_{S_k} = M_N = 0$. Denote by $\mu_\varphi \geq 0$ the strong convexity parameter of φ. Let $x^\star$ denote the unique solution to (3.2) satisfying $\psi^\star = \psi(x^\star)$. Proceeding as in the proof of Theorem 3.10, we obtain the following analogue to (3.42)

$$\begin{aligned}(1+\alpha\mu_\varphi)(1-\rho_1)\|x^{k+1}-x\|^2 - \|x^k - x\|^2 &\leq 2\alpha[\psi(x) - \psi(x^{k+1})] + 2\alpha e_k(x) \\ &+ \frac{\alpha}{\rho_2}\|\nabla f(x^k) - \nabla f_{S_k}(x^k) - v^k\|^2 - [1-(L+\rho_2)\alpha]\,\|x^{k+1}-x^k\|^2 \\ &+ \alpha^2\|h^{k+1}\|^2 + (1+\alpha\mu_\varphi)(\rho_1^{-1}-1)\|\hat{x}^{k+1}-x^{k+1}\|^2\end{aligned}$$

for $x \in \mathrm{dom}(\varphi)$, $\rho_1 \in (0,1)$, and $\rho_2 > 0$. Conditioned on x^k, we have $\mathbb{E}[e_k(x)] = 0$. By Proposition 3.6, it holds $\|h^{k+1}\|^2 \leq (1+\alpha\bar{L})^2 \frac{\bar{A}^2}{\mu_*^2 b}\varepsilon_k^2$ and $\|\hat{x}^{k+1} - x^{k+1}\|^2 \leq \frac{\alpha^2\bar{A}^2}{\mu_*^2 b}\varepsilon_k^2$ and we can again use the estimate $\varepsilon_k \leq (2+L)\delta_s\|\tilde{x}^s - x^\star\|$. Applying Corollary 3.14 (with $x = x^k, \tilde{x} = \tilde{x}^s$), conditioned on the history of iterates up to $x^k = x^{s,k}$, we have

$$\mathbb{E}\|\nabla f(x^k) - \nabla f_{S_k}(x^k) - v^k\|^2 \leq 4\bar{L}b^{-1}(\psi(x^k) - \psi^\star + \psi(\tilde{x}^s) - \psi^\star)$$

almost surely. At this point, we assume that the condition $1-(L+\rho_2)\alpha \geq 0$ holds (for the selected ρ_2 and α). Next, setting $\beta(\rho_1) := (1+\alpha\mu_\varphi)(1-\rho_1)$ and $x = x^\star$, we apply expectation conditioned on the history up to iterate $x^k = x^{s,k}$ and conclude

$$\begin{aligned}\beta(\rho_1)\mathbb{E}\|x^{k+1}-x^\star\|^2 &\leq \|x^k - x^\star\|^2 + 2\alpha\mathbb{E}[\psi^\star - \psi(x^{k+1})] \\ &\quad + 4\alpha\bar{L}(\rho_2 b)^{-1}(\psi(x^k) - \psi^\star + \psi(\tilde{x}^s) - \psi^\star) + \tilde{c}(\alpha)\delta_s^2\|\tilde{x}^s - x^\star\|^2,\end{aligned} \tag{3.43}$$

where $\tilde{c}(\alpha) := ((1+\alpha\bar{L})^2 + (1+\alpha\mu_\varphi)\rho_1^{-1})\frac{\alpha^2\bar{A}^2}{\mu_*^2 b}(2+L)^2$. Using $\|x^k - x^\star\|^2 \leq \frac{2}{\mu}(\psi(x^k) - \psi^\star)$ and $1 - \beta(\rho_1) = \rho_1 + \alpha\mu_\varphi(\rho_1 - 1) \leq \rho_1$, this yields

$$\begin{aligned}\beta(\rho_1)\mathbb{E}\|x^{k+1}-x^\star\|^2 &\leq \beta(\rho_1)\|x^k - x^\star\|^2 + 2\alpha\mathbb{E}[\psi^\star - \psi(x^{k+1})] \\ &+ \tilde{c}(\alpha)\delta_s^2\|\tilde{x}^s - x^\star\|^2 + \tfrac{4\alpha\bar{L}}{\rho_2 b}(\psi(x^k) - \psi^\star + \psi(\tilde{x}^s) - \psi^\star) + \tfrac{2\rho_1}{\mu}(\psi(x^k) - \psi^\star).\end{aligned} \tag{3.44}$$

We now require $\rho_1 \leq \min\{\frac{1}{2}, \frac{\alpha\mu\bar{L}}{\rho_2 b}\}$. Using (3.44) recursively for $k = 1, \dots, m-1$ and (3.43) for $k = 0$, applying expectation conditioned on the history up to $\tilde{x}^s$ and the tower property, we obtain

$$\begin{aligned}\beta(\rho_1)\mathbb{E}\|x^m - x^\star\|^2 + 2\alpha(1 - \tfrac{3\bar{L}}{\rho_2 b})\textstyle\sum_{k=1}^m \mathbb{E}[\psi(x^k) - \psi^\star] \\ \leq (1 + \tilde{c}(\alpha)m\delta_s^2)\|\tilde{x}^s - x^\star\|^2 + \tfrac{4\alpha\bar{L}(m+1)}{\rho_2 b}(\psi(\tilde{x}^s) - \psi^\star).\end{aligned}$$

Due to the convexity of ψ and by Option II, we can infer $\psi(\tilde{x}^{s+1}) \leq \frac{1}{m}\sum_{k=1}^m \psi(x^k)$ and hence, for $\rho_2 > 3\bar{L}b^{-1}$ it holds that

$$2\alpha(1 - \tfrac{3\bar{L}}{\rho_2 b})m\mathbb{E}[\psi(\tilde{x}^{s+1}) - \psi^\star] \leq (1 + \tilde{c}(\alpha)m\delta_s^2)\|\tilde{x}^s - x^\star\|^2 + \tfrac{4\alpha\bar{L}(m+1)}{\rho_2 b}(\psi(\tilde{x}^s) - \psi^\star).$$

Furthermore, the strong convexity of ψ again implies $\|\tilde{x}^s - x^\star\|^2 \leq \frac{2}{\mu}(\psi(\tilde{x}^s) - \psi^\star)$. We now set $\rho_2 = \frac{\bar{L}}{b}(\frac{4}{1-2\theta} + 3)$, i.e., $\frac{4\bar{L}}{\rho_2 b}(1 - \frac{3\bar{L}}{\rho_2 b})^{-1} = 1 - 2\theta$. This choice satisfies $\rho_2 > 3\bar{L}b^{-1}$ automatically and due to (3.20), we have $(L+\rho_2)\alpha \leq 1$. This yields

$$\mathbb{E}[\psi(\tilde{x}^{s+1}) - \psi^\star] \leq \left[\frac{1 + \tilde{c}(\alpha)m\delta_s^2}{\mu\alpha(1 - \frac{3\bar{L}}{\rho_2 b})m} + \frac{2\bar{L}(m+1)}{\rho_2 b(1 - \frac{3\bar{L}}{\rho_2 b})m}\right](\psi(\tilde{x}^s) - \psi^\star).$$

By the choice of ρ_2, it holds $\frac{2\bar{L}(m+1)}{\rho_2 bm}(1-\frac{3\bar{L}}{\rho_2 b})^{-1} \leq 1-2\theta$ for all $m \in \mathbb{N}$. Finally, if δ_s is sufficiently small and m is sufficiently large such that $\frac{1+\tilde{c}(\alpha)m\delta_s^2}{\mu\alpha m}(1-\frac{3\bar{L}}{\rho_2 b})^{-1} \leq \theta$, then we can conclude $\mathbb{E}[\psi(\tilde{x}^{s+1})-\psi^\star] \leq (1-\theta)\mathbb{E}[\psi(\tilde{x}^s)-\psi^\star]$. □

Remark 3. *In the case $\mu_\varphi > 0$, ρ_1 can be chosen small enough such that $\beta(\rho_1) \geq 1$ and we obtain the term $1-\frac{2\bar{L}}{\rho_2 b}$ instead of $1-\frac{3\bar{L}}{\rho_2 b}$ in the latter computations.*

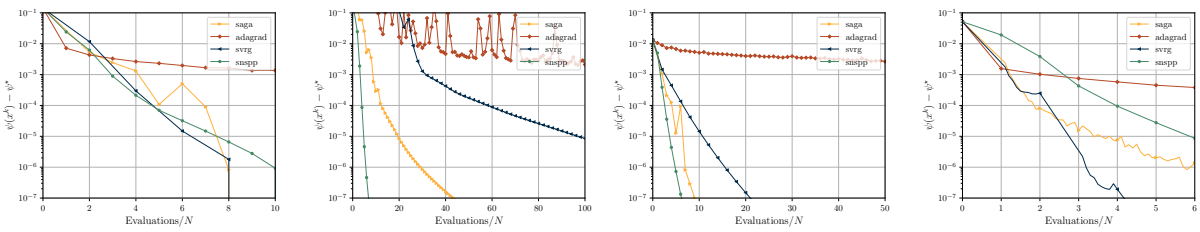
3.8.4 Parameter Choices

Here, we report details on the tuning procedure for Section 3.7.2 and Section 3.7.3. For each method and each dataset, we first identify a candidate interval of step sizes. We then choose a range of step sizes α on this interval (typically 5–7 values) and perform grid search over 2–3 different batch sizes b. For `SAGA`, we additionally try $b = 1$. We select the combination of α and b that performed best in terms of the objective function sub-optimality over three runs. We observe that `AdaGrad` typically requires larger batch sizes than `SNSPP`/`SVRG`, which might be due to a missing variance reduction mechanism in `AdaGrad`.

Dataset	`SNSPP`		`SAGA`		`SVRG`		`AdaGrad`	
	α	b	α	b	α	b	α	b
`mnist`	2.5	280	0.04	56	0.25	280	0.030	2800
`gisette`	7.0	240	$1.20{\cdot}10^{-3}$	1	0.022	50	0.028	240
`sido0`	30.0	50	0.20	10	0.733	50	0.0150	200
`covtype`	50.0	50	0.25	50	0.350	50	0.10	250
`student-t` ($\hat{\nu}=0.5$)	1.05	20	$2.50{\cdot}10^{-3}$	1	0.140	40	0.032	40
`student-t` ($\hat{\nu}=1$)	3.0	20	0.015	4	0.120	20	0.030	20
`student-t` ($\hat{\nu}=2$)	7.0	20	0.20	20	0.40	20	0.032	40
`student-t sido0`	5.5	200	0.004	1	0.0145	10	0.015	100

Table 3.2: Step size and batch size values for the experiments in Section 3.7.2 and Section 3.7.3.

3.8.5 Additional Plots



(a) `mnist` (b) `gisette` (c) `sido0` (d) `covtype`

Figure 3.6: Objective function convergence for the logistic regression datasets with respect to number of gradient evaluations. All settings are identical to Fig. 3.3.

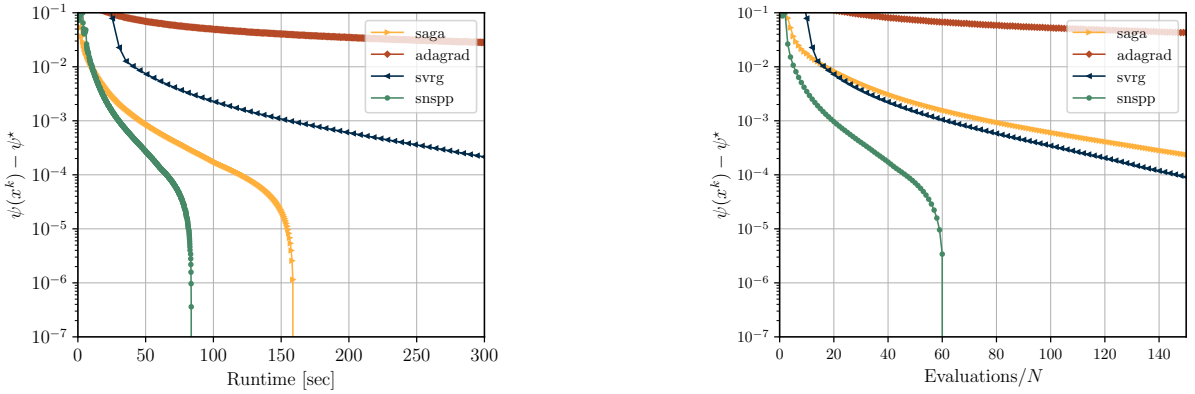


Figure 3.7: Convergence plot for logistic regression on the `madelon.2` dataset, with respect to runtime (left) and number of gradient evaluations (right).

3.9 Extension: Additional Loss and Regularization Functions

This section can be seen as a collection of calculations and references for i) the Fenchel conjugate of classical loss functions and ii) the proximal operator and its subdifferential for standard regularization terms.

3.9.1 Loss Functions and their Conjugate

In this section, we consider classical loss functions from statistical learning and compute their Fenchel conjugate. We denote the loss by $\ell : \mathbb{R}^m \to \mathbb{R},\ z \mapsto \ell(z)$. It is important to note that the argument z is the *model output* and not the vector of learnable parameters of the model. We denote the targets/labels by y which are typically given as (integer) labels for classification tasks or as real numbers for regression tasks. The general strategy will be to compute solutions of the problem

$$\ell^*(z) = \sup_{u \in \mathbb{R}^m} \langle u, z \rangle - \ell(u), \quad z \in \mathbb{R}^m. \tag{3.45}$$

If ℓ is convex, then the above is a concave maximization problem and solutions can be characterized by the necessary and sufficient first-order optimality condition. We consider loss functions for several tasks, namely regression and classification with two or more than two classes. For motivating considerations on the presented loss functions we refer to standard textbooks of statistical learning, e.g. [57].

For each loss function, we further check whether ℓ^* is strongly convex on its domain and essentially differentiable as these properties are needed for convergence of the semismooth Newton method presented in Algorithm 8.

Squared Loss. Given $y \in \mathbb{R}^m$, consider

$$\ell : \mathbb{R}^m \to \mathbb{R},\ \ell(z) := \|z - y\|^2.$$

Clearly, ℓ is strongly convex and it holds

$$\bar{u} = \operatorname*{arg\,max}_{u \in \mathbb{R}^m} \langle z, u \rangle - \ell(u) \iff z = \nabla \ell(\bar{u}) = 2(\bar{u} - y) \iff \bar{u} = \frac{z}{2} + y.$$

Plugging in $\bar{u}$ into the objective of (3.45), we get

$$\ell^*(z) = \langle z, \frac{z}{2} + y \rangle - \|\frac{z}{2} + y - y\|^2 = \tfrac{1}{4}\|z\|^2 + \langle y, z \rangle.$$

Moreover, it holds

$$\nabla \ell^*(z) = \frac{z}{2} + y, \quad \nabla^2 \ell^*(z) = \tfrac{1}{2}\mathbf{Id}_m.$$

As $\nabla^2 \ell^*(z) \succeq \frac{1}{2}$ for all z, ℓ^* is strongly convex. Further, $\mathrm{dom}(\ell^*) = \mathbb{R}^m$, hence ℓ^* is essentially differentiable.

Huber Loss. For $\mu > 0$ and $y \in \mathbb{R}$, the Huber loss is given by

$$\ell : \mathbb{R} \to \mathbb{R}, \quad \ell(z) := \begin{cases} \frac{(z-y)^2}{2\mu} & |z - y| \leq \mu, \\ |z - y| - \frac{\mu}{2} & \text{else.} \end{cases}$$

The derivative of the Huber loss is

$$\ell'(z) = \begin{cases} (z - y)/\mu & |z - y| \leq \mu, \\ \mathrm{sgn}(z - y) & \text{else.} \end{cases}$$

Again, consider

$$\bar{u} = \operatorname*{arg\,max}_{u \in \mathbb{R}} z \cdot u - \ell(u) \iff \begin{cases} \bar{u} = \mu z + y & |\bar{u} - y| \leq \mu, \\ \mathrm{sgn}(\bar{u} - y) = z & \text{else.} \end{cases}$$

From the first case, we get that $\bar{u} = \mu z + y$ is a solution to (3.45) if $|\mu z| \leq \mu \iff |z| \leq 1$. The second case implies that $z = \pm 1$ which we already covered in the first case, so we can disregard it. Plugging in $\bar{u}$ from the first case into the objective of (3.45), we obtain

$$\ell^*(z) = \begin{cases} z(\mu z + y) - \frac{\mu^2 z^2}{2\mu} = \frac{1}{2}\mu z^2 + yz & |z| \leq 1, \\ +\infty & \text{else.} \end{cases}$$

For $|z| < 1$, the derivatives are given by

$$(\ell^*)'(z) = \mu z + y, \quad (\ell^*)''(z) = \mu.$$

For $|z| < 1$, $|z| \to 1$, we have $|(\ell^*)'(z)| \not\to \infty$, hence ℓ^* is not essentially differentiable. It is strongly convex on $(-1, 1)$ as $\mu > 0$.

Pseudo-Huber Loss. For $\mu > 0$ and $y \in \mathbb{R}$, the pseudo-Huber loss [44] is given by

$$\ell : \mathbb{R} \to \mathbb{R}, \; z \mapsto \sqrt{\mu^2 + (z - y)^2} - \mu.$$

In contrast to the Huber loss, the above is arbitrarily often differentiable and $\ell'(z) = \frac{z-y}{\sqrt{\mu^2+(z-y)^2}}$. For details we refer to [44]. For given $z \in \mathbb{R}$, it holds

$$\bar{u} = \operatorname*{arg\,max}_{u \in \mathbb{R}} z \cdot u - \ell(u) \iff z - \ell'(\bar{u}) = 0 \iff z = \frac{\bar{u} - y}{\sqrt{\mu^2 + (\bar{u} - y)^2}}. \tag{3.46}$$

Squaring both sides in the last equality of (3.46) leads to the necessary condition

$$z^2(\mu^2 + (\bar{u} - y)^2) = (\bar{u} - y)^2.$$

Simplifying the terms, we obtain a quadratic equation in $\bar{u}$ with solutions $y \pm \frac{\mu|z|}{\sqrt{1-z^2}}$. From (3.46) it follows that z and $\bar{u} - y$ have the same sign. Therefore, (3.45) has a solution if and only if $z \in (-1, 1)$ and in this case, the solution is $\bar{u} = y + \frac{\mu z}{\sqrt{1-z^2}}$. Plugging in this $\bar{u}$ into the objective of (3.45) yields

$$\ell^*(z) = \begin{cases} yz + \frac{\mu z^2 - \mu}{\sqrt{1-z^2}} + \mu = yz + \mu - \mu\sqrt{1-z^2} & |z| < 1, \\ +\infty & \text{else.} \end{cases}$$

For $|z| < 1$, the derivatives are given by

$$(\ell^*)'(z) = y + \frac{\mu z}{\sqrt{1-z^2}}, \quad (\ell^*)''(z) = \frac{\mu}{(1-z^2)^{3/2}}.$$

For the psuedo-Huber loss, ℓ^* is essentially differentiable as well as strongly convex on $(-1, 1)$.

Logistic Loss. For two-class classification, where the label is given by $y \in \{-1, 1\}$, the logistic loss[5] is defined as

$$\ell : \mathbb{R} \to \mathbb{R}, \quad \ell(z) := \ln(1 + \exp(-yz)).$$

The derivative is $\ell'(z) = \frac{-y\exp(-yz)}{1+\exp(-yz)} = \frac{-y}{1+\exp(yz)}$ and we have

$$\bar{u} = \operatorname*{arg\,max}_{u \in \mathbb{R}} z \cdot u - \ell(u) \iff z\big(1 + \exp(y\bar{u})\big) = -y.$$

Using $y = 1/y$, if $z \neq 0$ we compute $\bar{u} = y\ln(\frac{-y-z}{z})$. Further, if $z = 0$ or $\frac{-y-z}{z} \leq 0 \iff -\frac{y}{z} \leq 1$ then (3.45) has no solution. Plugging in $\bar{u}$ into the objective of (3.45), we get

$$\ell^*(z) = \begin{cases} yz\ln(\frac{-y-z}{z}) + \ln(\frac{y+z}{y}) & -\frac{y}{z} > 1 \wedge z \neq 0, \\ +\infty & \text{else.} \end{cases}$$

Note that the domain of ℓ^* is equal to $(-1, 0)$ if $y = 1$ and $(0, 1)$ if $y = -1$. Computing the derivative (or using Danskin's theorem), we conclude that, for $-\frac{y}{z} > 1$, we have

$$(\ell^*)'(z) = y\ln\big(\frac{-y-z}{z}\big), \quad (\ell^*)''(z) = -\frac{1}{z(y+z)}.$$

One can easily verify that $(\ell^*)''(z) \geq 4$ for all z such that $-\frac{y}{z} > 1$ and hence ℓ^* is strongly convex on its domain. Further, it is easy to see that $|(\ell^*)'(z)| \to \infty$ if z approaches the boundary of $\mathrm{dom}(\ell^*)$; hence ℓ^* is essentially differentiable.

Squared Hinge Loss. For $y \in \{-1, 1\}$, consider

$$\ell : \mathbb{R} \to \mathbb{R}, \quad \ell(z) := \max\{0, 1 - yz\}^2.$$

[5] In `Pytorch`, this loss function is implemented in `torch.nn.SoftMarginLoss`.

It is easy to verify that ℓ is convex and continuously differentiable. It holds $\ell'(z) = 2y(yz - 1)\mathbb{1}_{yz\leq 1}$. As $y^2 = 1$, we have

$$\bar{u} = \operatorname*{arg\,max}_{u\in\mathbb{R}} z \cdot u - \ell(u) \iff z - \ell'(\bar{u}) = 0 \iff 0 = \begin{cases} 2(y - \bar{u}) + z & y\bar{u} \leq 1, \\ z & y\bar{u} > 1. \end{cases}$$

From the first case, $\bar{u} = \frac{z}{2} + y$ is a solution to (3.45) if $y\bar{u} = 1 + \frac{yz}{2} \leq 1 \iff yz \leq 0$. The second case does not yield new solutions as the first case includes $z = 0$. In the first case, we have $1 - y\bar{u} = -\frac{yz}{2} \geq 0$. Plugging in this $\bar{u}$ into the objective of (3.45) yields

$$\ell^*(z) = \begin{cases} \frac{z^2}{4} + yz & yz \leq 0, \\ +\infty & \text{else.} \end{cases}$$

Hence the domain of ℓ^* is given by $[0, \infty)$ if $y = -1$ and $(-\infty, 0]$ if $y = 1$. For $z \in \operatorname{int}(\operatorname{dom}(\ell^*))$, the derivatives of the Fenchel conjugate are given by

$$(\ell^*)'(z) = y + z/2, \quad (\ell^*)''(z) = \frac{1}{2}.$$

The soft-margin loss conjugate is strongly convex on its domain, but not essentially differentiable (e.g. let $z_j \to 0$, then $|(\ell^*)'(z_j)| \to 1$).

Cross Entropy. For Multi-class classification, the task is to predict the correct out of K classes, i.e. the label is given as $y \in \{1, \dots, K\}$. Given a vector of *scores* $z \in \mathbb{R}^K$, a standard approach is to model the probability that the predicted class is k with $\frac{\exp(z_k)}{\sum_{j=1}^K \exp(z_j)}$. Computing the negative log-likelihood, we obtain the loss function

$$\ell : \mathbb{R}^K \to \mathbb{R}, \quad \ell(z) := -\ln \frac{\exp(z_y)}{\sum_{j=1}^K \exp(z_j)} = -z_y + \ln \sum_{j=1}^K \exp(z_j). \tag{3.47}$$

The above is called cross entropy loss.[6] We remark that it is invariant under addition of a constant, i.e. $\ell(z + c\mathbf{1}_K) = \ell(z)$ for any $c \in \mathbb{R}$. Further, ℓ is related to the conjugate function of the negative entropy on the unit simplex and hence convex [9, Thm. 4.3, Sec. 4.4.11].

Computing $\nabla\ell(z) = \left(\sum_{j=1}^K \exp(z_j)\right)^{-1} \big(\exp(z_1), \dots, \exp(z_K)\big)^\top - \mathbf{e}_y$, we have

$$\bar{u} = \operatorname*{arg\,max}_{u\in\mathbb{R}^K} \langle u, z\rangle - \ell(u) \iff \frac{\exp(\bar{u}_l)}{\sum_{j=1}^K \exp(\bar{u}_j)} = \begin{cases} z_l + 1 & l = y, \\ z_l & \text{else.} \end{cases} \tag{3.48}$$

Summing on both sides, this implies $1 + \sum_{l=1}^K z_l = 1 \iff \sum_{l=1}^K z_l = 0$. Further, $\frac{\exp(\bar{u}_l)}{\sum_{j=1}^K \exp(\bar{u}_j)} > 0$ implies $z_l > -1$ if $l = y$ and $z_l > 0$ else. Denote

$$M(y) := \left\{ z \in \mathbb{R}^K \,\middle|\, z_l \begin{cases} > -1 & \text{if } l = y \\ > 0 & \text{else} \end{cases}, \sum_{j=1}^K z_j = 0 \right\}.$$

Now, assume $z \in M(y)$. Then, (3.48) is satisfied for $\bar{u}_l = \ln(z_l + 1)$ if $y = l$ and $\bar{u}_l = \ln(z_l)$ else. This shows that the domain of ℓ^* is equal to $M(y)$ which is not an open set. Hence, it is not

[6] See for example `torch.nn.CrossEntropyLoss` in `Pytorch`.

well suited for the semismooth Newton approach. We will develop an alternative in the next paragraph.

For the sake of completeness, we compute also ℓ^*: plugging in the derived $\bar{u}$ into (3.45), we obtain

$$\ell^*(z) = \begin{cases} (1+z_y)\ln(1+z_y) + \sum_{l\neq y} z_l \ln(z_l), & z \in M(y), \\ +\infty & \text{else.} \end{cases}$$

Reduced Cross Entropy. Again, consider K classes and $y \in \{1,\dots,K\}$. We have seen that the cross entropy loss is invariant to adding a constant. Hence, we can subtract from every component of the score vector z the last entry of z without loss of generality. This results in the score vector having a zero in the last component and effectively reducing the dimension by one. Thus, defining $z_y := 0$ if $y = K$, consider

$$\ell : \mathbb{R}^{K-1} \to \mathbb{R}, \quad \ell(z) := -z_y + \ln\left(1 + \sum_{j=1}^{K-1} \exp(z_j)\right).$$

This loss is used for example in multiclass logistic regression [58, Sec. 3.3] and we will call it *reduced cross entropy*. It holds

$$w := \begin{cases} \mathbf{e}_y & \text{if } y \in [K-1], \\ 0 & \text{else,} \end{cases} \qquad \nabla\ell(z) = -w + \frac{(\exp(z_1),\dots,\exp(z_{K-1}))^\top}{1+\sum_{j=1}^{K-1}\exp(z_j)}.$$

As the reduced cross entropy ℓ is equal to the cross entropy on the set of K-dimensional vectors where the last entry is zero, it is convex. Hence, for $y \in [K]$ and $z \in \mathbb{R}^{K-1}$ given, it holds

$$\begin{aligned} &\bar{u} = \operatorname*{arg\,max}_{u\in\mathbb{R}^{K-1}} \langle z, u\rangle - \ell(u) \\ \iff & z + w - \frac{(\exp(\bar{u}_1),\dots,\exp(\bar{u}_{K-1}))^\top}{1+\sum_{j=1}^{K-1}\exp(\bar{u}_j)} = 0 \\ \iff & \frac{1}{1+\sum_{j=1}^{K-1}\exp(\bar{u}_j)}\exp(\bar{u}_l) = \begin{cases} z_l + 1 & y = l,\ y \in [K-1], \\ z_l & \text{else.} \end{cases} \end{aligned} \tag{3.49}$$

A necessary condition for the above system to have a solution $\bar{u}$ is that $z_l > -1$ if $y = l,\ y \in [K-1]$ and $z_l > 0$ else. Moreover, as the sum of the left-hand side is smaller than one, this implies $\sum_{j=1}^{K-1} z_j < 0$ if $y \in [K-1]$ and $\sum_{j=1}^{K-1} z_j < 1$ if $y = K$. We summarize these conditions with

$$M(y) := \left\{ z \in \mathbb{R}^{K-1} \middle| z_l \begin{cases} > -1 & \text{if } y = l,\ y \in [K-1] \\ > 0 & \text{else} \end{cases}, \sum_{j=1}^{K-1} z_j \begin{cases} < 0 & \text{if } y \in [K-1] \\ < 1 & \text{if } y = K \end{cases} \right\}.$$

Assume for the following that $z \in M(y)$. Summing over l on both sides of (3.49), we obtain

$$\frac{\sum_{l=1}^{K-1}\exp(\bar{u}_l)}{1+\sum_{l=1}^{K-1}\exp(\bar{u}_l)} = \mathbb{1}_{y\in[K-1]} + \sum_{l=1}^{K-1} z_l.$$

Using $\frac{x}{1+x} = c \iff x = \frac{c}{1-c}$ and denoting $\bar{z} := \sum_{l=1}^{K-1} z_l$, we conclude

$$\sum_{l=1}^{K-1}\exp(\bar{u}_l) = \frac{\mathbb{1}_{y\in[K-1]} + \bar{z}}{\mathbb{1}_{y=K} - \bar{z}} =: c_{y,z}. \tag{3.50}$$

Plugging in (3.50) into (3.49), we obtain

$$\begin{aligned}\exp(\bar{u}_l) = (1+c_{y,z})\begin{cases} z_l+1 & y=l,\ y\in[K-1],\\ z_l & \text{else.}\end{cases}\\ \iff \bar{u}_l = \begin{cases}\ln\big((1+c_{y,z})(z_l+1)\big) & y=l,\ y\in[K-1],\\ \ln\big((1+c_{y,z})(z_l)\big) & \text{else.}\end{cases}\end{aligned} \tag{3.51}$$

Our derivation shows that this is the unique solution to (3.49) and hence Danskin's theorem implies that $\nabla(\ell^*)(z) = \bar{u}$ if $z \in M(y)$. Moreover, the domain of ℓ^* is given by $M(y)$. We will now compute ℓ^* and $\nabla^2(\ell^*)$ with case distinction:

$\mathbf{y} \in [\mathbf{K-1}]$: Plugging in $\bar{u}$ from (3.51) into the objective of (3.45), we obtain

$$\ell^*(z) = \sum_{l\neq y} z_l \ln(-\frac{z_l}{\bar{z}}) + (1+z_y)\ln(-\frac{1+z_y}{\bar{z}}) + \ln(-\bar{z}),$$

where we used that $1 + c_{y,z} = -\frac{1}{\bar{z}}$ in this case. Again from (3.51), using that $\ln(a/b) = \ln(a) - \ln(b)$, we compute the Hessian

$$\nabla^2\ell^*(z) = \operatorname{diag}(\{v_l\}_{l=1,\dots,K-1}) - \frac{1}{\bar{z}}\mathbf{1}_{K-1}\mathbf{1}_{K-1}^\top, \quad v_l = \begin{cases}1/(z_l+1) & y=l,\\ 1/z_l & \text{else.}\end{cases}$$

As $z \in M(y)$, we have $\nabla^2\ell^*(z) \succeq 1$. Note that if $\bar{z} \to 0^-$, then $c_{y,z} \to \infty$. As $\bar{z} \to 0^-$ and all $z_l \to 0$, $z_y \to -1$ can not happen simultaneously, we conclude $\|\nabla(\ell^*)(z)\| \to \infty$, if z approaches the boundary of $M(y)$.

$\mathbf{y} = \mathbf{K}$: Plugging in $\bar{u}$ from (3.51) into the objective of (3.45), we obtain

$$\ell^*(z) = \sum_{l=1}^{K-1} z_l \ln(\frac{z_l}{1-\bar{z}}) + \ln(1-\bar{z}),$$

where we used that $1 + c_{y,z} = \frac{1}{1-\bar{z}}$ in this case. Using $\ln(a/b) = \ln(a) - \ln(b)$, we compute the Hessian

$$\nabla^2\ell^*(z) = \operatorname{diag}((z_l^{-1})_{l=1,\dots,K-1}) + \frac{1}{1-\bar{z}}\mathbf{1}_{K-1}\mathbf{1}_{K-1}^\top.$$

As $z \in M(y)$, we have $\nabla^2\ell^*(z) \succeq 1$. Again, if $\bar{z} \to 1$, then $c_{y,z} \to \infty$. As $\bar{z} \to 1$ and all $z_l \to 0$ can not happen simultaneously, we conclude $\|\nabla(\ell^*)(z)\| \to \infty$, if z approaches the boundary of $M(y)$.

In conclusion, ℓ^* is strongly convex on its domain, and essentially differentiable.

3.9.2 Regularization Functions

In this section we consider classical regularization functions $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ used in statistical learning. Typically, these are used to avoid overfitting or to bias the solution towards a prior. One of the most prominent examples is the ℓ_1-norm which is used in order to obtain sparse solutions. It has been used for the Lasso [150], i.e. sparse regression, as well as in compressed sensing [31].

Again, we do not go into details here and refer instead to [57,58] for more background. We also refer to [9, Sec. 6.9] for additional prox-computations.

The purpose of this section is to summarize the proximal operator and its Clarke subdifferential for several classical regularization functions. In fact, instead of stating the complete set $\partial \mathrm{prox}_{\alpha\varphi}(x)$, for our purposes it is sufficient to compute one element of the subdifferential. `Python` implementations of most of the operators below can be found in [135].

Squared ℓ_2-norm. Let $\varphi(x) = \frac{\lambda}{2}\|x\|^2$ for some $\lambda > 0$. The squared Euclidean norm has been used in ridge regression [57] and is still among the most commonly used regularization techniques in modern deep learning (where it is often referred to as *weight decay* [77]).

For $\alpha > 0$, it is easy to compute

$$\mathrm{prox}_{\alpha\varphi}(x) = \frac{1}{1+\alpha\lambda}x, \quad \mathrm{env}^{\alpha}_{\varphi}(x) = \frac{\lambda}{2(1+\alpha\lambda)}\|x\|^2 \quad \partial\mathrm{prox}_{\alpha\varphi}(x) = \{\frac{1}{1+\alpha\lambda}\mathbf{Id}\}.$$

ℓ_1- and ℓ_2-norm. Let $\varphi(x) = \lambda\|x\|_1$ for some $\lambda > 0$. As this is multiplicative in λ we only compute prox_{φ} instead of $\mathrm{prox}_{\alpha\varphi}$. We have (cf. [9, Sec. 6.9])

$$\mathrm{prox}_{\varphi}(x) = \mathrm{sign}(x) \odot \max\{|x| - \lambda\mathbf{1}_n, 0\}, \quad \mathrm{Diag}(d_i)_{i=1,\dots,n} \in \partial\mathrm{prox}_{\varphi}(x),$$

where $d_i = 1$ if $|x_i| > \lambda$ and $d_i = 0$ else[7].

Let $\varphi(x) = \lambda\|x\|$ for some $\lambda > 0$. According to [9, Sec. 6.9], it holds

$$\mathrm{prox}_{\varphi}(x) = \big(1 - \frac{\lambda}{\max\{\|x\|, \lambda\}}\big)x = x - \mathrm{Proj}_{B_\lambda(0)}(x),$$

where $B_\lambda(0)$ is the λ-ball at 0. Due to [164, Rem. 3.1] we have $\mathbf{Id} - \Sigma \in \partial\mathrm{prox}_{\varphi}(x)$ for

$$\Sigma = \begin{cases} \frac{\lambda}{\|x\|}[\mathbf{Id} - \frac{xx^\top}{\|x\|^2}] & \text{if } \|x\| > \lambda, \\ \mathbf{Id} & \text{else.} \end{cases}$$

For both ℓ_1- and ℓ_2-norm the proximal operator is strongly semismooth [164, Lem. 2.1]. Further, for $\lambda_1, \lambda_2 > 0$, due to [164, Prop. 2.1] we have

$$\mathrm{prox}_{\lambda_1\|\cdot\|_1 + \lambda_2\|\cdot\|}(x) = \big(\mathrm{prox}_{\lambda_2\|\cdot\|} \circ \mathrm{prox}_{\lambda_1\|\cdot\|_1}\big)(x).$$

Hence, the proximal operator of the sum can be computed by composition of the individual proximal operators; this can be extended to settings where the ℓ_2-norm is computed over a partition of the components of x [164]. Due to [153, Prop. 2.16], the proximal operator of the sum is also strongly semismooth.

Total Variation. Let $\varphi(x) = \lambda\sum_{i=2}^{n}|x_i - x_{i-1}|$ for some $\lambda > 0$. The proximal operator $\mathrm{prox}_{\varphi}(x)$ is not known in closed form, but can be computed efficiently using Condat's algorithm [23]. Using a duality argument, one can even compute an element of $\partial\mathrm{prox}_{\varphi}(x)$ and extend the results to the regularizer $\varphi(x) = \lambda_1\|x\|_1 + \lambda_2\sum_{i=2}^{n}|x_i - x_{i-1}|$ for $\lambda_1, \lambda_2 > 0$. As the derivations of these formulas is rather technical, we refer to [85, 135, 163] for implementation details.

Due to φ being convex and piecewise affine, prox_{φ} is piecewise linear [126, Thm. 12.30] and thus strongly semismooth.

[7] Setting $d_i = 0$ in the corner case $|x_i| = \lambda$ leads to a sparser matrix and helps to reduce computation in our semismooth Newton method (cf. [84, Sec. 3.3]).

Indicator Functions. Let $C \subset \mathbb{R}^n$ be a nonempty, closed, convex set. Then, the proximal operator of the indicator function is equal to the projection, i.e. $\operatorname{prox}_{\mathbb{1}_C}(x) = \operatorname{Proj}_C(x)$, and in particular the projection is unique [9, Thm. 6.25]. The projection for standard constraint sets is given in [9, Sec. 6.4.6]. We refer to [86] for computing an element of the subdifferential of $\operatorname{Proj}_C(x)$, if C is a generalized matrix simplex.

3.10 Extension: Prox-linear Algorithm

When training machine learning models, the objective function is often the composition of two functions: one of them, for a given input y^{in}, computes a model output based on learnable parameters x (the variables to optimize for). We have called this function $\texttt{forward}_x(\cdot)$ in Section 2.1. The second mapping computes the loss, as a function of the model output $\texttt{forward}_x(y^{\text{in}})$ and a desired output y^{out}.[8]

For a training set of size N, we can formalize the loss function to be

$$f(x) := \frac{1}{N} \sum_{i=1}^{N} h_i(c_i(x)).$$

In the above, $c_i(x)$ denotes the function that produces a model output for the i-th input sample. The loss function measuring the quality of the model output is denoted by h_i. The dependence on the index i for h_i, c_i is typically given via $h_i(z) = h(z; y_i^{\text{out}})$ and $c_i(x) = c(x; y_i^{\text{in}})$. However, as $(y_i^{\text{in}},\ y_i^{\text{out}})_{i=1,\dots,N}$ are a given training set, we will only denote the dependency on the variable x. Adding a regularization function φ, we obtain the problem

$$\min_{x \in \mathbb{R}^n} f(x) + \varphi(x), \quad f(x) := \frac{1}{N} \sum_{i=1}^{N} h_i(c_i(x)). \tag{3.52}$$

One important observation is that the network architecture only affects c_i, while the loss function h_i is often simple: for example, the squared loss for regression or the cross entropy loss for classification. In particular, for both examples h_i is a convex function. It is the arbitrary choice of c_i, for example a multi-layer network with activation functions, which results in $h_i(c_i(x))$ being non-convex. This section discusses how this particular structure could be leveraged for solving (3.52); the main technique will be similar to what has been discussed before for obtaining a practical stochastic proximal point method.

Throughout this section, we consider problems of form (3.52) where we assume $h_i : \mathbb{R}^m \to \mathbb{R}$ to be convex and differentiable for some $m \in \mathbb{N}$, and $c_i : \mathbb{R}^n \to \mathbb{R}^m$ to be differentiable. We assume that the Fenchel conjugate h_i^* is twice continuously differentiable on $\operatorname{int}(\operatorname{dom}(h_i^*))$. Further, assume that $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is proper, closed, and convex. We use the convention $\nabla c_i(x) \in \mathbb{R}^{n \times m}$, i.e. ∇ denotes the transposed Jacobian.

3.10.1 Background and Related Work

We start by introducing the (deterministic) prox-linear algorithm [26, 35]. Its core idea is to compose h_i with the linearization of c_i around the current iterate and take a proximal step. This is in contrast to gradient descent which would linearize $h_i(c_i(\cdot))$ instead. In the following,

[8] What is used for y^{out} depends on whether we train a supervised learning task or not.

consider that the current iterate $x \in \mathbb{R}^n$ is given and we want to compute the next iterate x^+. For simplicity, the iteration index is suppressed in the notation.

Let $\alpha > 0$ denote the step size. The update of a (finite-sum) prox-linear algorithm is given by

$$\begin{aligned} x^+ &= \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \frac{1}{N} \sum_{i=1}^N h_i\Big(c_i(x) + \nabla c_i(x)^\top (y - x)\Big) + \varphi(y) + \frac{1}{2\alpha}\|y - x\|^2 \\ &= \operatorname{prox}_{\alpha\varphi}\Big(x - \frac{\alpha}{N} \sum_{i=1}^N \nabla c_i(x) \nabla h_i(c_i(x) + \nabla c_i(x)^\top (x^+ - x))\Big). \end{aligned}$$

In the stochastic setting, we can replace the full sum over $i = 1, \dots, N$ with a mini-batch S, i.e. a random tuple containing b elements of $[N]$. Let us introduce the map $\kappa : [b] \to [N]$ mapping the index of the mini-batch to the index of the data point.

The mini-batch version of the update then becomes

$$x^+ = \operatorname{prox}_{\alpha\varphi}\Big(x - \frac{\alpha}{b} \sum_{i=1}^b \nabla c_{\kappa(i)}(x) \nabla h_{\kappa(i)}(c_{\kappa(i)}(x) + \nabla c_{\kappa(i)}(x)^\top (x^+ - x))\Big). \tag{3.53}$$

The following section shows how to compute this *implicit* update efficiently. In essence, the semismooth Newton framework developed in the previous section can be used in very similar fashion for the prox-linear method.

Related work. The prox-linear method has been analyzed in [35,83], with its stochastic version being considered in [26]. We refer to the introduction of [35] for further references on the history of this method. A prox-linear method with variance reduction is analyzed in [162], with the difference that the finite sum is inside the argument of the outer function.

Practical consideration for the prox-linear method on how to compute (3.53) have been proposed in [131, 142]. Both of these works present very similar ideas to the ones here.[9] Thus, we do not consider the contents of this section to be particularly novel, but rather want to showcase how to extend the ideas developed earlier. Specifically, [131] solves the prox-linear subproblem (3.53) approximately with conjugate gradient (in the dual, similar to what we propose below) and obtain promising numerical performance.

3.10.2 Algorithmic Framework

The update (3.53) can be transformed into a dual subproblem, as we show next. We will then consider important special cases where solving the subproblem can be simplified. Using Fenchel duality [9, Thm. 4.20], we introduce for $i = 1, \dots, b$ the variable

$$\xi_i := \nabla h_{\kappa(i)}\big(c_{\kappa(i)}(x) + \nabla c_{\kappa(i)}(x)^\top (x^+ - x)\big) \iff \nabla h^*_{\kappa(i)}(\xi_i) = c_{\kappa(i)}(x) + \nabla c_{\kappa(i)}(x)^\top (x^+ - x).$$

Therefore, for $b \in [N]$, solve

$$x^+ = \operatorname{prox}_{\alpha\varphi}\Big(x - \frac{\alpha}{b} \sum_{i=1}^b \nabla c_{\kappa(i)}(x) \xi_i\Big), \tag{3.54}$$

$$\nabla h^*_{\kappa(i)}(\xi_i) = c_{\kappa(i)}(x) + \nabla c_{\kappa(i)}(x)^\top (x^+ - x), \quad i = 1, \dots, b. \tag{3.55}$$

[9] [142] appeared on arxiv shortly after our work [98], [131] was posted over one year later. However, the contents of this section are unpublished work and were not part of [98], even though the central idea is essentially the same.

For $\xi = (\xi_1, \dots, \xi_b)^\top,\ \xi_i \in \mathbb{R}^m$, we introduce the linear operator

$$\mathcal{C} : \mathbb{R}^{b\times m} \to \mathbb{R}^n, \quad \mathcal{C}[\xi] := \frac{1}{b}\sum_{i=1}^{b} \nabla c_{\kappa(i)}(x)\xi_i.$$

Note that in the case of $m = 1$ the operator $\mathcal{C}$ can be represented by a $n \times b$ matrix. Plugging in (3.54) into (3.55), we obtain the following system of nonlinear equations

$$\begin{aligned} &\mathcal{V}(\xi) = \big(\mathcal{V}_1(\xi), \dots, \mathcal{V}_b(\xi)\big) = 0, \\ &\mathcal{V}_i(\xi) := \nabla h^*_{\kappa(i)}(\xi_i) - c_{\kappa(i)}(x) - \nabla c_{\kappa(i)}(x)^\top \big(\mathrm{prox}_{\alpha\varphi}(x - \alpha\mathcal{C}[\xi]) - x\big). \end{aligned} \tag{3.56}$$

We aim to solve (3.56) with a globalized semismooth Newton method. An element of the Clarke subdifferential $\mathcal{W} \in \partial\mathcal{V}(\xi)$ is given by

$$\begin{aligned} \mathcal{W} &= \mathrm{diag}\left((\nabla^2 h^*_{\kappa(i)}(\xi_i))_{i=1,\dots,b}\right) + \frac{\alpha}{b}\Lambda, \\ \Lambda_{ij} &= \nabla c_{\kappa(i)}(x)^\top U \nabla c_{\kappa(j)}(x), \quad U \in \partial \mathrm{prox}_{\alpha\varphi}(x - \alpha\mathcal{C}[\xi]), \quad i,j = 1,\dots,b. \end{aligned}$$

Moreover, the (semismooth) Newton method can be globalized using Armijo line search, by the following observation: using that

$$\nabla \mathrm{env}_{\alpha\varphi}(y) = y - \mathrm{prox}_{\alpha\varphi}(y) \Longrightarrow \mathrm{prox}_{\alpha\varphi}(y) = \nabla\big[\frac{1}{2}\|y\|^2 - \mathrm{env}_{\alpha\varphi}(y)\big],$$

we have that $\mathcal{V}(\xi) = \nabla\mathcal{U}(\xi)$ for

$$\begin{aligned} \mathcal{U}(\xi) := &\sum_{i=1}^{b} h^*_{\kappa(i)}(\xi_i) - \sum_{i=1}^{b} \langle c_{\kappa(i)}(x), \xi_i\rangle \\ &+ \frac{b}{2\alpha}\Big(\|x - \alpha\mathcal{C}[\xi]\|^2 - 2\mathrm{env}_{\alpha\varphi}(x - \alpha\mathcal{C}[\xi])\Big) + \sum_{i=1}^{b} \langle \nabla c_{\kappa(i)}(x)^\top x, \xi_i\rangle. \end{aligned} \tag{3.57}$$

Altogether, solving (3.56) is equivalent to finding a stationary point of $\mathcal{U}$. Under appropriate assumptions on h_i^*, $\mathcal{U}$ can be guaranteed to be strongly convex, and convergence theory for the semismooth Newton scheme can be derived similar to Section 3.4.

Easy updates. Next, we consider the system of equations (3.54)–(3.55) for the following simplified setting:

i) Loss function $h_i(z) = \|z - y_i^{\mathrm{out}}\|^2$, where $y_i^{\mathrm{out}} \in \mathbb{R}^m$,

ii) squared ℓ_2-regularization $\varphi(x) = \frac{\lambda}{2}\|x\|^2$ for some $\lambda \geq 0$.

Note that this does not make any restrictions to the model architecture (encoded in c_i) except for the loss on the outputs of the final layer. Clearly, h_i is strongly convex and it is easy to compute $h_i^*(z) = \frac{1}{4}\|z\|^2 + \langle z, y_i^{\mathrm{out}}\rangle$ and $\nabla h_i^*(z) = \frac{1}{2}z + y_i^{\mathrm{out}}$. Further, it holds

$$\mathrm{prox}_{\alpha\varphi}(x) = \frac{1}{1+\alpha\lambda}x, \quad \mathrm{env}_{\alpha\varphi}(x) = \frac{\alpha\lambda}{2(1+\alpha\lambda)}\|x\|^2, \quad \partial\mathrm{prox}_{\alpha\varphi}(x) = \{\frac{1}{1+\alpha\lambda}\mathbf{Id}\}.$$

We will show that in this simplified setting, (3.56) is a *linear* system of equations. Plugging in ∇h_i^* and $\mathrm{prox}_{\alpha\varphi}$, we get

$$\begin{aligned}\mathcal{V}_i(\xi) &= \frac{1}{2}\xi_i + y_{\kappa(i)}^{\text{out}} - c_{\kappa(i)}(x) - \nabla c_{\kappa(i)}(x)^\top\big(\frac{1}{1+\alpha\lambda}(x - \alpha\mathcal{C}[\xi]) - x\big) \\ &= \frac{1}{2}\xi_i + y_{\kappa(i)}^{\text{out}} - c_{\kappa(i)}(x) + \frac{\alpha\lambda}{1+\alpha\lambda}\nabla c_{\kappa(i)}(x)^\top x + \frac{\alpha}{1+\alpha\lambda}\nabla c_{\kappa(i)}(x)^\top\mathcal{C}[\xi].\end{aligned} \tag{3.58}$$

Plugging in $\mathcal{C}[\xi]$ and rearranging we obtain

$$\begin{aligned}&\mathcal{V}_i(\xi) = 0 \iff \\ &\frac{1}{2}\xi_i + \frac{\alpha}{b(1+\alpha\lambda)}\sum_{j=1}^{b}\nabla c_{\kappa(i)}(x)^\top\nabla c_{\kappa(j)}(x)\xi_j = c_{\kappa(i)}(x) - y_{\kappa(i)}^{\text{out}} - \frac{\alpha\lambda}{1+\alpha\lambda}\nabla c_{\kappa(i)}(x)^\top x.\end{aligned} \tag{3.59}$$

Clearly, solving $\mathcal{V}(\xi) = (\mathcal{V}_1(\xi), \dots, \mathcal{V}_b(\xi)) = 0$ is a linear system of equations in ξ. It can be seen that the linear operator applied to ξ is positive definite and self-adjoint; hence, we can solve (3.59) with efficient methods such as conjugate gradient.

Closed-form updates. If $b = 1$, $m = 1$, and in the setting above, we can compute ξ in closed form. This assumption is of course very restrictive, but it serves to illustrate the update formula. In this case, as $b = 1$, we can simplify notation to $\kappa(i) \to i$ and $\xi_i \to \xi \in \mathbb{R}$. Further, $\mathcal{C}[\xi] = \nabla c_i(x)\xi$. Equation (3.58) becomes

$$\begin{aligned}&\frac{1}{2}\xi + y_i^{\text{out}} - c_i(x) + \frac{\alpha\lambda}{1+\alpha\lambda}\nabla c_i(x)^\top x + \frac{\alpha}{1+\alpha\lambda}\|\nabla c_i(x)\|^2\xi = 0 \\ \iff &\xi = (\frac{1}{2} + \frac{\alpha}{1+\alpha\lambda}\|\nabla c_i(x)\|^2)^{-1}\big[c_i(x) - y_i^{\text{out}} - \frac{\alpha\lambda}{1+\alpha\lambda}\nabla c_i(x)^\top x\big].\end{aligned}$$

Consequently, plugging ξ into (3.54), we have the update

$$x^+ = \frac{1}{1+\alpha\lambda}\Big[x - \alpha\frac{c_i(x) - y_i^{\text{out}} - \frac{\alpha\lambda}{1+\alpha\lambda}\nabla c_i(x)^\top x}{\frac{1}{2} + \frac{\alpha}{1+\alpha\lambda}\|\nabla c_i(x)\|^2}\nabla c_i(x)\Big]. \tag{3.60}$$

3.11 Conclusions and Open Questions

In this chapter, we present SNSPP, a variance-reduced stochastic proximal point method, using a semismooth Newton method to solve the subproblem in each update step. Most importantly, this is the first *practical* SPP method that can handle mini-batching, (nonsmooth) regularization terms and is implementable for many classical loss functions, including least squares, the logistic loss or a Student-t likelihood loss.

Further, SNSPP is novel in combining SPP with variance reduction and obtaining improved rates for constant step sizes for the weakly and strongly convex case, similar to the results of SVRG. Our experiments show that our proposed algorithm is competitive with SAGA and SVRG, and can be much better for problem instances where the dimension is large. Similar to what has been observed for SPP versus SGD without variance reduction, SNSPP is less sensitive to step-size choice than its variance-reduced stochastic proximal gradient competitors.

SNSPP employs the same variance reduction scheme as SVRG, computing the full gradient every once in a while. We have not covered other variance reduction schemes (for example the one of

SAGA), but it is clear how they can be immediately incorporated into the framework of SNSPP. However, for the theoretical results this change would necessitate different proof strategies.

While SNSPP is mostly designed for problems with generalized linear models, we have shown in Section 3.10 how the underlying idea can be extended to a more general composite problem structure. Investigating the theoretical and empirical performance of this method ís left for future work.

Chapter 4

A Stochastic Proximal Polyak Step Size

The chapter is mainly based on the article

[132] F. Schaipp, R. M. Gower, and M. Ulbrich, *A Stochastic Proximal Polyak Step Size*, Transactions on Machine Learning Research, (2023), https://openreview.net/forum?id=jWr41htaB3.

4.1 Introduction

When training machine learning models, a commonly known problem is how to select the step size (or learning rate). Not only does the practitioner need to choose the *value* of the learning rate, but also a *schedule*, meaning that the learning rate in general can change over the course of training. To complicate this matter even further, it has been shown in [137] that the benefit of learning rate schedules depends on the specific problem and optimization method, and while selecting a schedule helps on average, it can also lead to worse results for specific instances. The same study [137] concludes that picking an optimizer and tuning its hyperparameters performs equally well as compared to choosing the best among a set of optimization methods in their default setting.

Designing optimization methods that select the learning rate on the fly, for example based on the current progress in terms of loss function or gradient, could overcome these issues and reduce the need for tuning, or alternatively in light of [137], lead to better performance in default setting. We will refer to such techniques in the following as *adaptive learning rates*.[1]

A general principle in optimization is that methods which exploit the problem structure at hand are usually superior. However, widely used methods, such as `SGD` or `Adam`, do not fully make use of the inherent structure in deep learning. On the one hand, deep learning models can approximately interpolate the training data (i.e. reach a loss of zero) while generalizing well [51, 91]. In such situations, an *a priori* estimate of the optimal value of the underlying problem is available. Even the simpler fact that loss functions are typically non-negative, yielding a lower bound of the objective, is not used in `SGD` or `Adam`.

On the other hand, the stochastic loss function value computed at the current iterate is usually available at no extra cost but not used for training. This is due to the fact that in frameworks

[1] This is not to be confused with the notion of adaptivity for `Adam` or `Adagrad`, introduced in Section 2.6, where the learning rate is adapted to each coordinate.

such as `Pytorch`, gradients are computed with backpropagation after a forward pass. Both the current function value and optimal value are employed in the famously known *Polyak step size* [117], proposed for the subgradient method in 1987. In recent years, several works [6, 10, 59, 88] developed adaptive learning rate mechanisms for stochastic optimization reminiscent of Polyak's work.

An open question in this line of research is how to deal with additional objective function terms, such as regularization, for adaptive learning rates. This is important *precisely because* additional terms change the objective and its optimal value and these quantities are used for Polyak-type methods.

One of the most established regularization in deep learning is using the squared ℓ_2-norm (also called weight decay) in order to improve generalization [89]. To give a non-extensive list of examples, weight decay has been used for training the large language model `GPT-3` [18], for the vision transformer `ViT` [32], or Google's weather forecasting model `MetNet2` [40].

The subject matter of this chapter is to propose a stochastic Polyak step size for regularized problems. Our solution handles regularization in a *proximal* fashion instead of interpreting it as an additional loss function term. We will state convergence theory for general regularization terms, and show that the proposed method can be implemented easily for squared ℓ_2-norm regularization. Our experiments substantiate that stochastic Polyak step sizes indeed reduce the tuning effort as hypothesized above – but it matters how to handle regularization, and the proximal way we propose typically works better.

Problem setup. In this chapter, we consider problems of the form (`P`), namely

$$\min_{x\in\mathbb{R}^n} \psi(x), \quad \psi(x) := f(x) + \varphi(x).$$

Here, f is again given via (2.1), i.e. $f(x) = \mathbb{E}[f(x;S)]$, where $f(\cdot;s):\mathbb{R}^n\to\mathbb{R}$ is locally Lipschitz for all $s\in\mathcal{S}$ and $\varphi:\mathbb{R}^n\to\overline{\mathbb{R}}$ is a proper, closed, and convex regularization function. An important special case will be the unregularized problem, when $\varphi = 0$, given by

$$\min_{x\in\mathbb{R}^n} f(x), \quad f(x) = \mathbb{E}[f(x;S)]. \tag{4.1}$$

Throughout this chapter, we assume that Assumption 1 holds true. It is clear from the above that Polyak step sizes are only useful in situations where we know or can estimate the optimal value, or (as it will turn out) a lower bound thereof. As a result, we make the following assumption throughout the entire chapter.

Assumption 8. *For every $s\in\mathcal{S}$, $\inf_x f(x;s)$ is finite and there exists $C(s)$ satisfying $C(s)\leq \inf_x f(x;s)$.*

In many machine learning applications, the loss functions $f(\cdot;s)$ are non-negative and thus $C(s)=0$ is an appropriate choice. We also make the technical assumption that whenever we select $g\in\partial f(x;s)$ and $C(s)$, this selection is measurable with respect to $P(s)$, the associated probability measure of $\mathcal{S}$.

4.2 Background and Contributions

Polyak step size. For minimizing a convex, possibly non-differentiable function f, Polyak proposed in [117, Chapter 5.3] the step size

$$x^{k+1} = x^k - \alpha_k g_k, \quad \alpha_k = \frac{f(x^k) - \min f}{\|g_k\|^2}, \quad g_k \in \partial f(x^k) \setminus \{0\}.$$

This particular choice of α_k, requiring the knowledge of $\min f$, has been subsequently called the *Polyak step size* for the subgradient method. Recently, [10,88,109] adapted the Polyak step size to the stochastic setting: consider the (ER) case and assume that each f_i is differentiable and that a lower bound $C(s_i) \leq \inf_x f_i(x)$ is known for all $i \in [N]$. The method proposed by [88] is

$$x^{k+1} = x^k - \min\left\{\gamma_b, \frac{f_{i_k}(x^k) - C(S_{i_k})}{c\|\nabla f_{i_k}(x^k)\|^2}\right\}\nabla f_{i_k}(x^k), \tag{SPS$_{\max}$}$$

with hyper-parameters $c, \gamma_b > 0$ and where in each iteration i_k is drawn from $\{1, \dots, N\}$ uniformly at random. It is important to note that the initial work [88] used $C(s_i) = \inf f_i$; later, [109] established theory for SPS$_{\max}$ for the more general case of $C(s_i) \leq \inf_x f_i(x)$ and allowing for mini-batching. Other works analyzed the Polyak step size in the convex, smooth setting [59] and in the convex, smooth and stochastic setting [118]. Further, the stochastic Polyak step size is closely related to stochastic model-based proximal point [6] as well as stochastic bundle methods [111].

Contribution. We propose a proximal version of the stochastic Polyak step size, called ProxSPS, which explicitly handles regularization functions. Our proposal is based crucially on the fact that the stochastic Polyak step size can be motivated with stochastic proximal point for a truncated linear model of the objective function (we explain this in detail in Section 4.3). Our method has closed-form updates for squared ℓ_2-regularization. We provide theoretical guarantees for ProxSPS for any closed, proper, and convex regularization function (including indicator functions for constraints). Our main results, Theorem 4.7 and Theorem 4.8, also give new insights for SPS$_{\max}$, in particular showing exact convergence for convex and non-convex settings.

Lower bounds and regularization. Methods such as SPS$_{\max}$ need to estimate a lower bound $C(s)$ for each loss function $f(\cdot; s)$. Though $\inf_x f(x; s)$ can be precomputed in some restricted settings, in practice the lower bound $C(s) = 0$ is used for non-negative loss functions.[2] The tightness of the choice $C(s)$ is further reflected in the constant $\sigma^2 := \min f - \mathbb{E}[C(S)]$, which affects the convergence guarantees of SPS$_{\max}$ [109].

Contribution. For regularized problems (P) and if φ is differentiable, the current proposal of SPS$_{\max}$ would add φ to every loss function $f(\cdot; s)$. In this case, for non-negative regularization terms, such as the squared ℓ_2-norm, the lower bound $C(s) = 0$ is always loose. Indeed, if $\varphi \geq 0$, then $\inf_{x \in \mathbb{R}^n}(f(x; s) + \varphi(x)) \geq \inf_{x \in \mathbb{R}^n} f(x; s)$ and this inequality is strict in most practical scenarios. For our proposed method ProxSPS, we now need only estimate a lower bound for the loss $f(x; s)$ and not for the composite function $f(x; s) + \varphi(x)$. Further, ProxSPS decouples the adaptive step size for the gradient of the loss from the regularization (we explain this in detail in Section 4.4.1 and Fig. 4.1).

Proximal and adaptive methods. The question on how to handle regularization terms has also been posed for other families of adaptive methods. For Adam [74] with ℓ_2-regularization it has

[2] See for instance https://github.com/IssamLaradji/sps.

been observed that it generalizes worse and is harder to tune than `AdamW` [89] which uses weight decay. Further, `AdamW` can be seen as an approximation to a proximal version of `Adam` [169].[3] On the other hand, [88] showed that – without regularization – default hyperparameter settings for $\texttt{SPS}_{\max}$ give very encouraging results on matrix factorization and image classification tasks. This is promising since it suggests that $\texttt{SPS}_{\max}$ is an *adaptive* method, and can work well across varied tasks without the need for extensive hyperparameter tuning.

Contribution. We show that by handling ℓ_2-regularization using a proximal step, our resulting `ProxSPS` is less sensitive to hyperparameter choice as compared to $\texttt{SPS}_{\max}$. This becomes apparent in matrix factorization problems, where `ProxSPS` converges for a much wider range of regularization parameters and learning rates, while $\texttt{SPS}_{\max}$ is more sensitive to these settings. We also show similar results for image classification over the `CIFAR10` and `Imagenet32` dataset when using a `ResNet` model, where, compared to `AdamW`, our method is less sensitive with respect to the regularization parameter.

The remainder of our paper is organized as follows: we will first recall how the stochastic Polyak step size, in the case of $\varphi = 0$, can be derived using the model-based approach of [6,26] and how this is connected to $\texttt{SPS}_{\max}$. We then derive `ProxSPS` based on the connection to model-based methods, and present our theoretical results, based on the proof techniques in [26].

4.3 A Model-based Viewpoint for the Unregularized Case

In this section, consider problems of form (4.1), i.e. the regularization term φ is zero. Let us recall some important insights from Section 2.7: for solving (4.1), model-based stochastic proximal point in each iteration constructs a model $f_x(\cdot; s)$ approximating $f(\cdot; s)$ locally around x. With $S_k \sim P$ being drawn at random, the update is computed as

$$x^{k+1} = \arg\min_{y \in \mathbb{R}^n} f_{x^k}(y; S_k) + \frac{1}{2\alpha_k}\|y - x^k\|^2. \tag{4.2}$$

We have shown in Section 2.7 that for the *linear model*, $f_x(y; s) = f(x; s) + \langle g, y - x\rangle$ where $g \in \partial f(x; s)$, update (4.2) is equal to `SGD`.

For the *truncated model*, $f_x(y; s) = \max\{f(x; s) + \langle g, y - x\rangle, C(s)\}$ where $g \in \partial f(x; s)$ and $C(s) \leq \inf_{z \in \mathbb{R}^n} f(z; s)$, (4.2) results in the update

$$x^{k+1} = x^k - \min\left\{\alpha_k, \frac{f(x^k; S_k) - C(S_k)}{\|g_k\|^2}\right\} g_k, \quad g_k \in \partial f(x^k, S_k). \tag{4.3}$$

A proof for this statement is given in Lemma 4.11; the complete method is stated in Algorithm 9. This inherent connection between the truncated model and the stochastic Polyak step size (`SPS`) is not a new insight and has been pointed out in several works (including [6,88] and [10, Prop. 1]). Regarding the name `SPS`, it should be pointed out that this acronym (and variations of it) have been used for stochastic Polyak-type methods in slightly different ways [51, 88]. For instance consider again the $\texttt{SPS}_{\max}$ method

$$x^{k+1} = x^k - \min\left\{\gamma_b, \frac{f_{i_k}(x^k) - C(s_{i_k})}{c\|\nabla f_{i_k}(x^k)\|^2}\right\} \nabla f_{i_k}(x^k), \tag{$\texttt{SPS}_{\max}$}$$

[3] For `SGD` treating ℓ_2-regularization as a part of the loss can be seen to be equivalent to its proximal version (cf. Section 4.8.5).

Algorithm 9 SPS

Require: $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$.

1: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
2: Sample S_k and set $C_k := C(S_k)$.
3: Choose $g_k \in \partial f(x^k; S_k)$. If $g_k = 0$, set $x^{k+1} = x^k$. Otherwise, set

$$x^{k+1} = x^k - \gamma_k g_k, \quad \gamma_k = \min\left\{\alpha_k, \frac{f(x^k; S_k) - C_k}{\|g_k\|^2}\right\}. \tag{4.4}$$

4: **end for**
5: **return** x^K

where $c, \gamma_b > 0$. Clearly, for $c = 1$ and $\alpha_k = \gamma_b$, update (4.4) is identical to $\texttt{SPS}_{\max}$. With this in mind, we can interpret the hyperparameter γ_b in $\texttt{SPS}_{\max}$ simply as a step size for the model-based stochastic proximal point step. For the parameter c on the other hand, the model-based approach motivates the choice $c = 1$. In this article, we will focus on this natural choice $c = 1$ which also reduces the amount of hyperparameter tuning. However, we should point out that, in the strongly convex case, $c = 1/2$ gives the best rate of convergence in [88].

4.4 The Regularized Case

Now we consider regularized problems of the form (P), i.e.

$$\min_{x \in \mathbb{R}^n} \psi(x), \quad \psi(x) = f(x) + \varphi(x),$$

where $\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, closed, λ-strongly convex function for $\lambda \geq 0$ (we allow $\lambda = 0$). For $s \in \mathcal{S}$, denote by $\psi_x(\cdot; s)$ a stochastic model of the objective ψ at x. We aim to analyze algorithms with the update

$$x^{k+1} = \arg\min_{x \in \mathbb{R}^n} \psi_{x^k}(x; S_k) + \frac{1}{2\alpha_k}\|x - x^k\|^2, \tag{4.5}$$

where $S_k \sim P$ and $\alpha_k > 0$. Naively, if we know a lower bound $\tilde{C}(s)$ of $f(\cdot; s) + \varphi(\cdot)$, the truncated model could be constructed for the function $f(x; s) + \varphi(x)$, resulting in

$$\psi_x(y; s) = \max\{f(x; s) + \varphi(x) + \langle g + u, y - x\rangle, \tilde{C}(s)\}, \quad g \in \partial f(x; s), \quad u \in \partial\varphi(x). \tag{4.6}$$

In fact, [6] and [88] work in the setting of unregularized problems and hence their approaches would handle regularization in this way. What we propose instead, is to only truncate a linearization of the loss $f(x; s)$, yielding the model

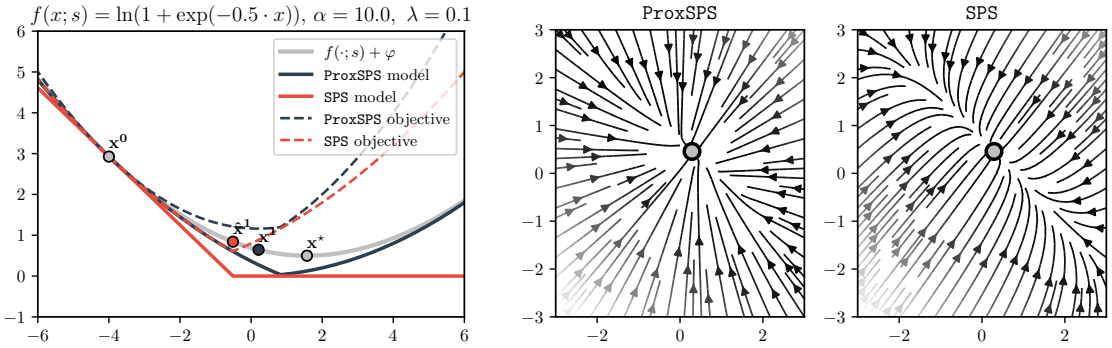
$$\psi_x(y; s) = f_x(y; s) + \varphi(y), \quad f_x(y; s) = \max\{f(x; s) + \langle g, y - x\rangle, C(s)\}, \quad g \in \partial f(x; s). \tag{4.7}$$

Solving (4.5) with the model in (4.7) results in

$$x^{k+1} = \arg\min_{y \in \mathbb{R}^n} \max\{f(x^k; S_k) + \langle g_k, y - x^k\rangle, C(S_k)\} + \varphi(y) + \frac{1}{2\alpha_k}\|y - x^k\|^2. \tag{4.8}$$

The resulting model-based stochastic proximal point method is given in Algorithm 10 [4]. Lemma 4.12 shows that, if prox_φ is known, update (4.8) can be computed by minimizing a strongly convex

[4] For $\varphi = 0$, Algorithm 10 is identical to Algorithm 9.



(a) Regularized logistic loss.

(b) Regularized squared loss with $\alpha_k = 1,\ \lambda = 1$.

Figure 4.1: a) `SPS` refers to model (4.6) whereas `ProxSPS` refers to (4.7). We plot the corresponding model $\psi_{x^0}(y;s)$ and the objective function of (4.5). x^1 (resp. $\hat{x}^1$) denotes the new iterate for `ProxSPS` (resp. `SPS`), $x^\star$ is the minimizer of $f(\cdot;s)+\varphi$. b) Streamlines of the vector field $V(x^k) := x^{k+1} - x^k$, for $f(x) = \|Ax-b\|^2$ and for the deterministic update, i.e. $f(x;s) = f(x)$. `ProxSPS` refers to update (4.11) and `SPS` refers to (4.10). The circle marks the minimizer of $f(x) + \frac{\lambda}{2}\|x\|^2$.

function over a compact one-dimensional interval. The relation to the proximal operator of φ motivates the name `ProxSPS`. Further, the `ProxSPS` update (4.8) has a closed form solution when φ is the squared ℓ_2-norm, as we detail in the next section.

Algorithm 10 `ProxSPS`

Require: $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$.

1: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
2: Sample S_k and set $C_k := C(S_k)$.
3: Choose $g_k \in \partial f(x^k; S_k)$. Update x^{k+1} according to (4.8).
4: **end for**
5: **return** x^K

4.4.1 The Special Case of ℓ_2-regularization

When $\varphi(x) = \frac{\lambda}{2}\|x\|^2$ for some $\lambda > 0$, `ProxSPS` (4.8) has a closed form solution as we show next in Lemma 4.1. For this lemma, recall that the proximal operator of $\varphi(x) = \frac{\lambda}{2}\|x\|^2$ is given by $\operatorname{prox}_{\alpha\varphi}(x) = \frac{1}{1+\alpha\lambda}x$ for all $\alpha > 0,\ x \in \mathbb{R}^n$.

Lemma 4.1. *Let $\varphi(x) = \frac{\lambda}{2}\|x\|^2$ and let $g \in \partial f(x;s)$ and $C(s) \leq \inf_{z\in\mathbb{R}^n} f(z;s)$ hold for all $s \in \mathcal{S}$. For $\psi_x(y;s) = f_x(y;s) + \varphi(y)$ with $f_x(y;s) = \max\{f(x;s) + \langle g, y-x\rangle, C(s)\}$ consider update (4.8), i.e.*

$$x^{k+1} = \arg\min_{y\in\mathbb{R}^n} \psi_{x^k}(y;S_k) + \frac{1}{2\alpha_k}\|y-x^k\|^2.$$

Denote $C_k := C(S_k)$ *and let* $g_k \in \partial f(x^k; S_k)$. *Define*

$$\tau_k^+ := \begin{cases} 0 & \text{if } g_k = 0, \\ \min\left\{\alpha_k, \left(\frac{(1+\alpha_k\lambda)(f(x^k;S_k)-C_k)-\alpha_k\lambda\langle g_k, x^k\rangle}{\|g_k\|^2}\right)_+\right\} & \text{else.} \end{cases}$$

Update (4.8) *is given by*

$$x^{k+1} \;=\; \frac{1}{1+\alpha_k\lambda}\Big(x^k - \tau_k^+ g_k\Big) = \operatorname{prox}_{\alpha_k\varphi}(x^k - \tau_k^+ g_k). \tag{4.9}$$

Proof. Rearrange to $f_x(y;s) = \big(f(x;s) - C(s) + \langle g, y-x\rangle\big)_+ + C(s)$ and drop the constant term $C(S_k)$ in the objective of (4.8). We can apply Lemma 4.10 with $c \leftarrow f(x^k;S_k) - C(S_k)$, $a \leftarrow g_k$, $y_0 \leftarrow x^k$, $\beta \leftarrow \alpha_k$ and $\mathbf{D} \leftarrow \mathbf{Id}_n$. The statement follows from (4.32) with $\tau_k^+ \leftarrow \tau$. □

Remark 4. *In the published version [132], we provide a more direct proof for the above lemma.*

The update (4.9) can be naturally decomposed into two steps, one stochastic gradient step with an adaptive stepsize, that is $\bar{x}^{k+1} = x^k - \tau_k^+ g_k$ followed by a proximal step $x^{k+1} = \operatorname{prox}_{\alpha_k\varphi}(\bar{x}^{k+1})$. This decoupling into two steps, makes it easier to interpret the effect of each step, with τ_k^+ adjusting for the scale/curvature and the following proximal step shrinking the resulting parameters. There is no clear separation of tasks if we apply the `SPS` method to the regularized problem, as we see next.

Algorithm 11 `ProxSPS` for $\varphi = \frac{\lambda}{2}\|\cdot\|^2$

Require: $x^0 \in \mathbb{R}^n$, step sizes $\alpha_k > 0$.

1: **for** $k = 0, 1, 2, \ldots, K-1$ **do**

2: Sample S_k and set $C_k := C(S_k)$.

3: Choose $g_k \in \partial f(x^k; S_k)$. If $g_k = 0$, set $x^{k+1} = \frac{1}{1+\alpha_k\lambda}x^k$. Otherwise, set

$$x^{k+1} = \frac{1}{1+\alpha_k\lambda}\left[x^k - \min\left\{\alpha_k, \left(\frac{(1+\alpha_k\lambda)(f(x^k;S_k)-C_k)-\alpha_k\lambda\langle g_k, x^k\rangle}{\|g_k\|^2}\right)_+\right\} g_k\right].$$

4: **end for**

5: **return** x^K

4.4.2 Comparing the Model of `SPS` and `ProxSPS`

For simplicity, assume again the discrete sample space setting (`ER`) with differentiable loss functions f_i and let $\varphi = \frac{\lambda}{2}\|\cdot\|^2$. Clearly, the composite problem (`P`) can be transformed to an instance of (4.1) by setting $\ell_i(x) := f_i(x) + \frac{\lambda}{2}\|x\|^2$ and solving $\min_x \ell(x)$ with $\ell(x) := \frac{1}{N}\sum_{i=1}^N \ell_i(x)$. Assume that a lower bound $\underline{\ell}_i \le \inf_x \ell_i(x)$ is known. In this case (4.6) becomes

$$\psi_x(y; s_i) = \max\Big\{f_i(x) + \tfrac{\lambda}{2}\|x\|^2 + \langle \nabla f_i(x) + \lambda x, y - x\rangle,\; \underline{\ell}_i\Big\}.$$

Due to Lemma 4.11, if $\nabla f_{i_k}(x^k) + \lambda x^k \neq 0$, the update (4.5) is given by

$$x^{k+1} = x^k - \min\left\{\alpha_k, \frac{f_{i_k}(x^k) + \frac{\lambda}{2}\|x^k\|^2 - \underline{\ell}_{i_k}}{\|\nabla f_{i_k}(x^k) + \lambda x^k\|^2}\right\}(\nabla f_{i_k}(x^k) + \lambda x^k). \tag{4.10}$$

We refer to this method, which is using model (4.6), as `SPS`. On the other hand, using model (4.7) and if $\nabla f_{i_k}(x^k) \neq 0$, the update of `ProxSPS` (4.9) is

$$x^{k+1} = \frac{1}{1+\alpha_k\lambda}\Big[x^k - \min\left\{\alpha_k, \left(\frac{(1+\alpha_k\lambda)(f_{i_k}(x^k)-C(s_{i_k}))-\alpha_k\lambda\langle\nabla f_{i_k}(x^k),x^k\rangle}{\|\nabla f_{i_k}(x^k)\|^2}\right)_+\right\}\nabla f_{i_k}(x^k)\Big]. \quad (4.11)$$

In Fig. 4.1a, we illustrate the two models (4.6) (denoted by `SPS`) and (4.7) (denoted by `ProxSPS`) for the logistic loss with squared ℓ_2-regularization. We can see that the `ProxSPS` model is a much better approximation of the (stochastic) objective function as it still captures the quadratic behaviour of φ. Furthermore, as noted in the previous section, `ProxSPS` decouples the step size of the gradient and of the shrinkage, and hence the update direction depends on α_k. In contrast, the update direction of `SPS` does not depend on α_k, and the regularization effect is intertwined with the adaptive step size. Another way to see that the model (4.7) on which `ProxSPS` is based on is a more accurate model as compared to the `SPS` model (4.6), is that the resulting vector field of `ProxSPS` takes a more direct route to the minimum, as illustrated in Fig. 4.1b.

Update (4.11) needs to compute the term $\langle\nabla f_{i_k}(x^k), x^k\rangle$ while (4.10) needs to evaluate $\|x^k\|^2$. Other than that, the computational costs are roughly identical. For (4.11), a lower bound $\underline{\ell}_i$ is required. For non-negative loss functions, in practice both $\underline{\ell}_i$ and $C(s_i)$ are often set to zero, in which case (4.7) will be a more accurate model as compared to (4.6). [5]

4.5 Convergence Analysis

For the convergence analysis of Algorithm 10, we can work with the following assumption on φ.

Assumption 9. *$\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is a proper, closed, λ-strongly convex function with $\lambda \geq 0$.*

Throughout this section we consider model (4.7), i.e. for $g \in \partial f(x; s)$, let

$$\psi_x(y; s) = f_x(y; s) + \varphi(y), \quad f_x(y; s) = \max\{f(x; s) + \langle g, y - x\rangle, C(s)\}.$$

Let us first state a lemma on important properties of the truncated model:

Lemma 4.2. *Consider $f_x(y; s) = \max\{f(x; s)+\langle g, y-x\rangle, C(s)\}$, where $g \in \partial f(x; s)$ is arbitrary and $C(s) \leq \inf_{z\in\mathbb{R}^n} f(z; s)$. Then, it holds:*

(i) The mapping $y \mapsto f_x(y; s)$ is convex.

(ii) For all $x \in \mathbb{R}^n$, it holds $f_x(x; s) = f(x; s)$. If $f(\cdot; s)$ is ρ_s–weakly convex for all $s \in \mathcal{S}$, then

$$f_x(y; s) \leq f(y; s) + \tfrac{\rho_s}{2}\|y - x\|^2 \quad \text{for all } x, y \in \mathbb{R}^n.$$

Proof. (i) The maximum over a constant and linear term is convex.

(ii) Recall that $C(s) \leq f(y; s)$ for all $y \in \mathbb{R}^n$. Therefore, $f_x(x; s) = \max\{C(s), f(x; s)\} = f(x; s)$. From weak convexity of $f(\cdot; s)$ it follows $f(x; s) + \langle g, y - x\rangle \leq f(y; s) + \frac{\rho_s}{2}\|y - x\|^2$ and therefore

$$f_x(y; s) \leq \max\{C(s), f(y; s) + \tfrac{\rho_s}{2}\|y - x\|^2\} = f(y; s) + \tfrac{\rho_s}{2}\|y - x\|^2 \quad \text{for all } y \in \mathbb{R}^n.$$

□

[5] For single element sampling, $\inf \ell_i$ can sometimes be precomputed (e.g. regularized logistic regression, see [88, Appendix D]). But even in this restricted setting it is not clear how to estimate $\inf \ell_i$ when using mini-batching.

4.5.1 Globally Bounded Subgradients

In this section, we show that the results for stochastic model-based proximal point methods in [26] can be immediately applied to our specific model – even though this model has not been explicitly analyzed in their article. This, however, requires assuming that the subgradients are bounded.

Proposition 4.3. *Let Assumption 1 and Assumption 9 hold and assume that there is an open, convex set U containing* dom φ. *Let $f(\cdot; s)$ be ρ_s–weakly convex for all $s \in \mathcal{S}$ and let $\rho = \mathbb{E}[\rho_S]$. Assume that there exists $G_s \in \mathbb{R}_+$ for all $s \in \mathcal{S}$, such that $\mathsf{G} := \sqrt{\mathbb{E}[G_S^2]} < \infty$ and*

$$\|g(x; s)\| \leq G_s \quad \forall g(x; s) \in \partial f(x; s), \ \forall x \in U. \tag{4.12}$$

Then, $\psi_x(y; s)$ (given in (4.7)) satisfies the following:

(B1) It is possible to generate infinitely many i.i.d. realizations $S_0, S_1, \ldots$ from $\mathcal{S}$.

(B2) It holds $\mathbb{E}[f_x(x; S)] = f(x)$ and $\mathbb{E}[f_x(y; S)] \leq f(y) + \frac{\rho}{2}\|y - x\|^2$ for all $x, y \in \mathbb{R}^n$.

(B3) The mapping $\psi_x(\cdot; s) = f_x(\cdot; s) + \varphi(\cdot)$ is convex for all $x \in \mathbb{R}^n$ and all $s \in \mathcal{S}$.

(B4) For all $x, y \in U$ and $s \in \mathcal{S}$, it holds $f_x(x; s) - f_x(y; s) \leq G_s\|x - y\|$.

Proof. The properties (B1)–(B4) are identical to (B1)–(B4) in [26, Assum. B], setting $r = \varphi$, $f_x(\cdot, \xi) = f_x(\cdot; s)$, $\eta = 0$, $\tau = \rho$, $\mathsf{L} = \mathsf{G}$, and $L(\xi) = G_s$. (B1) is identical to Assumption 1. (B2) holds due to Lemma 4.2, (ii), applying expectation and using the definition of f, i.e. $f(x) = \mathbb{E}[f(x; S)]$. (B3) holds due to Lemma 4.2, (i) and convexity of φ. For (B4), taking $g \in \partial f(x; s)$ and $x, y \in U$, we have

$$f_x(x; s) - f_x(y; s) \leq f(x; s) - f(x; s) - \langle g, y - x\rangle \leq \|g\|\|y - x\| \leq G_s\|x - y\|.$$

□

Corollary 4.4 (Weakly convex case)**.** *Let the assumptions of Proposition 4.3 hold with $\rho_s > 0$ for all $s \in \mathcal{S}$. Let $\rho = \mathbb{E}[\rho_S] < \infty$ and let $\Delta \geq \operatorname{env}_\psi^{1/(2\rho)}(x^0) - \min \psi$. Let $\{x^k\}_{k=0,\ldots,K}$ be generated by Algorithm 10 for constant step sizes $\alpha_k = \left(2\rho + \sqrt{\frac{4\rho\mathsf{G}^2 K}{\Delta}}\right)^{-1}$. Then, it holds*

$$\mathbb{E}\|\nabla \operatorname{env}_\psi^{1/(2\rho)}(x_\sim^K)\|^2 \leq \frac{8\rho\Delta}{K} + 16\mathsf{G}\sqrt{\frac{\rho\Delta}{K}},$$

where $x_\sim^K$ is uniformly drawn from $\{x^0, \ldots, x^{K-1}\}$.

Proof. The claim follows from Proposition 4.3 and [26, Thm. 4.3], (4.16) setting $\eta = 0$, $\bar{\rho} = 2\rho$, $T = K - 1$ and $\beta_t = \alpha_k^{-1}$. □

Corollary 4.5 ((Strongly) convex case)**.** *Let the assumptions of Proposition 4.3 hold with $\rho_s = 0$ for all $s \in \mathcal{S}$. Let $\lambda > 0$ and $x^\star = \arg\min_x \psi(x)$. Let $\{x^k\}_{k=0,\ldots,K}$ be generated by Algorithm 10 for step sizes $\alpha_k = \frac{2}{\lambda(k+1)}$. Then, it holds*

$$\mathbb{E}\Big[\psi\Big(\tfrac{2}{(K+1)(K+2)-2}\sum_{k=1}^{K}(k+1)x^k\Big) - \psi(x^\star)\Big] \leq \frac{\lambda}{(K+1)^2}\|x^0 - x^\star\|^2 + \frac{8\mathsf{G}^2}{\lambda(K+1)}.$$

Proof. As $\rho_s = 0$ and hence $\rho = 0$, we have that [26, Assum. B] is satisfied with $\tau = 0$ (in the notation of [26], see Proposition 4.3). Moreover, by Lemma 4.2, (i) and λ–strong convexity of φ, we have λ–strong convexity of $\psi_x(\cdot; s)$. The claim follows from Proposition 4.3 and [26, Thm. 4.5] setting $\mu = \lambda$, $T = K - 1$ and $\beta_t = \alpha_k^{-1}$. □

4.5.2 Lipschitz Smoothness

Assumption (4.12), i.e. having globally bounded subgradients, is strong: it implies Lipschitz continuity of f (cf. [26, Lem. 4.1]) and simple functions such as the squared loss do not satisfy this. Therefore, we provide additional guarantees for the smooth case, without the assumption of globally bounded gradients.

The following result, similar to [26, Lem. 4.2], is the basic inequality for the subsequent convergence analysis.

Lemma 4.6. *Let Assumption 9 hold. Let x^{k+1} be given by (4.8) and ψ_{x^k} be given in (4.7). For every $x \in \mathbb{R}^n$ it holds*

$$(1 + \alpha_k\lambda)\|x^{k+1} - x\|^2 \leq \|x^k - x\|^2 - \|x^{k+1} - x^k\|^2 + 2\alpha_k\big(\psi_{x^k}(x; S_k) - \psi_{x^k}(x^{k+1}; S_k)\big). \quad (4.13)$$

Moreover, it holds

$$\psi_{x^k}(x^{k+1}; S_k) \geq f(x^k; S_k) + \langle g_k, x^{k+1} - x^k\rangle + \varphi(x^{k+1}). \quad (4.14)$$

Proof. The objective of (4.8) is given by $\Psi_k(y) := \psi_{x^k}(y; S_k) + \frac{1}{2\alpha_k}\|y - x^k\|^2$. Using Lemma 4.2, (i) and λ-strong convexity of φ, $\Psi_k(y)$ is $(\lambda + \frac{1}{\alpha_k})$–strongly convex. As x^{k+1} is the minimizer of $\Psi_k(y)$, for all $x \in \mathbb{R}^n$ we have

$$\begin{aligned}
&\Psi_k(x) \geq \Psi_k(x^{k+1}) + \frac{1 + \alpha_k\lambda}{2\alpha_k}\|x^{k+1} - x\|^2 \iff \\
&(1 + \alpha_k\lambda)\|x^{k+1} - x\|^2 \leq \|x^k - x\|^2 - \|x^{k+1} - x^k\|^2 + 2\alpha_k\big(\psi_{x^k}(x; S_k) - \psi_{x^k}(x^{k+1}; S_k)\big).
\end{aligned}$$

Moreover, by definition of $f_x(y; s)$ in (4.7) we have

$$\psi_{x^k}(x^{k+1}; S_k) = f_{x^k}(x^{k+1}; S_k) + \varphi(x^{k+1}) \geq f(x^k; S_k) + \langle g_k, x^{k+1} - x^k\rangle + \varphi(x^{k+1}).$$

□

We will work in the setting of differentiable loss functions with bounded gradient noise.

Assumption 10. *The mapping $f(\cdot; s)$ is differentiable for all $s \in \mathcal{S}$ and there exists $\beta \geq 0$ such that*

$$\mathbb{E}\|\nabla f(x; S) - \nabla f(x)\|^2 \leq \beta \quad \text{for all } x \in \mathbb{R}^n. \quad (4.15)$$

The assumption of bounded gradient noise (4.15) (in the differentiable setting) is indeed a weaker assumption than (4.12) since $\mathbb{E}[\nabla f(x; S)] = \nabla f(x)$ and

$$\mathbb{E}\|\nabla f(x; S) - \nabla f(x)\|^2 \leq \beta \iff \mathbb{E}\|\nabla f(x; S)\|^2 \leq \|\nabla f(x)\|^2 + \beta.$$

Remark 5. *Assumption 10 (and the subsequent theorems) could be adapted to the case where $f(\cdot; s)$ is weakly convex but non-differentiable: for fixed $x \in \mathbb{R}^n$, due to [11, Prop. 2.2] and [26, Lem. 2.1] it holds*

$$\mathbb{E}[\partial f(x;S)] = \mathbb{E}\Big[\partial\big(f(x;S) + \frac{\rho_S}{2}\|x\|^2\big) - \rho_S x\Big] = \partial f(x) + \rho x - \mathbb{E}[\rho_S x] = \partial f(x),$$

where we used $\rho = \mathbb{E}[\rho_S]$. Hence, for $g_s \in \partial f(x;s)$ we have $\mathbb{E}[g_S] \in \partial f(x)$ and (4.15) is replaced by

$$\mathbb{E}\|g_S - \mathbb{E}[g_S]\|^2 \leq \beta \quad \text{for all } x \in \mathbb{R}^n.$$

However, as we will still require that f is Lipschitz-smooth, we present our results for the differentiable setting.

The proof of the subsequent theorems can be found in Section 4.8.2 and Section 4.8.3.

Theorem 4.7. *Let Assumption 9 and Assumption 10 hold. Let $f(\cdot; s)$ be convex for all $s \in \mathcal{S}$ and let f be L–smooth. Let $x^\star \in \arg\min_{x\in\mathbb{R}^n} \psi(x)$ and let $\theta > 1$. Let $\{x^k\}_{k=0,\dots,K}$ be generated by Algorithm 10 for step sizes $\alpha_k > 0$ such that*

$$\alpha_k \leq \frac{1 - 1/\theta}{L}. \tag{4.16}$$

Then, it holds

$$(1 + \alpha_k\lambda)\mathbb{E}\|x^{k+1} - x^\star\|^2 \leq \mathbb{E}\|x^k - x^\star\|^2 + 2\alpha_k\mathbb{E}[\psi(x^\star) - \psi(x^{k+1})] + \theta\beta\alpha_k^2. \tag{4.17}$$

Moreover, we have:

a) If $\lambda > 0$ and $\alpha_k = \frac{1}{\lambda(k+k_0)}$ with $k_0 \geq 1$ large enough such that (4.16) is fulfilled, then

$$\mathbb{E}\Big[\psi\Big(\tfrac{1}{K}\sum_{k=0}^{K-1} x^{k+1}\Big) - \psi(x^\star)\Big] \leq \frac{\lambda k_0}{2K}\|x^0 - x^\star\|^2 + \frac{\theta\beta(1 + \ln K)}{2\lambda K}. \tag{4.18}$$

b) If $\lambda = 0$ and $\alpha_k = \frac{\alpha}{\sqrt{k+1}}$ with $\alpha \leq \frac{1-1/\theta}{L}$, then

$$\mathbb{E}\Big[\psi\Big(\tfrac{1}{\sum_{k=0}^{K-1}\alpha_k}\sum_{k=0}^{K-1}\alpha_k x^{k+1}\Big) - \psi(x^\star)\Big] \leq \frac{\|x^0 - x^\star\|^2}{4\alpha(\sqrt{K+1} - 1)} + \frac{\theta\beta\alpha(1 + \ln K)}{4(\sqrt{K+1} - 1)}. \tag{4.19}$$

c) If f is μ–strongly convex with $\mu \geq 0$,[6] and $\alpha_k = \alpha$ fulfilling (4.16), then

$$\mathbb{E}\|x^K - x^\star\|^2 \leq (1 + \alpha(\mu + 2\lambda))^{-K}\|x^0 - x^\star\|^2 + \frac{\theta\beta\alpha}{\mu + 2\lambda}. \tag{4.20}$$

Remark 6. *If $\lambda > 0$, for the decaying step sizes in item a) we get a rate of $\tilde{\mathcal{O}}(\frac{1}{K})$ if $\lambda > 0$. In the strongly convex case in item c), for constant step sizes, we get a linear convergence upto a neighborhood of the solution. Note that the constant on the right-hand side of (4.20) can be forced to be small using a small α. Further, the rate (4.20) has a 2λ term, instead of λ. This slight improvement in the rate occurs because we do not linearize φ in the* `ProxSPS` *model.*

[6] Note that as $f(\cdot; s)$ is convex, so is f, and that we allow $\mu = 0$ here.

Theorem 4.8. *Let Assumption 9 and Assumption 10 hold. Let $f(\cdot; s)$ be ρ_s–weakly convex for all $s \in \mathcal{S}$ and let $\rho := \mathbb{E}[\rho_S] < \infty$. Let f be L–smooth[7] and assume that $\inf \psi > -\infty$. Let $\{x^k\}_{k\geq 0}$ be generated by Algorithm 10. For $\theta > 1$, under the condition*

$$\eta \in \begin{cases} (0, \frac{1}{\rho - \lambda}) & \text{if } \rho > \lambda \\ (0, \infty) & \text{else} \end{cases}, \qquad \alpha_k \leq \frac{1 - \theta^{-1}}{L + \eta^{-1}}, \tag{4.21}$$

it holds

$$\sum_{k=0}^{K-1} \alpha_k \mathbb{E}\|\nabla \mathrm{env}_\psi^\eta(x^k)\|^2 \leq \frac{2(\mathrm{env}_\psi^\eta(x^0) - \inf \psi)}{1 - \eta(\rho - \lambda)} + \frac{\beta\theta}{\eta(1 - \eta(\rho - \lambda))} \sum_{k=0}^{K-1} \alpha_k^2. \tag{4.22}$$

Moreover, for the choice $\alpha_k = \frac{\alpha}{\sqrt{k+1}}$ and with $\alpha \leq \frac{1-\theta^{-1}}{L+\eta^{-1}}$, we have

$$\min_{k=0,\ldots,K-1} \mathbb{E}\|\nabla \mathrm{env}_\psi^\eta(x^k)\|^2 \leq \frac{\mathrm{env}_\psi^\eta(x^0) - \inf \psi}{\alpha(1 - \eta(\rho - \lambda))(\sqrt{K+1} - 1)} + \frac{\beta\theta}{2\eta(1 - \eta(\rho - \lambda))} \frac{\alpha(1 + \ln K)}{(\sqrt{K+1} - 1)}.$$

If instead we choose $\alpha_k = \frac{\alpha}{\sqrt{K}}$ and with $\alpha \leq \sqrt{K}\frac{1-\theta^{-1}}{L+\eta^{-1}}$, we have

$$\mathbb{E}\|\nabla \mathrm{env}_\psi^\eta(x_\sim^K)\|^2 \leq \frac{2(\mathrm{env}_\psi^\eta(x^0) - \inf \psi)}{\alpha(1 - \eta(\rho - \lambda))\sqrt{K}} + \frac{\beta\theta}{\eta(1 - \eta(\rho - \lambda))} \frac{\alpha}{\sqrt{K}},$$

where $x_\sim^K$ is uniformly drawn from $\{x^0, \ldots, x^{K-1}\}$.

Comparison to Existing Theory. Recalling that Algorithm 9 is equivalent to $\texttt{SPS}_{\max}$ with $c = 1$ and $\gamma_b = \alpha_k$, we can apply Theorem 4.7 and Theorem 4.8 for the unregularized case where $\varphi = 0$ and hence obtain new theory for $\texttt{SPS}_{\max}$. We start by summarizing the main theoretical results for $\texttt{SPS}_{\max}$ given in [88, 109]: in the (ER) setting, consider the interpolation constant $\sigma^2 = \mathbb{E}[f(x^\star; S) - C(S)] = \frac{1}{N}\sum_{i=1}^N f_i(x^\star) - C(s_i)$. If f_i is L_i-smooth and convex, [109, Thm. 3.1] proves convergence to a neighborhood of the solution, i.e. the iterates $\{x^k\}$ of $\texttt{SPS}_{\max}$ satisfy

$$\mathbb{E}[f(\bar{x}^K) - f(x^\star)] \leq \frac{\|x^0 - x^\star\|^2}{\alpha K} + \frac{2\gamma_b\sigma^2}{\alpha}, \tag{4.23}$$

where $\bar{x}^K := \frac{1}{K}\sum_{k=0}^{K-1} x^k$, $\alpha := \min\{\frac{1}{2cL_{\max}}, \gamma_b\}$, and $L_{\max} := \max_{i\in[N]} L_i$.[8] For the nonconvex case, if f_i is L_i-smooth and under suitable assumptions on the gradient noise, [88, Thm. 3.8] states that, for constants c_1 and c_2, we have

$$\min_{k=1,\ldots,K} \mathbb{E}\|\nabla f(x^k)\|^2 \leq \frac{1}{c_1 K} + c_2. \tag{4.24}$$

The main advantage of these results is that γ_b can be held constant; furthermore in the convex setting (4.23), the choice of γ_b requires no knowledge of the smoothness constants L_i. For both results however, we can not directly conclude that the right-hand side goes to zero as $K \to \infty$ as there is an additional constant. Choosing γ_b sufficiently small does not immediately solve this as c_1, α and c_2 all go to zero as γ_b goes to zero.

[7] As f is ρ–weakly convex, this implies $\rho \leq L$.

[8] The theorem also handles the mini-batch case but, for simplicity, we state the result for sampling a single i_k in each iteration.

Our results complement this by showing exact convergence for the (weakly) convex case, i.e. without constants on the right-hand side. This comes at the cost of an upper bound on the step sizes α_k which depends on the smoothness constant L. For exact convergence, it is important to use decreasing step sizes α_k: Theorem 4.8 shows that the gradient of the Moreau envelope converges to zero at the rate $\mathcal{O}(1/\sqrt{K})$ for the choice of $\alpha_k = \frac{\alpha}{\sqrt{K}}$.[9] Another minor difference to [88] is that we do not need to assume Lipschitz-smoothness for all $f(\cdot; s)$ and work instead with the (more general) assumption of weak convexity. However, we still need to assume Lipschitz smoothness of f.

Another variant of $\texttt{SPS}_{\max}$, named `DecSPS`, has been proposed in [109]: for unregularized problems (4.1) it is given by

$$x^{k+1} = x^k - \hat{\gamma}_k g_k, \quad \hat{\gamma}_k = \frac{1}{c_k} \min\left\{ \frac{f(x^k; S_k) - C_k}{\|g_k\|^2}, c_{k-1}\hat{\gamma}_{k-1} \right\} \tag{DecSPS}$$

where $\{c_k\}_{k\geq 0}$ is an increasing sequence. In the (ER) setting, if all f_i are Lipschitz-smooth and strongly convex, `DecSPS` converges with a rate of $\mathcal{O}(\frac{1}{\sqrt{K}})$, without knowledge of the smoothness or convexity constants (cf. [109, Thm. 5.5]). However, under these assumptions, the objective f is strongly convex and the optimal rate is $\mathcal{O}(\frac{1}{K})$, which we achieve up to a logarithmic factor in Theorem 4.7, (4.18). Moreover, for `DecSPS` no guarantees are given for nonconvex problems.

For regularized problems, the constant in (4.23) is problematic if σ^2 (computed for the regularized loss) is moderately large. We refer to Section 4.9.3 where we show that this can easily happen. For `ProxSPS`, our theoretical results Theorem 4.7 and Theorem 4.8 are not affected by this as they do not depend on the size of σ^2. To the best of our knowledge, this is the first work to show theory for the stochastic Polyak step size in a setting that explicitly considers regularization. Moreover, our results also cover the case of non-smooth or non-real-valued regularization φ where the theory in [88] can not be applied.

4.6 Numerical Experiments

Throughout we denote Algorithm 9 by `SPS` and Algorithm 11 with `ProxSPS`. For all experiments we use `Pytorch` [113][10].

4.6.1 General Parameter Setting

For `SPS` and `ProxSPS` we always use $C(s) = 0$ for all $s \in \mathcal{S}$. For α_k, we use the following schedules:

- `constant`: set $\alpha_k = \alpha_0$ for all k and some $\alpha_0 > 0$.
- `sqrt`: set $\alpha_k = \frac{\alpha_0}{\sqrt{j}}$ for all iterations k during epoch j.

As we consider problems with ℓ_2-regularization, for `SPS` we handle the regularization term by incorporating it into all individual loss functions, as depicted in (4.10). With $\varphi = \frac{\lambda}{2}\|\cdot\|^2$ for $\lambda \geq 0$, we denote by ζ_k the *adaptive step size* term of the following algorithms:

[9] Notice that α_k then depends on the total number of iterations K and hence one would need to fix K before starting the method.

[10] The code for our experiments and an implementation of `ProxSPS` can be found at https://github.com/fabian-sp/ProxSPS.

- for SPS we have $\zeta_k := \frac{f(x^k;S_k)+\frac{\lambda}{2}\|x^k\|^2}{\|g_k+\lambda x^k\|^2}$ (cf. (4.10) with $\underline{\ell}_{i_k} = 0$),
- for ProxSPS we have $\zeta_k := \left(\frac{(1+\alpha_k\lambda)f(x^k;S_k)-\alpha_k\lambda\langle g_k,x^k\rangle}{\|g_k\|^2}\right)_+$ and thus $\tau_k^+ = \min\{\alpha_k, \zeta_k\}$ (cf. Lemma 4.1 with $C(S_k) = 0$).

4.6.2 Regularized Matrix Factorization

Problem description. For $A \in \mathbb{R}^{q\times p}$, consider the problem

$$\min_{W_1\in\mathbb{R}^{r\times p}, W_2\mathbb{R}^{q\times r}} \mathbb{E}_{y\sim N(0,I)}\|W_2W_1y - Ay\|^2 = \min_{W_1\in\mathbb{R}^{r\times p}, W_2\mathbb{R}^{q\times r}} \|W_2W_1 - A\|_F^2.$$

For the above problem, $\texttt{SPS}_{\max}$ has shown superior performance than other methods in the numerical experiments of [88]. The problem can can be turned into a (nonconvex) empirical risk minimization problem by drawing N samples $\{y^{(1)}, \dots, y^{(N)}\}$. Denote $b^{(i)} := Ay^{(i)}$. Adding squared norm regularization with $\lambda \geq 0$ (cf. [146]), we obtain the problem

$$\min_{W_1\in\mathbb{R}^{r\times p}, W_2\mathbb{R}^{q\times r}} \frac{1}{N}\sum_{i=1}^{N}\|W_2W_1y^{(i)} - b^{(i)}\|^2 + \tfrac{\lambda}{2}\big(\|W_1\|_F^2 + \|W_2\|_F^2\big). \tag{4.25}$$

This fits the format of (P), where $x = (W_1, W_2)$, using a finite sample space $\mathcal{S} = \{s_1, \dots, s_N\}$, $f(x; s_i) = \|W_2W_1y^{(i)} - Ay^{(i)}\|^2$, and $\varphi = \frac{\lambda}{2}\|\cdot\|_F^2$. Clearly, zero is a lower bound of $f(\cdot; s_i)$ for all $i \in [N]$. We investigate ProxSPS for problems of form (4.25) on synthetic data. For details on the experimental procedure, we refer to Section 4.9.1.

Discussion. Here, we discuss results for the setting matrix-fac1 in Table 4.1 in the Appendix. We first fix $\lambda = 0.001$ and consider the three methods SPS, ProxSPS and SGD. Fig. 4.2 shows the objective function over 50 epochs, for both step size schedules sqrt and constant, and several initial values α_0. For the constant schedule, we observe that ProxSPS converges quickly for all initial values while SPS is unstable. Note that for SGD we need to pick much smaller values for α_0 in order to avoid divergence (SGD diverges for large α_0). SPS for large α_0 is unstable, while for small α_0 we can expect similar performance to SGD (as γ_k is capped by $\alpha_k = \alpha_0$). However, in the regime of small α_0, convergence will be very slow. Hence, one of the main advantages of SPS, namely that its step size can be chosen constant and moderately large (compared to SGD), is not observed here. ProxSPS fixes this by admitting a larger range of initial step sizes, all of which result in fast convergence, and therefore is more robust than SGD and SPS with respect to the tuning of α_0.

For the sqrt schedule, we observe in Fig. 4.2 that SPS can be stabilized by reducing the values of α_k over the course of the iterations. However, for large α_0 we still see instability in the early iterations, whereas ProxSPS does not show this behaviour. We again observe that ProxSPS is less sensitive with respect to the choice of α_0 as compared to SGD. The empirical results also confirm our theoretical statement, showing exact convergence if α_k is decaying in the order of $1/\sqrt{k}$. From Fig. 4.3, we can make similar observations for the validation error, defined as $\frac{1}{N_{\text{val}}}\sum_{i=1}^{N_{\text{val}}}\|W_2W_1y^{(i)} - b_{\text{val}}^{(i)}\|^2$, where $b_{\text{val}}^{(i)}$ are the $N_{\text{val}} = N$ measurements from the validation set (cf. Section 4.9.1 for details).

We now consider different values for λ and only consider the sqrt schedule, as we have seen that for constant step sizes, SPS would not work for large step sizes and be almost identical to SGD for small step sizes. Fig. 4.4 shows the objective function and validation error. Again, we can observe that SPS is unstable for large initial values α_0 for all $\lambda \geq 10^{-4}$. On the other

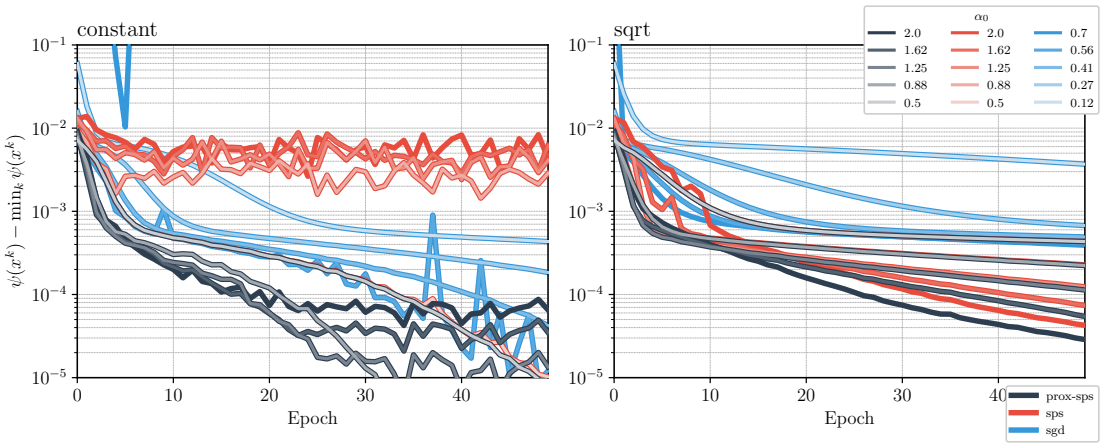


Figure 4.2: Objective function for the Matrix Factorization problem (4.25), with `constant` (left) and `sqrt` (right) step size schedule and several choices of initial values. Here $\min_k \psi(x^k)$ is the best objective function value found over all methods and all iterations.

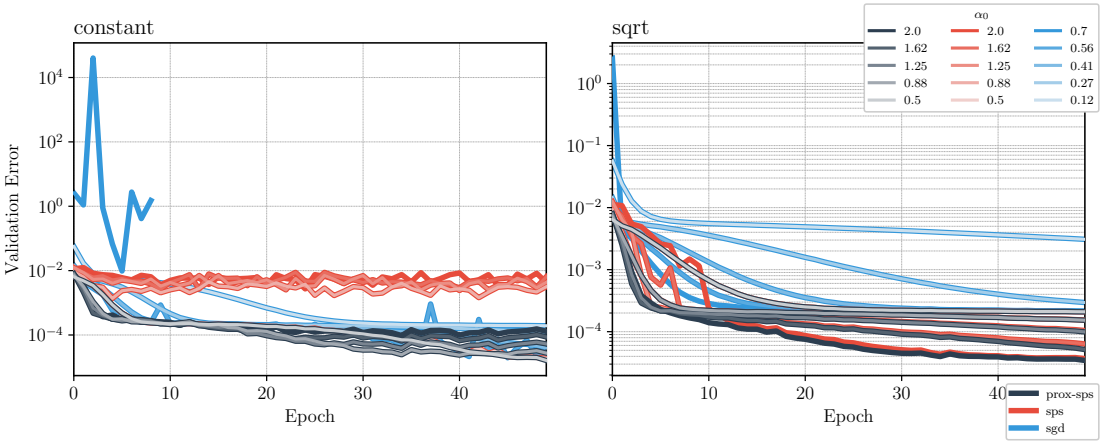


Figure 4.3: Validation error for the Matrix Factorization problem (4.25), with `constant` (left) and `sqrt` (right) step size schedule and several choices of initial values.

hand, `ProxSPS` has a good performance for a wide range of $\alpha_0 \in [1, 10]$ if λ is not too large. Indeed, `ProxSPS` convergence only starts to deteriorate when both α_0 and λ are very large. For $\alpha_0 = 1$, the two methods give almost identical results. Finally, in Fig. 4.5a we plot the validation error as a function of λ (taking the median over the last ten epochs). The plot shows that the best validation error is obtained for $\lambda = 10^{-4}$ and for large α_0. With `SPS` the validation error is higher, in particular for large α_0 and λ. Fig. 4.5b shows that `ProxSPS` leads to smaller norm of the iterates, hence a more effective regularization. Finally, we plot the actual step sizes for both methods in Fig. 4.6. We observe that the adaptive step size ζ_k (Definition at end of Section 4.6.1) is typically larger and has more variance for `SPS` than `ProxSPS`, in particular for large λ. This increased variance might explain why `SPS` is unstable when α_0 is large: the actual step size is the minimum between α_k and ζ_k and hence both terms being large could lead to instability. On the other hand, if $\alpha_0 = 1$, the plot confirms that `SPS` and `ProxSPS` are almost identical methods as $\zeta_k > \alpha_k$ for most iterations. In Section 4.9.1 of the supplementary material, we provide additional numerical results which confirm the above findings in the for the setting `matrix-fac2` of Table 4.1.

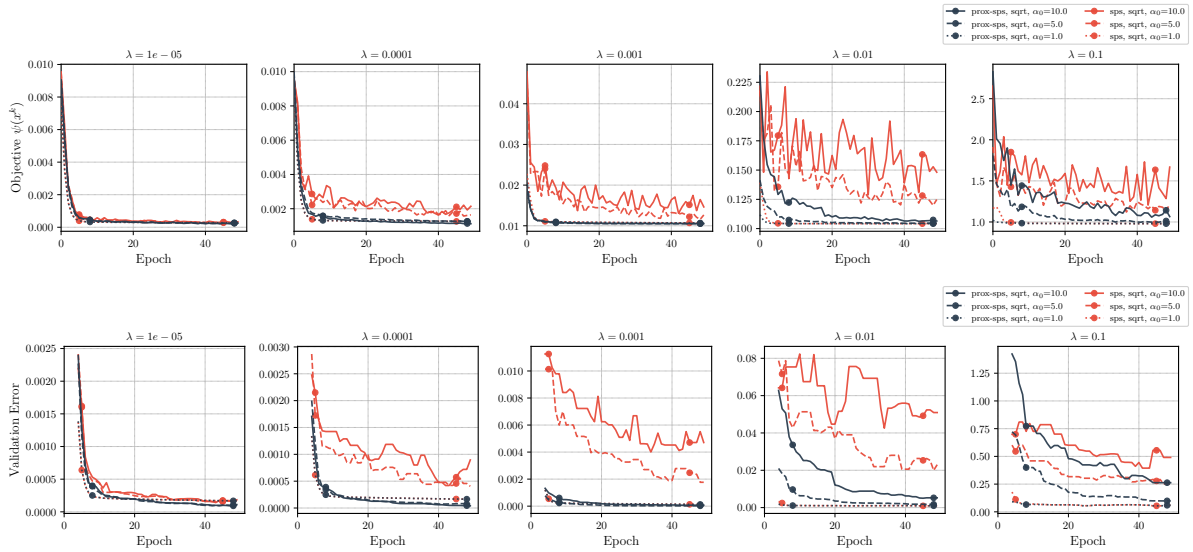


Figure 4.4: Objective function value and validation error over the course of optimization. For the validation error, we plot a rolling median over five epochs in order to avoid clutter.

4.6.3 Regularized Matrix Completion

Consider an unknown matrix of interest $W \in \mathbb{R}^{d_1 \times d_2}$. Factorizing $W \approx U^\top V$ with $U \in \mathbb{R}^{r \times d_1}$, $V \in \mathbb{R}^{r \times d_2}$, we can estimate the entries of matrix W as

$$\hat{W}_{ij} = u_i^\top v_j + b_i^U + b_j^V, \quad i \in [d_1],\ j \in [d_2], \tag{4.26}$$

where u_i is the i-th column of U and v_j is the j-th column of V, and $b^U \in \mathbb{R}^{d_1}$, $b^V \in \mathbb{R}^{d_2}$ are bias terms [121].

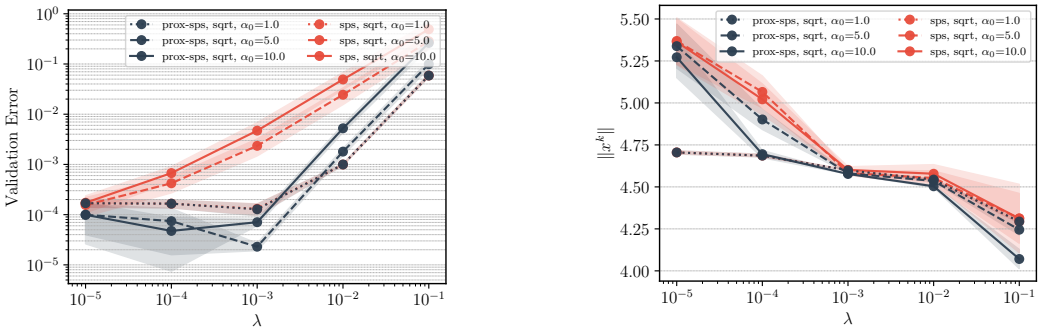
We can interpret this as an empirical risk minimization problem as follows: let $\mathcal{T}$ be the set of indices (i, j) where W_{ij} is known. With $\hat{W}_{ij}$ as in (4.26) for trainable parameters (U, V, b^U, b^V), the (regularized) problem is then given as

$$\min_{U,V,b^U,b^V} \frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} (W_{ij} - \hat{W}_{ij})^2 + \frac{\lambda}{2} \|(U, V, b^U, b^V)\|^2.$$

We use a dataset containing air quality measurements of a sensor network over one month. This dataset has been studied in [121].[11] The dataset contains measurements from 130 sensors over 720 timestamps, hence $d_1 = 130$, $d_2 = 720$. In total, there are 56158 nonzero measurements (the rest was missing data or removed due to being an outlier). We split the nonzero measurements into a training set of size $|\mathcal{T}| = 44926 \approx 0.8 \cdot 56158$ and the rest as a validation set. We standardize training and validation set using mean and variance of the training set. We set $r = 24$ and use batch size 128. The validation error is defined as the root mean squared error on the elements of the validation set (which is not used for training).

Discussion. The results are plotted in Fig. 4.7 and Fig. 4.16a. For all methods, we use a constant step size α_k. `ProxSPS` achieves the smallest error on the validation set for the two smaller values of λ. For the largest λ, `ProxSPS`, `SPS` and `SGD` are almost identical for $\alpha_0 = 5$,

[11] The dataset can be downloaded from https://github.com/andresgiraldo3312/DMF/blob/main/DatosEliminados/Ventana_Eli_mes1.csv.



(a) Validation error

(b) Model norm $\sqrt{\|W_1\|^2 + \|W_2\|^2}$

Figure 4.5: Validation error and model norm as a function of the regularization parameter λ. Shaded area is one standard deviation (computed over ten independent runs). For all values, we take the median over epochs $[40, 50]$.

but `SGD` with $\alpha_0 = 1$ is the best method. However, comparing all tested values of λ, Fig. 4.16a shows that `ProxSPS` obtains the smallest error. Again, from the lower plot in Fig. 4.7 we can observe that `ProxSPS` produces iterates with smaller norm.

4.6.4 Deep Networks for Image Classification

We train a `ResNet56` and `ResNet110` model [60] on the `CIFAR10` dataset. We use the data loading and preprocessing procedure and network implementation from `https://github.com/akamaster/pytorch_resnet_cifar10`. The loss function is the cross-entropy loss of the true image class with respect to the predicted class probabilities, being the output of the `ResNet56` network. We add $\frac{\lambda}{2}\|x\|^2$ as regularization term, where x consists of all learnable parameters of the model. We do not use batch normalization.[12] The `CIFAR10` dataset consists of 60,000 images, each of size 32×32, from ten different classes. We use the `Pytorch` split into 50,000 training and 10,000 test examples and use a batch size of 128. For `AdamW`, we set the weight decay parameter to λ and set all other hyperparameters to its default. We use the `AdamW`-implementation from `https://github.com/zhenxun-zhuang/AdamW-Scale-free` as it does not – in contrast to the `Pytorch` implementation – multiply the weight decay parameter with the learning rate, which leads to better comparability to `SPS` and `ProxSPS` for identical values of λ. For `SPS` and `ProxSPS` we use the `sqrt`-schedule and $\alpha_0 = 1$. We run each method repeatedly using (the same) three different seeds for the dataset shuffling.

Discussion. For `Resnet56`, from the bottom plot in Fig. 4.8, we observe that both `SPS` and `ProxSPS` work well with `ProxSPS` leading to smaller weights. For $\lambda = 5e-4$, the progress of `ProxSPS` stagnates after roughly 25 epochs. This can be explained by looking at the adaptive step size term ζ_k in Fig. 4.10a: as it decays over time we have $\tau_k^+ = \zeta_k \ll \alpha_k$. Since every iteration of `ProxSPS` shrinks the weights by a factor $\frac{1}{1+\alpha_k\lambda}$, this leads to a bias towards zero. This suggests that we should choose α_k roughly of the order of ζ_k, for example by using the values of ζ_k from the previous epoch.

[12] The reason for deactivating batch norm is that the interplay between weight decay/ℓ_2-regularization and batch norm is still poorly understood: the output of batch norm layers are by construction invariant to a scaling of the weights hence and regularization becomes ineffective [161]. Further, experiments in [169] show that for `CIFAR10`, it is often the case that with batch norm, the best test accuracy is reached for $\lambda = 0$, i.e. no regularization.

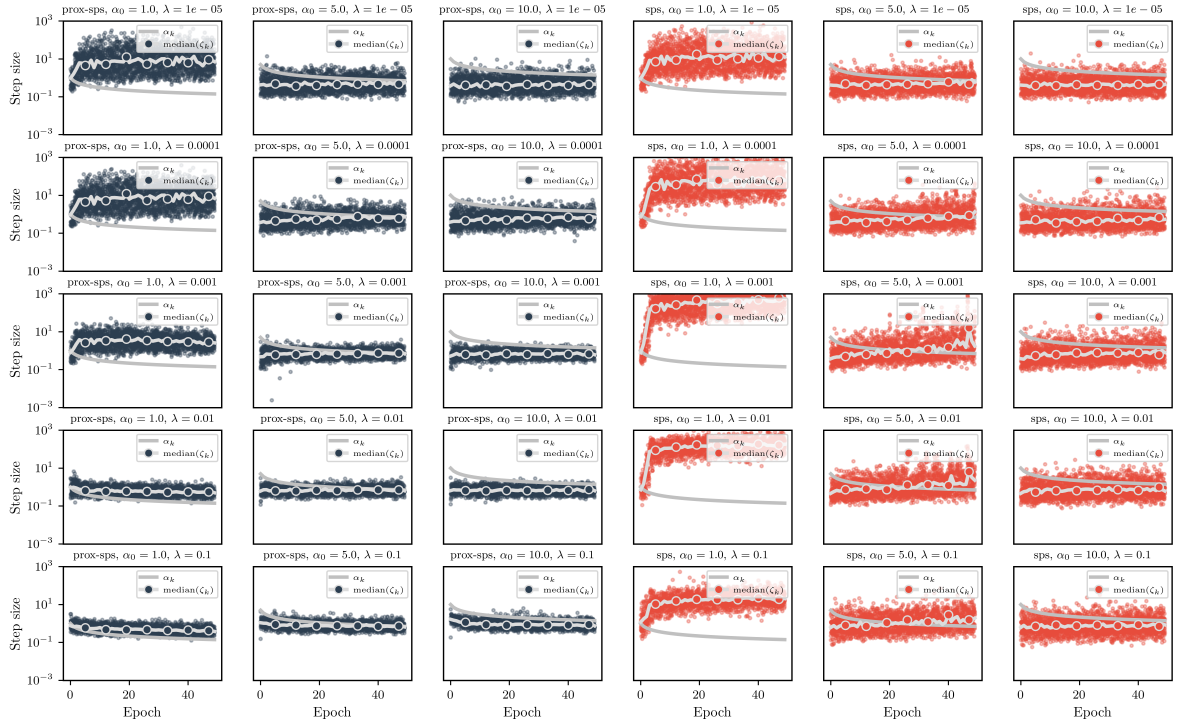


Figure 4.6: Adaptive step size selection for `SPS` and `ProxSPS`. We plot ζ_k (see definition in Section 4.6.1) as dots for each iteration as well as their median over each epoch. For this plot, we use the results of only one of the ten runs.

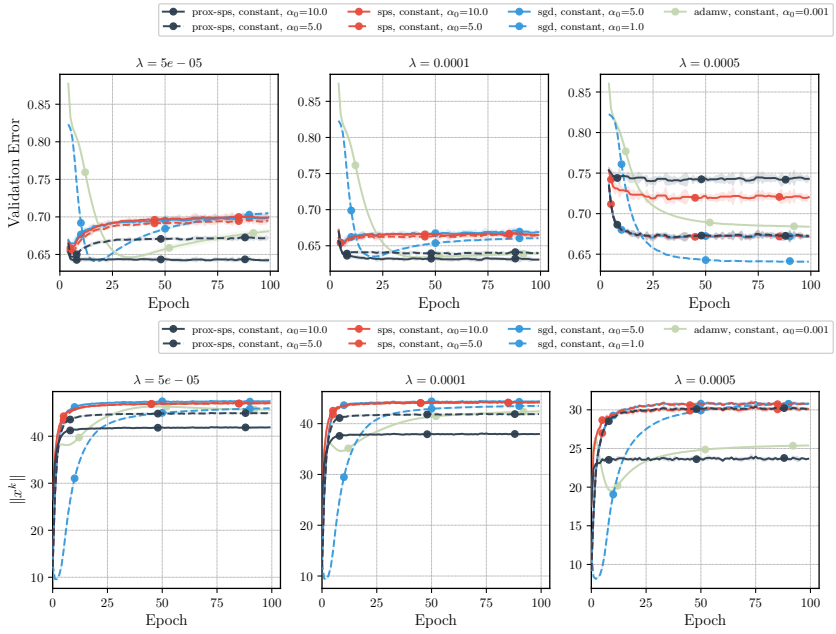


Figure 4.7: `Matrix Completion`: Validation error (top) and model norm (bottom) for three values of the regularization parameter λ. Validation error is plotted as five-epoch running median. Shaded area is two standard deviations over three independent runs.

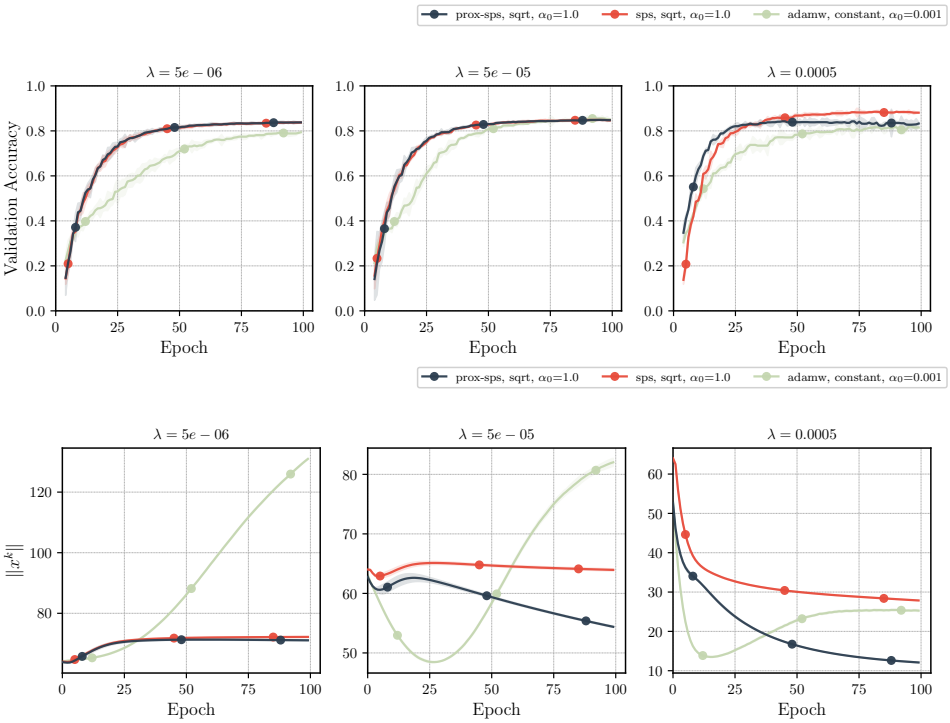


Figure 4.8: `ResNet56`: (Top): Validation accuracy and model norm for three values of the regularization parameter λ. Validation accuracy is defined as the ratio of correctly labeled images on the validation set (i.e. *Top-1 accuracy*), plotted as five-epoch running median. (Bottom): By $\|x^k\|$ we denote the norm of all learnable parameters at the k-th iteration. Shaded area is two standard deviations over three independent runs.

For the larger model `Resnet110` however, `SPS` does not make progress for a long time because the adaptive step size is very small (see Fig. 4.9 and Fig. 4.10b). `ProxSPS` does not share this issue and performs well after a few initial epochs. For larger values of λ, the training is also considerably faster than for `AdamW`. Generally, we observe that `ProxSPS` (and `SPS` for `Resnet56`) performs well in comparison to `AdamW`. This is achieved without extensive hyperparameter tuning (in particular this suggests that setting $c = 1$ in $\texttt{SPS}_{\max}$ leads to good results and reduces tuning).

In order to test the performance of `ProxSPS` on a larger dataset, we trained a `ResNet110` *with* batch norm on `Imagenet32`, containing over one million training images. The plots and experimental details can be found in Section 4.9.2. From Fig. 4.14, we conclude that `SPS` and `ProxSPS` perform equally well in this experiment. Both `SPS` and `ProxSPS` are less sensititve with respect to the regularization parameter λ than `AdamW` and the adaptive step size leads to faster learning in the initial epochs compared to `SGD`.

4.7 Conclusions and Open Questions

In this chapter, we propose and analyze `ProxSPS`, a proximal version of the stochastic Polyak step size. We arrive at `ProxSPS` by using the framework of stochastic model-based proximal point methods. We then use this framework to argue that the resulting model of `ProxSPS` is a better approximation as compared to the model used by `SPS` when using regularization. Our theoretical results cover a wide range of optimization problems, including convex and non-convex

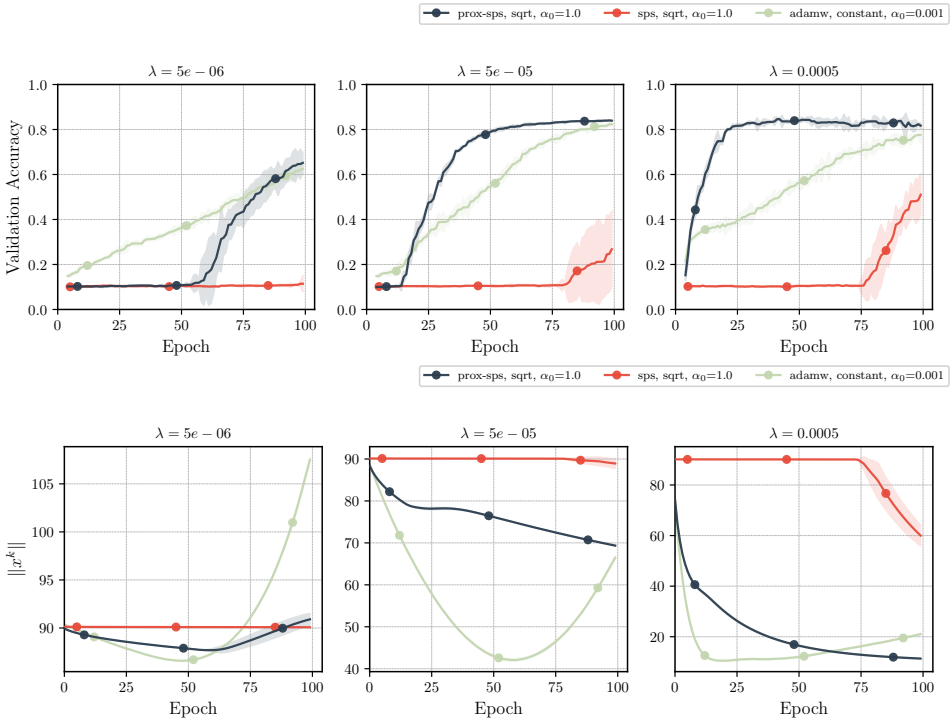
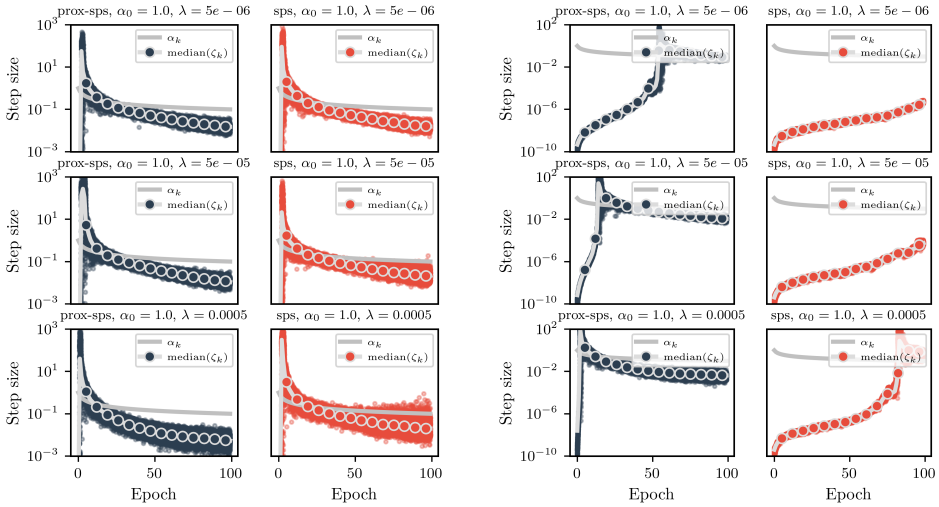


Figure 4.9: `ResNet110`: Validation accuracy as five-epoch running median (top) and model norm (bottom) for three values of λ. Shaded area is two standard deviations over three independent runs.



(a) `ResNet56` (b) `ResNet110`

Figure 4.10: Adaptive step sizes for `SPS` and `ProxSPS`. See definition of ζ_k in Section 4.6.1. For this plot, we use the results of only one of the three runs.

settings.

Extensive experiments for matrix factorization, matrix completion and image classification compare `ProxSPS`, `SPS`, `SGD` and `AdamW` when using ℓ_2-regularization. In particular, we find that `SPS` can be very hard to tune when using ℓ_2-regularization, and in contrast, `ProxSPS` performs well for a wide choice of step sizes and regularization parameters. Finally, for our experiments on image classification, we find that `ProxSPS` is competitive to `AdamW`, whereas `SPS` might collapse for larger models. At the same time `ProxSPS` produces smaller weights in the trained neural network. This may help to reduce the memory footprint of the resulting network, or suggest which weights can be pruned.

To end this chapter, we discuss two questions which are yet to be answered. First, we observe that in the theory of the Polyak step size for unregularized problems, the quantity $\sigma^2 = \mathbb{E}[f(x^\star; S) - C(S)]$ appears naturally. For regularized problems (or in a nonconvex setting), it might be better to use a different notion of interpolation. One natural option would be to look at the gradient variance, i.e. $\hat{\sigma}^2 := \mathbb{E}\|\nabla f(x^\star; S) - \nabla f(x^\star)\|^2$ (see [46, Def. 4.16]). In our current proofs, neither σ^2 nor $\hat{\sigma}^2$ appear; one can ask if it possible to improve our theory of `ProxSPS` for problems where $\hat{\sigma}$ is small or zero.

Second, in practice the most successful methods typically use momentum (e.g. `Adam` or `SGD` with heavy-ball momentum). At this point it is unclear (i) how to incorporate momentum in `ProxSPS`, and (ii) how to combine the Polyak step size with preconditioning techniques used for example in `Adam`. We will give an answer to both points in the next chapter.

4.8 Supplementary Material and Missing Proofs

4.8.1 Update Lemmas for the Truncated Model

We provide two technical lemmas which are crucial for deriving the central method of this (and the next) chapter. It should be remarked that during the work on the contents of this chapter, the proof for the update formula was slightly more direct (see [132, Lem. 9]). It turned out later that the update formula could be further generalized; this led to some of the results presented in Chapter 5.

Lemma 4.9. *Let $y_0, a \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Let $\beta > 0$. The solution to*

$$y^+ = \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \underbrace{\Big(c + \langle a, y - y_0\rangle\Big)_+}_{=:\phi(y)} + \frac{1}{2\beta}\|y - y_0\|^2 \tag{4.27}$$

is given by

$$y^+ = y_0 - \tau a, \quad \tau := \begin{cases} 0 & \text{if } a = 0, \\ \min\{\beta, \frac{(c)_+}{\|a\|^2}\} & \text{else.} \end{cases} \tag{4.28}$$

Moreover we have

$$\phi(y^+) = \begin{cases} c - \tau\|a\|^2, & \text{if } c \geq 0, \\ 0 & \text{else.} \end{cases} \tag{4.29}$$

Proof. Clearly, the objective of (4.27) is strongly convex and therefore there exists a unique solution, which can be characterized by the necessary and sufficient first-order optimality conditions

$$0 = ta + \beta^{-1}(y^+ - y_0), \quad t \in \partial(\cdot)_+(c + \langle a, y^+ - y_0\rangle). \tag{4.30}$$

The subdifferential $\partial(\cdot)_+(u)$ is given by $\{0\}$ if $u < 0$, $\{1\}$ if $u > 0$, and $[0, 1]$ if $u = 0$. First, consider $a = 0$. Then, it clearly holds $y^+ = y_0$ and $\phi(y^+) = (c)_+$ which shows (4.29).

Now, consider $a \neq 0$. We distinguish three cases:

(i) Suppose $c < 0$. Then, $y^+ = y_0$ satisfies (4.30) with $t = 0$. In this case, $(c)_+ = 0$ implies $\tau = 0$.

(ii) Suppose $\frac{c}{\|a\|^2} > \beta$. Let $\bar{y} := y_0 - \beta a$ and hence $c + \langle a, \bar{y} - y_0\rangle > 0 \iff c - \beta\|a\|^2 > 0 \iff \frac{c}{\|a\|^2} > \beta$. Then $y^+ = \bar{y}$ satisfies (4.30) with $t = 1$. As $\beta > 0$, hence $c > 0$ and $\tau = \beta$. Further, $\phi(y^+) = (c + \langle a, y^+ - y_0\rangle)_+ = (c - \beta\|a\|^2)_+ = c - \beta\|a\|^2$, equation (4.29) holds.

(iii) If neither $c < 0$ nor $\frac{c}{\|a\|^2} > \beta$ holds, then (4.30) can only be satisfied if $c + \langle a, y^+ - y_0\rangle = 0$ holds (i.e. $\phi(y^+) = 0$). Due to, $c - t\beta\|a\|^2 = c + \langle a, y^+ - y_0\rangle = 0 \iff t = \frac{c}{\beta\|a\|^2}$, we propose $\bar{y} = y_0 - t\beta a$ with $t = \frac{c}{\beta\|a\|^2}$ as a solution candidate. Indeed, as $c \geq 0$ we have $t \geq 0$ and $\frac{c}{\|a\|^2} \leq \beta$ implies $t \leq 1$. Hence, (4.30) is satisfied for $y^+ = \bar{y}$ and we have $\tau = \frac{c}{\|a\|^2}$. From $c - \tau\|a\|^2 = c - c = 0$, (4.29) holds as well.

As we have derived a solution for all possible cases, and the solution is unique, the proof is complete. □

Using the result above, we can prove a more general version.

Lemma 4.10. *Let $y_0,\ a \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Let $\beta > 0$ and $\lambda \geq 0$. Let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be a symmetric, positive definite matrix. The solution to*

$$y^+ = \operatorname*{arg\,min}_{y\in\mathbb{R}^n} \underbrace{\Big(c + \langle a, y - y_0\rangle\Big)_+}_{=:\phi(y)} + \frac{1}{2\beta}\|y - y_0\|^2_{\mathbf{D}} + \frac{\lambda}{2}\|y\|^2_{\mathbf{D}} \tag{4.31}$$

is given by

$$y^+ = \frac{1}{1+\lambda\beta}\big[y_0 - \tau\mathbf{D}^{-1}a\big], \quad \tau := \begin{cases} 0 & \text{if } a = 0, \\ \min\Big\{\beta, \frac{\big((1+\lambda\beta)c - \lambda\beta\langle a, y_0\rangle\big)_+}{\|a\|^2_{\mathbf{D}^{-1}}}\Big\} & \text{else.} \end{cases} \tag{4.32}$$

Furthermore, it holds

$$\phi(y^+) = \begin{cases} c - \frac{\lambda\beta}{1+\lambda\beta}\langle a, y_0\rangle - \frac{\tau}{1+\lambda\beta}\|a\|^2_{\mathbf{D}^{-1}} & \text{if } (1+\lambda\beta)c - \lambda\beta\langle a, y_0\rangle \geq 0, \\ 0 & \text{else.} \end{cases} \tag{4.33}$$

Proof. If $a = 0$, it is easy to see that the solution is given by $\frac{\mathbf{D}}{\beta}[(1+\lambda\beta)y^+ - y_0] = 0 \iff y^+ = \frac{1}{1+\lambda\beta}y_0$ and hence (4.32) holds. Further, clearly we have $\phi(y^+) = (c)_+$.

Now suppose $a \neq 0$. Denoting with $\text{cst.}(y)$ terms that are constant in y, we complete the squares as follows

$$\begin{aligned} \frac{\lambda}{2}\|y\|^2_{\mathbf{D}} + \frac{1}{2\beta}\|y - y_0\|^2_{\mathbf{D}} &= \frac{1}{2\beta}\|y\|^2_{(1+\lambda\beta)\mathbf{D}} - \frac{1}{\beta}\langle y, \mathbf{D}y_0\rangle + \text{cst.}(y) \\ &= \frac{1}{2\beta}\|y\|^2_{(1+\lambda\beta)\mathbf{D}} - \frac{1}{\beta}\langle y, (1+\lambda\beta)\mathbf{D}\tfrac{y_0}{1+\lambda\beta}\rangle + \text{cst.}(y) \\ &= \frac{1}{2\beta}\|y - \tfrac{1}{1+\lambda\beta}y_0\|^2_{(1+\lambda\beta)\mathbf{D}} + \text{cst.}(y), \end{aligned}$$

Using the above, (4.31) is equivalent to

$$\begin{aligned} y^+ &= \operatorname*{arg\,min}_{y\in\mathbb{R}^n} \phi(y) + \frac{1}{2\beta}\|y - \tfrac{1}{1+\lambda\beta}y_0\|^2_{(1+\lambda\beta)\mathbf{D}} \\ &= \operatorname*{arg\,min}_{y\in\mathbb{R}^n} \Big(c + \langle a, y - \tfrac{1}{1+\lambda\beta}y_0\rangle + \big(\tfrac{1}{1+\lambda\beta} - 1\big)\langle a, y_0\rangle\Big)_+ + \frac{1}{2\beta}\|y - \tfrac{1}{1+\lambda\beta}y_0\|^2_{(1+\lambda\beta)\mathbf{D}}. \end{aligned}$$

Let $\hat{c} := c + (\frac{1}{1+\lambda\beta} - 1)\langle a, y_0\rangle = c - \frac{\lambda\beta}{1+\lambda\beta}\langle a, y_0\rangle$. With this definition, (4.31) is equivalent to

$$y^+ = \operatorname*{arg\,min}_{y\in\mathbb{R}^n} \Big(\hat{c} + \langle a, y - \tfrac{1}{1+\lambda\beta}y_0\rangle\Big)_+ + \frac{1}{2\beta}\|y - \tfrac{1}{1+\lambda\beta}y_0\|^2_{(1+\lambda\beta)\mathbf{D}}.$$

Changing variables with $z^+ = \mathbf{D}^{1/2}y^+$, $z = \mathbf{D}^{1/2}y$, and $z_0 = \mathbf{D}^{1/2}y_0$ gives

$$z^+ = \operatorname*{arg\,min}_{z\in\mathbb{R}^n} \underbrace{\Big(\hat{c} + \langle \mathbf{D}^{-1/2}a, z - \tfrac{1}{1+\lambda\beta}z_0\rangle\Big)_+}_{=:\hat{\phi}(z)} + \frac{(1+\lambda\beta)}{2\beta}\|z - \tfrac{1}{1+\lambda\beta}z_0\|^2. \tag{4.34}$$

Applying Lemma 4.9 with $y_0 \leftarrow \frac{1}{1+\lambda\beta} z_0,\ c \leftarrow \hat{c},\ a \leftarrow \mathbf{D}^{-1/2}a, \beta \leftarrow \frac{\beta}{1+\lambda\beta}$ gives

$$z^+ = \frac{1}{1+\lambda\beta} z_0 - \underbrace{\min\left\{\frac{\beta}{1+\lambda\beta}, \frac{(\hat{c})_+}{\|a\|^2_{\mathbf{D}^{-1}}}\right\}}_{=:\hat{\tau}} \mathbf{D}^{-1/2}a.$$

Changing variables back using $y^+ = \mathbf{D}^{-1/2}z^+$, substituting $\hat{c} = c - \frac{\lambda\beta}{1+\lambda\beta}\langle a, y_0\rangle$ and re-arranging the above gives

$$\begin{aligned} y^+ &= \frac{1}{1+\lambda\beta} y_0 - \min\left\{\frac{\beta}{1+\lambda\beta}, \frac{\left(c - \frac{\lambda\beta}{1+\lambda\beta}\langle a, y_0\rangle\right)_+}{\|a\|^2_{\mathbf{D}^{-1}}}\right\}\mathbf{D}^{-1}a \\ &= \frac{1}{1+\lambda\beta}\left[y_0 - \min\left\{\beta, \frac{\left((1+\lambda\beta)c - \lambda\beta\langle a, y_0\rangle\right)_+}{\|a\|^2_{\mathbf{D}^{-1}}}\right\}\mathbf{D}^{-1}a\right]. \end{aligned}$$

Now, let us prove (4.33). First, simple calculations using the change of variable shows that

$$\begin{aligned} \hat{\phi}(z^+) &= \left(c - \tfrac{\lambda\beta}{1+\lambda\beta}\langle a, y_0\rangle + \langle a, y^+ - \tfrac{1}{1+\lambda\beta} y_0\rangle\right)_+ \\ &= \left(c + \langle a, y^+ - y_0\rangle\right)_+ = \phi(y^+). \end{aligned}$$

First, assume that $(1+\lambda\beta)c - \lambda\beta\langle a, y_0\rangle \geq 0 \iff \hat{c} \geq 0$. In this case, from applying Lemma 4.9 after (4.34), we get that

$$\phi(y^+) = \hat{\phi}(z^+) = \hat{c} - \hat{\tau}\|a\|^2_{\mathbf{D}^{-1}} = c - \tfrac{\lambda\beta}{1+\lambda\beta}\langle a, y_0\rangle - \tfrac{\tau}{1+\lambda\beta}\|a\|^2_{\mathbf{D}^{-1}}.$$

In the other case, when $\hat{c} < 0$, then $\phi(y^+) = \hat{\phi}(z^+) = 0$. □

4.8.2 Proof of Theorem 4.7

For the following proofs, denote by $\mathcal{F}_k$ the filtration that is generated by the history of all S_j for $j = 0, \ldots, k-1$.

Proof of Theorem 4.7. In the proof, we will denote $g_k = \nabla f(x^k; S_k)$. We apply Lemma 4.6, (4.13) with $x = x^\star$. Due to Lemma 4.2 (ii) and convexity of $f(\cdot; s)$ it holds

$$\psi_{x^k}(x^\star; S_k) \leq f(x^\star; S_k) + \varphi(x^\star).$$

Together with (4.14), we have

$$\begin{aligned} (1+\alpha_k\lambda)\|x^{k+1} - x^\star\|^2 \leq \|x^k - x^\star\|^2 - \|x^{k+1} - x^k\|^2 + 2\alpha_k[\varphi(x^\star) - \varphi(x^{k+1})] \\ + 2\alpha_k\big[f(x^\star; S_k) - f(x^k; S_k) - \langle g_k, x^{k+1} - x^k\rangle\big]. \end{aligned} \tag{4.35}$$

Smoothness of f yields

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

Consequently,

$$\begin{aligned} &-\langle g_k, x^{k+1} - x^k\rangle = f(x^k) - f(x^k) - \langle g_k, x^{k+1} - x^k\rangle \\ &\leq f(x^k) - f(x^{k+1}) + \langle \nabla f(x^k) - g_k, x^{k+1} - x^k\rangle + \tfrac{L}{2}\|x^{k+1} - x^k\|^2 \\ &\leq f(x^k) - f(x^{k+1}) + \frac{\theta\alpha_k}{2}\|\nabla f(x^k) - g_k\|^2 + \frac{1}{2\theta\alpha_k}\|x^{k+1} - x^k\|^2 + \tfrac{L}{2}\|x^{k+1} - x^k\|^2. \end{aligned}$$

for any $\theta > 0$, where we used Young's inequality in the last step. Plugging into (4.35) gives

$$(1+\alpha_k\lambda)\|x^{k+1}-x^\star\|^2 \le \|x^k-x^\star\|^2 + \big[\alpha_k L + \tfrac{1}{\theta} - 1\big]\|x^{k+1}-x^k\|^2 + 2\alpha_k[\varphi(x^\star)-\varphi(x^{k+1})] \\ + 2\alpha_k\big[f(x^\star;S_k) - f(x^k;S_k) + f(x^k) - f(x^{k+1})\big] + \theta\alpha_k^2\|\nabla f(x^k) - g_k\|^2.$$

Applying conditional expectation, we have $\mathbb{E}[f(x^\star;S_k)|\mathcal{F}_k] = f(x^\star)$ and

$$\mathbb{E}[-f(x^k;S_k) + f(x^k)|\mathcal{F}_k] = 0, \quad \mathbb{E}[\|\nabla f(x^k) - g_k\|^2|\mathcal{F}_k] \le \beta.$$

Moreover, by assumption, $\alpha_k L + \frac{1}{\theta} - 1 \le 0$. Altogether, applying total expectation yields

$$(1+\alpha_k\lambda)\mathbb{E}\|x^{k+1}-x^\star\|^2 \le \mathbb{E}\|x^k-x^\star\|^2 + 2\alpha_k\mathbb{E}[\psi(x^\star)-\psi(x^{k+1})] + \theta\beta\alpha_k^2$$

which proves (4.17).

Proof of a): let $\alpha_k = \frac{1}{\lambda(k+k_0)}$. Denote $\Delta_k := \mathbb{E}\|x^k - x^\star\|^2$. Rearranging and summing (4.17), we have

$$\sum_{k=0}^{K-1}\mathbb{E}[\psi(x^{k+1})-\psi(x^\star)] \le \sum_{k=0}^{K-1}\Big[\tfrac{1}{2\alpha_k}\Delta_k - \tfrac{1+\alpha_k\lambda}{2\alpha_k}\Delta_{k+1} + \tfrac{\theta\beta\alpha_k}{2}\Big].$$

Plugging in α_k, we have $\frac{1+\alpha_k\lambda}{2\alpha_k} = \frac{\lambda(k+k_0)}{2} + \frac{\lambda}{2}$ and thus

$$\sum_{k=0}^{K-1}\mathbb{E}[\psi(x^{k+1})-\psi(x^\star)] \le \sum_{k=0}^{K-1}\Big[\tfrac{\lambda(k+k_0)}{2}\Delta_k - \tfrac{\lambda(k+1+k_0)}{2}\Delta_{k+1}\Big] + \tfrac{\theta\beta}{2}\sum_{k=0}^{K-1}\tfrac{1}{\lambda(k+k_0)}.$$

Dividing by K and using convexity of ψ,[13] we have

$$\mathbb{E}\Big[\psi\Big(\tfrac{1}{K}\sum_{k=0}^{K-1}x^{k+1}\Big) - \psi(x^\star)\Big] \le \frac{\lambda k_0}{2K}\|x^0-x^\star\|^2 + \tfrac{\theta\beta}{2\lambda K}\sum_{k=0}^{K-1}\tfrac{1}{k+k_0}.$$

Finally, as $k_0 \ge 1$, we estimate $\sum_{k=0}^{K-1}\frac{1}{k+k_0} \le \sum_{k=0}^{K-1}\frac{1}{k+1} \le 1+\ln K$ by Lemma 4.13 and obtain (4.18).

Proof of b): Similar to the proof above, we rearrange and sum (4.17) from $k = 0,\dots,K-1$, and obtain

$$\sum_{k=0}^{K-1}\alpha_k\mathbb{E}[\psi(x^{k+1})-\psi(x^\star)] \le \frac{\|x^0-x^\star\|^2}{2} + \frac{\theta\beta\sum_{k=0}^{K-1}\alpha_k^2}{2}.$$

We divide by $\sum_{k=0}^{K-1}\alpha_k$ and use convexity of ψ in order to obtain the left-hand side of (4.19). Moreover, by Lemma 4.13 we have

$$\sum_{k=0}^{K-1}\alpha_k \ge 2\alpha(\sqrt{K+1}-1), \quad \sum_{k=0}^{K-1}\alpha_k^2 \le \alpha^2(1+\ln K).$$

Plugging in the above estimates, gives

$$\mathbb{E}\Big[\psi\Big(\tfrac{1}{\sum_{k=0}^{K-1}\alpha_k}\sum_{k=0}^{K-1}\alpha_k x^{k+1}\Big) - \psi(x^\star)\Big] \le \frac{\|x^0-x^\star\|^2}{4\alpha(\sqrt{K+1}-1)} + \frac{\theta\beta\alpha(1+\ln K)}{4(\sqrt{K+1}-1)}.$$

[13] By assumption f is convex and therefore ψ is convex.

Proof of c): If f is μ–strongly–convex, then ψ is $(\lambda+\mu)$–strongly convex and

$$\psi(x^\star) - \psi(x^{k+1}) \leq -\tfrac{\mu+\lambda}{2}\|x^{k+1} - x^\star\|^2.$$

From (4.17), with $\alpha_k = \alpha$, we get

$$(1+\alpha(\mu+2\lambda))\mathbb{E}\|x^{k+1} - x^\star\|^2 \leq \mathbb{E}\|x^k - x^\star\|^2 + \theta\beta\alpha^2.$$

Doing a recursion of the above from $k = 0, \ldots, K-1$ gives

$$\mathbb{E}\|x^K - x^\star\|^2 \leq (1+\alpha(\mu+2\lambda))^{-K}\|x^0 - x^\star\|^2 + \theta\beta\alpha^2 \sum_{k=1}^{K}(1+\alpha(\mu+2\lambda))^{-k}$$

Using the geometric series, $\sum_{k=1}^{K}(1+\alpha(\mu+2\lambda))^{-k} \leq \frac{1+\alpha(\mu+2\lambda)}{\alpha(\mu+2\lambda)} - 1 = \frac{1}{\alpha(\mu+2\lambda)}$, and thus

$$\mathbb{E}\|x^K - x^\star\|^2 \leq (1+\alpha(\mu+2\lambda))^{-K}\|x^0 - x^\star\|^2 + \frac{\theta\beta\alpha}{\mu+2\lambda}.$$

□

4.8.3 Proof of Theorem 4.8

Proof of Theorem 4.8. In the proof, we will denote $g_k = \nabla f(x^k; S_k)$. By assumption f is ρ-weakly convex and hence ψ is $(\rho-\lambda)$-weakly convex if $\rho > \lambda$ and convex if $\rho \leq \lambda$. Hence, $\hat{x}^k := \operatorname{prox}_{\eta\psi}(x^k)$ is well-defined for $\eta < 1/(\rho-\lambda)$ if $\rho > \lambda$ and for any $\eta > 0$ else. Note that $\hat{x}^k$ is $\mathcal{F}_k$–measurable. We apply Lemma 4.6, (4.13) with $x = \hat{x}^k$. Due to Lemma 4.2 (ii) it holds

$$\psi_{x^k}(\hat{x}^k; S_k) = f_{x^k}(\hat{x}^k; S_k) + \varphi(\hat{x}^k) \leq f(\hat{x}^k; S_k) + \tfrac{\rho_{S_k}}{2}\|\hat{x}^k - x^k\|^2 + \varphi(\hat{x}^k).$$

Together with (4.14), this gives

$$\begin{aligned}(1+\alpha_k\lambda)\|x^{k+1} - \hat{x}^k\|^2 \leq &(1+\alpha_k\rho_{S_k})\|x^k - \hat{x}^k\|^2 - \|x^{k+1} - x^k\|^2 \\ &+ 2\alpha_k\Big(\varphi(\hat{x}^k) - \varphi(x^{k+1}) + f(\hat{x}^k; S_k) - f(x^k; S_k) - \langle g_k, x^{k+1} - x^k\rangle\Big)\end{aligned}$$

Analogous to the proof of Theorem 4.7, due to Lipschitz smoothness, for all $\theta > 0$ we have

$$\begin{aligned}-f(x^k; S_k) - \langle g_k, x^{k+1} - x^k\rangle \leq &-f(x^k; S_k) + f(x^k) \\ &- f(x^{k+1}) + \tfrac{\theta\alpha_k}{2}\|\nabla f(x^k) - g_k\|^2 + \big[\tfrac{1}{2\theta\alpha_k} + \tfrac{L}{2}\big]\|x^{k+1} - x^k\|^2.\end{aligned}$$

Plugging in gives

$$\begin{aligned}(1+\alpha_k\lambda)\|x^{k+1} - \hat{x}^k\|^2 \leq &(1+\alpha_k\rho_{S_k})\|x^k - \hat{x}^k\|^2 + 2\alpha_k\Big(\varphi(\hat{x}^k) - \varphi(x^{k+1})\Big) \\ &+ 2\alpha_k\big(f(\hat{x}^k; S_k) - f(x^k; S_k) + f(x^k) - f(x^{k+1}) + \tfrac{\theta\alpha_k}{2}\|\nabla f(x^k) - g_k\|^2\big) \\ &+ \big[\tfrac{1}{\theta} + \alpha_k L - 1\big]\|x^{k+1} - x^k\|^2.\end{aligned}$$

It holds $\mathbb{E}[f(\hat{x}^k; S_k) - f(x^k; S_k)|\mathcal{F}_k] = f(\hat{x}^k) - f(x^k)$ and $\mathbb{E}[\psi(\hat{x}^k)|\mathcal{F}_k] = \psi(\hat{x}^k)$. By Assumption 10, we have $\mathbb{E}[\|g_k - \nabla f(x^k)\|^2|\mathcal{F}_k] \leq \beta$. Altogether, taking conditional expectation yields

$$\begin{aligned}(1+\alpha_k\lambda)\mathbb{E}[\|x^{k+1} - \hat{x}^k\|^2|\mathcal{F}_k] \leq &(1+\alpha_k\rho)\|x^k - \hat{x}^k\|^2 + 2\alpha_k\mathbb{E}\big[\psi(\hat{x}^k) - \psi(x^{k+1})|\mathcal{F}_k\big] \\ &+ \alpha_k^2\theta\beta + \big[\tfrac{1}{\theta} + \alpha_k L - 1\big]\mathbb{E}[\|x^{k+1} - x^k\|^2|\mathcal{F}_k].\end{aligned}$$

Next, the definition of the proximal operator implies that almost surely

$$\psi(\hat{x}^k) + \tfrac{1}{2\eta}\|\hat{x}^k - x^k\|^2 \leq \psi(x^{k+1}) + \tfrac{1}{2\eta}\|x^{k+1} - x^k\|^2,$$

and hence

$$\mathbb{E}\big[\psi(\hat{x}^k) - \psi(x^{k+1})|\mathcal{F}_k\big] \leq \mathbb{E}\big[\tfrac{1}{2\eta}\|x^{k+1} - x^k\|^2 - \tfrac{1}{2\eta}\|\hat{x}^k - x^k\|^2|\mathcal{F}_k\big].$$

Altogether, we have

$$\begin{aligned}(1+\alpha_k\lambda)\mathbb{E}[\|x^{k+1} - \hat{x}^k\|^2|\mathcal{F}_k] \leq (1+\alpha_k(\rho-\eta^{-1}))\|x^k - \hat{x}^k\|^2 \\ + \alpha_k^2\theta\beta + \big[\tfrac{1}{\theta} + \alpha_k L + \alpha_k\eta^{-1} - 1\big]\mathbb{E}[\|x^{k+1} - x^k\|^2|\mathcal{F}_k].\end{aligned}$$

From assumption (4.21), we can drop the last term. Now, we aim for a recursion in env_ψ^η. Using that

$$\frac{1+\alpha_k(\rho-\eta^{-1})}{1+\alpha_k\lambda} = \frac{1+\alpha_k\lambda-\alpha_k\lambda+\alpha_k(\rho-\eta^{-1})}{1+\alpha_k\lambda} = 1 + \frac{\alpha_k(\rho-\eta^{-1}-\lambda)}{1+\alpha_k\lambda} \leq 1+\alpha_k(\rho-\eta^{-1}-\lambda),$$

we get

$$\begin{aligned}\mathbb{E}[\mathrm{env}_\psi^\eta(x^{k+1})|\mathcal{F}_k] &\leq \mathbb{E}[\psi(\hat{x}^k) + \frac{1}{2\eta}\|x^{k+1} - \hat{x}^k\|^2|\mathcal{F}_k] \\ &\leq \underbrace{\psi(\hat{x}^k) + \frac{1}{2\eta}\|x^k - \hat{x}^k\|^2}_{=\mathrm{env}_\psi^\eta(x^k)} + \frac{1}{2\eta}\big[\alpha_k(\rho-\eta^{-1}-\lambda)\big]\|x^k - \hat{x}^k\|^2 + \frac{\alpha_k^2}{2\eta}\theta\beta.\end{aligned}$$

Now using $\|x^k - \hat{x}^k\| = \eta\|\nabla\mathrm{env}_\psi^\eta(x^k)\|$ we conclude

$$\mathbb{E}[\mathrm{env}_\psi^\eta(x^{k+1})|\mathcal{F}_k] \leq \mathrm{env}_\psi^\eta(x^k) + \frac{\eta}{2}\big[\alpha_k(\rho-\eta^{-1}-\lambda)\big]\|\nabla\mathrm{env}_\psi^\eta(x^k)\|^2 + \frac{\alpha_k^2}{2\eta}\theta\beta.$$

Due to (4.21), we have $\eta^{-1} + \lambda - \rho > 0$. Taking expectation and unfolding the recursion by summing over $k = 0, \ldots, K-1$, we get

$$\sum_{k=0}^{K-1} \tfrac{\alpha_k}{2}(1-\eta(\rho-\lambda))\mathbb{E}\|\nabla\mathrm{env}_\psi^\eta(x^k)\|^2 \leq \mathrm{env}_\psi^\eta(x^0) - \mathbb{E}[\mathrm{env}_\psi^\eta(x^K)] + \sum_{k=0}^{K-1}\frac{\alpha_k^2}{2\eta}\theta\beta.$$

Now using that $\mathrm{env}_\psi^\eta(x^K) \geq \inf\psi$ almost surely, we finally get

$$\sum_{k=0}^{K-1}\alpha_k\mathbb{E}\|\nabla\mathrm{env}_\psi^\eta(x^k)\|^2 \leq \frac{2(\mathrm{env}_\psi^\eta(x^0) - \inf\psi)}{1-\eta(\rho-\lambda)} + \frac{\beta\theta}{\eta(1-\eta(\rho-\lambda))}\sum_{k=0}^{K-1}\alpha_k^2, \tag{4.36}$$

which proves (4.22). Now choose $\alpha_k = \frac{\alpha}{\sqrt{k+1}}$ and divide (4.36) by $\sum_{k=0}^{K-1}\alpha_k$. Using Lemma 4.13 for $\sum_{k=0}^{K-1}\alpha_k$ and $\sum_{k=0}^{K-1}\alpha_k^2$, we have

$$\min_{k=0,\ldots,K-1}\mathbb{E}\|\nabla\mathrm{env}_\psi^\eta(x^k)\|^2 \leq \frac{\mathrm{env}_\psi^\eta(x^0) - \inf\psi}{\alpha(1-\eta(\rho-\lambda))(\sqrt{K+1}-1)} + \frac{\beta\theta}{2\eta(1-\eta(\rho-\lambda))}\frac{\alpha(1+\ln K)}{(\sqrt{K+1}-1)}.$$

Choosing $\alpha_k = \frac{\alpha}{\sqrt{K}}$ instead, we can identify the left-hand-side of (4.36) as $\alpha\sqrt{K}\mathbb{E}\|\nabla\mathrm{env}_\psi^\eta(x_\sim^K)\|^2$. Dividing by $\alpha\sqrt{K}$ and using $\sum_{k=0}^{K-1}\alpha_k^2 = \alpha^2$, we obtain

$$\mathbb{E}\|\nabla\mathrm{env}_\psi^\eta(x_\sim^K)\|^2 \leq \frac{2(\mathrm{env}_\psi^\eta(x^0) - \inf\psi)}{\alpha(1-\eta(\rho-\lambda))\sqrt{K}} + \frac{\beta\theta}{\eta(1-\eta(\rho-\lambda))}\frac{\alpha}{\sqrt{K}}.$$

□

4.8.4 Auxiliary Lemmas

Lemma 4.11. *Consider the model* $f_x(y;s) := \max\{f(x;s) + \langle g, y-x\rangle, C(s)\}$ *where* $g \in \partial f(x;s)$ *and* $C(s) \leq \inf_{z\in\mathbb{R}^n} f(z;s)$ *holds for all* $s \in \mathcal{S}$*. Then, update* (4.2) *is given as*

$$x^{k+1} = x^k - \gamma_k g_k, \quad \gamma_k = \begin{cases} 0 & \text{if } g_k = 0, \\ \min\left\{\alpha_k, \frac{f(x^k;S_k)-C(S_k)}{\|g_k\|^2}\right\} & \text{else.} \end{cases}$$

where $g_k \in \partial f(x^k; S_k)$*.*

Proof. We apply Lemma 4.1 with $\lambda = 0$. As $f(x^k;S_k) \geq C(S_k)$, we have that τ_k^+ (from Lemma 4.1) is equal to γ_k. □

Lemma 4.12. *Let* $c \in \mathbb{R}, a, x^0 \in \mathbb{R}^n$ *and* $\beta > 0$ *and let* $\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ *be proper, closed, convex. The solution to*

$$y^+ = \operatorname*{arg\,min}_{y\in\mathbb{R}^n} \quad \big(c + \langle a, y\rangle\big)_+ + \varphi(y) + \frac{1}{2\beta}\|y - x^0\|^2 \tag{4.37}$$

is given by

$$y^+ = \begin{cases} \operatorname{prox}_{\beta\varphi}(x^0 - \beta a), & \text{if } c + \langle a, \operatorname{prox}_{\beta\varphi}(x^0 - \beta a)\rangle > 0, \\ \operatorname{prox}_{\beta\varphi}(x^0), & \text{if } c + \langle a, \operatorname{prox}_{\beta\varphi}(x^0)\rangle < 0, \\ \operatorname{prox}_{\beta\varphi}(x^0 - \beta u a) & \text{else, for } u \in [0,1] \text{ s.th. } c + \langle a, \operatorname{prox}_{\beta\varphi}(x^0 - \beta u a)\rangle = 0. \end{cases} \tag{4.38}$$

Remark 7. *The first two conditions can not hold simultaneously due to uniqueness of the solution. If neither of the conditions of the first two cases are satisfied, we have to find the root of* $u \mapsto c + \langle a, \operatorname{prox}_{\beta\varphi}(x^0 - \beta u a)\rangle$ *for* $u \in [0,1]$*. Due to strong convexity of the objective in* (4.37)*, we know that there exists a root and hence* y^+ *can be found efficiently with bisection.*

Proof. The objective of (4.37) is strongly convex and hence there exists a unique solution. Due to [9, Thm. 3.63], y is the solution to (4.37) if and only if it satisfies first-order optimality, i.e.

$$\exists u \in \partial(\cdot)_+(c + \langle a, y\rangle) : \ 0 \in ua + \partial\varphi(y) + \frac{1}{\beta}(y - x^0). \tag{4.39}$$

Now, as $y = \operatorname{prox}_{\beta\varphi}(z) \iff 0 \in \partial\varphi(y) + \frac{1}{\beta}(y - z)$, it holds

$$\begin{aligned} (4.39) &\iff \exists u \in \partial(\cdot)_+(c + \langle a, y\rangle) : \ 0 \in \partial\varphi(y) + \frac{1}{\beta}(y - (x^0 - \beta u a)) \\ &\iff \exists u \in \partial(\cdot)_+(c + \langle a, y\rangle) : \ y = \operatorname{prox}_{\beta\varphi}(x^0 - \beta u a). \end{aligned}$$

We distinguish three cases:

1. Let $\bar{y} := \operatorname{prox}_{\beta\varphi}(x^0 - \beta a)$ and suppose that $c + \langle a, \bar{y}\rangle > 0$. Then $\partial(\cdot)_+(c + \langle a, \bar{y}\rangle) = \{1\}$ and hence $\bar{y}$ satisfies (4.39) with $u = 1$. Hence, $y^+ = \bar{y}$.
2. Let $\bar{y} := \operatorname{prox}_{\beta\varphi}(x^0)$ and suppose that $c + \langle a, \bar{y}\rangle < 0$. Then $\partial(\cdot)_+(c + \langle a, \bar{y}\rangle) = \{0\}$ and hence $\bar{y}$ satisfies (4.39) with $u = 0$. Hence, $y^+ = \bar{y}$.
3. If neither the condition of the first nor of the second case of (4.38) are satisfied, then, as (4.39) is a necessary condition for the solution y^+, it must hold $c + \langle a, y^+\rangle = 0$. Hence, there exists a $u \in \partial(\cdot)_+(c + \langle a, y^+\rangle) = [0,1]$ such that
$$c + \langle a, \operatorname{prox}_{\beta\varphi}(x^0 - u\beta a)\rangle = 0.$$

□

Lemma 4.13. *For any $K \geq 1$ it holds*

$$\sum_{k=0}^{K-1} \tfrac{1}{k+1} = 1 + \sum_{k=1}^{K-1} \tfrac{1}{k+1} \leq 1 + \int_0^{K-1} \tfrac{1}{s+1} ds = 1 + \ln K,$$
$$\sum_{k=0}^{K-1} \tfrac{1}{\sqrt{k+1}} \geq \int_0^{K} \tfrac{1}{\sqrt{s+1}} ds = 2\sqrt{K+1} - 2.$$

4.8.5 Model Equivalence for SGD and ℓ_2-regularization

In the unregularized case, the SGD update

$$x^{k+1} = x^k - \alpha_k g_k, \quad g_k \in \partial f(x^k; S_k),$$

can be seen as solving (4.2) with the model

$$f_x(y; s) = f(x; s) + \langle g, y - x \rangle, \quad g \in \partial f(x; s).$$

Now, consider again the regularized problem (P) with $\varphi(x) = \frac{\lambda}{2}\|x\|^2$ and update (4.5) .
On the one hand, the model $\psi_x(y; s) = f(x; s) + \varphi(x) + \langle g + \lambda x, y - x \rangle$ with $g \in \partial f(x; s)$ yields

$$x^{k+1} = x^k - \alpha_k(g_k + \lambda x^k) = (1 - \alpha_k\lambda)x^k - \alpha_k g_k. \tag{4.40}$$

On the other hand, the model $\psi_x(y; s) = f(x; s) + \langle g, y - x \rangle + \varphi(y)$ with $g \in \partial f(x; s)$ results in

$$x^{k+1} = \operatorname{prox}_{\alpha_k\varphi}(x^k - \alpha_k g_k) = \frac{1}{1 + \alpha_k\lambda}\big[x^k - \alpha_k g_k\big] = (1 - \frac{\alpha_k}{1 + \alpha_k\lambda}\lambda)x^k - \frac{\alpha_k}{1 + \alpha_k\lambda} g_k. \tag{4.41}$$

Running (4.40) with step sizes $\alpha_k = \beta_k$ is equivalent to running (4.41) with step sizes $\frac{\alpha_k}{1+\alpha_k\lambda} = \beta_k \iff \alpha_k = \frac{\beta_k}{1-\beta_k\lambda}$. In this sense, standard SGD can be seen to be equivalent to proximal SGD for ℓ_2–regularized problems.

4.9 Supplementary Material on Numerical Experiments

4.9.1 Matrix Factorization

Synthetic data generation. We consider the experimental setting of the deep matrix factorization experiments in [88], but with an additional regularization. We generate data in the following way: first sample $B \in \mathbb{R}^{q \times p}$ with uniform entries in the interval $[0, 1]$. Then choose $\upsilon \in \mathbb{R}$ (which will be our targeted inverse condition number) and compute $A = DB$ where D is a diagonal matrix with entries from 1 to υ (equidistant on a logarithmic scale)[14]. In order to investigate the impact of regularization, we generate a noise matrix E with uniform entries in $[-\varepsilon, \varepsilon]$ and set $\tilde{A} := A \odot (1 + E)$. We then sample $y^{(i)} \sim N(0, I)$ and compute the targets $b^{(i)} = \tilde{A}y^{(i)}$. A validation set of identical size is created by the same mechanism, but computing its targets, denoted by $b^{(i)}_{\text{val}}$, via the original matrix A instead of $\tilde{A}$. The validation set contains

Name	p	q	N	υ	r	ε
`matrix-fac1`	6	10	1000	`1e-5`	4	0
`matrix-fac2`	6	10	1000	`1e-5`	10	0.05

Table 4.1: Matrix factorization synthetic datasets.

$N_{\text{val}} = N$ samples.

Model and general setup. Problem (4.25) can be interpreted as a two-layer neural network without activation functions. We train the network using the squared distance of the model output and $b^{(i)}$ (averaged over a mini-batch) as the loss function. We run 50 epochs for different methods, step size schedules and values of λ. For each different instance, we do ten independent runs: each run has the identical training set and initialization of W_1 and W_2, but different shuffling of the training set and different samples $y^{(i)}$ for the validation set. In order to allow a fair comparison, all methods have identical train and validation sets across all runs. All metrics are averaged over the ten runs. We always use a batch size of 20.

Plots for `matrix-fac2`. We plot additional results for Matrix Factorization, namely for the setting `matrix-fac2` of Table 4.1, see Fig. 4.11, Fig. 4.12, and Fig. 4.13. The results are qualitatively very similar to the setting `matrix-fac1`.

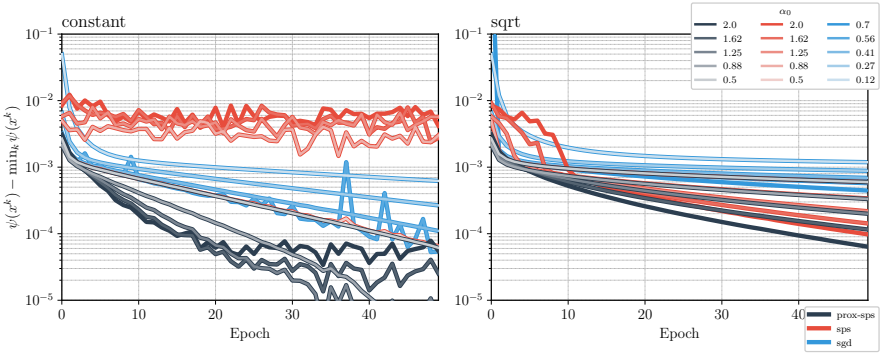


Figure 4.11: Objective function for the Matrix Factorization problem (4.25), with `constant` (left) and `sqrt` (right) step size schedule and several choices of initial values. Here $\min_k \psi(x^k)$ is the best objective function value found over all methods and all iterations.

[14]Note that [88] uses entries from 1 to υ on a *linear* scale which, in our experiments, did not result in large condition numbers even if υ is very small.

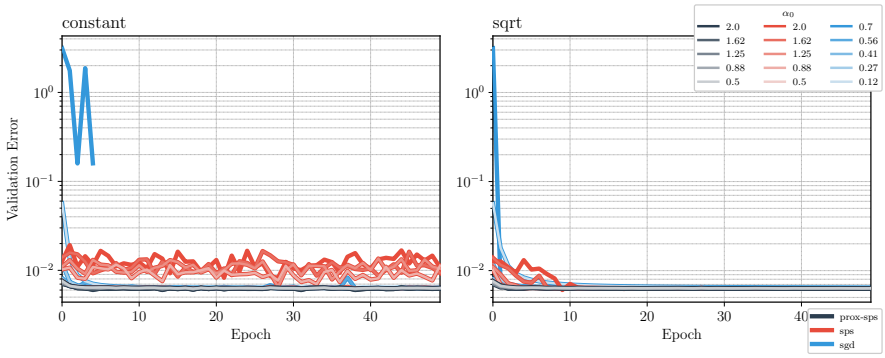


Figure 4.12: Validation error for the Matrix Factorization problem (4.25), with `constant` (left) and `sqrt` (right) step size schedule and several choices of initial values.

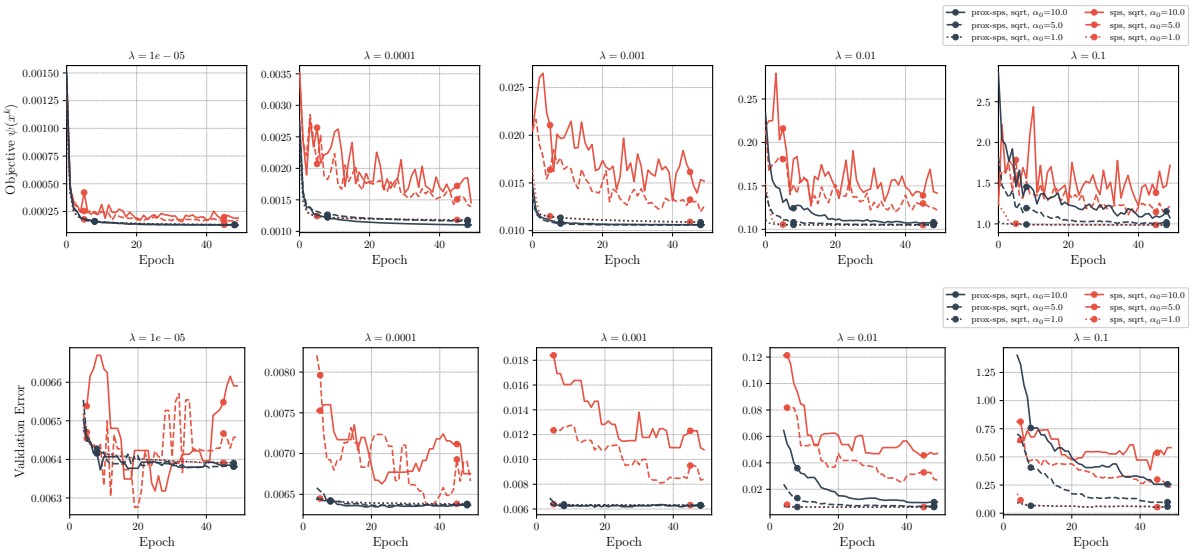


Figure 4.13: Objective function value and validation error over the course of optimization. For the validation error, we plot a rolling median over five epochs in order to avoid clutter.

4.9.2 `Imagenet32` Experiment

`Imagenet32` contains 1,28 million training and 50,000 test images of size 32×32, from 1,000 classes. We train the same `ResNet110` as described in Section 4.6.4 with two differences: we exchange the output dimension of the final layer to 1,000 and activate batch norm. We use batch size 512. For this experiment we only run one repetition, simply due to the fact that a single run takes almost one day to run on a GPU.

Similar to the setup in Section 4.6.4, we run all methods for three different values of λ. For `AdamW`, we use a constant learning rate 0.001, for `SGD`, `SPS`, and `ProxSPS` we use the `sqrt`-schedule and $\alpha_0 = 1$. The validation accuracy and model norm are plotted in Fig. 4.14: we can observe that all methods perform similarly well in terms of accuracy. However, `AdamW` is more sensitive with respect to the choice of λ and the norm of its iterates differs significantly from the other methods. Further, using an adaptive step size is advantageous: from Fig. 4.15, we see that the adaptive step size is active in the initial iterations, which leads to a faster learning of `(Prox)SPS` in the initial epochs compared to `SGD`.

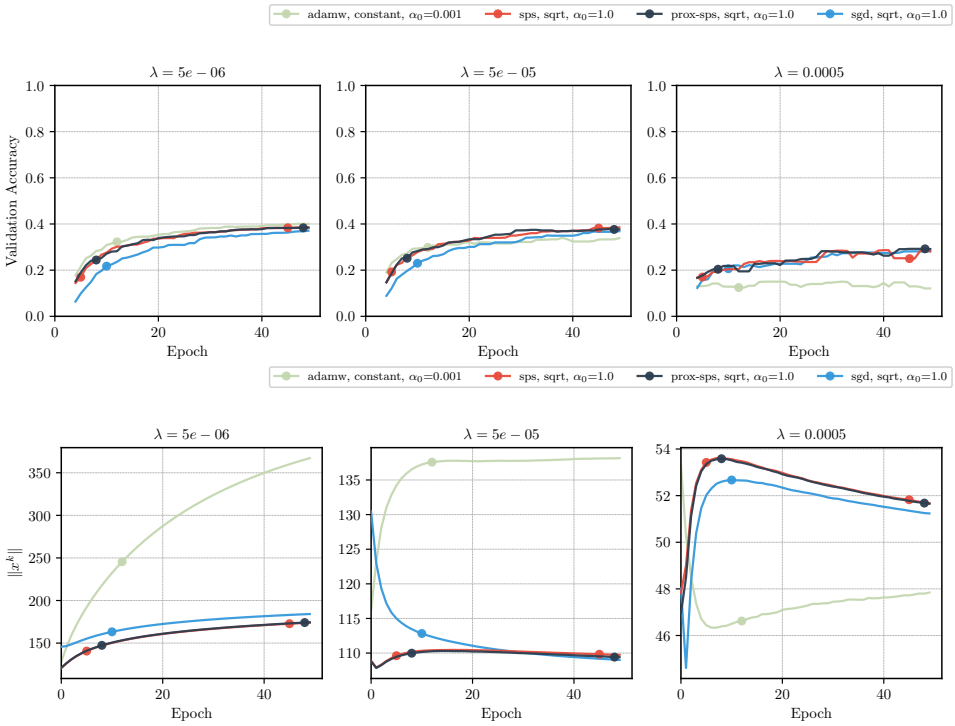


Figure 4.14: `ResNet110` for `Imagenet32`: Validation accuracy as five-epoch running median (top) and model norm (bottom) for three values of λ.

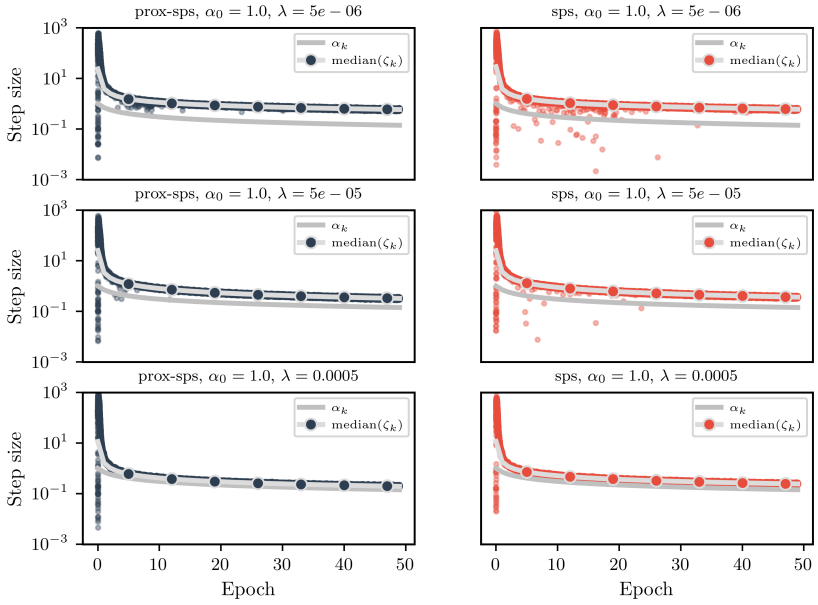
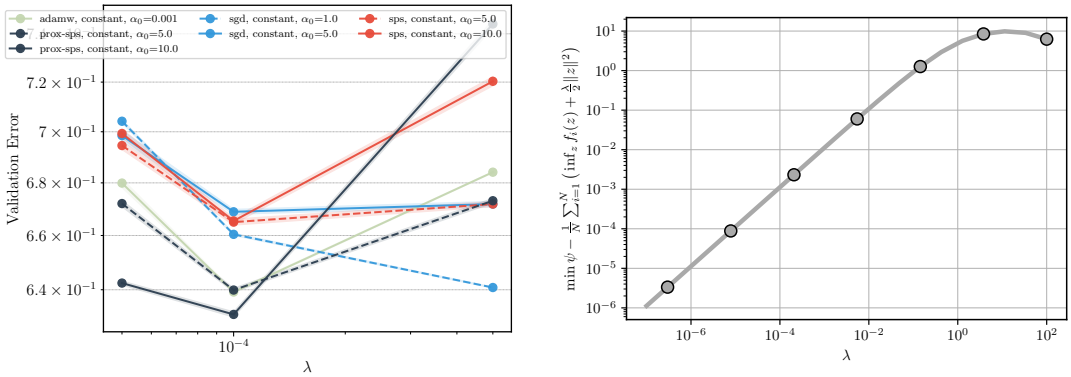


Figure 4.15: `ResNet110` for `Imagenet32`: Adaptive step sizes for `SPS` and `ProxSPS`.



(a) (b)

Figure 4.16: (a) `Matrix Completion`: Validation error as a function of the regularization parameter λ. Shaded area is one standard deviation (computed over three independent runs). For all values, we take the median over epochs $[90, 100]$. (b) Interpolation constant for a ridge regression problem for varying regularization parameter λ. See Section 4.9.3 for details.

4.9.3 Interpolation Constant

We illustrate how the interpolation constant σ^2 behaves if it would be computed for the regularized loss $\ell_i(x) = f_i(x) + \frac{\lambda}{2}\|x\|^2$ (cf. also Section 4.4.2). We do a simple ridge regression experiment. Let $A \in \mathbb{R}^{N\times n}$ be a matrix with row vectors $a_i \in \mathbb{R}^n$, $i \in [N]$. We set $N = 80$, $n = 100$ and generate $\hat{x} \in \mathbb{R}^n$ with entries drawn uniformly from $[0, 1]$. We compute $b = A\hat{x}$. In this case, we have $f_i(x) = \frac{1}{2}(a_i^\top x - b_i)^2$ and $f(x) = \frac{1}{N}\sum_{i=1}^N f_i(x)$.

If one would apply the theory of $\texttt{SPS}_{\max}$ for the regularized loss functions ℓ_i with estimates $\underline{\ell}_i = 0$, the constant $\sigma^2 = \big(\min_{x\in\mathbb{R}^n} f(x) + \varphi(x)\big) - \frac{1}{N}\sum_{i=1}^N \inf_z \ell_i(z)$ determines the size of the constant term in the convergence results of [88, 109]. We compute $\min_{x\in\mathbb{R}^n} f(x)+\varphi(x)$ by solving the ridge regression problem. Further, the minimizer of ℓ_i is given by $(a_i a_i^\top + \lambda \mathbf{Id})^{-1} a_i b_i$. We plot σ^2 for varying λ in Fig. 4.16b to verify that σ^2 grows significantly if λ becomes large (even if the loss could be interpolated perfectly, i.e. $\inf_x f(x) = 0$). We point out that the constant σ^2 does not appear in our convergence results Theorem 4.7 and Theorem 4.8, and hence these results are not negatively affected when σ^2 is large.

Chapter 5

Momentum Models for Adaptive Learning Rates

The chapter is mainly based on the article

[134] F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower, *MoMo: Momentum Models for Adaptive Learning Rates*, 41st International Conference on Machine Learning, 2024, https://proceedings.mlr.press/v235/schaipp24a.html.

The original idea for this project emerged during a meeting with Robert Gower and Aaron Defazio during the author's research visit at the Flatiron Institute, New York, in fall 2022. The experimental results presented below have been a truly collaborative effort: the `Imagenet32` experiments were mainly run by Michael Eickenberg, the German-to-English translation experiment was conducted by Ruben Ohana and the `DLRM` results are due to Aaron Defazio. The idea for the online lower bound estimation was initially proposed by Robert M. Gower.

5.1 Introduction

The computational cost of training machine learning models can be attributed to a large portion to hyperparameter tuning [25]. This is because for a new task, for example on an unknown dataset and/or architecture, many different training runs need to be conducted in order to determine the best hyperparameters for model architecture, loss function or data processing. However, for each of those runs, the optimal learning rate might be different. Selecting the best learning rate (schedule) for each setting individually will often exceed the computational/time budget. As a consequence, from a practitioners perspective, an optimizer that performs reasonably well in default settings across many different problems is appealing.

The goal of this chapter is to develop methods which are superior when it comes to out-of-the-box performance (or having only small tuning budgets). We again consider problems of the form (P) with $\varphi = 0$, given by

$$\min_{x \in \mathbb{R}^n} f(x), \quad f(x) := \mathbb{E}[f(x; S)], \tag{5.1}$$

where x denotes the learnable parameters, $S \in \mathcal{S}$ represents the distribution of input data, and $f(x; s)$ is the loss function for $s \in \mathcal{S}$. For a formal description of this setup we refer to Section 2.2.

Chapter 4 gave a glimpse on how to tackle the fundamental problem of learning rate tuning: our philosophy is to design adaptive learning-rate methods which makes use of the specific problem structure at hand, that is, minimizing the expectation of non-negative loss functions. The resulting method `ProxSPS`, or `SPS` for the unregularized case, can be summarized as *linearize and truncate* in order to obtain a model of $f(x; s)$. However, since our objective is to minimize $f(x) = \mathbb{E}[f(x; S)]$, can we construct a model directly for $f(x)$, and not $f(x; s)$? That is, in each iteration $k \in \mathbb{N}$, select a model m_k and step size $\alpha_k > 0$, and update

$$x^{k+1} = \arg\min_{x \in \mathbb{R}^n} m_k(x) + \frac{1}{2\alpha_k}\|x - x^k\|^2. \tag{5.2}$$

The model m_k needs to satisfy two properties: first, it should be a good approximation of the loss $f(x)$, at least locally around x^k. Second, the update (5.2) should be easy to compute, ideally in closed form.

The model-based stochastic optimization literature [6, 26] mainly focuses on models which use only the last sample S_k (cf. Section 2.7). One approach beyond this, is to use the pointwise maximum of past linearizations as a model of $f(x)$. In iteration $k \in \mathbb{N}$, this would lead to

$$m_k(x) = \max_{j=1,\dots,k} \left\{ f(x^j; S_j) + \langle \nabla f(x^j; S_j), x - x^j \rangle \right\},$$

which is resembling a bundle method [111][1]. It has the advantage that it uses past information to better approximate the loss, at the cost of update (5.2) not being computable in closed form. In fact, [111] proposes to solve (5.2) iteratively via a dual formulation.

In this chapter, we take a different route, proposing a model m_k which uses loss values and gradients from the entire history of iterations, *and* where (5.2) can be computed in closed form. One of the main insights is to use a *weighted average* of linearizations, followed by truncation. This will lead to the central method of this chapter, called `MoMo`.

Problem setup. We assume throughout that $f(x; s) \geq 0$ for all $x \in \mathbb{R}^n$, $s \in \mathcal{S}$, which is the case for most loss functions[2]. We also assume that $f(\cdot; s)$ is a continuously differentiable function for all $s \in \mathcal{S}$, that there exists a solution $x^\star$ to (5.1) and denote the optimal value by $f^\star := f(x^\star) \in \mathbb{R}$.

Remark 8. *For what follows, in particular the derivation of the `MoMo` methods, the assumption on differentiability is not essential. We could instead assume $f(\cdot; s)$ to be locally Lipschitz and use an element of the Clarke subdifferential instead (as we did in Chapter 4). We choose to make the assumption on differentiability merely to keep notation simple because the contents of this chapter are focusing more on application aspects.*

5.2 Background and Contributions

Momentum and model-based methods. The update formula of many stochastic methods such as `SGD` can be interpreted by taking a proximal step with respect to a model of the objective function (cf. Section 2.7 and [6, 26]). Independently of this, (heavy-ball) momentum [116, 139] is incorporated into many methods in order to boost performance.

[1] For simplicity we take the sum over $1, \dots, k$ but [111] is more general in using arbitrary bundles.

[2] We choose zero as a lower bound for simplicity. Our methods could handle any constant lower bound similarly to Chapter 4.

Contributions. Here we give a new interpretation of momentum, namely that it can be motivated as a model of the objective function $f(x)$ by averaging sampled loss functions. This allows us to naturally combine momentum with other model-based techniques.

Lower bounds and truncated models. One of the main advantages of the model-based viewpoint [6, 26] is that it illustrates how to use knowledge of a lower bound of the function via truncation. Methods using this truncated model are often easier to tune [6, 95, 132].

Contributions. By combining the model-based viewpoint of momentum with a truncated model we arrive at our new `MoMo` method. Since we are interested in loss functions, we can use zero as a lower bound estimate in many learning tasks. However, for some tasks such as training transformers, the minimal loss is often non-zero. If the non-zero lower bound is known, we can straightforwardly incorporate it into our model. For unknown lower bound values we also develop new online estimates of a lower bound in Section 5.6. Our estimates can be applied to any stochastic momentum-based method, and thus may be of independent interest. Our main influence for this technique was D-adaptation [28] which develops an online estimate of the distance to the solution which is used to set the learning rate.

Adaptive methods. In practice, tuning the learning rate is intricate and computationally expensive. `Adam` [74] (or `AdamW` [89]) is often easier to tune and now being used routinely to train DNNs across a variety of tasks. This success of `Adam` has incentivised the development of many new (adaptive) learning-rate techniques, including approaches based on coin-betting [107, 108], variants of `AdaGrad` [28, 36], and stochastic line search [155]. Recent work also combines parameter-free coin betting methods with truncated models [19].

Contributions. Our new adaptive learning rate can be combined with any momentum based method, and even allows for a preconditioner to be used. For example, `Adam` is a momentum method that makes use of a preconditioner (cf. Section 2.6). By using this viewpoint, together with a lower bound, we derive `MoMo-Adam`, which is a variant of `Adam` that uses our adaptive learning rates.

Polyak step sizes. For convex, non-smooth optimization, Polyak proposed an adaptive step size using the current objective function value $f(x^k)$ and the optimal value $f^\star$ [117]. Recently, the Polyak step size has been adapted to the stochastic setting [10, 51, 88, 109]. For example, we have previously introduced $\texttt{SPS}_{\max}$ [88], given by

$$x^{k+1} = x^k - \min\left\{\gamma_b, \frac{f(x^k; S_k) - C(S_k)}{c\|\nabla f(x^k; S_k)\|^2}\right\}\nabla f(x^k; S_k),$$

where $c, \gamma_b > 0$ and $C(S_k) \leq \inf_z f(z; S_k)$. The stochastic Polyak step size is closely related to stochastic model-based proximal point methods (see Chapter 4 and [6, 111]).

Contributions. Our proposed method `MoMo` can be seen as an extension of the Polyak step size that also incorporates momentum. This follows from the viewpoint of the Polyak step size [111, 132] as a truncated model-based method. In particular, `MoMo` with no momentum is equal to `SPS` (cf. Algorithm 9). It has been previously an open question how to ideally combine momentum and Polyak steps: for instance, [7] shows that knowledge of $f^\star$ can replace knowledge of the strong convexity parameter for choosing the momentum parameter. Concurrently to the work presented in this chapter, [156] proposed a momentum version of Polyak's step size, effectively replacing the stochastic gradient by the momentum gradient in $\texttt{SPS}_{\max}$.

Numerical findings. We find that `MoMo` consistently improves the sensitivity with respect to hyperparameter choice as compared to `SGD-M` (`SGD` with momentum). The same is true for `MoMo-Adam` compared to `Adam`. Our experiments cover image classification tasks on `MNIST`, `CIFAR10`, `CIFAR100` and `Imagenet`, a recommender system on the `Criteo` dataset, an encoder-decoder transformer for the translation task `IWSLT14`, and a diffusion model.

Furthermore, we find that the adaptive learning rate of `MoMo`(`-Adam`) for some tasks automatically performs a warm-up at the beginning of training and a decay in later iterations, two techniques often used in order to improve training [147].

5.3 Model-Based Momentum Methods

Let us recall that the `SGD` update is (5.2) with $m_k(x) = f(x^k; S_k) + \langle \nabla f(x^k; S_k), x - x^k \rangle$. This has two issues: first, it approximates a single stochastic function $f(x; S_k)$, as opposed to the full loss $f(x)$. Second, this model can be negative even though our loss functions are always non-negative. Here, we develop a model directly for $f(x)$, and not $f(x; S_k)$, which also takes into account lower bounds on the function value.

5.3.1 Model-Based Viewpoint of Momentum

Suppose we have sampled inputs $S_1, \dots, S_k$ and past iterates $x^1, \dots, x^k$. We can use these samples to build a better model of $f(x)$ by averaging past function evaluations as follows

$$f(x) = \mathbb{E}[f(x; S)] \;\approx\; \tfrac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} f(x; S_j), \tag{5.3}$$

where $\rho_{j,k} \geq 0$ and $\rho_k := \sum_{j=1}^{k} \rho_{j,k}$. Thus, the $\rho_k^{-1}\rho_{j,k}$ represent a discrete probability mass function over the previous samples. The obvious choice for $\rho_{j,k}$ would be uniform averaging, however, as we will show, a more natural choice will be an exponentially weighted average in order to put more weight on recent iterates. The issue with (5.3) is that it is expensive to evaluate $f(x; S_j)$ for $j = 1, \dots, k$, which we would need to do at every iteration. Instead, we approximate each $f(x; S_j)$ by linearizing $f(x; S_j)$ around x^j, which was the point where it was last evaluated. This leads to the further approximation

$$f(x; S_j) \approx f(x^j; S_j) + \langle \nabla f(x^j; S_j), x - x^j \rangle, \quad \text{for } j = 1, \dots, k. \tag{5.4}$$

Using (5.3) and the linear approximations in (5.4) we can approximate $f(x)$ as follows

$$f(x) \approx \tfrac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} \big[f(x^j; S_j) + \langle \nabla f(x^j; S_j), x - x^j \rangle \big] = m_k(x). \tag{5.5}$$

If we plug in the above model $m_k(x)$ into (5.2), then the resulting update is:

$$d_k := \sum_{j=1}^{k} \rho_{j,k} \nabla f(x^j; S_j), \quad x^{k+1} = x^k - \tfrac{\alpha_k}{\rho_k} d_k. \tag{5.6}$$

Now choosing $\rho_{j,k}$ such that d_k is an exponentially weighted average (with factor $\beta \in [0, 1)$), (5.6) is equal to `SGD-M`, given by

$$d_k = \beta d_{k-1} + (1 - \beta) \nabla f(x^k; S_k), \quad x^{k+1} = x^k - \alpha_k d_k.$$

This gives a new viewpoint of (heavy-ball) momentum. Next we incorporate a lower bound into this model so that, much like the loss function, it cannot become negative.

5.3.2 Deriving MoMo

Since we know the loss is lower-bounded by zero, we will also impose a lower bound on the model (5.5). Though we could use zero, we will use an estimate $f_\star^k \geq 0$ of the lower bound to allow for cases where $f^\star$ may be far from zero. Imposing a lower bound of $f_\star^k$ gives the following model

$$f(x) \approx \max\left\{\tfrac{1}{\rho_k}\sum_{j=1}^{k}\rho_{j,k}\left[f(x^j;S_j)+\langle\nabla f(x^j;S_j),x-x^j\rangle\right], f_\star^k\right\} =: m_k(x). \tag{5.7}$$

For overparametrized machine-learning models the minimum value $f(x^\star)$ is often close to zero [51, 91]. Thus, choosing $f_\star^k = 0$ in every iteration will work well (as we verify later in our experiments). For tasks where $f_\star^k = 0$ is too loose of a bound, in Section 5.6 we develop an online estimate for $f_\star^k$ based on available information. Using the model (5.7), we now compute the proximal update

$$x^{k+1} = \arg\min_{y\in\mathbb{R}^n} m_k(y) + \frac{1}{2\alpha_k}\|y-x^k\|^2. \tag{5.8}$$

Because $m_k(y)$ is a simple piece-wise linear function, update (5.8) has a closed form solution, as shown in the following lemma.

Lemma 5.1 (MoMo update)**.** *Let*

$$d_k := \sum_{j=1}^{k}\rho_{j,k}\nabla f(x^j;S_j), \quad \bar{f}_k := \sum_{j=1}^{k}\rho_{j,k}f(x^j;S_j), \quad \gamma_k := \sum_{j=1}^{k}\rho_{j,k}\langle\nabla f(x^j;S_j),x^j\rangle. \tag{5.9}$$

Define

$$h_k := \sum_{j=1}^{k}\rho_{j,k}[f(x^j;S_j)+\langle\nabla f(x^j;S_j),x^k-x^j\rangle] = \bar{f}_k + \langle d_k, x^k\rangle - \gamma_k. \tag{5.10}$$

Using model (5.7)*, the closed form solution to* (5.8) *is*

$$x^{k+1} = x^k - \underbrace{\min\left\{\frac{\alpha_k}{\rho_k}, \frac{\left(\bar{f}_k+\langle d_k,x^k\rangle-\gamma_k-\rho_k f_\star^k\right)_+}{\|d_k\|^2}\right\}}_{=:\tau_k} d_k. \tag{5.11}$$

In the above, we define $\tau_k := 0$ *if* $d_k = 0$*. Moreover, it holds*

$$m_k(x^{k+1}) = \begin{cases}\frac{1}{\rho_k}[h_k - \tau_k\|d_k\|^2] & \text{if } h_k \geq \rho_k f_\star^k,\\ f_\star^k & \text{else.}\end{cases} \tag{5.12}$$

Proof. We have that

$$m_k(y) = \max\left\{\rho_k^{-1}(h_k+\langle d_k,y-x^k\rangle), f_\star^k\right\} = \left(\rho_k^{-1}(h_k+\langle d_k,y-x^k\rangle)-f_\star^k\right)_+ + f_\star^k. \tag{5.13}$$

Using (5.13), dropping the constant term $f_\star^k$, and multiplying with ρ_k, problem (5.8) is equivalent to

$$x^{k+1} = \arg\min_{y \in \mathbb{R}^n} \Big(h_k + \langle d_k, y - x^k \rangle - \rho_k f_\star^k \Big)_+ + \frac{\rho_k}{2\alpha_k} \|y - x^k\|^2.$$

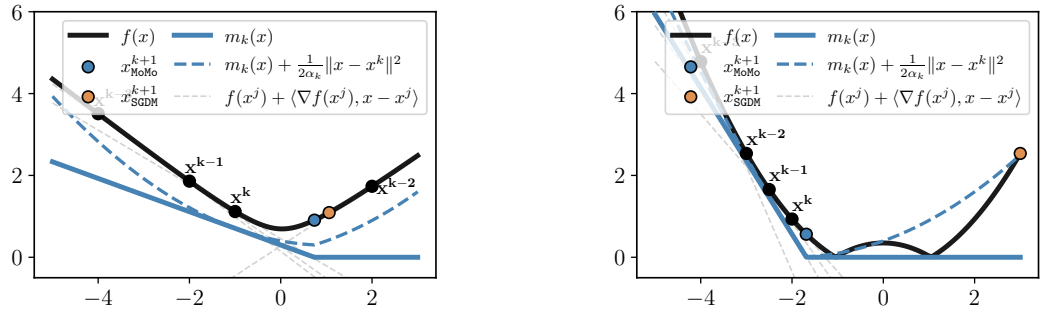
Applying Lemma 4.9 with $\beta \leftarrow \rho_k^{-1}\alpha_k$, $c \leftarrow h_k - \rho_k f_\star^k$, $a \leftarrow d_k$ and $y_0 \leftarrow x^k$ gives the result. Equation (5.12) follows from (4.29), and using that

$$m_k(x^{k+1}) = f_\star^k + \frac{1}{\rho_k}\Big(h_k + \langle d_k, y - x^k \rangle - \rho_k f_\star^k \Big)_+ = f_\star^k + \frac{1}{\rho_k}\phi(x^{k+1}).$$

□

Finally, it remains to select the averaging coefficients $\rho_{j,k}$. Here we will use an exponentially weighted average that places more weight on recent samples. Aside from working well in practice on countless real-world examples, using an exponentially weighted average can be motivated in the setting of the proximal update (5.8). Recent iterates will most likely have gradients and loss values that are closer to the current iterate. Since we want the model $m_k(y)$ to be a reasonably accurate approximation of $f(y)$ when y is close to the current iterate, this justifies using averaging weights $\rho_{j,k}$ that put more weight on the recent iterates, i.e. $\rho_{j,k}$ is big for j close to k.

We give two options for exponentially weighted averaging next.



(a) Convex loss

(b) Non-convex loss

Figure 5.1: Illustration of the `MoMo` model (blue curves) for two different loss functions with learning rate $\alpha_k = 5$. Due to truncation, the new iterate of `MoMo` (blue point) is closer to the minimum than `SGD-M` (orange point). The right plot shows how `MoMo` takes a small step when gradients are steep, whereas `SGD-M` takes a large step and ends up far from the solution as a result.

5.3.3 The Coefficients $\rho_{j,k}$: To Bias or not to Bias

We now choose $\rho_{j,k} \geq 0$ such that we can update $\bar{f}_k$, d_k and γ_k in (5.9) on the fly, storing only two scalars and one vector.

Exponentially Weighted Average. Let $\beta \in [0, 1)$. Starting with $\rho_{1,1} = 1$, and for $k \geq 2$ define $\rho_{j,k} = \beta\rho_{j,k-1}$ for $j \leq k-1$ and $\rho_{j,k} = 1 - \beta$ for $j = k$. Then, $\rho_k = \sum_{j=1}^k \rho_{j,k} = 1$ for all $k \in \mathbb{N}$

and the quantities in (5.9) are exponentially weighted averages. A proof is given in Lemma 5.7. As a consequence, we can update $\bar{f}_k$, d_k and γ_k on the fly as given in lines 3–5 in Algorithm 12. Combining update (5.11) and the fact that $\rho_k = 1$, we obtain Algorithm 12, which we call `MoMo`.

Algorithm 12 `MoMo`: Model-based Momentum method.

Require: $x^1 \in \mathbb{R}^n$, $\alpha_k > 0$, $\beta \in [0,\, 1)$, $(f_\star^k)_{k\in\mathbb{N}} \subset \mathbb{R}$.
Default settings: $\alpha_k = 1$, $\beta = 0.9$ and $(f_\star^k)_{k\in\mathbb{N}} = 0$.

1: **Initialize:** $\bar{f}_0 = f(x^1; S_1),\ d_0 = \nabla f(x^1; S_1),\ \gamma_0 = \langle d_0, x^1\rangle$
2: **for** $k = 1, \dots, K-1$ **do**
3: $\quad \bar{f}_k \ = (1-\beta) f(x^k; S_k) + \beta \bar{f}_{k-1}$
4: $\quad \gamma_k = (1-\beta)\langle \nabla f(x^k; S_k), x^k\rangle + \beta \gamma_{k-1}$
5: $\quad d_k \ = (1-\beta)\nabla f(x^k; S_k) + \beta d_{k-1}$
6: $\quad$ Set $\tau_k = 0$ if $d_k = 0$ and else $\tau_k = \min\left\{\alpha_k, \frac{(\bar{f}_k + \langle d_k, x^k\rangle - \gamma_k - f_\star^k)_+}{\|d_k\|^2}\right\}$
7: $\quad$ Update

$$x^{k+1} = x^k - \tau_k d_k.$$

8: **end for**
9: **return** x^K

Remark 9. *The* adaptive learning rate τ_k *in* (5.11) *determines the size of the step and can vary in each iteration even if* α_k *is constant. The* (user-specified) learning rate α_k *is used to cap the adaptive learning rate* τ_k*.*

Remark 10 (Complexity)**.** *`MoMo` has the same order of iteration complexity and memory footprint as `SGD-M`. Indeed, `MoMo` only stores two additional scalars* γ_k *and* $\bar{f}_k$ *as compared to `SGD-M`. The additional* $\mathcal{O}(d)$ *costs are two inner products on lines 4 and 6, and one vector norm on line 6.*

Let us mention a special case: For $\beta = 0$ (no momentum), we have $\gamma_k = \langle \nabla f(x^k; S_k), x^k\rangle = \langle d_k, x^k\rangle$ and $\bar{f}_k = f(x^k; S_k)$. Consequently $h_k = f(x^k; S_k)$, and in this special case, `MoMo` is equivalent[3] to $\texttt{SPS}_{\max}$.

Fig. 5.1 illustrates how the `MoMo` model (5.8) approximates a convex function (left) and a non-convex function (right). In both examples, the resulting update $x^{k+1}_{\texttt{MoMo}}$ in Fig. 5.1 attains a lower loss, as compared to a step of `SGD-M`. However, this is a toy example and only serves to illustrate how `MoMo` can result in a different learning dynamic than `SGD-M`; a proper numerical comparison will follow later.

Averaging with Bias Correction. Alternatively, we can choose $\rho_{j,k} = (1-\beta)\beta^{k-j}$ for $j = 1, \dots, k$, as it is used in `Adam` [74]. This gives $\rho_k = 1 - \beta^k \neq 1$. We discuss this choice for `MoMo` in Section 5.9.2 and will use it later for `MoMo-Adam`. We have not experienced any significant performance difference in our experiments between switching bias correction on or off.

[3] This equivalence requires setting $\gamma_b \leftarrow \alpha_k,\ c \leftarrow 1$, and assuming $f_\star^k = C(S_{i_k})$.

5.4 Weight Decay and Preconditioning

Next, we show how to include weight decay [77] in the `MoMo` framework. Weight decay can be seen as adding a squared ℓ_2-regularization to the objective function, in other words, instead of (5.1) we solve

$$\min_{x\in\mathbb{R}^n} f(x) + \frac{\lambda}{2}\|x\|^2, \tag{5.14}$$

where $f(x)$ is again the loss function and $\lambda \geq 0$.

The technique to adapt the model-based framework to include weight decay is essentially what has been presented in Chapter 4. That is, we build a model m_k for the loss f and keep the ℓ_2-regularization outside of the model. Hence, equation (5.8) is modified to

$$x^{k+1} = \arg\min_{y\in\mathbb{R}^n} m_k(y) + \frac{\lambda}{2}\|y\|^2 + \frac{1}{2\alpha_k}\|y - x^k\|^2. \tag{5.15}$$

Finally, for the proximal term $\frac{1}{2\alpha_k}\|y - x^k\|^2$ the Euclidean norm may often not be best suited. Many popular methods such as `AdaGrad` or `Adam` are based on using a preconditioner for the proximal step. Hence, we allow for an arbitrary norm defined by a symmetric, positive definite matrix $\mathbf{D}_k \in \mathbb{R}^{n\times n}$, i.e. $\|x\|^2_{\mathbf{D}_k} = \langle x, \mathbf{D}_k x\rangle$. We can now use $\mathbf{D}_k$ to change the metric within our proximal method via

$$x^{k+1} = \arg\min_{y\in\mathbb{R}^n} m_k(y) + \frac{\lambda}{2}\|y\|^2_{\mathbf{D}_k} + \frac{1}{2\alpha_k}\|y - x^k\|^2_{\mathbf{D}_k}. \tag{5.16}$$

This update (5.16) still enjoys a closed form solution, as we show next.

Lemma 5.2. *Using model* (5.7)*, the closed form solution to* (5.16) *is given by*

$$\tau_k = \min\left\{\frac{\alpha_k}{\rho_k}, \frac{\big((1+\alpha_k\lambda)(\bar{f}_k - \rho_k f_\star^k - \gamma_k) + \langle d_k, x^k\rangle\big)_+}{\|d_k\|^2_{\mathbf{D}_k^{-1}}}\right\}, \tag{5.17}$$

$$x^{k+1} = \frac{1}{1+\alpha_k\lambda}\left[x^k - \tau_k \mathbf{D}_k^{-1} d_k\right]. \tag{5.18}$$

In the above, we define $\tau_k := 0$ if $d_k = 0$. Moreover, it holds

$$m_k(x^{k+1}) = \begin{cases} \frac{1}{\rho_k}\big[h_k - \frac{\alpha_k\lambda}{1+\alpha_k\lambda}\langle d_k, x^k\rangle - \frac{\tau_k}{1+\alpha_k\lambda}\|d_k\|^2_{\mathbf{D}_k^{-1}}\big] & \text{if } h_k - \rho_k f_\star^k \geq \frac{\alpha_k\lambda\langle d_k, x^k\rangle}{1+\alpha_k\lambda}, \\ f_\star^k & \text{else.} \end{cases} \tag{5.19}$$

Proof. We use again (5.13). Dropping the constant term $f_\star^k$, and multiplying with ρ_k, problem (5.16) is equivalent to

$$x^{k+1} = \arg\min_{y\in\mathbb{R}^n}\Big(h_k + \langle d_k, y - x^k\rangle - \rho_k f_\star^k\Big)_+ + \frac{\rho_k}{2\alpha_k}\|y - x^k\|^2_{\mathbf{D}_k} + \frac{\rho_k\lambda}{2}\|y\|^2_{\mathbf{D}_k}.$$

Now applying Lemma 4.10 with $y_0 \leftarrow x^k$, $a \leftarrow d_k$, $c \leftarrow h_k - \rho_k f_\star^k$, $\lambda \leftarrow \rho_k\lambda$, $\beta \leftarrow \rho_k^{-1}\alpha_k$ and $\mathbf{D} \leftarrow \mathbf{D}_k$, we obtain the update. Equation (5.19) follows again from $m_k(x^{k+1}) = f_\star^k + {\rho_k}^{-1}\phi(x^{k+1})$ and (4.33). □

Lemma 5.2 shows how to incorporate weight decay in `MoMo`: we replace line 7 in Algorithm 12 by (5.17)-(5.18) with $\mathbf{D}_k = \mathbf{Id}_n$ and $\rho_k = 1$. In this case, if $\beta = 0$ (no momentum) then `MoMo` with weight decay recovers `ProxSPS` as special case, the proximal version of the stochastic Polyak step size presented in Chapter 4.

Deriving `MoMo-Adam`. Using Lemma 5.2 we can obtain an `Adam`-version of `MoMo` by defining $\mathbf{D}_k$ as the diagonal preconditioner of `Adam`. Recall the notation $\mathbf{1}_n$ for a n-dimensional vector of ones, and $\mathrm{Diag}(v)$ for a diagonal matrix with diagonal entries $v \in \mathbb{R}^n$. We use $\odot$ and $\sqrt{v}$ for elementwise multiplication and square-root operations. Denoting $g_k = \nabla f(x^k; S_k)$, choose

$$v_k = (1-\beta_2)v_{k-1} + \beta_2(g_k \odot g_k), \quad \mathbf{D}_k = \mathrm{Diag}(\varepsilon \mathbf{1}_n + \sqrt{v_k/(1-\beta_2)^k}),$$

where $\beta_2 \in [0,1)$, $\varepsilon > 0$. Using this diagonal preconditioner together with Lemma 5.2 gives Algorithm 13, called `MoMo-Adam`. Note that here we choose $\rho_{j,k} = (1-\beta)\beta^{k-j}$ (cf. Section 5.3.3) which gives the standard averaging scheme of `Adam`.

Algorithm 13 `MoMo-Adam`: Adaptive learning rates for `Adam`.

Require: $x^1 \in \mathbb{R}^n$, $\alpha_k > 0$, $\beta_1, \beta_2 \in [0,\ 1)$, $\varepsilon > 0$, $\lambda \geq 0$, and $(f_\star^k)_{k\in\mathbb{N}} \subset \mathbb{R}$.
Default settings: $\alpha_k = 10^{-2}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\varepsilon = 10^{-8}$, and $(f_\star^k)_{k\in\mathbb{N}} = 0$.

Initialize: $\bar{f}_0 = 0$, $d_0 = 0$, $\gamma_0 = 0$, and $v_0 = 0$.
for $k = 1, \dots, K-1$ **do**
  Compute $g_k = \nabla f(x^k; S_k)$ and set $d_k = (1-\beta_1)g_k + \beta_1 d_{k-1}$.
  $v_k = \beta_2 v_{k-1} + (1-\beta_2)(g_k \odot g_k)$
  $\mathbf{D}_k = \mathrm{Diag}\left(\varepsilon \mathbf{1}_n + \sqrt{v_k/(1-\beta_2^k)}\right)$
  $\bar{f}_k = (1-\beta_1)f(x^k; S_k) + \beta_1 \bar{f}_{k-1}$
  $\gamma_k = (1-\beta_1)\langle g_k, x^k\rangle + \beta_1 \gamma_{k-1}$
  Set $\tau_k = 0$ if $d_k = 0$ and else $\tau_k = \min\left\{\frac{\alpha_k}{1-\beta_1^k}, \frac{\left((1+\lambda\alpha_k)(\bar{f}_k - \gamma_k - (1-\beta_1^k)f_\star^k) + \langle d_k, x^k\rangle\right)_+}{\|d_k\|^2_{\mathbf{D}_k^{-1}}}\right\}$
  Update

$$x^{k+1} = \frac{1}{1+\alpha_k\lambda}\left[x^k - \tau_k \mathbf{D}_k^{-1} d_k\right]$$

end for
return x^K

Extensions. We focused on `SGD-M` and `Adam` as they are the most widely used methods in the community. From Lemma 5.2 it is however obvious how to derive a `MoMo` version for any preconditioned momentum method. In fact, many recently proposed variations of `Adam`, often claiming to be superior in performance, fall into this category. For example, consider `AdaBelief` [168], which is equal to `Adam` up to the choice of preconditioner, given by

$$v_k = \beta_2 v_{k-1} + (1-\beta_2)(g_k - d_k)^2 + \varepsilon,$$
$$\mathbf{D}_k = \mathrm{Diag}\left(\varepsilon \mathbf{1}_n + \sqrt{v_k/(1-\beta_2^k)}\right).$$

Applying Lemma 5.2 with this preconditioner would lead to `MoMo-AdaBelief`.

5.5 Convergence Analysis

In this section, we present a convergence theorem for `MoMo` in the convex case. We first show why proving convergence is non-trivial, in the sense that one can not simply apply the theory of model-based stochastic optimization established in [6, 26]. If we compute the model value at the current iterate $m_k(x^k)$, we observe that for (5.7) we have $\mathbb{E}[m_k(x^k)] \neq f(x^k)$ due to the

averaging of linearizations. From this we conclude that the standard model-based theory can not be applied (cf. Proposition 4.3, (B2)).

Thus, we need to take an alternative path: below, we prove a *monotonicity result* for the convex case, i.e. we show that `MoMo` reduces the distance to the solution in every iteration (almost surely), if one chooses $f_\star^k$ in a specific way. In particular, the iterates of `MoMo` are almost surely contained in a bounded set. This is surprising, since no such result exists for standard `SGD` to the best of our knowledge [46]. Finally, we will show a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{K}})$ in the convex case if an interpolation condition holds in Theorem 5.4. Importantly, this does not require neither smoothness nor globally bounded gradients, due to the fact that the iterates are almost surely bounded.

Let us define the quantity

$$\bar{f}_\star^k := \frac{1}{\rho_k} \sum_{j=1}^{k} \rho_{j,k} f(x^\star; S_j). \tag{5.20}$$

It is somewhat clear that it will be only possible to prove convergence if $f_\star^k$ is chosen appropiately: it turns out (from the proof of the below lemma) that the choice $f_\star^k = \bar{f}_\star^k$ allows for a monotonous decrease of the distance to the solution in the convex setting.

Lemma 5.3. *Consider the iterates generated by the general `MoMo` update (cf. Lemma 5.2) with no weight decay ($\lambda = 0$) and with $f_\star^k = \bar{f}_\star^k$ (defined as in (5.20)). That is, if $d_k = 0$ set $\tau_k = 0$ and else*

$$\tau_k = \min\Big\{\frac{\alpha_k}{\rho_k}, \frac{(\bar{f}_k - \rho_k \bar{f}_\star^k - \gamma_k + \langle d_k, x^k\rangle)_+}{\|d_k\|^2_{\mathbf{D}_k^{-1}}}\Big\},$$
$$x^{k+1} = x^k - \tau_k \mathbf{D}_k^{-1} d_k.$$

Let $f(\cdot; s)$ be convex for each $s \in \mathcal{S}$ and let $x^\star \in \arg\min_{x\in\mathbb{R}^n} f(x)$. Then, it holds almost surely

$$\|x^{k+1} - x^\star\|^2_{\mathbf{D}_k} \leq \|x^k - x^\star\|^2_{\mathbf{D}_k} - \tau_k (h_k - \rho_k \bar{f}_\star^k)_+. \tag{5.21}$$

To prove convergence, we rely on the following interpolation assumption:

$$f(x^\star; s) = \inf_x f(x; s) = f^\star \quad \text{for all } s \in \mathcal{S}. \tag{5.22}$$

Theorem 5.4. *Let $f(\cdot; s)$ be convex for every $s \in \mathcal{S}$ and let $x^\star \in \arg\min_{x\in\mathbb{R}^n} f(x)$. Assume that (5.22) holds. For $K \in \mathbb{N}$, let $(x^k)_{k\in[K]}$ be the iterates of Algorithm 12 (`MoMo`) with $f_\star^k = f^\star$ for all $k \in [K]$ and assume that $d_k \neq 0$ for all $k \in [K]$.[4] Define*

$$B := \{x \in \mathbb{R}^n \mid \|x - x^\star\| \leq \|x^1 - x^\star\|\}.$$

Assume that there exists $G > 0$ such that $\|\nabla f(x; s)\|^2 \leq G^2 < \infty$ for all $x \in B$, $s \in \mathcal{S}$. Further, assume that

$$\alpha_k \geq \frac{(1-\beta)}{G^2} \max_{x\in B,\ s\in\mathcal{S}} [f(x; s) - f^\star]. \tag{5.23}$$

Then, it holds

$$\min_{k=1,\dots,K} \mathbb{E}[f(x^k) - f^\star] \leq \frac{G\|x^1 - x^\star\|}{\sqrt{K(1-\beta)}}. \tag{5.24}$$

[4] The unlikely case $d_k = 0$ could be circumvented, for example, by adding a resampling step of g_k if $d_k = 0$.

The proofs are given in Section 5.9.1. We make a few remarks on the above theorem. Assumption (5.23) on α_k is somewhat uncommon, since typical results require an upper bound on the step size. Our result in particular allows $\alpha_k = \infty$, but using α_k as a learning rate cap can be useful in practice (where other assumptions of the theorem may not be satisfied).

We stress that unlike for standard results for `SGD` in the convex, non-(Lipschitz-)smooth case, we do not need to assume *globally* bounded gradients, but instead only boundedness on a compact set. Theorem 5.4 is similar to [88, Thm. C.1], with the major difference that here we allow for momentum. The best constant in (5.24) is achieved by $\beta = 0$, i.e. no momentum. Thus, while empirically momentum helps in most cases, we can not show a *theoretical* improvement at this time. Second, we remark that the improvement of not assuming globally bounded gradients could be achieved analogously for the proof of [88] based on the result in Lemma 5.3.

5.6 Estimating a Lower Bound

So far, we have assumed that lower-bound estimates $(f_\star^k)$ are given with $f_\star^k = 0$ being the default. However, $f_\star^k = 0$ might not always be a tight estimate of $f^\star$ (e.g. when training transformers). For such situations, we derive an online estimate of the lower bound next. In particular, for convex functions we will derive a lower bound for $\bar{f}_\star^k$. Though this is not the optimal value $f(x^\star)$ of our original loss function, it is a reasonable approximation. In particular, it is an unbiased estimator of $f^\star$ since

$$\mathbb{E}[\bar{f}_\star^k] \;=\; \frac{1}{\rho_k}\sum_{j=1}^{k}\rho_{j,k}\mathbb{E}[f(x^\star;S_j)] \;=\; \frac{1}{\rho_k}\sum_{j=1}^{k}\rho_{j,k}f(x^\star) \;=\; f(x^\star). \tag{5.25}$$

Further, we motivated our method using the analogous approximation of $f(x)$ in (5.3). The following lemma derives an estimate $f_\star^k \geq 0$ for $\bar{f}_\star^k$ given in (5.20) by using readily available information for any momentum-based method, in particular Algorithm 12 and Algorithm 13 with $\lambda = 0$.

Lemma 5.5. *Let $f(x;s)$ be convex in x for all $s \in \mathcal{S}$. Let the sequence $(x^k)_k$ be generated by $x^{k+1} = x^k - \tau_k \mathbf{D}_k^{-1} d_k$ for any step size $\tau_k > 0$ and symmetric, positive definite $\mathbf{D}_k$. Further, let*

$$\eta_k := \prod_{j=2}^{k} \lambda_{\min}\big(\mathbf{D}_j^{-1}\mathbf{D}_{j-1}\big), \quad \textit{and} \quad h_k := \bar{f}_k + \langle d_k, x^k\rangle - \gamma_k. \tag{5.26}$$

It follows that $\bar{f}_\star^k \geq f_\star^{k+1}$ where

$$f_\star^{k+1} := \tfrac{1}{2\eta_k\tau_k\rho_k}\Big(\sum_{j=1}^{k} 2\eta_j\tau_j\big(h_j - \frac{1}{2}\tau_j\|d_j\|^2_{\mathbf{D}_j^{-1}}\big) - \|x^1 - x^\star\|^2_{\mathbf{D}_1} - 2\sum_{j=1}^{k-1}\eta_j\tau_j\rho_j\bar{f}_\star^j\Big) \tag{5.27}$$

$$= \frac{2\eta_{k-1}\tau_{k-1}\rho_{k-1}(f_\star^k - \bar{f}_\star^{k-1}) - \eta_k\tau_k^2\|d_k\|^2_{\mathbf{D}_k^{-1}} + 2\eta_k\tau_k h_k}{2\eta_k\tau_k\rho_k}. \tag{5.28}$$

Bootstrapping by using $f_\star^k \approx \bar{f}_\star^{k-1}$, this yields for $k \geq 2$ that

$$f_\star^{k+1} \approx \tfrac{1}{\rho_k}\left(h_k - \tfrac{1}{2}\tau_k\|d_k\|^2_{\mathbf{D}_k^{-1}}\right). \tag{5.29}$$

The proof is given in Section 5.9 in the Supplementary Material. We want to give a few remarks on the above result. First, note that $f_\star^{k+1}$ in (5.27) depends on $\|x^1 - x^\star\|^2_{\mathbf{D}_1}$, which we do not know in general. We circumvent this by using the bootstrapping approximation (5.29) where $\|x^1 - x^\star\|^2_{\mathbf{D}_1}$ does not appear. We can initialize $f_\star^1 = 0$ for most loss functions (or use an available initial guess for the optimal value). Clearly, (5.29) is only an approximation of (5.28) and therefore not guaranteed to be a lower bound of $\bar{f}_\star^k$ – though we will verify in experiments that this heuristic yields accurate estimates of the optimal value.

We can now introduce `MoMo` or `MoMo-Adam` with online lower bound $f_\star^k$, which we call `MoMo`* and `MoMo-Adam`*. We add one more precautionary measure, because we want to avoid the step size τ_k in (5.17) to be zero. That is, by examining (5.17) we aim to prevent that

$$(1 + \alpha_k\lambda)\rho_k f_\star^k \geq (1 + \alpha_k\lambda)(\bar{f}_k - \gamma_k) + \langle d_k, x^k\rangle =: h_k^\lambda. \tag{5.30}$$

Hence, in each iteration of `MoMo`* or `MoMo-Adam`*, we call the `ResetStar` routine in Algorithm 14 *before* the update of x^{k+1}. The idea is to check if $f_\star^k$ has become too large such that (5.30) is satisfied; if this is the case, reset $f_\star^k$ to be sufficiently small.[5] After updating x^{k+1}, we update $f_\star^{k+1}$ with `EstimateStar` routine in Algorithm 15, according to Lemma 5.5. For completeness, we give the full algorithm of `MoMo`* in Algorithm 17 in the Supplementary Material.

We show for a simple numerical example how the values of $f_\star^k$ produced by `MoMo(-Adam)`* converge to the true $f^\star$ in Section 5.10.4.

A `Pytorch` implementation of all `MoMo` methods we discussed is provided in [133].

Algorithm 14 `ResetStar`

Require: $f_\star^k, \alpha_k, \lambda, \rho_k, h_k^\lambda$
1: **if** (5.30) **then**
2: $\quad f_\star^k = \max\left\{\frac{1}{2}[(1 + \alpha_k\lambda)\rho_k]^{-1}h_k^\lambda, f_\star^1\right\}$
3: **end if**
4: **return** $f_\star^k$

Algorithm 15 `EstimateStar`

Require: $\bar{f}_k, x^k, \gamma_k, \tau_k, d_k, \mathbf{D}_k, \rho_k$
1: $h_k = \bar{f}_k + \langle d_k, x_k\rangle - \gamma_k$
2: $f_\star^{k+1} = \max\left\{\rho_k^{-1}\left(h_k - \frac{1}{2}\tau_k\|d_k\|^2_{\mathbf{D}_k^{-1}}\right), f_\star^1\right\}$
3: **return** $f_\star^{k+1}$

5.7 Numerical Experiments

Here we investigate how using the `MoMo` adaptive learning rate can improve the stability of both `SGD-M` and `Adam`. To do this, for each task and model, we do a learning-rate sweep for both `SGD-M`, `Adam`, `MoMo` and `MoMo-Adam` and compare the resulting validation score for each learning rate.

Our experiments will focus on the sensitivity with respect to the learning rate choice. This is motivated from the insights in [137] who showed that most optimization methods perform equally well when being tuned, however for practical use a tuning budget needs to be considered.

For `MoMo` and `MoMo-Adam`, note that the effective step size (cf. (5.17)) has the form

$$\tau_k = \min\{\tfrac{\alpha_k}{\rho_k}, \zeta_k\} \quad \text{with} \quad \zeta_k := \frac{\left((1+\alpha_k\lambda)(\bar{f}_k - \rho_k f_\star^k - \gamma_k) + \langle d_k, x^k\rangle\right)_+}{\|d_k\|^2_{\mathbf{D}_k^{-1}}}. \tag{5.31}$$

[5] In the case that $h_k^\lambda < 0$, we can not guarantee that $\tau_k > 0$ with this resetting procedure.

We refer to Algorithm 12, line 7 and Algorithm 13, line 9 for the exact formula for `MoMo` and `MoMo-Adam`. For `MoMo` we have that $\rho_k = 1, \mathbf{D}_k = \mathbf{Id}$. We will refer to α_k as the *(user-specified) learning rate* and to τ_k as the *adaptive learning rate*. In all experiments with weight decay, when we write `Adam` this actually refers to its decoupled weight decay version `AdamW`.

5.7.1 Zero as Lower Bound

First, we compare the `MoMo` methods to `SGD-M` and `Adam` for problems where zero is a reasonable estimate of the optimal value $f^\star$. In this section, we set $f_\star^k = 0$ for all $k \in \mathbb{N}$ for `MoMo`(-`Adam`).

Models and Datasets. We do the following tasks (more details in Section 5.10.2).

- `ResNet110` for `CIFAR100`, `ResNet20`, `VGG16`, `ViT` for `CIFAR10`,
- `DLRM` for `Criteo` Kaggle Display Advertising Challenge [151],
- `MLP` for `MNIST`: two hidden layers of size 100 and `ReLU`.

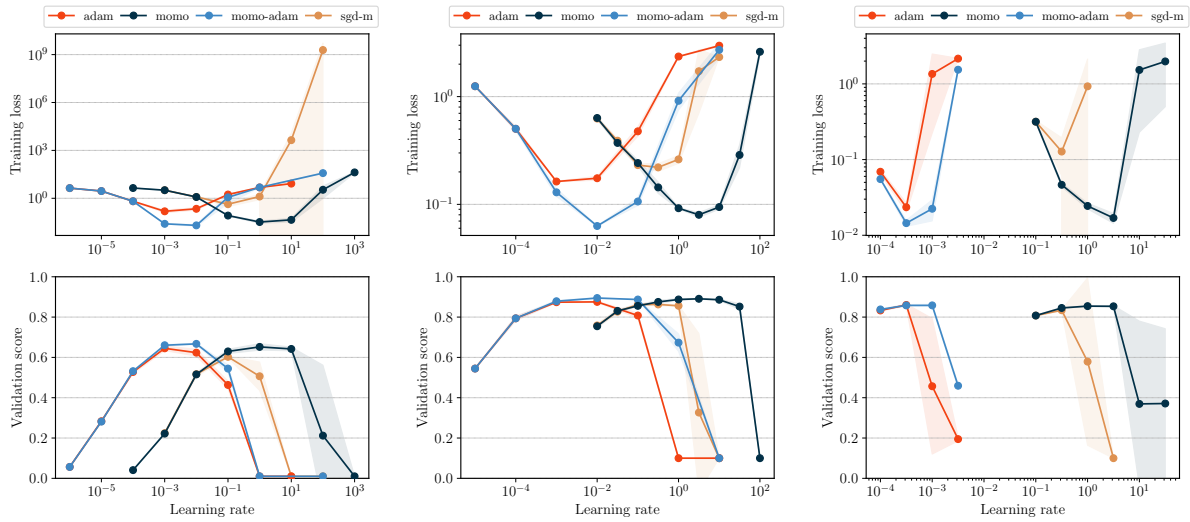
Parameter Settings. We use default choices for momentum parameter $\beta = 0.9$ for `MoMo` and `SGD-M`, and $(\beta_1, \beta_2) = (0.9, 0.999)$ for `MoMo-Adam` and `Adam` respectively. We set $\lambda = 0$, i.e. no weight decay. In the experiments of this section, we always report averaged values over three seeds (five for `DLRM`).

Discussion. We run `MoMo`, `MoMo-Adam`, `Adam` and `SGD-M`, for a fixed number of epochs (cf. Section 5.10.2), using a constant learning rate $\alpha_k = \alpha_0$. The plots in Fig. 5.2 and Fig. 5.3 show the final training loss (top) and accuracy on the validation set (bottom) of each method when varying the learning rate α_0 (training curves for the best runs can be found in Figs. 5.4 and 5.5). We observe that for small learning rates `MoMo` behaves identically to `SGD-M`. This is expected, since for small α_0, we have $\tau_k = \alpha_0$ (see (5.31)) and the update of `MoMo` is equal to `SGD-M`. The same argument applies to the comparison of `Adam` and `MoMo-Adam`.

For larger learning rates, we observe that `MoMo` and `MoMo-Adam` improve the training loss and validation accuracy, but `SGD-M` and `Adam` decline in performance or even fail to converge. Most importantly, `MoMo`(-`Adam`) consistently extend the range of "good" learning rates by over one order of magnitude. Further, `MoMo`(-`Adam`) achieve the overall best validation accuracy for all problems except `DLRM` and `ViT`, where the gap to the best score is minute or within the standard deviation of running multiple seeds (see Table 5.1).

This advantage can be explained with the adaptivity of the step size of `MoMo`(-`Adam`). In Fig. 5.6a, we plot the adaptive term ζ_k (cf. (5.31)) for `MoMo` on a `ResNet20`. For $\alpha_0 \in [1, 10]$, we have $\tau_k = \zeta_k$ in many iterations, which means that the effective learning rate is adaptive even though α_k is constant. We observe two phenomena: firstly, in Fig. 5.6a, `MoMo` is doing an automatic learning rate decay *without any user choice for a learning-rate schedule*. Secondly, in the very first iterations, `MoMo` is doing a warm-up of the learning rate as $\tau_k = \zeta_k$ starts very small, but quickly becomes large. Both of these dynamics of τ_k seemingly improve performance and stability. We also obtain faster initial training progress of `MoMo`(-`Adam`) (cf. Fig. 5.4 and Fig. 5.5).

We also compare to `AdaBelief` and `AdaBound` in Fig. 5.10. However, `MoMo`(-`Adam`) still turns out to be the best method. We want to stress again that our approach would also allow to



(a) `ResNet110` for `CIFAR100` (b) `ResNet20` for `CIFAR10` (c) `ViT` for `CIFAR10`

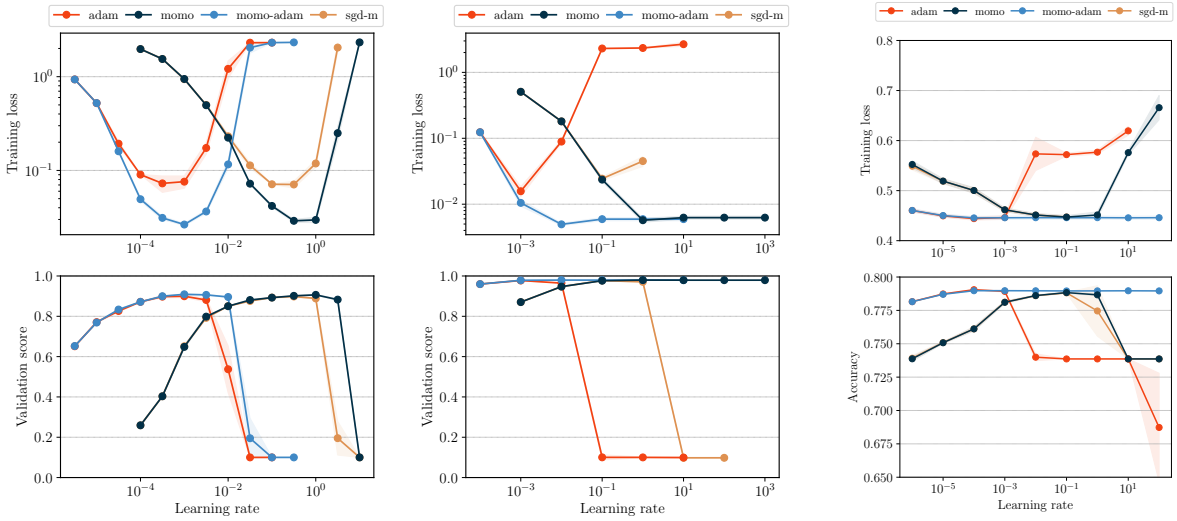
Figure 5.2: Training loss (top row) and validation accuracy (bottom row) after a fixed number of epochs, for varying (constant) learning rate α_0. Shaded area depicts two standard deviations over 3 runs.

derive a `MoMo`-version for any of these variations of `Adam`. We further compare to `SGD-M` with an exponentially decaying schedule for `CIFAR100`.

Using a Learning-rate Schedule. We present additional experiments, training a vision transformer (`ViT`) on `Imagenet-1k`, as well as a diffusion model (with `UNet` architecture) on the `Smithsonian Butterflies` dataset. Here we only compare `MoMo-Adam` to `AdamW`. This is in order to keep the computational expense within reasonable limits[6], and that `Adam(W)` is the prevalent method for these tasks. Experiment details are in Section 5.10.2.

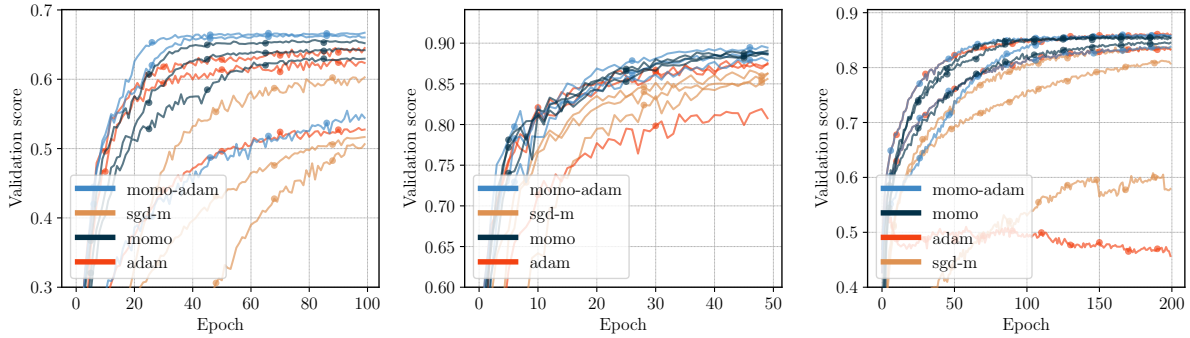
What distinguishes these experiments to previous ones, is that we use `MoMo-Adam` and `AdamW` with a learning-rate schedule for α_k. For both tasks, it is standard practice to use a warmup (from a very small value to a specified base value α_{base}) followed by cosine decay [32]. We vary the value of α_{base} and investigate again how sensitive the methods are with respect to this choice.

[6]For example, one single run for `ViT` training on `Imagenet-1k` takes ten hours on four A100-GPUs with 64 CPUs, and we conduct 24 runs in total.



(a) `VGG16` for `CIFAR10` (b) `MLP` for `MNIST` (c) `DLRM` for `Criteo`

Figure 5.3: Training loss (top row) and validation accuracy (bottom row) after a fixed number of epochs, for varying (constant) learning rate α_0. Shaded area depicts two standard deviations over 3 runs.



(a) `ResNet110` for `CIFAR100` (b) `ResNet20` for `CIFAR10` (c) `ViT` for `CIFAR10`

Figure 5.4: Validation score over training, we plot, for each method, the three choices of α_0 that lead to the best validation score (compare to Fig. 5.2).

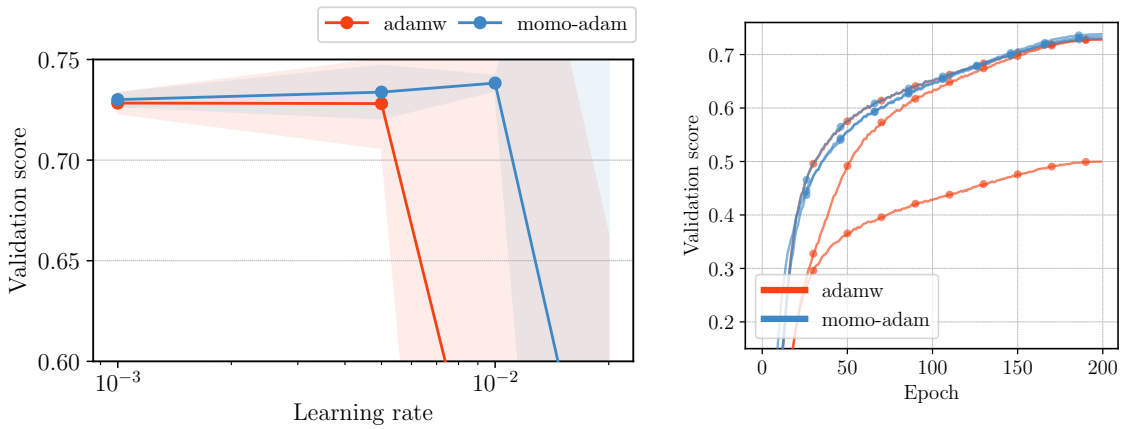
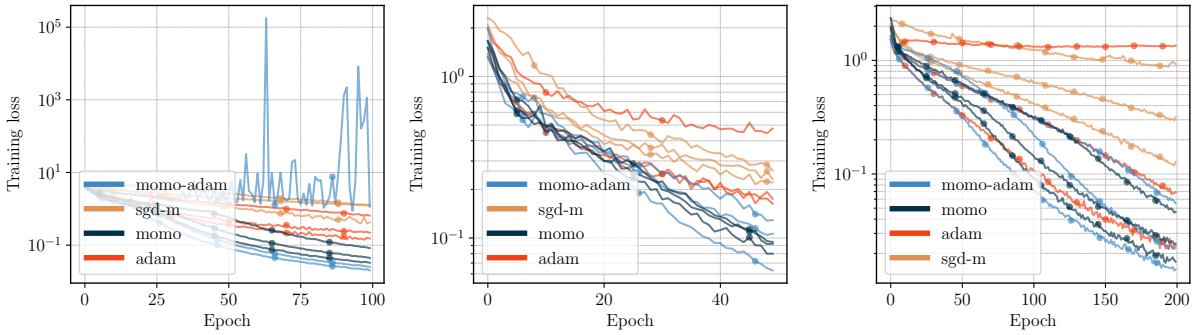


Figure 5.7: `ViT` for `Imagenet-1k`. Left: Final validation set accuracy (top-1) for different base learning-rate values. Right: Training curves for the three best learning-rate values for both methods.



(a) `ResNet110` for `CIFAR100` (b) `ResNet20` for `CIFAR10` (c) `ViT` for `CIFAR10`

Figure 5.5: Training loss over training, we plot, for each method, the three choices of α_0 that lead to the best validation score.

From Fig. 5.7, we see that `MoMo-Adam` (i) works on a larger range of base learning-rate values and (ii) reaches a higher accuracy for the best value of α_{base} (here 0.01).

The results for the diffusion model are similar and presented in Figs. 5.11 and 5.12 in the Appendix. To summarize, we reach the same conclusion as previously: `MoMo-Adam` is generally easier to tune as it works for a wider range of learning rates, it stabilizes training, and it can improve the best model accuracy.

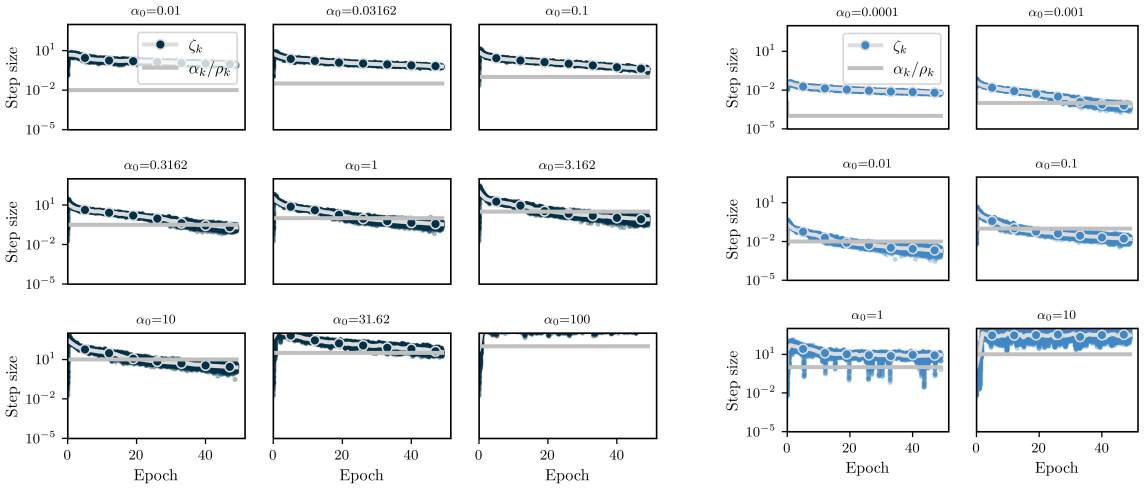
For all of the above tasks, the (training) loss converges to values below 0.5. Next, we consider two problems where the final training loss is significantly above zero. In such situations, we find that `MoMo` methods with $f_\star^k = 0$ are less likely to make use of the adaptive term ζ_k for the step size. As a consequence, `MoMo` with $f_\star^k = 0$ will yield little or no improvement. To see an improvement we need to employ the online estimation of a lower bound for `MoMo` given in Lemma 5.5.

5.7.2 Online Lower Bound Estimation

We now consider image classification on `Imagenet32` and training a transformer for German-to-English translation. For both problems, the optimal value $f^\star$ is far away from zero and hence we use `MoMo` with a known estimate of $f^\star$ or with the online estimation developed in Section 5.6. More details on models and datasets can be found in Section 5.10.2.

`Imagenet32` Classification. We train a `ResNet18` for `Imagenet32` and give the resulting validation accuracy in Fig. 5.8a for weight decay $\lambda = 0$. We show the results for $\lambda = 10^{-4}$ in Fig. 5.9a. We run `MoMo(-Adam)` first with constant lower bound $f_\star^k = 0$ and an *oracle* value $f_\star^k = 0.9$. Further, we run `MoMo(-Adam)`* (indicated by the suffix *-star* in the plots), which correspond to `MoMo(-Adam)` using the estimate $f_\star^k$ given by Lemma 5.5 (cf. Algorithm 17). We compare to `SGD-M` and `AdamW` as baseline. For all methods, we use a constant learning rate $\alpha_k = \alpha_0$ and vary the value of α_0.

First, observe that setting $f_\star^k = 0$ leads to similar performance as the baseline method (in particular it is never worse). Next, observe that the tighter lower bound $f_\star^k = 0.9$ leads to improvement for all learning rates. Finally, the online estimated lower bound widens the range of learning rate with good accuracy by an order of magnitude and leads to small improvements in top accuracy.



(a) `MoMo` (b) `MoMo-Adam`

Figure 5.6: `ResNet20` for `CIFAR10`. Adaptive learning rate of `MoMo` (left) and `MoMo-Adam` (right). The colored dots represent the term ζ_k in each iteration. The grey line represents the user-specified learning rate α_k/ρ_k (note that $\rho_k = 1$ for `MoMo` and $\rho_k \approx 1$ except for the first few iterations in `MoMo-Adam`). The minimum of the grey line and the dots is the adaptive learning rate $\tau_k = \min\{\frac{\alpha_k}{\rho_k}, \zeta_k\}$ in each iteration. The silver line with colored markers is the median over the values of ζ_k in each epoch.

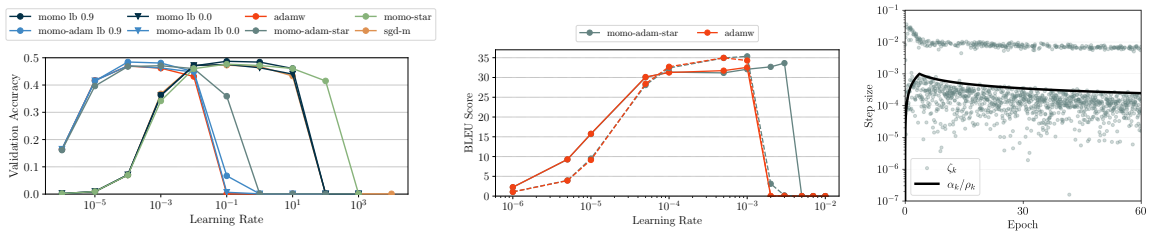
Transformer for German-to-English Translation. We consider the task of neural machine translation from German to English by training an encoder-decoder transformer architecture [154] from scratch on the `IWSLT14` dataset. We use a transformer with six encoder and decoder blocks from `fairseq` [110]. We consider two settings, namely dropout of 0.1 and 0.3. According to standard practice, we fine-tune the hyperparameters of the baseline `AdamW`: for the learning-rate schedule α_k, we use a linear warm-up of 4000 iterations from zero to a given value α_{base} followed by an inverse square-root decay (cf. Fig. 5.8b for an example curve and the adaptive step sizes). All other settings are reported in Section 5.10.2. `MoMo-Adam`* uses the same hyperparameter settings as `AdamW`.

Fig. 5.8b shows the BLEU score[7] after training 60 epochs when varying the initial learning rate α_0: `MoMo-Adam`* is on par or better than `AdamW` on the full range of initial learning rates and for both dropout values. While the improvement is not as substantial as for previous examples, we remark that for this particular task we compare to a fine-tuned configuration of `AdamW`.

5.8 Conclusions and Open Questions

In this chapter, we propose a way to combine two fundamental ideas: the Polyak step size and heavy-ball momentum. We achieve this by introducing momentum as a certain model of the objective function, more specifically by averaging linearizations of the stochastic loss functions around past iterates. We call the resulting method `MoMo`. Second, the model-based approach offers an easy way to include preconditioning, thus allowing us to propose `MoMo-Adam`. More

[7]The BLEU score indicates how similar the candidate sentence, i.e. the output of the trained model, is to a list of reference sentences. The higher the similarity, the higher is the BLEU score.



(a) `ResNet18` for `Imagenet32` (b) Encoder-decoder transformer for `IWSLT14`

Figure 5.8: Validation accuracy over a range of learning rates α_0. (a) `Imagenet32` without weight decay ($\lambda = 0$). (b) Left: `IWSLT14` translation task with dropout 0.1 (**plain**) or 0.3 (**dashed**). The x-axis is the final learning rate α_{base} after warm-up. Right: Learning rate schedule (**black**) and adaptive step sizes (**grey dots**) of `MoMo-Adam`* for $\alpha_{\text{base}} = 10^{-3}$ and dropout 0.3.

broadly, we can derive adaptive learning rates for all preconditioned momentum methods within one single framework.

The results of our experiments reveal one clear and consistent message: `MoMo` and `MoMo-Adam` extend the range of learning rates that lead to the best training results, often by over an order of magnitude. The `MoMo` methods can further lead to improved model performance when comparing with a fixed tuning budget. We demonstrate these advantages across a range of tasks, model architectures and datasets.

Finally, we discuss briefly open questions and topics for future work: first, the current convergence guarantees of `MoMo` are limited to convex problems with interpolation. Naturally, nonconvex convergence results, or less restrictive assumptions on interpolation, would be satisfactory. Given that the existing proof techniques for `SGD-M` often rely on the connection to iterate averaging [139] and do not leave much room for adaptive step sizes, this might be a difficult undertaking.

Second, in this work we can only provide a limited understanding of *why* the adaptive learning rates we proposed are so effective. Our motivation of using a known lower bound, thus improving the model, does not reveal why the adaptive term in `MoMo` performs an automatic warmup and decay, as we observed across many experiments. One possible explanation for this phenomenon is adaptivity to local curvature; this conjecture stems from the fact that for μ-strongly convex, and L-smooth f, the term $\frac{f(x)-\min f}{\|\nabla f(x)\|^2}$ lies in the interval $[\frac{1}{2L}, \frac{1}{2\mu}]$, and thus could be related to the inverse curvature. We leave the analysis of this connection between Polyak-type step sizes and the loss landscape of neural networks for future work.

5.9 Supplementary Material and Missing Proofs

Proof of Lemma 5.5. After switching the index $k \to j$, the update rule is

$$x^{j+1} = x^j - \tau_j \mathbf{D}_j^{-1} d_j,$$

where τ_j is the step size. Subtracting $x^\star$ from both sides, taking norms and expanding the squares we have that

$$\|x^{j+1} - x^\star\|_{\mathbf{D}_j}^2 = \|x^j - x^\star\|_{\mathbf{D}_j}^2 - 2\tau_j \langle d_j, x^j - x^\star \rangle + \tau_j^2 \|d_j\|_{\mathbf{D}_j^{-1}}^2. \tag{5.32}$$

Now let $\delta_{j+1} := \lambda_{\min}\big(\mathbf{D}_{j+1}^{-1}\mathbf{D}_j\big)$ and note that for every vector $v \in \mathbb{R}^n$ we have that

$$\delta_{j+1}\|v\|_{\mathbf{D}_{j+1}}^2 \leq \|v\|_{\mathbf{D}_j}^2. \tag{5.33}$$

Indeed this follows since

$$\begin{aligned}\|v\|_{\mathbf{D}_j}^2 &= v^\top \mathbf{D}_j v = v^\top \mathbf{D}_{j+1}^{1/2}\big(\mathbf{D}_{j+1}^{-1/2}\mathbf{D}_j\mathbf{D}_{j+1}^{-1/2}\big)\mathbf{D}_{j+1}^{1/2} v \\ &\geq \lambda_{\min}\big(\mathbf{D}_{j+1}^{-1}\mathbf{D}_j\big)\|v\|_{\mathbf{D}_{j+1}}^2 = \delta_{j+1}\|v\|_{\mathbf{D}_{j+1}}^2.\end{aligned}$$

For simplicity, denote $\nabla f_l := \nabla f(x^l; S_l),\ f_l := f(x^l; S_l)$. We have that

$$\begin{aligned}\langle d_j, x^j - x^\star \rangle &= \sum_{l=1}^{j} \rho_{l,j} \langle \nabla f_l, x^j - x^\star \rangle \\ &= \sum_{l=1}^{j} \rho_{l,j}\big(\langle \nabla f_l, x^j - x^l \rangle + \langle \nabla f_l, x^l - x^\star \rangle\big) \\ &\geq \sum_{l=1}^{j} \rho_{l,j}\big(\langle \nabla f_l, x^j - x^l \rangle + f_l - f(x^\star; S_l)\big) \qquad \text{(by convexity of } f(\cdot; s)\text{)} \\ &= \bar{f}_j + \langle d_j, x^j \rangle - \gamma_j - \sum_{l=1}^{j} \rho_{l,j} f(x^\star; S_l) \ = \ h_j - \rho_j \bar{f}_\star^j.\end{aligned} \tag{5.34}$$

Using (5.33) together with (5.34) in (5.32) gives

$$\begin{aligned}\delta_{j+1}\|x^{j+1} - x^\star\|_{\mathbf{D}_{j+1}}^2 &\leq \|x^{j+1} - x^\star\|_{\mathbf{D}_j}^2 \\ &= \|x^j - x^\star\|_{\mathbf{D}_j}^2 - 2\tau_j \langle d_j, x^j - x^\star \rangle + \tau_j^2 \|d_j\|_{\mathbf{D}_j^{-1}}^2 \\ &\leq \|x^j - x^\star\|_{\mathbf{D}_j}^2 - 2\tau_j(h_j - \rho_j \bar{f}_\star^j) + \tau_j^2 \|d_j\|_{\mathbf{D}_j^{-1}}^2.\end{aligned} \tag{5.35}$$

Now we will perform a weighted telescoping. We multiply the above by $\eta_j > 0$ such that $\delta_{j+1}\eta_j = \eta_{j+1}$, thus $\eta_j = \eta_1 \prod_{l=2}^{j} \delta_l$. In doing so, we obtain

$$\eta_{j+1}\|x^{j+1} - x^\star\|_{\mathbf{D}_{j+1}}^2 \leq \eta_j \|x^j - x^\star\|_{\mathbf{D}_j}^2 - 2\eta_j\tau_j(h_j - \rho_j \bar{f}_\star^j) + \eta_j \tau_j^2 \|d_j\|_{\mathbf{D}_j^{-1}}^2.$$

Summing up from $j = 1, \ldots, k$ and telescoping we have that

$$\begin{aligned}0 &\leq \eta_{k+1}\|x^{k+1} - x^\star\|_{\mathbf{D}_{k+1}}^2 \\ &\leq \eta_1 \|x^1 - x^\star\|_{\mathbf{D}_1}^2 - 2\sum_{j=1}^{k} \eta_j\tau_j(h_j - \rho_j \bar{f}_\star^j) + \sum_{j=1}^{k} \eta_j \tau_j^2 \|d_j\|_{\mathbf{D}_j^{-1}}^2.\end{aligned} \tag{5.36}$$

Re-arranging the above, choosing $\eta_1 = 1$ and isolating $\bar{f}_\star^k$ gives

$$2\eta_k\tau_k\rho_k\bar{f}_\star^k \geq 2\sum_{j=1}^{k}\eta_j\tau_j h_j - \|x^1 - x^\star\|_{\mathbf{D}_1}^2 - \sum_{j=1}^{k}\eta_j\tau_j^2\|d_j\|_{\mathbf{D}_j^{-1}}^2 - 2\sum_{j=1}^{k-1}\eta_j\tau_j\rho_j\bar{f}_\star^j.$$

Dividing through by $2\eta_k\tau_k\rho_k$ gives the main result. Finally the recurrence follows since, for $k \geq 2$ we have that

$$\begin{aligned}
f_\star^{k+1} &= \frac{2\sum_{j=1}^{k}\eta_j\tau_j h_j - \|x^1 - x^\star\|_{\mathbf{D}_1}^2 - \sum_{j=1}^{k}\eta_j\tau_j^2\|d_j\|_{\mathbf{D}_j^{-1}}^2 - 2\sum_{j=1}^{k-1}\eta_j\tau_j\rho_j\bar{f}_\star^j}{2\eta_k\tau_k\rho_k} \\
&= \frac{\eta_{k-1}\tau_{k-1}\rho_{k-1}}{\eta_k\tau_k\rho_k}\underbrace{\frac{2\sum_{j=1}^{k-1}\eta_j\tau_j h_j - \|x^1 - x^\star\|_{\mathbf{D}_1}^2 - \sum_{j=1}^{k-1}\eta_j\tau_j^2\|d_j\|_{\mathbf{D}_j^{-1}}^2 - 2\sum_{j=1}^{k-2}\eta_j\tau_j\rho_j\bar{f}_\star^j}{2\eta_{k-1}\tau_{k-1}\rho_{k-1}}}_{=f_\star^k} \\
&\quad + \frac{\eta_{k-1}\tau_{k-1}\rho_{k-1}}{\eta_k\tau_k\rho_k}\frac{2\eta_k\tau_k h_k - \eta_k\tau_k^2\|d_k\|_{\mathbf{D}_k^{-1}}^2 - 2\eta_{k-1}\tau_{k-1}\rho_{k-1}\bar{f}_\star^{k-1}}{2\eta_{k-1}\tau_{k-1}\rho_{k-1}} \\
&= \frac{2\eta_{k-1}\tau_{k-1}\rho_{k-1}(f_\star^k - \bar{f}_\star^{k-1}) - \eta_k\tau_k^2\|d_k\|_{\mathbf{D}_k^{-1}}^2 + 2\eta_k\tau_k h_k}{2\eta_k\tau_k\rho_k}.
\end{aligned}$$

Now bootstrapping by using $f_\star^k \approx \bar{f}_\star^{k-1}$ gives the result. □

5.9.1 Convergence Proofs

We discovered the following interpretation, namely that the adaptive step size term in `MoMo` minimizes an upper bound of the distance to the solution at the new iterate, after reading the concurrent work [156].

Lemma 5.6. *Let $f(\cdot; s)$ be convex for every $s \in \mathcal{S}$. Let $h_k := \bar{f}_k + \langle d_k, x^k\rangle - \gamma_k$ where $d_k, \bar{f}_k$, and γ_k are defined in (5.9). Consider the iterates $x^{k+1} = x^k - \tau_k\mathbf{D}_k^{-1}d_k$ for any sequence of step sizes $(\tau_k)_k \subset \mathbb{R}_{\geq 0}$. Let $x^\star \in \arg\min_{x\in\mathbb{R}^n} f(x)$.*

Then, we have the upper bound

$$\|x^{k+1} - x^\star\|_{\mathbf{D}_k}^2 \leq \|x^k - x^\star\|_{\mathbf{D}_k}^2 - 2\tau_k(h_k - \rho_k\bar{f}_\star^k) + \tau_k^2\|d_k\|_{\mathbf{D}_k^{-1}}^2. \tag{5.37}$$

Moreover, if $d_k \neq 0$, the right-hand side of (5.37) attains its minimum over the set $\tau_k \in \mathbb{R}_{\geq 0}$ at

$$\bar{\tau}_k \;=\; \frac{(h_k - \rho_k\bar{f}_\star^k)_+}{\|d_k\|_{\mathbf{D}_k^{-1}}^2}, \tag{5.38}$$

which is the adaptive term in the step size of `MoMo`.

Proof. Subtracting $x^\star$ from both sides, taking norms and expanding the squares we have that

$$\|x^{k+1} - x^\star\|_{\mathbf{D}_k}^2 = \|x^k - x^\star\|_{\mathbf{D}_k}^2 - 2\tau_k\langle d_k, x^k - x^\star\rangle + \tau_k^2\|d_k\|_{\mathbf{D}_k^{-1}}^2. \tag{5.39}$$

We use again the notation $\nabla f_j := \nabla f(x^j; S_j),\ f_j := f(x^j; S_j)$. Now using that

$$\begin{aligned}
\langle d_k, x^k - x^\star\rangle &= \sum_{j=1}^k \rho_{j,k}\langle \nabla f_j, x^k - x^\star\rangle \\
&= \sum_{j=1}^k \rho_{j,k}\big(\langle \nabla f_j, x^k - x^j\rangle + \langle \nabla f_j, x^j - x^\star\rangle\big) \\
&\geq \sum_{j=1}^k \rho_{j,k}\big(\langle \nabla f_j, x^k - x^j\rangle + f_j - f(x^\star; S_j)\big) \qquad \text{(by convexity of } f(\cdot; S_j)\text{)} \\
&= \langle d_k, x^k\rangle - \gamma_k + \sum_{j=1}^k \rho_{j,k}(f_j - f(x^\star; S_j)) = h_k - \rho_k \bar{f}_\star^k. \qquad (5.40)
\end{aligned}$$

Using (5.40) in (5.39) gives

$$\begin{aligned}
\|x^{k+1} - x^\star\|_{\mathbf{D}_k}^2 &= \|x^k - x^\star\|_{\mathbf{D}_k}^2 - 2\tau_k\langle d_k, x^k - x^\star\rangle + \tau_k^2\|d_k\|_{\mathbf{D}_k^{-1}}^2 \\
&\leq \|x^k - x^\star\|_{\mathbf{D}_k}^2 - 2\tau_k(h_k - \rho_k\bar{f}_\star^k) + \tau_k^2\|d_k\|_{\mathbf{D}_k^{-1}}^2.
\end{aligned}$$

For $d_k \neq 0$, minimizing the right-hand side of the above in $\tau_k \in \mathbb{R}$, the optimality condition is

$$-2(h_k - \rho_k\bar{f}_\star^k) + 2\bar{\tau}_k\|d_k\|_{\mathbf{D}_k^{-1}}^2 = 0 \iff \bar{\tau}_k = \frac{(h_k - \rho_k\bar{f}_\star^k)}{\|d_k\|_{\mathbf{D}_k^{-1}}^2}.$$

Restricting to $\tau_k \geq 0$, we finally arrive at (5.38). □

Proof of Lemma 5.3. Under the assumptions of this lemma, (5.37) holds. As in Lemma 5.6, we denote $h_k = \bar{f}_k + \langle d_k, x^k\rangle - \gamma_k$.

First, consider the case that $d_k = 0$. Then, $\tau_k = 0$ and the statement clearly holds.

Now, assume that $d_k \neq 0$ and $\tau_k = \frac{(h_k - \rho_k\bar{f}_\star^k)_+}{\|d_k\|_{\mathbf{D}_k^{-1}}^2}$. Inserting this τ_k back in (5.37) we have

$$\begin{aligned}
\|x^{k+1} - x^\star\|_{\mathbf{D}_k}^2 &\leq \|x^k - x^\star\|_{\mathbf{D}_k}^2 - 2\frac{(h_k - \rho_k\bar{f}_\star^k)_+}{\|d_k\|_{\mathbf{D}_k^{-1}}^2}(h_k - \rho_k\bar{f}_\star^k) + \frac{(h_k - \rho_k\bar{f}_\star^k)_+^2}{\|d_k\|_{\mathbf{D}_k^{-1}}^2} \\
&= \|x^k - x^\star\|_{\mathbf{D}_k}^2 - \frac{(h_k - \rho_k\bar{f}_\star^k)_+^2}{\|d_k\|_{\mathbf{D}_k^{-1}}^2} \\
&= \|x^k - x^\star\|_{\mathbf{D}_k}^2 - \tau_k(h_k - \rho_k\bar{f}_\star^k)_+. \qquad (5.41)
\end{aligned}$$

Here we used that $a(a)_+ = (a)_+^2$ for any $a \in \mathbb{R}$.

On the other hand, if we have $\tau_k = \frac{\alpha_k}{\rho_k}$, then (5.37) yields

$$\|x^{k+1} - x^\star\|_{\mathbf{D}_k}^2 \leq \|x^k - x^\star\|_{\mathbf{D}_k}^2 + \frac{\alpha_k}{\rho_k}\big[-2(h_k - \rho_k\bar{f}_\star^k) + \frac{\alpha_k}{\rho_k}\|d_k\|_{\mathbf{D}_k^{-1}}^2\big]. \qquad (5.42)$$

In this case $\frac{\alpha_k}{\rho_k} \leq \frac{(h_k - \rho_k\bar{f}_\star^k)_+}{\|d_k\|_{\mathbf{D}_k^{-1}}^2}$ and hence $\frac{\alpha_k}{\rho_k}\|d_k\|_{\mathbf{D}_k^{-1}}^2 \leq (h_k - \rho_k\bar{f}_\star^k)_+$. Further, it must hold $(h_k - \rho_k\bar{f}_\star^k) = (h_k - \rho_k\bar{f}_\star^k)_+$ as $\alpha_k > 0$. We get

$$\begin{aligned}
\|x^{k+1} - x^\star\|_{\mathbf{D}_k}^2 &\leq \|x^k - x^\star\|_{\mathbf{D}_k}^2 - \frac{\alpha_k}{\rho_k}(h_k - \rho_k\bar{f}_\star^k) \\
&= \|x^k - x^\star\|_{\mathbf{D}_k}^2 - \tau_k(h_k - \rho_k\bar{f}_\star^k)_+ \quad (\tau_k = \frac{\alpha_k}{\rho_k}). \qquad (5.43)
\end{aligned}$$

Now, if $\tau_k = \min\{\frac{\alpha_k}{\rho_k}, \frac{(h_k - \rho_k \bar{f}_\star^k)_+}{\|d_k\|^2_{\mathbf{D}_k^{-1}}}\}$, either (5.41) or (5.43) is true, and hence we have

$$\|x^{k+1} - x^\star\|^2_{\tilde{\mathbf{D}}_k} \leq \|x^k - x^\star\|^2_{\tilde{\mathbf{D}}_k} - \tau_k(h_k - \rho_k \bar{f}_\star^k)_+.$$

□

Proof of Theorem 5.4. Recall that Algorithm 12 is Lemma 5.2 with $\rho_k = 1$, $\mathbf{D}_k = \mathbf{Id}$ and $\lambda = 0$. The key quantity is $h_k = \bar{f}_k + \langle d_k, x^k \rangle - \gamma_k$. Let us denote $g_k = \nabla f(x^k; S_k)$. Further, denote with $\mathcal{F}_k$ the filtration generated by $\{S_1, \ldots, S_{k-1}\}$.

Step 1. We first show by induction that $h_k - f^\star \geq 0$ for all $k \in \mathbb{N}$. For $k = 1$, we have $h_1 = f(x^1; S_1) \geq f^\star$ due to the initialization in Algorithm 12, which implies $\langle d_1, x^1 \rangle - \gamma_1 = 0$, and (5.22). Now, for $k \geq 2$, assume that $h_{k-1} - f^\star \geq 0$. Rewrite as

$$\begin{aligned} h_k &= \beta\big[\bar{f}_{k-1} + \langle d_{k-1}, x^k \rangle - \gamma_{k-1}\big] + (1-\beta)\big[f(x^k; S_k) + \langle g_k, x^k \rangle - \langle g_k, x^k \rangle\big] \\ &= \beta\big[\bar{f}_{k-1} + \langle d_{k-1}, x^{k-1} \rangle - \gamma_{k-1} + \langle d_{k-1}, x^k - x^{k-1} \rangle\big] + (1-\beta) f(x^k; S_k) \\ &= \beta h_{k-1} + \beta \langle d_{k-1}, x^k - x^{k-1} \rangle + (1-\beta) f(x^k; S_k). \end{aligned}$$

Using the update rule $x^k = x^{k-1} - \tau_{k-1} d_{k-1}$ in the above gives

$$h_k = \beta(h_{k-1} - \tau_{k-1}\|d_{k-1}\|^2) + (1-\beta) f(x^k; S_k). \tag{5.44}$$

From definition of τ_{k-1}, we have

$$\tau_{k-1}\|d_{k-1}\|^2 \leq (h_{k-1} - f_\star^{k-1})_+ = (h_{k-1} - f^\star)_+ = h_{k-1} - f^\star,$$

where the last equality is the induction hypothesis. Re-arranging the above, we get

$$h_{k-1} - \tau_{k-1}\|d_{k-1}\|^2 \geq f^\star. \tag{5.45}$$

Plugging this inequality into (5.44) gives

$$h_k \geq \beta f^\star + (1-\beta) f(x^k; S_k) \geq f^\star,$$

due to $\beta \in [0, 1)$ and $f(x^k; S_k) \geq f^\star$. This completes the induction, and we have further shown that

$$h_k - f^\star \geq (1-\beta)\big(f(x^k; S_k) - f^\star\big). \tag{5.46}$$

Step 2. Due to (5.22) and $\rho_k = 1$, it holds $\bar{f}_k^\star = f^\star = f_\star^k$. Hence, the assumptions of Lemma 5.3 are satisfied and we can apply (5.21), which implies in particular that the iterates (x^k) are almost surely contained in the bounded set B. By assumption, we conclude that $\|g_j\|^2 \leq G^2$ for all $j \leq k$. Using Jensen for the discrete probability measure induced by $\rho_{j,k}$, we have

$$\|d_k\|^2 = \|\sum_{j=1}^k \rho_{j,k} g_j\|^2 \leq \sum_{j=1}^k \rho_{j,k}\|g_j\|^2 \leq G^2.$$

We will again distinct two cases: first, assume $\tau_k = \frac{h_k - f^\star}{\|d_k\|^2}$ (by Step 1, we can omit $(\cdot)_+$). From (5.21), we have

$$\begin{aligned} \|x^{k+1} - x^\star\|^2 &\leq \|x^k - x^\star\|^2 - \frac{(h_k - f^\star)^2}{\|d_k\|^2} \\ &\leq \|x^k - x^\star\|^2 - \frac{(h_k - f^\star)^2}{G^2} \\ &\leq \|x^k - x^\star\|^2 - \frac{(1-\beta)^2 (f(x^k; S_k) - f^\star)^2}{G^2}. \end{aligned}$$

Now, consider the other case, namely $\tau_k = \alpha_k$ (recall $\rho_k = 1$). From (5.21) we obtain

$$\begin{aligned}\|x^{k+1} - x^\star\|^2 &\leq \|x^k - x^\star\|^2 - \alpha_k(h_k - f^\star) \\ &\leq \|x^k - x^\star\|^2 - \alpha_k(1-\beta)(f(x^k;S_k) - f^\star) \\ &\leq \|x^k - x^\star\|^2 - \frac{(1-\beta)^2(f(x^k;S_k) - f^\star)^2}{G^2}\end{aligned}$$

where we used (5.46), the fact that $\alpha_k \geq \frac{(1-\beta)(f(x^k;S_k)-f^\star)}{G^2}$ by assumption and that $f(x^k;S_k) - f^\star \geq 0$ due to (5.22). In both cases, hence almost surely conditioned on $\mathcal{F}_k$, it holds

$$\|x^{k+1} - x^\star\|^2 \leq \|x^k - x^\star\|^2 - \frac{(1-\beta)^2(f(x^k;S_k) - f^\star)^2}{G^2}.$$

Now, we apply conditional expectation. Jensen's inequality yields

$$\mathbb{E}[(f(x^k;S_k) - f^\star)^2 \mid \mathcal{F}_k] \geq (\mathbb{E}[f(x^k;S_k) - f^\star \mid \mathcal{F}_k])^2 = (f(x^k) - f^\star)^2.$$

Thus, we get

$$\mathbb{E}[\|x^{k+1} - x^\star\|^2 \mid \mathcal{F}_k] \leq \|x^k - x^\star\|^2 - \frac{(1-\beta)^2(f(x^k) - f^\star)^2}{G^2}.$$

Step 3. Taking full expectation, using the law of total expectation, suming over $k = 1, \ldots, K$, dividing by K and re-arranging gives

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[(f(x^k) - f^\star)^2] \leq \frac{G^2\|x^1 - x^\star\|^2}{K(1-\beta)^2}. \tag{5.47}$$

Now, due to Jensen's inequality we have $\mathbb{E}[(f(x^k) - f^\star)^2] \geq \mathbb{E}[f(x^k) - f^\star]^2$ and because the square-root is concave, it holds

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[f(x^k) - f^\star] \leq \sqrt{\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[f(x^k) - f^\star]^2}.$$

Using the above together with (5.47), we obtain

$$\min_{k=1,\ldots,K}\mathbb{E}[f(x^k) - f^\star] \leq \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}[f(x^k) - f^\star] \leq \frac{G\|x^1 - x^\star\|}{\sqrt{K}(1-\beta)}.$$

□

5.9.2 Notes on the Averaging Coefficients

Lemma 5.7. *Let $\beta \in [0,1)$. Let $\rho_{1,1} = 1$, and for $k \geq 2$ let*

$$\rho_{j,k} = \begin{cases}\beta\rho_{j,k-1}, & j \leq k-1, \\ 1-\beta, & j = k.\end{cases}$$

Then, $\sum_{j=1}^{k}\rho_{j,k} = 1$ holds for all $k \in \mathbb{N}$. Further, for an arbitrary sequence $(u_j)_{j\in\mathbb{N}} \subset \mathbb{R}^m$, $m \in \mathbb{N}$, consider the weighted sum

$$\bar{u}_k := \sum_{j=1}^{k}\rho_{j,k}u_j.$$

Then, if $\bar{u}_0 := u_1$ it holds $\bar{u}_k = (1-\beta)u_k + \beta\bar{u}_{k-1}$ for all $k \in \mathbb{N}$.

Proof. We prove that $\sum_{j=1}^{k} \rho_{j,k} = 1$ holds for all $k \in \mathbb{N}$ by induction. For the base case $k = 1$, we have $\rho_{1,1} = 1$ by definition. Assuming that $\sum_{j=1}^{k-1} \rho_{j,k-1} = 1$, we have

$$\sum_{j=1}^{k} \rho_{j,k} = \rho_{k,k} + \sum_{j=1}^{k-1} \rho_{j,k} = 1 - \beta + \beta \sum_{j=1}^{k-1} \rho_{j,k-1} = 1 - \beta + \beta = 1.$$

Consequently, we have $\bar{u}_1 = \rho_{11} u_1 = u_1$, and for $k \geq 2$,

$$\begin{aligned} \bar{u}_k &= \sum_{j=1}^{k} \rho_{j,k} u_j = (1-\beta) u_k + \sum_{j=1}^{k-1} \beta \rho_{j,k-1} u_j = (1-\beta) u_k + \beta \sum_{j=1}^{k-1} \rho_{j,k-1} u_j \\ &= (1-\beta) u_k + \beta \bar{u}_{k-1}. \end{aligned}$$

□

For the choice of $\rho_{j,k}$ in Lemma 5.7, unrolling the recursion, for $k \geq 2$ we obtain the explicit formula

$$\rho_{j,k} = \begin{cases} (1-\beta)\beta^{k-j}, & j \geq 2 \\ \beta^{k-1}, & j = 1. \end{cases} \tag{5.48}$$

Averaging with Bias Correction. Choosing $\rho_{j,k} = (1-\beta)\beta^{k-j}$, we have $\rho_{j,k} = \beta \rho_{j,k-1}$, and $\rho_{k,k} = 1 - \beta$. Hence, we can update $\bar{f}_k = (1-\beta) f(x^k, s_k) + \beta \bar{f}_{k-1}$ and analogously for d_k, γ_k. However, this choice does not satisfy $\sum_{j=1}^{k} \rho_{j,k} = 1$. Indeed using the geometric series gives

$$\rho_k = (1-\beta) \sum_{j=0}^{k-1} \beta^j = 1 - \beta^k.$$

This fact motivates scaling by the factor of $1-\beta^k$ which was termed *debiasing* in `Adam`. Combined with Lemma 5.1, this alternative averaging scheme leads to a variant of `MoMo` with bias correction, presented in Algorithm 16. As the two presented choices of $\rho_{j,k}$ are very similar, we would not expect major differences in their performance (cf. Remark 11) – and indeed did also not observe any such experimentally.

Algorithm 16 `MoMo-Bias`: Model-based Momentum with bias correction.

Require: $x^1 \in \mathbb{R}^n$, $\beta \in [0,\ 1)$, $\alpha_k > 0$, $(f_\star^k)_{k \in \mathbb{N}} \subset \mathbb{R}$.

1: **Initialize:** $\bar{f}_0 = 0,\ d_0 = 0$, and $\gamma_0 = 0$.
2: **for** $k = 1, \ldots, K-1$ **do**
3: $\quad \bar{f}_k = (1-\beta) f(x^k; S_k) + \beta \bar{f}_{k-1}$
4: $\quad d_k = (1-\beta) \nabla f(x^k; S_k) + \beta d_{k-1}$
5: $\quad \gamma_k = (1-\beta) \langle \nabla f(x^k; S_k), x^k \rangle + \beta \gamma_{k-1}$
6: $\quad x^{k+1} = x^k - \min \left\{ \frac{\alpha_k}{1-\beta^k}, \frac{(\bar{f}_k - (1-\beta^k) f_\star^k + \langle d_k, x^k \rangle - \gamma_k)_+}{\|d_k\|^2} \right\} d_k.$
7: **end for**
8: **return** x^K

Remark 11. *Algorithm 16 differs from Algorithm 12 only in two steps: first, the quantities $\bar{f}_0,\ d_0,\ \gamma_0$ are initialized at zero. Secondly, we use $\frac{\alpha_k}{1-\beta^k}$ instead of α_k and $(1-\beta^k) f_\star^k$ instead of $f_\star^k$ in line 6. As $\beta \in [0, 1)$, for late iteration number k, we can expect that both methods behave very similarly.*

5.9.3 Comparison of MoMo-Adam to AdamW

Algorithm 13 naturally compares to AdamW [89]. Note that the update of AdamW (in the notation of Algorithm 13) can be written as

$$x^{k+1} = (1 - \alpha_k \lambda) x^k - \frac{\alpha_k}{1 - \beta_1^k} \mathbf{D}_k^{-1} d_k,$$

Compared to Algorithm 13, line 9, the weight decay of AdamW is not done dividing the whole expression by $\frac{1}{1+\alpha_k\lambda}$, but instead multiplying only x^k with $1 - \alpha_k\lambda$. This is a first-order Taylor approximation [169]: for α small it holds $\frac{1}{1+\alpha\lambda} \approx 1 - \alpha\lambda$ and $\frac{\alpha}{1+\alpha\lambda} \approx \alpha$. If we would want to adapt this approximation, line 9 would be replaced with

$$x^{k+1} = (1 - \lambda\alpha_k) x^k - \min\left\{\tfrac{\alpha_k}{1-\beta_1^k}, \tfrac{((1+\lambda\alpha_k)(\bar{f}_k - (1-\beta_1^k) f_\star^k - \gamma_k) + \langle d_k, x^k\rangle)_+}{\|d_k\|^2_{\mathbf{D}_k^{-1}}}\right\} \mathbf{D}_k^{-1} d_k. \tag{5.49}$$

However, the results of [169] suggest that this approximation has almost no impact on the empirical performance.

5.9.4 Implementation details on MoMo*

Here we give the complete pseudocode for MoMo*, that is the MoMo method that uses the estimator for $f_\star^k$ given in Lemma 5.5.

Algorithm 17 MoMo*: Adaptive learning rates and online estimation of $f^\star$.

Require: $x^1 \in \mathbb{R}^n$, $\beta \in [0,\ 1)$, $\alpha_k > 0$, $\lambda \geq 0$, $f_\star^1 \subset \mathbb{R}$.

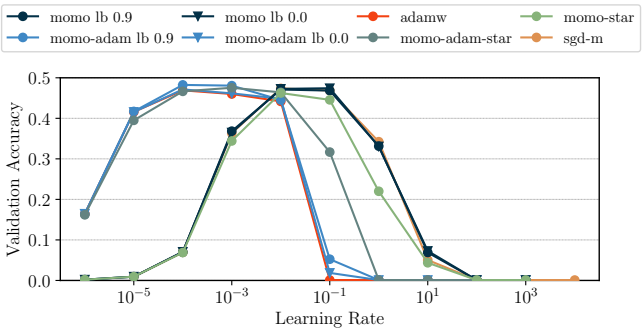
1: **Initialize:** $\bar{f}_0 = f(x^1; S_1)$, $d_0 = \nabla f(x^1; S_1)$ and $\gamma_0 = \langle \nabla f(x^1; S_1), x^1 \rangle$.
2: **for** $k = 1, \ldots, K-1$ **do**
3: $\quad \bar{f}_k = (1-\beta) f(x^k; S_k) + \beta \bar{f}_{k-1}$
4: $\quad \gamma_k = (1-\beta)\langle \nabla f(x^k; S_k), x^k \rangle + \beta \gamma_{k-1}$
5: $\quad d_k = (1-\beta) \nabla f(x^k; S_k) + \beta d_{k-1}$
6: $\quad h_k^\lambda = (1 + \alpha_k \lambda)(\bar{f}_k - \gamma_k) + \langle d_k, x^k \rangle$
7: $\quad f_\star^k = \texttt{ResetStar}(f_\star^k, \alpha_k, \lambda, 1, h_k^\lambda)$
8: $\quad x^{k+1} = \frac{1}{1+\alpha_k\lambda}\left[x^k - \min\left\{\alpha_k, \frac{((1+\alpha_k\lambda)(\bar{f}_k - f_\star^k - \gamma_k) + \langle d_k, x^k\rangle)_+}{\|d_k\|^2}\right\} d_k\right]$
9: $\quad f_\star^{k+1} = \texttt{EstimateStar}(\bar{f}_k, x^k, \gamma_k, \tau_k, d_k, \mathbf{Id}, 1)$.
10: **end for**
11: **return** x^K

5.10 Supplementary Material on Numerical Experiments

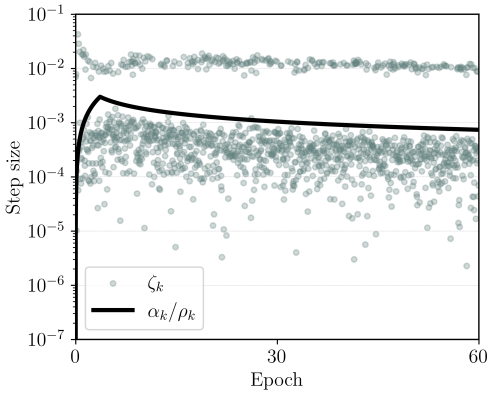
5.10.1 Experimental Setup of Section 5.7.1

We use default choices for momentum parameter $\beta = 0.9$ for MoMo and SGD-M, and $(\beta_1, \beta_2) = (0.9, 0.999)$ for MoMo-Adam and Adam respectively. We set $\lambda = 0$, i.e. no weight decay.

For SGD-M we set the dampening parameter equal to the momentum parameter. Like this, SGD-M does an exponentially-weighted average of past gradients and hence is comparable to MoMo for identical learning rate and momentum. For all other hyperparameters we use the Pytorch default values for Adam and SGD-M (unless explicitly stated otherwise).



(a) `ResNet18` for `Imagenet32`



(b) Encoder-decoder transformer for `IWSLT14`

Figure 5.9: Left: Validation accuracy of a `ResNet18` for `Imagenet32` with weight decay $\lambda = 10^{-4}$. Right: Encoder-decoder transformer for `IWSLT14`. Learning rate schedule (**black**) and adaptive step sizes (**grey dots**) of `MoMo-Adam`* for $\alpha_{\text{base}} = 3 \cdot 10^{-3}$ and dropout 0.1. The black line represents the user-specified learning-rate schedule, using linear warm-up followed by an inverse square-root decay.

5.10.2 Models and Datasets

`ResNet` for `CIFAR` [60]

Used for `ResNet20` for `CIFAR10` and `ResNet110` for `CIFAR100`. We adapt the last layer output size to $\{10, 100\}$ according to the used dataset. We run 50 epochs for `ResNet20` and 100 epochs for `ResNet110`, both with batch size 128.

Model `https://github.com/akamaster/pytorch_resnet_cifar10/blob/master/resnet.py`

`VGG16` for `CIFAR10` [144]

A deep network with 16 convolutional layers. We run 50 epochs with batch size 128.

Model `https://github.com/chengyangfu/pytorch-vgg-cifar10/blob/master/vgg.py`

`ViT` for `CIFAR10` [32]

A small vision transformer, based on the hyperparameter setting proposed in `github.com/kentaroy47/vision-transformers-cifar10`. In particular, we set the patch size to four. We run 200 epochs with batch size 512.

Model `https://github.com/lucidrains/vit-pytorch`

`ResNet18` for `Imagenet32` [60]

`Imagenet32` is a downsampled version of `Imagenet-1k` to images of 32×32 pixels. We adapt the last layer output size to 1000. We run 45 epochs with batch size 128.

Model `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py`

`DLRM` for `Criteo` [151]

`DLRM` is an industry-scale model with over 300 million parameters. the `Criteo` dataset contains

approximately 46 million training samples. We run 300k iterations with batch size 128.

Dataset https://kaggle.com/c/criteo-display-ad-challenge
Model https://github.com/facebookresearch/dlrm

IWSLT14 [110]

The IWSLT14 dataset consists of over 160,000 German and English sentence pairs. We use a transformer with six encoder and decoder blocks from fairseq. The training loss is the cross-entropy loss with label smoothing of 0.1. We use weight decay of $\lambda = 10^{-4}$ (although we noticed that weight decay does not influence the performance of MoMo-Adam), momentum parameters $(\beta_1, \beta_2) = (0.9, 0.98)$. We train for 60 epochs.

Model https://github.com/facebookresearch/fairseq

UNet for Smithsonian Butterflies [128]

The Smithsonian Butterflies dataset contains 1,000 images of butterflies. We train a diffusion model, using a UNet2D architecture from the Huggingface library. For both constant learning-rate schedule as well as cosine decay we use a warmup period, where the learning rate is increased linearly over 500 steps from zero to the final value. For MoMo-Adam we use no weight decay. For Adam we tried both $\lambda \in \{0, 0.01\}$ but did not observe major differences; we display the results for $\lambda = 0.01$ (the default value). We train for 50 epochs with batch size 16.[8]

Dataset https://huggingface.co/datasets/huggan/smithsonian_butterflies_subset
Model https://huggingface.co/docs/diffusers/main/en/api/models/unet2d

ViT for Imagenet-1k [32]

We train a vision transformer for image classification on the full Imagenet-1k dataset. We use the timm library for training and select the vit_tiny_patch16_224 model. We use the settings reported in [28]; the only exception is that when increasing the learning rate α, we decrease the weight decay λ by the same factor, such that $\alpha \cdot \lambda = 10^{-4}$ for all runs. By standard practice, we use a warmup period of five epochs (starting at 10^{-5} with epoch-wise steps) to a base learning rate α_{base}, followed by a cosine decay. We train for 200 epochs with batch size 512. The loss function is the binary cross entropy loss with label smoothing of 0.1 (also used in [28]).

Model timm/models/vision_transformer.py

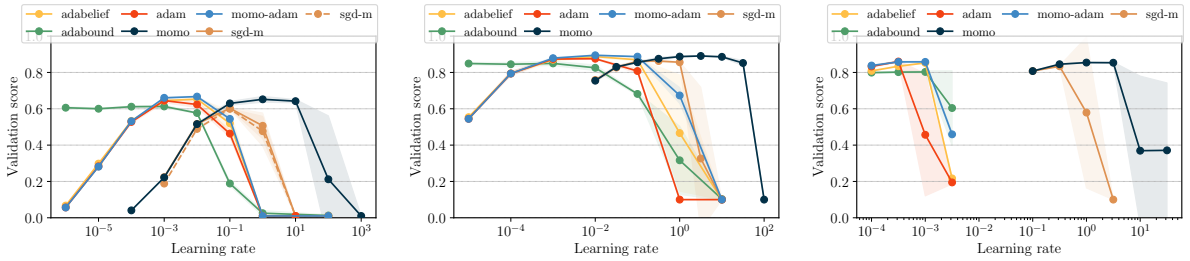
5.10.3 Additional Experiments

We present additional comparisons in Fig. 5.10, including AdaBelief [168] and AdaBound [90]. For ResNet110 on CIFAR100, we also tried SGD-M with a learning-rate schedule that decays by a factor of 0.7 every 30 epochs. This apparently does not yield improvements.

[8] The training script is adapted from https://colab.research.google.com/github/huggingface/notebooks/blob/main/diffusers_doc/en/pytorch/basic_training.ipynb.

	MoMo	MoMo-Adam	SGD-M	Adam(W)
ResNet110 for CIFAR100	65.21 ± 1.61	**66.71** ± 0.31	60.28 ± 0.36	64.5 ± 1.14
ResNet20 for CIFAR10	89.07 ± 0.2	**89.45** ± 0.17	86.27 ± 0.67	87.54 ± 0.26
ViT for CIFAR10	85.43 ± 0.19	85.81 ± 0.57	83.39 ± 0.28	**86.02** ± 0.44
VGG16 for CIFAR10	90.64 ± 0.18	**90.9** ± 0.17	89.81 ± 0.43	89.95 ± 0.67
MLP for MNIST	**97.97** ± 0.08	97.96 ± 0.12	97.73 ± 0.12	97.75 ± 0.06
DLRM for Criteo	78.83 ± 0.038	78.98 ± 0.036	78.81 ± 0.041	**79.05** ± 0.014
ResNet18 for Imagenet32	**47.66**⋆	47.54⋆	47.38	46.98
IWSLT14 (dp 0.1)	N/A	**33.63**⋆	N/A	32.56
IWSLT14 (dp 0.3)	N/A	**35.34**⋆	N/A	34.97
ViT for Imagenet-1k	N/A	**73.83** ± 0.36	N/A	72.83 ± 0.51

Table 5.1: Validation score (with one standard deviation) for the best learning rate choice for each method among the ones displayed in Section 5.7. Best method in bold. Symbol "⋆" indicates usage of online lower bound, otherwise MoMo(-Adam) used with $f_\star^k = 0$.



(a) ResNet110 for CIFAR100 (b) ResNet20 for CIFAR10 (c) ViT for CIFAR10

Figure 5.10: Additional comparisons to AdaBelief and AdaBound. For CIFAR100 (left plot), the dashed line of SGD-M displays result of using a learning-rate schedule that decays by a factor of 0.7 every 30 epochs.

We present the results for the UNet for Smithsonian Butterflies experiment in Figs. 5.11 and 5.12. We try a constant learning rate, as well as a cosine-decay schedule (both schedules with warmup). The cosine decay works better in general.

5.10.4 Illustrative Example of Online Lower Bound Estimation

We show how our online estimation of $f_\star^k$, derived in Section 5.6 and Lemma 5.5, work for a simple example. Consider a regression problem, with synthetic matrix $A \in \mathbb{R}^{200\times 10}$ and $b \in \mathbb{R}^{200}$. We solve the problem $\min_{x\in\mathbb{R}^{10}} \sum_{i=1}^{200} \frac{1}{2}\|a_i^\top x - b_i\|^2$, where a_i are the rows of A. The data is generated in a way such that there exists $\hat{x}$ with $b = A\hat{x}$ and hence the optimal value is $f^\star = 0$.

We now run MoMo(-Adam) with lower bound estimate $f_\star^k = -10$ in all iterations, and MoMo(-Adam)⋆ with initialization $f_\star^1 = -10$. Clearly, this is not a tight estimate of the optimal value $f^\star$. From Fig. 5.14a, we see that online estimation of $f_\star^k$, used in MoMo(-Adam)⋆, improves stability of the training compared to plain MoMo(-Adam) where a constant value $f_\star^k = -10$ is used. From Fig. 5.14b, we also see that the online values of $f_\star^k$ converge to $f^\star = 0$.

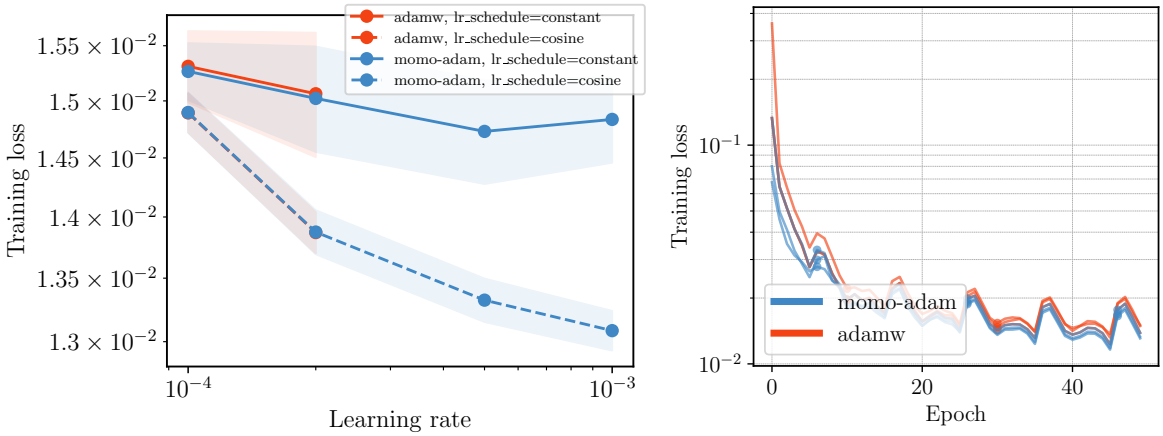


Figure 5.11: Left: Stability over learning rate, for constant and cosine decay schedule. Values are averaged over three repetitions with different seeds, shaded area depicts one standard deviation. Missing points for `Adam` mean that at least one of the three repetitions diverged (or results in `NaN` loss). Right: Training loss curve for the best three settings (across all learning rates and both schedules) for each method.

(a) $\alpha_{\text{base}} = 0.0001$ (b) $\alpha_{\text{base}} = 0.0005$ (c) $\alpha_{\text{base}} = 0.001$

Figure 5.12: Generated images of the `UNet` diffusion model at the end of training with `MoMo-Adam`. We display three different base learning rates for the cosine-decay schedule (i.e. the displayed value of α_{base} corresponds to the x-axis in the left plot of Fig. 5.11). Note that when training with `Adam`, the images in (a) look very similar, but for (b) and (c), `Adam` diverges and thus the model generates no useful images.

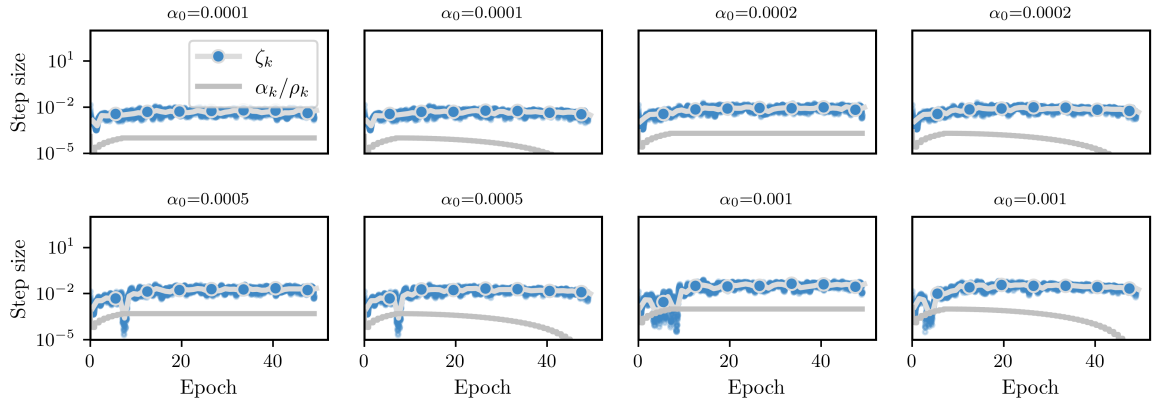
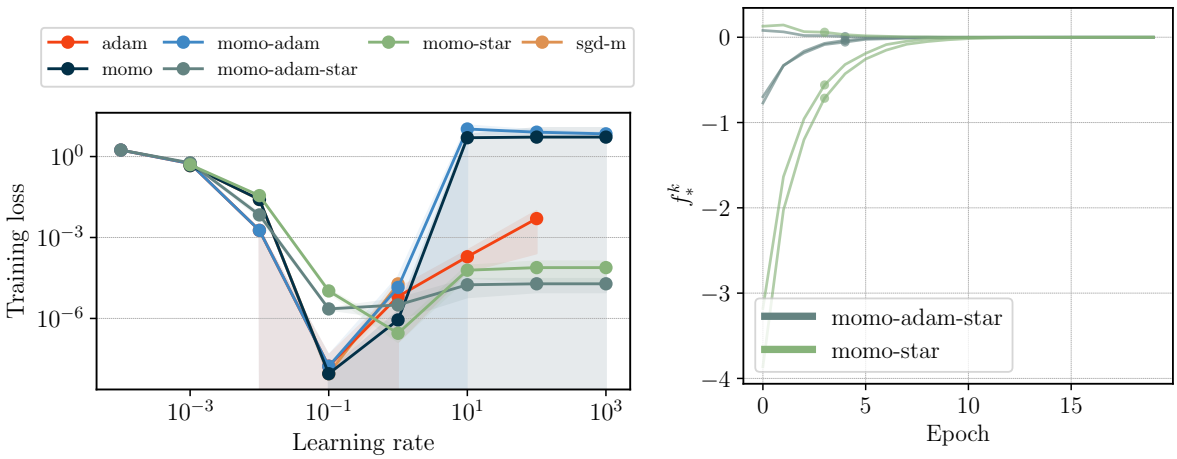


Figure 5.13: `UNet` for `Smithsonian Butterflies`: Adaptive learning rates of `MoMo-Adam` for the constant and cosine decay schedule (grey line), and for different base learning-rate values (indicated by α_0 in the title of each plot).



(a) Training loss

(b) Online estimation of $f^\star = 0$

Figure 5.14: Illustrative example of online lower bound estimation. For all `MoMo` methods, we initialize $f^1_\star = -10$. Left: Training loss for varying (constant) learning rate α_0. Right: Value of $f^k_\star$ over training, one line corresponds to one choice of α_0. We plot per method the three values of α_0 that lead to smallest training loss.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from tensorflow.org. [Cited on page 12]

[2] A. Ajalloeian and S. U. Stich, *On the convergence of SGD with biased gradients*, (2020). [Cited on page 15]

[3] S. Amari, *A theory of adaptive pattern classifiers*, IEEE Transactions on Electronic Computers, EC-16 (1967), pp. 299–307. [Cited on page 12]

[4] A. Aravkin, M. P. Friedlander, F. J. Herrmann, and T. van Leeuwen, *Robust inversion, dimensionality reduction, and randomized sampling*, Math Program, 134 (2012), pp. 101–125. [Cited on page 43]

[5] H. Asi and J. C. Duchi, *The importance of better models in stochastic optimization*, Proc. Natl. Acad. Sci. U. S. A., 116 (2019), pp. 22924–22930. [Cited on page 24]

[6] ——, *Stochastic (approximate) proximal point methods: convergence, optimality, and adaptivity*, SIAM J Optim, 29 (2019), pp. 2257–2290. [Cited on pages 2, 24, 25, 26, 28, 30, 31, 32, 39, 40, 68, 69, 70, 71, 102, 103, and 109]

[7] M. Barré, A. Taylor, and A. d'Aspremont, *Complexity guarantees for Polyak steps with momentum*, in Proceedings of Thirty Third Conference on Learning Theory, J. Abernethy and S. Agarwal, eds., vol. 125 of Proceedings of Machine Learning Research, PMLR, 09–12 Jul 2020, pp. 452–478. [Cited on page 103]

[8] H. H. Bauschke and P. L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, Springer, New York, 2011. With a foreword by Hédy Attouch. [Cited on pages 5, 6, 7, and 30]

[9] A. Beck, *First-order methods in optimization*, vol. 25 of MOS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Optimization Society, Philadelphia, PA, 2017. [Cited on pages 5, 6, 7, 8, 10, 32, 34, 44, 57, 60, 61, 62, and 94]

[10] L. Berrada, A. Zisserman, and M. P. Kumar, *Training neural networks for and by interpolation*, (2019). [Cited on pages 68, 69, 70, and 103]

[11] D. P. Bertsekas, *Stochastic optimization problems with nondifferentiable cost functionals*, J Optim Theory Appl, 12 (1973), pp. 218–231. [Cited on pages 15 and 77]

[12] ——, *Incremental proximal methods for large scale convex optimization*, Math Program, 129 (2011), pp. 163–195. [Cited on pages 2 and 28]

[13] J. Bolte, A. Daniilidis, and A. Lewis, *Tame functions are semismooth*, Math Program, 117 (2009), pp. 5–19. [Cited on page 30]

[14] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *A family of variable metric proximal methods*, Math Program, 68 (1995), pp. 15–47. [Cited on page 31]

[15] L. Bottou, *Large-scale machine learning with stochastic gradient descent*, in Proceedings of COMPSTAT'2010, Physica-Verlag/Springer, Heidelberg, 2010, pp. 177–186. [Cited on page 1]

[16] L. Bottou, F. E. Curtis, and J. Nocedal, *Optimization methods for large-scale machine learning*, SIAM Rev, 60 (2018), pp. 223–311. [Cited on pages 1, 13, 15, 18, and 19]

[17] S. Boyd and E. K. Ryu, *Stochastic proximal iteration: a non-asymptotic improvement upon stochastic gradient descent*. 2014. [Cited on pages 2 and 28]

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, *Language models are few-shot learners*, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Cited on pages 1, 12, and 68]

[19] K. Chen, A. Cutkosky, and F. Orabona, *Implicit parameter-free online learning with truncated linear models*, in Proceedings of The 33rd International Conference on Algorithmic Learning Theory, S. Dasgupta and N. Haghtalab, eds., vol. 167 of Proceedings of Machine Learning Research, PMLR, 29 Mar–01 Apr 2022, pp. 148–175. [Cited on page 103]

[20] L. Chizat and F. Bach, *On the global convergence of gradient descent for over-parameterized models using optimal transport*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018. [Cited on page 2]

[21] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, *Flexible, high performance convolutional neural networks for image classification*, in Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI'11, Barcelona, Catalonia, Spain, 2011, AAAI Press, p. 1237–1242. [Cited on page 13]

[22] F. H. Clarke, *Optimization and nonsmooth analysis*, Canadian Mathematical Society Series of Monographs and Advanced Texts, John Wiley & Sons, Inc., New York, 1983. A Wiley-Interscience Publication. [Cited on pages 5, 6, and 15]

[23] L. Condat, *A direct algorithm for 1-d total variation denoising*, IEEE Signal Processing Letters, 20 (2013), pp. 1054–1057. [Cited on page 60]

[24] M. Cranmer, A. Sanchez Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, *Discovering symbolic models from deep learning with inductive*

biases, in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., vol. 33, Curran Associates, Inc., 2020, pp. 17429–17442. [Cited on page 1]

[25] G. E. Dahl, F. Schneider, Z. Nado, N. Agarwal, C. S. Sastry, P. Hennig, S. Medapati, R. Eschenhagen, P. Kasimbeg, D. Suo, J. Bae, J. Gilmer, A. L. Peirson, B. Khan, R. Anil, M. Rabbat, S. Krishnan, D. Snider, E. Amid, K. Chen, C. J. Maddison, R. Vasudev, M. Badura, A. Garg, and P. Mattson, *Benchmarking neural network training algorithms*, (2023). [Cited on pages 2 and 101]

[26] D. Davis and D. Drusvyatskiy, *Stochastic model-based minimization of weakly convex functions*, SIAM J Optim, 29 (2019), pp. 207–239. [Cited on pages 2, 6, 8, 14, 17, 18, 24, 25, 26, 28, 30, 31, 32, 39, 40, 61, 62, 70, 75, 76, 77, 102, 103, and 109]

[27] A. Defazio, F. Bach, and S. Lacoste-Julien, *Saga: A fast incremental gradient method with support for non-strongly convex composite objectives*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds., vol. 27, Curran Associates, Inc., 2014, pp. 1646–1654. [Cited on pages 2, 19, 21, 30, and 31]

[28] A. Defazio and K. Mishchenko, *Learning-rate-free learning by D-adaptation*, in Proceedings of the 40th International Conference on Machine Learning, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds., vol. 202 of Proceedings of Machine Learning Research, PMLR, 23–29 Jul 2023, pp. 7449–7479. [Cited on pages 2, 103, and 127]

[29] A. Défossez, L. Bottou, F. Bach, and N. Usunier, *A simple convergence proof of Adam and Adagrad*, Transactions on Machine Learning Research, (2022). [Cited on page 22]

[30] U. K. Deiters and R. Macías-Salinas, *Calculation of Densities from Cubic Equations of State: Revisited*, Industrial & Engineering Chemistry Research, 53 (2014), pp. 2529–2536. [Cited on page 43]

[31] D. L. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2006), pp. 1289–1306. [Cited on page 59]

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, in 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, OpenReview.net, 2021. [Cited on pages 1, 13, 68, 114, 126, and 127]

[33] D. Drusvyatskiy, *The proximal point method revisited*, (2017). [Cited on page 28]

[34] D. Drusvyatskiy and A. S. Lewis, *Error bounds, quadratic growth, and linear convergence of proximal methods*, Math Oper Res, 43 (2018), pp. 919–948. [Cited on page 48]

[35] D. Drusvyatskiy and C. Paquette, *Efficiency of minimizing compositions of convex functions and smooth maps*, Math Program, 178 (2019), pp. 503–558. [Cited on pages 14, 61, and 62]

[36] J. Duchi, E. Hazan, and Y. Singer, *Adaptive subgradient methods for online learning and stochastic optimization*, J Mach Learn Res, 12 (2011), pp. 2121–2159. [Cited on pages 2, 22, and 103]

[37] J. C. Duchi and F. Ruan, *Stochastic methods for composite and weakly convex optimization problems*, SIAM J Optim, 28 (2018), pp. 3229–3259. [Cited on pages 2 and 24]

[38] R. Durrett, *Probability—theory and examples*, Cambridge University Press, Cambridge, 2019. [Cited on page 49]

[39] J. Eckstein and D. P. Bertsekas, *On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math Program, 55 (1992), pp. 293–318. [Cited on page 28]

[40] L. Espeholt, S. Agrawal, C. Sønderby, M. Kumar, J. Heek, C. Bromberg, C. Gazen, R. Carver, M. Andrychowicz, J. Hickey, A. Bell, and N. Kalchbrenner, *Deep learning for twelve hour precipitation forecasts*, Nat Commun, 13 (2022). [Cited on pages 1 and 68]

[41] F. Facchinei, *Minimization of* SC^1 *functions and the Maratos effect*, Oper Res Lett, 17 (1995), pp. 131–137. [Cited on pages 34 and 35]

[42] F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems. Vol. I*, Springer Series in Operations Research, Springer-Verlag, New York, 2003. [Cited on page 5]

[43] ——, *Finite-dimensional variational inequalities and complementarity problems. Vol. II*, Springer Series in Operations Research, Springer-Verlag, New York, 2003. [Cited on pages 5, 9, and 33]

[44] K. Fountoulakis and J. Gondzio, *A second-order method for strongly convex* ℓ_1*-regularization problems*, Math Program, 156 (2016), pp. 189–219. [Cited on page 55]

[45] K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*, Biol Cybernet, 36 (1980), pp. 193–202. [Cited on page 13]

[46] G. Garrigos and R. M. Gower, *Handbook of convergence theorems for (stochastic) gradient methods*, (2023). [Cited on pages 18, 21, 87, and 110]

[47] S. Ghadimi and G. Lan, *Stochastic first- and zeroth-order methods for nonconvex stochastic programming*, SIAM J Optim, 23 (2013), pp. 2341–2368. [Cited on pages 15 and 31]

[48] S. Ghadimi, G. Lan, and H. Zhang, *Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization*, Math Program, 155 (2016), pp. 267–305. [Cited on pages 14, 15, 18, 29, 30, and 31]

[49] R. Goebel and R. T. Rockafellar, *Local strong convexity and local Lipschitz continuity of the gradient of convex functions*, J Convex Anal, 15 (2008), pp. 263–270. [Cited on pages 8 and 30]

[50] E. Gorbunov, F. Hanzely, and P. Richtarik, *A unified theory of SGD: Variance reduction, sampling, quantization and coordinate descent*, in Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, S. Chiappa and R. Calandra, eds., vol. 108 of Proceedings of Machine Learning Research, PMLR, 26–28 Aug 2020, pp. 680–690. [Cited on page 19]

[51] R. Gower, O. Sebbouh, and N. Loizou, *SGD for structured nonconvex functions: Learning rates, minibatching and interpolation*, in Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, A. Banerjee and K. Fukumizu, eds.,

vol. 130 of Proceedings of Machine Learning Research, PMLR, 13–15 Apr 2021, pp. 1315–1323. [Cited on pages 67, 70, 103, and 105]

[52] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik, *SGD: General analysis and improved rates*, in Proceedings of the 36th International Conference on Machine Learning, K. Chaudhuri and R. Salakhutdinov, eds., vol. 97 of Proceedings of Machine Learning Research, PMLR, 09–15 Jun 2019, pp. 5200–5209. [Cited on pages 21 and 22]

[53] R. M. Gower, M. Schmidt, F. Bach, and P. Richtárik, *Variance-reduced methods for machine learning*, Proceedings of the IEEE, 108 (2020), pp. 1968–1983. [Cited on page 19]

[54] O. Güler, *On the convergence of the proximal point algorithm for convex minimization*, SIAM J Control Optim, 29 (1991), pp. 403–419. [Cited on page 28]

[55] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, *Result analysis of the NIPS 2003 feature selection challenge*, in Advances in Neural Information Processing Systems, L. Saul, Y. Weiss, and L. Bottou, eds., vol. 17, MIT Press, 2005, pp. 545–552. [Cited on page 39]

[56] F. Hanzely, K. Mishchenko, and P. Richtarik, *Sega: Variance reduction via gradient sketching*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018, pp. 2082–2093. [Cited on pages 2 and 31]

[57] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, Springer Series in Statistics, Springer, New York, second ed., 2009. Data mining, inference, and prediction. [Cited on pages 12, 38, 54, and 60]

[58] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity*, vol. 143 of Monographs on Statistics and Applied Probability, CRC Press, Boca Raton, FL, 2015. The lasso and generalizations. [Cited on pages 2, 12, 58, and 60]

[59] E. Hazan and S. Kakade, *Revisiting the Polyak step size*, (2019). [Cited on pages 3, 68, and 69]

[60] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. [Cited on pages 13, 83, and 126]

[61] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of convex analysis*, Grundlehren Text Editions, Springer-Verlag, Berlin, 2001. Abridged version of ıt Convex analysis and minimization algorithms. I [Springer, Berlin, 1993; MR1261420 (95m:90001)] and ıt II [ibid.; MR1295240 (95m:90002)]. [Cited on page 5]

[62] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen, *Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data*, Appl Math Optim, 11 (1984), pp. 43–56. [Cited on page 30]

[63] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, O. Vinyals, J. Rae, and L. Sifre, *An empirical analysis of compute-optimal large language model training*, in Advances in Neural Information Processing Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., vol. 35, Curran Associates, Inc., 2022, pp. 30016–30030. [Cited on page 1]

[64] P. J. Huber, *Robust statistics*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, Inc., New York, 1981. [Cited on page 43]

[65] C. Irrgang, N. Boers, M. Sonnewald, E. A. Barnes, C. Kadow, J. Staneva, and J. Saynisch-Wagner, *Towards neural Earth system modelling by integrating artificial intelligence in Earth system science*, Nature Machine Intelligence, 3 (2021), pp. 667–674. [Cited on page 1]

[66] A. G. Ivakhnenko, V. G. Lapa, and R. N. Mcdonough, *Cybernetics and forecasting techniques*, 1967. [Cited on page 12]

[67] M. Ivgi, O. Hinder, and Y. Carmon, *DoG is SGD's best friend: A parameter-free dynamic step size schedule*, in Proceedings of the 40th International Conference on Machine Learning, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds., vol. 202 of Proceedings of Machine Learning Research, PMLR, 23–29 Jul 2023, pp. 14465–14499. [Cited on page 2]

[68] S. J. Reddi, S. Sra, B. Poczos, and A. J. Smola, *Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization*, in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds., Curran Associates, Inc., 2016, pp. 1145–1153. [Cited on pages 2, 20, 21, 29, 30, 31, 36, 45, and 46]

[69] A. Jacot, F. Gabriel, and C. Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds., vol. 31, Curran Associates, Inc., 2018. [Cited on page 2]

[70] H. Jiang and L. Q. Qi, *Local uniqueness and convergence of iterative methods for nonsmooth variational inequalities*, J Math Anal Appl, 196 (1995), pp. 314–331. [Cited on page 33]

[71] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in Advances in Neural Information Processing Systems, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds., vol. 26, Curran Associates, Inc., 2013, pp. 315–323. [Cited on pages 2, 19, and 20]

[72] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, *Highly accurate protein structure prediction with alphafold*, Nature, 596 (2021), pp. 583–589. [Cited on page 1]

[73] J. Kiefer and J. Wolfowitz, *Stochastic estimation of the maximum of a regression function*, Ann. Math. Statistics, 23 (1952), pp. 462–466. [Cited on pages 11 and 14]

[74] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2015. [Cited on pages 1, 2, 22, 23, 69, 103, and 107]

[75] K. Koh, S.-J. Kim, and S. Boyd, *An interior-point method for large-scale l_1-regularized logistic regression*, J Mach Learn Res, 8 (2007), pp. 1519–1555. [Cited on page 38]

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds., vol. 25, Curran Associates, Inc., 2012. [Cited on page 13]

[77] A. Krogh and J. Hertz, *A simple weight decay can improve generalization*, in Advances in Neural Information Processing Systems, J. Moody, S. Hanson, and R. Lippmann, eds., vol. 4, Morgan-Kaufmann, 1991. [Cited on pages 2, 60, and 108]

[78] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia, *Learning skillful medium-range global weather forecasting*, Science, (2023). [Cited on page 1]

[79] K. L. Lange, R. J. A. Little, and J. M. G. Taylor, *Robust statistical modeling using the t distribution*, J Amer Statist Assoc, 84 (1989), pp. 881–896. [Cited on page 43]

[80] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time Series*, MIT Press, Cambridge, MA, USA, 1998, p. 255–258. [Cited on page 13]

[81] Y. LeCun, Y. Bengio, and G. Hinton, *Deep Learning*, Nature, 521 (2015), pp. 436–444. [Cited on pages 12 and 13]

[82] Y. LeCun and C. Cortes, *MNIST handwritten digit database*, (2010). [Cited on page 38]

[83] A. S. Lewis and S. J. Wright, *A proximal method for composite minimization*, Math Program, 158 (2015), pp. 501–546. [Cited on page 62]

[84] X. Li, D. Sun, and K.-C. Toh, *A highly efficient semismooth Newton augmented Lagrangian method for solving lasso problems*, SIAM J Optim, 28 (2018), pp. 433–458. [Cited on pages 2, 28, 29, 32, 34, and 60]

[85] ——, *On efficiently solving the subproblems of a level-set method for fused lasso problems*, SIAM J Optim, 28 (2018), pp. 1842–1866. [Cited on page 60]

[86] Y. Lin and S. Hu, *B-subdifferential of the projection onto the generalized spectraplex*, J Optim Theory Appl, 192 (2022), pp. 702–724. [Cited on page 61]

[87] S. L. Lohr, *Sampling: design and analysis*, Brooks/Cole, Cengage Learning, Boston, MA, second ed., 2010. [Cited on pages 32 and 45]

[88] N. Loizou, S. Vaswani, I. Hadj Laradji, and S. Lacoste-Julien, *Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence*, in Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, A. Banerjee and K. Fukumizu, eds., vol. 130 of Proceedings of Machine Learning Research, PMLR, 13–15 Apr 2021, pp. 1306–1314. [Cited on pages 3, 68, 69, 70, 71, 74, 78, 79, 80, 95, 96, 99, 103, and 111]

[89] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, in 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019. [Cited on pages 1, 2, 23, 68, 70, 103, and 125]

[90] L. Luo, Y. Xiong, Y. Liu, and X. Sun, *Adaptive gradient methods with dynamic bound of learning rate*, in Proceedings of the 7th International Conference on Learning Representations, May 2019. [Cited on page 127]

[91] S. Ma, R. Bassily, and M. Belkin, *The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning*, in Proceedings of the 35th International Conference on Machine Learning, J. Dy and A. Krause, eds., vol. 80 of Proceedings of Machine Learning Research, PMLR, 10–15 Jul 2018, pp. 3325–3334. [Cited on pages 3, 21, 67, and 105]

[92] B. Martinet, *Regularisation d'inéquations variationelles par approximations successives*, Rev. Française Inf. Rech. Oper., (1970), pp. 154–159. [Cited on pages 27 and 31]

[93] ——, *Determination approchée d'un point fixe d'une application pseudo-contractante*, C.R. Acad. Sci. Paris, 274 (1972), pp. 163–165. [Cited on pages 27 and 31]

[94] H. B. McMahan and M. Streeter, *Adaptive bound optimization for online convex optimization*, Proceedings of the 23rd Annual Conference on Learning Theory (COLT) 2010, (2010). [Cited on pages 2 and 22]

[95] S. Y. Meng and R. M. Gower, *A model-based method for minimizing CVaR and beyond*, in Proceedings of the 40th International Conference on Machine Learning, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds., vol. 202 of Proceedings of Machine Learning Research, PMLR, 23–29 Jul 2023, pp. 24436–24456. [Cited on page 103]

[96] R. Mifflin, *Semismooth and semiconvex functions in constrained optimization*, SIAM J Control Optim, 15 (1977), pp. 959–972. [Cited on page 9]

[97] A. Milzarek, *Numerical methods and second order theory for nonsmooth problems*, PhD thesis, Technische Universität München, 2016. [Cited on pages 7, 30, and 33]

[98] A. Milzarek, F. Schaipp, and M. Ulbrich, *A semismooth Newton stochastic proximal point algorithm with variance reduction*, SIAM J Optim, 34 (2024), pp. 1157–1185. [Cited on pages 3, 27, and 62]

[99] A. Milzarek, X. Xiao, S. Cen, Z. Wen, and M. Ulbrich, *A stochastic semismooth Newton method for nonsmooth nonconvex optimization*, SIAM J Optim, 29 (2019), pp. 2916–2948. [Cited on pages 22 and 31]

[100] J.-J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bull Soc Math France, 93 (1965), pp. 273–299. [Cited on page 8]

[101] E. Moulines and F. Bach, *Non-asymptotic analysis of stochastic approximation algorithms for machine learning*, in Advances in Neural Information Processing Systems, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, eds., vol. 24, Curran Associates, Inc., 2011. [Cited on page 22]

[102] D. Needell, N. Srebro, and R. Ward, *Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm*, Math Program, 155 (2016), pp. 549–573. [Cited on page 22]

[103] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, *Robust stochastic approximation approach to stochastic programming*, SIAM J Optim, 19 (2008), pp. 1574–1609. [Cited on pages 15 and 18]

[104] A. S. Nemirovsky and D. B. a. Yudin, *Problem complexity and method efficiency in optimization*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1983. Translated from the Russian and with a preface by E. R. Dawson, Wiley-Interscience Series in Discrete Mathematics. [Cited on page 1]

[105] Y. Nesterov, *Gradient methods for minimizing composite functions*, Math Program, 140 (2013), pp. 125–161. [Cited on pages 29 and 48]

[106] Y. Nesterov, *Lectures on convex optimization*, vol. 137 of Springer Optimization and Its Applications, Springer, Cham, 2018. Second edition of [MR2142598]. [Cited on page 5]

[107] F. Orabona and D. Pál, *Coin betting and parameter-free online learning*, in Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds., 2016, pp. 577–585. [Cited on pages 2 and 103]

[108] F. Orabona and T. Tommasi, *Training deep networks without learning rates through coin betting*, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017. [Cited on page 103]

[109] A. Orvieto, S. Lacoste-Julien, and N. Loizou, *Dynamics of SGD with stochastic Polyak stepsizes: Truly adaptive variants and convergence to exact solution*, in Advances in Neural Information Processing Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., vol. 35, Curran Associates, Inc., 2022, pp. 26943–26954. [Cited on pages 3, 69, 78, 79, 99, and 103]

[110] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, *fairseq: A fast, extensible toolkit for sequence modeling*, in Proceedings of NAACL-HLT 2019: Demonstrations, 2019. [Cited on pages 117 and 127]

[111] A. Paren, L. Berrada, R. P. K. Poudel, and M. P. Kumar, *A stochastic bundle method for interpolation*, J Mach Learn Res, 23 (2022), pp. 1–57. [Cited on pages 69, 102, and 103]

[112] L. A. Parente, P. A. Lotito, and M. V. Solodov, *A class of inexact variable metric proximal point algorithms*, SIAM J Optim, 19 (2008), pp. 240–260. [Cited on page 31]

[113] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, in Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035. [Cited on pages 12 and 79]

[114] A. Patrascu and I. Necoara, *Nonasymptotic convergence of stochastic proximal point methods for constrained convex optimization*, J Mach Learn Res, 18 (2017), pp. Paper No. 198, 42. [Cited on pages 2 and 28]

[115] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, J Mach Learn Res, 12 (2011), pp. 2825–2830. [Cited on page 38]

[116] B. T. Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics, 4 (1964), pp. 1–17. [Cited on pages 3 and 102]

[117] B. T. Polyak, *Introduction to optimization*, Translations Series in Mathematics and Engineering, Optimization Software, Inc., Publications Division, New York, 1987. Translated from the Russian, With a foreword by Dimitri P. Bertsekas. [Cited on pages 3, 68, 69, and 103]

[118] M. Prazeres and A. M. Oberman, *Stochastic gradient descent with Polyak's learning rate*, J Sci Comput, 89 (2021), pp. Paper No. 25, 16. [Cited on page 69]

[119] L. Q. Qi and J. Sun, *A nonsmooth version of Newton's method*, Math Program, 58 (1993), pp. 353–367. [Cited on page 9]

[120] S. J. Reddi, S. Kale, and S. Kumar, *On the convergence of Adam and beyond*, in International Conference on Learning Representations, 2018. [Cited on page 22]

[121] L. Rivera-Muñoz, A. Giraldo-Forero, and J. Martinez-Vargas, *Deep matrix factorization models for estimation of missing data in a low-cost sensor network to measure air quality*, Ecol Inform, 71 (2022), p. 101775. [Cited on page 82]

[122] H. Robbins and S. Monro, *A stochastic approximation method*, Ann. Math. Statistics, 22 (1951), pp. 400–407. [Cited on pages 1, 11, 14, and 19]

[123] R. T. Rockafellar, *Convex analysis*, Princeton Mathematical Series, No. 28, Princeton University Press, Princeton, N.J., 1970. [Cited on pages 5, 6, and 8]

[124] ——, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math Oper Res, 1 (1976), pp. 97–116. [Cited on pages 2, 27, 28, and 31]

[125] ——, *Monotone operators and the proximal point algorithm*, SIAM J Control Optim, 14 (1976), pp. 877–898. [Cited on pages 2, 27, and 28]

[126] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*, vol. 317 of Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1998. [Cited on pages 5, 6, 7, 29, and 60]

[127] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022. [Cited on page 13]

[128] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, in Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 234–241. [Cited on page 127]

[129] L. Rosasco, S. Villa, and B. C. Vũ, *Convergence of stochastic proximal gradient algorithm*, Appl Math Optim, 82 (2020), pp. 891–917. [Cited on page 18]

[130] F. Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review, 65 6 (1958), pp. 386–408. [Cited on page 12]

[131] V. Roulet and M. Blondel, *Dual gauss-newton directions for deep learning*, (2023). [Cited on page 62]

[132] F. Schaipp, R. M. Gower, and M. Ulbrich, *A stochastic proximal Polyak step size*, Transactions on Machine Learning Research, (2023). Reproducibility Certification. [Cited on pages 3, 67, 73, 88, and 103]

[133] F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower, *Pytorch implementation of MoMo methods*. https://github.com/fabian-sp/MoMo, 2023. Accessed: 2023-09-15. [Cited on pages 3 and 112]

[134] F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower, *MoMo: Momentum models for adaptive learning rates*, in Proceedings of the 41st International Conference on Machine Learning, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, eds., vol. 235 of Proceedings of Machine Learning Research, PMLR, 21–27 Jul 2024, pp. 43542–43570. [Cited on pages 3 and 101]

[135] F. Schaipp, O. Vlasovets, and C. L. Müller, *Gglasso - a Python package for general graphical lasso computation*, Journal of Open Source Software, 6 (2021), p. 3865. [Cited on page 60]

[136] M. Schmidt, N. Le Roux, and F. Bach, *Minimizing finite sums with the stochastic average gradient*, Math Program, 162 (2017), pp. 83–112. [Cited on page 19]

[137] R. M. Schmidt, F. Schneider, and P. Hennig, *Descending through a crowded valley - benchmarking deep learning optimizers*, in Proceedings of the 38th International Conference on Machine Learning, M. Meila and T. Zhang, eds., vol. 139 of Proceedings of Machine Learning Research, PMLR, 18–24 Jul 2021, pp. 9367–9376. [Cited on pages 2, 39, 67, and 112]

[138] B. Schölkopf and A. J. Smola, *Learning with Kernels: support vector machines, regularization, optimization, and beyond*, Adaptive computation and machine learning series, MIT Press, 2002. [Cited on page 12]

[139] O. Sebbouh, R. M. Gower, and A. Defazio, *Almost sure convergence rates for stochastic gradient descent and stochastic heavy ball*, in Proceedings of Thirty Fourth Conference on Learning Theory, M. Belkin and S. Kpotufe, eds., vol. 134 of Proceedings of Machine Learning Research, PMLR, 15–19 Aug 2021, pp. 3935–3971. [Cited on pages 102 and 118]

[140] S. Shalev-Shwartz and T. Zhang, *Stochastic dual coordinate ascent methods for regularized loss minimization*, J Mach Learn Res, 14 (2013), p. 567–599. [Cited on page 19]

[141] A. Shapiro, *On concepts of directional differentiability*, J Optim Theory Appl, 66 (1990), pp. 477–487. [Cited on page 33]

[142] A. Shtoff, *Efficient implementation of incremental proximal-point methods*, (2022). [Cited on page 62]

[143] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, *A sparse-group Lasso*, J Comput Graph Statist, 22 (2013), pp. 231–245. [Cited on pages 2 and 43]

[144] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2015. [Cited on pages 13 and 126]

[145] S. Sra, S. Nowozin, and S. J. Wright, *Optimization for Machine Learning*, The MIT Press, 2011. [Cited on page 1]

[146] N. Srebro, J. Rennie, and T. Jaakkola, *Maximum-margin matrix factorization*, in Advances in Neural Information Processing Systems, L. Saul, Y. Weiss, and L. Bottou, eds., vol. 17, MIT Press, 2004. [Cited on page 80]

[147] R.-Y. Sun, *Optimization for deep learning: An overview*, J Oper Res Soc China, 8 (2020), pp. 249–294. [Cited on page 104]

[148] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, *On the importance of initialization and momentum in deep learning*, in Proceedings of the 30th International Conference on Machine Learning, S. Dasgupta and D. McAllester, eds., vol. 28 of Proceedings of Machine Learning Research, Atlanta, Georgia, USA, 17–19 Jun 2013, PMLR, pp. 1139–1147. [Cited on page 3]

[149] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to sequence learning with neural networks*, in Advances in Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, eds., vol. 27, Curran Associates, Inc., 2014. [Cited on page 1]

[150] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. Roy. Statist. Soc. Ser. B, 58 (1996), pp. 267–288. [Cited on pages 12, 42, and 59]

[151] J.-B. Tien and O. Chapelle, *Display advertising challenge*, 2014. [Cited on pages 113 and 126]

[152] P. Toulis, T. Horel, and E. M. Airoldi, *The proximal robbins–monro method*, Journal of the Royal Statistical Society Series B: Statistical Methodology, 83 (2020), pp. 188–212. [Cited on pages 2 and 28]

[153] M. Ulbrich, *Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces*, vol. 11 of MOS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Optimization Society, Philadelphia, PA, 2011. [Cited on pages 5, 9, 30, 33, 34, and 60]

[154] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds., vol. 30, Curran Associates, Inc., 2017. [Cited on pages 13 and 117]

[155] S. Vaswani, A. Mishkin, I. H. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, *Painless stochastic gradient: Interpolation, line-search, and convergence rates*, in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, eds., 2019, pp. 3727–3740. [Cited on page 103]

[156] X. Wang, M. Johansson, and T. Zhang, *Generalized Polyak step size for first order optimization with momentum*, in Proceedings of the 40th International Conference on Machine Learning, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds., vol. 202 of Proceedings of Machine Learning Research, PMLR, 23–29 Jul 2023, pp. 35836–35863. [Cited on pages 103 and 120]

[157] L. Xiao and T. Zhang, *A proximal stochastic gradient method with progressive variance reduction*, SIAM J Optim, 24 (2014), pp. 2057–2075. [Cited on pages 2, 20, 21, 27, 29, 30, 31, 32, 37, and 45]

[158] L. Yang, D. Sun, and K.-C. Toh, SDPNAL+*: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints*, Math Program Comput, 7 (2015), pp. 331–366. [Cited on page 29]

[159] M. Yang, A. Milzarek, Z. Wen, and T. Zhang, *A stochastic extra-step quasi-newton method for nonsmooth nonconvex optimization*, Math Program, (2021). [Cited on page 49]

[160] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, *S4l: Self-supervised semi-supervised learning*, in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1476–1485. [Cited on page 12]

[161] G. Zhang, C. Wang, B. Xu, and R. B. Grosse, *Three mechanisms of weight decay regularization*, in 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, 2019. [Cited on pages 2 and 83]

[162] J. Zhang and L. Xiao, *Stochastic variance-reduced prox-linear algorithms for nonconvex composite optimization*, Math Program, (2021). [Cited on page 62]

[163] N. Zhang, Y. Zhang, D. Sun, and K.-C. Toh, *An efficient linearly convergent regularized proximal point algorithm for fused multiple graphical Lasso problems*, SIAM J Math Data Sci, 3 (2021), pp. 524–543. [Cited on page 60]

[164] Y. Zhang, N. Zhang, D. Sun, and K.-C. Toh, *An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems*, Math Program, 179 (2020), pp. 223–263. [Cited on pages 2, 29, 38, and 60]

[165] ——, *A proximal point dual Newton algorithm for solving group graphical Lasso problems*, SIAM J Optim, 30 (2020), pp. 2197–2220. [Cited on page 28]

[166] P. Zhao and T. Zhang, *Stochastic optimization with importance sampling for regularized loss minimization*, in Proceedings of the 32nd International Conference on Machine Learning, F. Bach and D. Blei, eds., vol. 37 of Proceedings of Machine Learning Research, Lille, France, 07–09 Jul 2015, PMLR, pp. 1–9. [Cited on page 18]

[167] X.-Y. Zhao, D. Sun, and K.-C. Toh, *A Newton-CG augmented Lagrangian method for semidefinite programming*, SIAM J Optim, 20 (2010), pp. 1737–1765. [Cited on pages 2, 28, 29, 32, 34, and 35]

[168] J. Zhuang, T. Tang, Y. Ding, S. Tatikonda, N. C. Dvornek, X. Papademetris, and J. S. Duncan, *Adabelief optimizer: Adapting stepsizes by the belief in observed gradients*, in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., 2020. [Cited on pages 109 and 127]

[169] Z. Zhuang, M. Liu, A. Cutkosky, and F. Orabona, *Understanding AdamW through proximal methods and scale-freeness*, Transactions on Machine Learning Research, (2022). [Cited on pages 23, 70, 83, and 125]