IMT School for Advanced Studies, Lucca
Lucca, Italy

Quasi-Newton methods for solving nonsmooth optimization problems in learning and control

PhD Program in Systems Science

Track in Computer Science and Systems Engineering

Curriculum in Learning and Control

XXXVI Cycle

By

Adeyemi Damilare Adeoye

2025

The dissertation of Adeyemi Damilare Adeoye is approved.

PhD Program Coordinator: Prof. Alberto Bemporad, IMT School for Advanced Studies Lucca

Advisor: Prof. Alberto Bemporad, IMT School for Advanced Studies Lucca

The dissertation of Adeyemi Damilare Adeoye has been reviewed by:

Prof. Mikael Johansson, KTH Royal Institute of Technology, Stockholm, Sweden

Prof. Panagiotis Patrinos, KU Leuven, Leuven, Belgium

IMT School for Advanced Studies Lucca
2025

to my dearly beloved late mother;

to my family.

Contents

List of Figures

List of Tables

Acknowledgements

I am deeply and sincerely grateful to my supervisor, Prof. Alberto Bemporad, for exercising his professional prowess, knowledge and experience in helping to shape my PhD and academic path. It has been a great pleasure learning about different control and numerical optimization tools and techniques, art of programming and software development, research methodologies, and writing, through his guidance and support. His dedication and belief in my abilities have been a great source of motivation and inspiration. I feel very fortunate to have had the opportunity to work with him, and I am grateful for the trust he placed in me. I would like to acknowledge the funding received from the European Commission under the ERC Advanced Grant (COMPACT – Computational Model Predictive and Adaptive Control Tools) for completing the writing of this thesis.

I would like to thank Prof. Philipp Christian Petersen for the fruitful six months spent with his amazing research group at the University of Vienna, Austria. I am grateful for the immense support, invaluable guidance, and mentorship I received from him before and during my PhD. I thank the Faculty of Mathematics people there who created a welcoming environment and made my stay in Vienna truly special. I am grateful for the opportunity to learn from them, both in academic discussions and social gatherings. Thanks to Prof. Radu Ioan Boț for his group's optimization seminar and for the engaging discussion on the interesting open problems he was thinking about. I would like to acknowledge the Erasmus+ Traineeship fund received from the European Union through IMT Lucca, which made this experience possible.

I am grateful to the reviewers of my thesis, Prof. Mikael Johansson, Prof. Panagiotis Patrinos, and Asst. Prof. Puya Latafat, for their time, effort, and very insightful feedback, which has helped to improve the quality of the writing and presentation.

I would also like to thank all the visiting researchers and Professors of the DYSCO (Dynamical Systems, Control, and Optimization) research group with whom I exchanged scientific ideas during their visit. I am grateful for the continued engagement and support. Thanks to Prof. Stephen Boyd and his PhD student, Max, for a week-long discussion on parametric convex optimization, and for sparking interesting ideas in this line.

I thank all the administrative staff members of IMT Lucca for being very responsive and professional with helping to handle and navigate so many important aspects of my PhD and legal stay in Italy. I would like to thank all colleagues, Professors and all academic staff members of IMT Lucca, and collaborators who played significant roles in this rewarding journey. I greatly appreciate all the help and support I received from IMT community without hesitation when I asked. Some discussions have also helped in improving the presentation of some results in this thesis; such contributions are acknowledged in the respective chapters. I would like to thank the IMT football team for the enjoyable moments and physical activity and to the table tennis players for the engaging and competitive matches.

I would like to express my immense gratitude to all family members and friends who were supportive in diverse ways throughout this journey. I thank my loving wife and son for their unconditional love, companionship, patience, sacrifice, encouragement and support. I am grateful to my late mother for being the strongest human support of my life and aca-

demic pursuit. Unfortunately, you left when I was just about halfway into my PhD and didn't wait to witness this day which you've so much anticipated, but I'm sure you would be proud of me. I hope to continue to make you proud; you remain my hero and my best exemplar of a smart, hardworking and prayerful person who could achieve great things. I thank my ever-supportive uncle, Dr. J.O.F., my loving dad, his friend, Mr. O.A., and all my siblings.

Vita

Dec 04, 1993	Born, Ilorin, Nigeria
2016	Bachelor of Science in Mathematics Final mark: 4.59/5.0 First Class Honours University of Ilorin, Ilorin, Nigeria
2018	Master of Science in Mathematical Sciences Final mark: 70-80% Good Pass African Institute for Mathematical Sciences, Limbe, Cameroon
2020	Master of Science in Mathematical Sciences (Machine Intelligence) Final mark: 65-75% Good Pass African Institute for Mathematical Sciences, Kigali, Rwanda
2020–Now	PhD in Systems Science IMT School for Advanced Studies, Lucca, Italy
2023–2024	Visiting PhD Student (Erasmus+ Trainee) Faculty of Mathematics, University of Vienna Vienna, Austria

Publications

1. Adeyemi D. Adeoye and Alberto Bemporad. "SCORE: approximating curvature information under self-concordant regularization". In: *Computational Optimization and Applications* 86.2 (2023), pp. 599–626.
2. Adeyemi D. Adeoye and Alberto Bemporad. "An Inexact Sequential Quadratic Programming Method for Learning and Control of Recurrent Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 36.2 (2025), pp. 2762–2776. DOI: 10.1109/TNNLS.2024.3354855.
3. Adeyemi D. Adeoye and Alberto Bemporad. "Self-concordant Smoothing for Large-Scale Convex Composite Optimization". In: *arXiv preprint arXiv:2309.01781* (2024). Submitted.
4. Adeyemi D. Adeoye, Philipp Christian Petersen, and Alberto Bemporad. "Regularized Gauss-Newton for Optimizing Overparameterized Neural Networks". In: *arXiv preprint arXiv:2404.14875* (2024). Submitted.
5. Mikalai Korbit et al. "Exact Gauss-Newton Optimization for Training Deep Neural Networks". In: *arXiv preprint arXiv:2405.14402* (2024). Submitted.
6. Adeyemi D Adeoye and Alberto Bemporad. *SC-Reg: Training Overparameterized Neural Networks under Self-Concordant Regularization*. Tech. rep. IMT School for Advanced Studies Lucca, 2021.
7. Adeyemi Damilare Adeoye and Philipp Petersen. *A Deep Neural Network Optimization Method Via A Traffic Flow Model*. Tech. rep. African Institue for Mathematical Sciences, Rwanda, 2021.
8. Jos U Abubakar and AD Adeoye. "Effects of radiative heat and magnetic field on blood flow in an inclined tapered stenosed porous artery". In: *Journal of Taibah University for Science* 14.1 (2020), pp. 77–86.
9. Adeyemi Damilare Adeoye. "Blood Flow in an Inclined Tapered Stenosed Porous Artery under the Influence of Magnetic Field and Heat Transfer". MA thesis. African Institue for Mathematical Sciences, Cameroon, 2018.

Presentations

1. A.D. Adeoye, "Newton-type Optimization Methods for Model Learning and Control in Nonlinear Dynamical Systems," (Virtual Seminar) at *Mathematics and Computer Science (MCS) Division, Argonne National Laboratory (ANL)*, Lemont, Illinois, USA, 2025.
2. A.D. Adeoye, "Optimization of Neural Networks with an Explicit Regularization: Generalized Gauss-Newton Method," (Poster) at *Applied Harmonic Analysis and Machine Learning Summer School, Department of Mathematics, University of Genova*, Genova, Italy, 2024.
3. A.D. Adeoye, "Self-concordant Regularization in Machine Learning," at *Mathematics of Machine Learning Group Seminar, University of Vienna*, Vienna, Austria, 2023.
4. A.D. Adeoye, "Self-concordant Regularization in Machine Learning," (Virtual) at *DYSCO Research Unit Seminar, IMT Lucca*, Lucca, Italy, 2023.
5. A.D. Adeoye, "Self-concordant Regularization for Convex Composite Optimization," at *Mathematics of Machine Learning Research Group, Faculty of Mathematics, University of Vienna*, Vienna, Austria, 2023.
6. A.D. Adeoye, "Inexact SQP for Neural Network-Based Identification of Nonlinear Dynamical Systems," at *DYSCO Research Unit Mini-Symposium, IMT Lucca*, Lucca, Italy, 2023.
7. A.D. Adeoye, "SCORE: Approximating Curvature Information under Self-Concordant Regularization," (Virtual with Poster) at *Eastern European Machine Learning (EEML) Summer School*, Vilnius, Lithuania, 2022.

Abstract

This thesis is concerned with the design and analysis of some quasi-Newton methods for solving optimization problems in machine learning and control of nonlinear dynamical systems. The proposed algorithms are designed to exploit approximate second-order information to improve convergence rates, stability, and generalization of the learned models and control policies. The thesis is organized into two main parts. In the first part, we present a generalized Gauss-Newton algorithm that uses an adaptive step-size selection strategy and preserves the affine-invariant property of Newton's method. This algorithm significantly reduces the computational cost of Gauss-Newton methods, particularly in mini-batch supervised learning. We then extend this with a proximal method for nonsmooth convex composite optimization, resulting in two new algorithms. In the second part, we treat learning and control problems in the training of neural networks. First, we present a rigorous theoretical study of the generalized Gauss-Newton algorithm for the optimization of feedforward neural networks. This study establishes a non-asymptotic guarantee for the convergence of feedforward neural networks with a general explicit regularizer. Then, an inexact sequential quadratic programming framework is proposed for optimal control in recurrent neural networks, using a two-stage approach for system identification and optimal control policy selection. Several practical applications of all the proposed algorithms are demonstrated through numerical experiments.

Part I

Motivation, context and structure

Chapter 1

Introduction

Optimization problems are ubiquitous in fields such as machine learning, control, and signal processing. Many real-world applications pose challenging optimization problems that are nonsmooth, nonconvex, and high-dimensional. First-order optimization methods, like gradient descent, often suffer from slow convergence, getting trapped in suboptimal local minima, or quickly reverting to poor solution estimates. Particularly in the context of learning and control, these challenges can lead to poor generalization, instability, and suboptimal performance of the learned models and control policies. Second-order optimization methods, which exploit curvature information, can help overcome these challenges by providing faster convergence rates and more accurate solutions. However, the computational overhead of computing and storing the full Hessian matrix (the matrix of second-order partial derivatives) can be prohibitive, especially in high-dimensional problems. Quasi-Newton methods, which replace the Hessian matrix with its approximations, are popular alternatives to full second-order methods. These methods are computationally efficient in a wide range of optimization and control problems.

1.1 Structure and overview

In this thesis, we design and analyze practical quasi-Newton optimization methods for learning and control problems. We focus on (i) nonsmooth optimization in machine learning for several data-based modeling applications, and (ii) optimization-based approaches for optimal control problems. The contributions herein are spread across two parts of the thesis: Part II and Part III. Part II is made up of Chapters 2–3, and Part III is made up of Chapters 4–5. Each of these chapters are written to be self-contained, with a clear problem statement, motivation, methodology, theoretical analysis, and numerical experiments. However, in Part I (this chapter), we provide a general overview of the problems considered in the thesis and the motivation behind the proposed algorithms. The thesis' main contributions are thus structured as follows.

Part II: Solving nonsmooth convex composite optimization problems with generalized Gauss-Newton methods and self-concordant smoothing. This part, the associated appendices (Appendices 2.A, 2.B, 3.A), and the conclusions from the respective chapters in Chapter 6 are based on:

- Adeyemi D. Adeoye and Alberto Bemporad. "SCORE: approximating curvature information under self-concordant regularization". In: *Computational Optimization and Applications* 86.2 (2023), pp. 599–626.
- Adeyemi D. Adeoye and Alberto Bemporad. "Self-concordant Smoothing for Large-Scale Convex Composite Optimization". In: *arXiv preprint arXiv:2309.01781* (2024). Submitted.

The main contributions of this part are summarized below.

(a) We introduce self-concordant regularization (SCORE), a regularization and optimization framework for smooth convex problems which incorporates Newton-decrement into damped Newton-type iterates. Under this framework, we present a generalized Gauss-Newton (GGN) algorithm, called GGN-SCORE, that updates the optimization variables in a *pseudo*-online (stochastic) manner. The proposed

algorithm exploits the structure of the regularized Gauss-Newton matrix to significantly reduce associated computational overhead. GGN-SCORE demonstrates how to speed up convergence while also improving model generalization for practical machine learning applications. Numerical experiments show the efficiency of our method and its fast convergence, which compare favorably against baseline first-order and quasi-Newton methods. Additional experiments involving nonconvex (overparameterized) neural network training problems show that the proposed method is promising for nonconvex optimization, motivating further study of the approach specifically for understanding the optimization landscape of deep neural networks.

(b) We extend and generalize the notion of SCORE to nonsmooth convex optimization problems by embedding proximal operators to handle nonsmooth regularizers. Based on this extension, we propose two new algorithms: (i) a proximal Newton algorithm, and (ii) a proximal GGN algorithm. We provide theoretical guarantees for the convergence of these algorithms and demonstrate their efficiency through numerical experiments on several problems relevant to machine learning, signal processing, and control.

Part III: Theoretical guarantees and practical frameworks for solving supervised learning and control problems in neural networks using quasi-Newton methods. This part, the associated appendices (Appendices 4.A, 4.B, 4.C), and the conclusions from the respective chapters in Chapter 6 are based on:

- Adeyemi D. Adeoye, Philipp Christian Petersen, and Alberto Bemporad. "Regularized Gauss-Newton for Optimizing Overparameterized Neural Networks". In: *arXiv preprint arXiv:2404.14875* (2024). Submitted.

- Adeyemi D. Adeoye and Alberto Bemporad. "An Inexact Sequential Quadratic Programming Method for Learning and Control of Recurrent Neural Networks". In: *IEEE Transactions on Neural Networks and*

Learning Systems 36.2 (2025), pp. 2762–2776. DOI: 10.1109/TNNLS.2024.3354855.

The main contributions of this part are summarized below.

(a) Under practical boundedness conditions on the regularized GGN matrices and the smoothness of the hidden-layer activation function, we establish a non-asymptotic guarantee for the last-iterate convergence of a two-layer neural network predictions to the outputs of a given target function. We also quantify the decay of the regularized objective function over the course of the training iterations which, under standard theoretical setups, guarantees the global convergence the GGN. Through empirical simulations, we show how this rather general framework relate generalization and stability of the optimized neural network with the structural properties of the class of regularizers despite possibly breaking the direct relation between GGN and the neural tangent kernel (NTK) regression.

(b) We present a quasi-Newton optimization framework based on a two-stage approach to solving optimal control problems in the training of neural networks. In the first stage, the true system dynamics are approximated using a specific type of RNN whose training problem is regarded as a finite-horizon optimal control problem. We describe how the resulting regularized quadratic programming (QP) subproblems are solved efficiently, and how they incorporate approximate second-order information into the learning process. In particular, we propose an inexact sequential quadratic programming (SQP) algorithm to solve the QP subproblems efficiently. Under mild assumptions, we establish the convergence of the proposed SQP algorithm. Our framework and proposed algorithms use (approximate) second-order information in the optimization process of the two stages to learn accurate models and control policies that help in real-time decision-making under uncertainty and partial observability. Through numerical experiments on a reinforcement learning (RL) problem and an industrial control setting, we demonstrate the practical applications of the proposed methods and discuss possible

extensions for future research. Notably, the proposed RNN learning framework is shown to outperform existing methods in terms of convergence speed and approximation accuracy.

1.1.1 Open-source implementations

We provide open-source implementations of most of the proposed algorithms in this thesis in a Julia [41] package. Due to the goal of developing a unified framework for solving optimization problems in learning and control, the Julia package is designed to be modular and extensible, allowing for easy integration of new algorithms and problem formulations. Markedly, the Julia package unifies algorithms for solving *Self-Concordant Smooth Optimization* (SCSO) problems, where we use "smooth" synonymously with "regularized" in the context of this thesis. It is available on GitHub at `https://github.com/adeyemiadeoye/SelfConcordantSmoothOptimization.jl`.

NB: It is worth mentioning that when revising some parts of the subsequent sections of this chapter for a first submission, a writing tool based on a Large Language Model (LLM) was used to rewrite some sentences, correct some grammatical errors and improve the writing of an original early draft where the main structure and content were already in place with appropriate references (when they're known). The goal is to make the introduction clearer, simpler and less technical for a broader audience; we hope this language technology has helped in making the introduction a good entry point into the main (technical) parts of the thesis for readers (as the expert reviews suggest). The technology was not used to generate any new content or ideas (which expert sources debate its ability to do currently), but rather to assist in the revision/writing process.
Also, the Julia package has since undergone some refactoring with the aid of the LLM technology, with no changes to the implemented algorithms.

1.2 Nonlinear programming problem

In this section, we provide a very general setup of the optimization problems considered in this thesis. We want to describe the basic and essential components that are common to all the optimization problems considered

throughout. The specific assumptions and notations for the objective functions, constraints, and optimization variables are tailored to the specific problem formulations and described in each chapter.

We aim to solve optimization problems of the form

$$\min_{x \in \mathcal{C}} f(x), \tag{1.1}$$

where $x \in \mathbb{R}^n$ is the optimization variable, and the set $\mathcal{C} \subseteq \mathbb{R}^n$ represents possible constraints on the optimization variables, such as bounds on the variables or linear equality and inequality constraints. The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is possibly nonsmooth, nonconvex and/or high-dimensional, depending on the target application. Irrespective of the specific form of f and $\mathcal{C}$, the goal is to find a minimizer $x^* \in \mathcal{C}$ such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{C}$. The optimization problems considered in this thesis arise in machine learning, control, and signal processing applications, where the objective function f is typically a loss function or a cost function that needs to be minimized to learn accurate models or control policies. In some problems, such as *structural risk minimization* in machine learning, the constraint set $\mathcal{C}$ serves the purpose of regularizing the optimization problem and preventing overfitting of the learned models [232]. In this case, the constraints are often imposed via an *explicit* regularization term $g : \mathbb{R}^n \to \mathbb{R}$, leading to the regularized optimization problem

$$\min_{x \in \mathbb{R}^n} \mathcal{L}(x) \triangleq f(x) + \lambda g(x), \tag{1.2}$$

where $\lambda > 0$ is a regularization parameter that controls the trade-off between the objective term f and the regularization term g. This regularization term penalizes complex models and encourages simpler solutions, thereby improving the generalization capability of the learned models.

When modeling large-scale dynamical systems and designing control systems, simpler and lightweight models are often preferred. To reduce the model complexity, an optimization problem of the form (1.2) is typically solved, where the regularization term g enforces (structured) sparsity on the model parameters. For example, g can be a sparse group lasso

penalty that encourages the selection of a small subset of states or inputs that are considered more important in the dynamical model (see, e.g., [32]). In MPC problems, $\mathcal{C}$ represents model dynamics and operational constraints that need to be satisfied by the control policies [44]. Here, the constraints are imposed directly on the optimization variables x, and an algorithm is sought to find a feasible solution that satisfies the constraints while minimizing the cost function f. Typical examples of this setup are presented in Section 1.3 and Chapter 5.

Each chapter in Parts II and III of this thesis deals with a specific type of constraint set $\mathcal{C}$. In Chapter 2, we consider smooth optimization problems with self-concordant regularizers, with the main goal of presenting the proposed optimization scheme in a comprehensive manner, and demonstrating its efficiency in solving practical machine learning problems. This scheme is extended to handle more general nonsmooth optimization problems in the subsequent chapters, with the goal of providing *a unified approach to solving optimization problems in learning and control*.

1.3 Learning and control problems

In common learning strategies, the optimization problem is formulated as a data-driven modeling problem, where the optimization variables are the model parameters, and the constraints (typically enforced through (1.2)) define desired structural properties of the model. When deeper structures and higher complexities are involved in the model learning, particularly in the context of neural networks, the learning algorithms of choice are often *gradient-based* that use the chain rule of differentiation to compute gradient terms. This process is known as *backpropagation* [203], and it is central to modern deep learning algorithms [102]. Backpropagation has been directly linked to an optimal control formalism in which constraints in the problem are represented as model dynamics [132]. Under this lens, we introduce the learning and control aspects of the optimization problems considered in this thesis.

1.3.1 Supervised learning with neural networks

Consider a data-generating system represented by $x^+ \triangleq h(x,u,t)$, where $x \in \mathbb{R}^{n_x}$ is the current state, $x^+ \in \mathbb{R}^{n_x}$ is the next state, $u \in \mathbb{R}^{n_u}$ is the control input, and $t \in \mathbb{R}$ is the discrete-time index. The system dynamics are (partially) unknown, but we have access to a dataset of (noisy) measurements $\{(x_i,u_i,y_i)\}_{i=1}^N$, where $y_i \in \mathbb{R}^{n_y}$ are the outputs of the system. We aim to learn an approximate model $\mathcal{M}$ that provides real-time predictions of the system outputs given the current state and control input. Suppose $\mathcal{M}$ is a neural network with L layers defined by

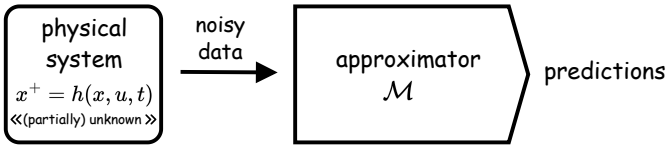


Figure 1: Data-driven approximation of a dynamical system.

$$\mathcal{M}(x,u;\theta) \triangleq$$
$$\sigma_L(W^{(L)}\sigma_{L-1}(W^{(L-1)}\cdots\sigma_1(W^{(1)}[x,u]^\top + b^{(1)})\cdots + b^{(L-1)}) + b^{(L)}), \tag{1.3}$$

where for each $l = 1,\ldots,L$, $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{n_l}$ are the weights and biases of the $l-$th layer, respectively, and $\sigma_l : \mathbb{R} \to \mathbb{R}$ is an element-wise nonlinear hidden map, called the *activation function*. Let $\theta_l \triangleq \{W^{(l)}, b^{(l)}\}$ be the parameters of the $l-$th layer, then $\mathbb{R}^n \ni \theta \triangleq vec(\theta_1,\ldots,\theta_L)$ represents the stacked vector of all parameters in the network. If $\hat{y}_i = \mathcal{M}(x_i,u_i;\theta)$ denotes the predictions of the neural network, then the model parameters θ are learned by minimizing the regularized empirical risk

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(Y_i,\hat{y}_i) + \lambda g(\theta), \tag{1.4}$$

where $Y_i \triangleq [y_i, x^+]^\top \in \mathbb{R}^{n_Y}$, $\ell\colon \mathbb{R}^{n_Y} \times \mathbb{R}^{n_Y} \to \mathbb{R}$ measures the discrepancy between the predicted and true outputs, $g\colon \mathbb{R}^n \to \mathbb{R}$ is a regularization term, and $\lambda > 0$ is a regularization parameter.

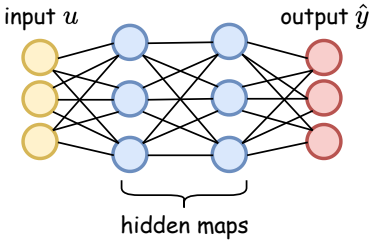


Figure 2: A neural network schematic.

The neural network (1.3) is called a *feedforward neural network* (FNN). Because FNNs are universal approximators, they can learn complex functions and model the system dynamics effectively [117]. However, since they lack memory (recurrent connections or feedback loops), they may not be suitable for modeling dynamical systems with memory or feedback without special modifications. On the other hand, *recurrent neural networks* (RNNs) are designed to handle such systems [203]. A vanilla RNN in state-space form is given by

$$\begin{aligned} x_{t+1} &= \mathcal{M}_x(x_t, u_t; \theta_x), \\ \hat{y}_t &= \mathcal{M}_y(x_t; \theta_y), \end{aligned} \tag{1.5}$$

where θ_x and θ_y are the parameters of the state and output models, respectively.

The optimization problems associated with training RNNs are more complex than those for FNNs as the model involves hidden states and a sequence of optimization problems are solved over time. However, the training problem of both FNNs and RNNs are finite-horizon optimal control problems, as we briefly discuss below. Hence, developing efficient optimization algorithms that solve problems coming from both (supervised) learning and control applications forms the core of subsequent chapters. In Chapter 5, we propose a specialized algorithm that solve the training problem of RNNs efficiently. All the algorithms proposed in

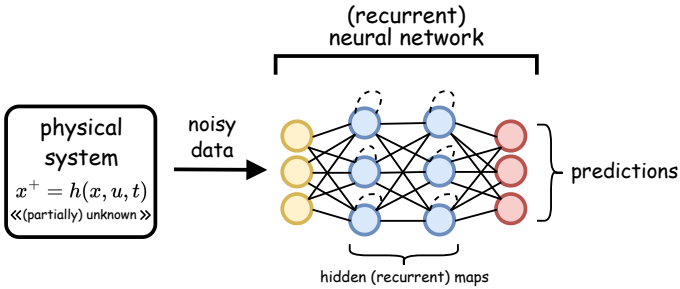


Figure 3: Schematic of a dynamical system approximated by a neural network.

Chapters 2–4 can handle the training problem of FNNs.

1.3.2 Finite-horizon optimal control problems in supervised learning

Optimal control

Consider a discrete-time system described by the state-space model

$$x_{t+1} = h(x_t, u_t), \tag{1.6}$$

where $x_t \in \mathbb{R}^{n_x}$ is the state, $u_t \in \mathbb{R}^{n_u}$ is the control input, and $h\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the system dynamics in discrete time. For simplicity of our presentation, we ignore possible disturbances, constraints, and uncertainties in the system dynamics (1.6), but we remark that these are important considerations in practice, and are taken into account in the subsequent relevant chapters.

In *finite-horizon* optimal control, the goal is to find a sequence of control inputs $\{u_t\}_{t=0}^{T-1}$ that minimizes a cost function

$$J(x_0, \{u_t\}_{t=0}^{T-1}) \triangleq \sum_{t=0}^{T-1} \ell(x_t, u_t) + \ell_T(x_T), \tag{1.7}$$

over a finite horizon T, where $\ell\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is the *stage cost*, and $\ell_T\colon \mathbb{R}^{n_x} \to \mathbb{R}$ is the *terminal cost*. The optimization problem is formulated as

$$\begin{aligned} \min_{\{u_t\}_{t=0}^{T-1}} \quad & J(x_0, \{u_t\}_{t=0}^{T-1}) \\ \text{s.t.} \quad & x_{t+1} = h(x_t, u_t), \quad t = 0, \dots, T-1 \\ & x_T \in \mathcal{X}_T, \\ & x_0 = x_{\text{init}}, \end{aligned} \tag{1.8}$$

where $J\colon \mathbb{R}^{n_x} \times \mathbb{R}^{Tn_u} \to \mathbb{R}$ is defined by (1.7), $\mathcal{X}_T$ is the *terminal constraint* set and x_{init} is the initial state of the system. In case disturbances, constraints, and uncertainties are present in the system dynamics, the optimization problem (1.8) is modified to include them accordingly.

The finite-horizon optimal control problem (1.7)–(1.8) can be solved using dynamic programming or direct optimization methods [44]. For a broader and more detailed presentation, including *infinite-horizon* optimal control problems, we refer the reader to [40, 44].

Feedforward neural network training as an optimal control problem

By explicitly writing the equality constraints in (1.8) as

$$\begin{aligned} x_1 &= h(x_0, u_0), \\ x_2 &= h(x_1, u_1), \\ &\vdots \\ x_T &= h(x_{T-1}, u_{T-1}), \end{aligned}$$

the optimal control problem (1.7)–(1.8) becomes more easily seen as a nonlinear programming (optimization) problem with variables $\{u_t\}_{t=0}^{T-1}$ and $\{x_t\}_{t=0}^{T}$. Consider eliminating the intermediate state variables $\{x_t\}_{t=1}^{T-1}$ by successive substitution, so that only $\{u_t\}_{t=0}^{T-1}$ and x_0 remain as optimization variables. While the optimization problem still depends on the initial state x_0 which determines the optimal control sequence $\{u_t^*\}_{t=0}^{T-1}$, there are now only $T+1$ decision variables to be optimized. This approach is called the *sequential* optimal control approach, and it is precisely

what backpropagation amounts to when gradient-based optimization algorithms are used to train FNNs by minimizing the regularized loss (1.4) [132, 93]. To make it more explicit, consider the FNN model (1.3) with L layers, and write this optimization problem as

$$\begin{aligned} \min_{\theta,\{z_l\}_{l=1}^{L-1}} \quad & \mathcal{L}(\theta) \\ \text{s.t.} \quad & a_0 = z_0, \\ & a_l = W^{(l)} z_{l-1} + b^{(l)}, \quad l = 1, \dots, L-1, \\ \text{with} \quad & z_l = \sigma_l(a_l), \quad l = 1, \dots, L-1, \end{aligned} \tag{1.9}$$

where $\mathcal{L}(\theta)$ is defined by (1.4) and $z_0 \in \mathbb{R}^{n_0 \times N}$ is the input data. Problem (1.9) is an L-stage finite-horizon optimal control problem, and the optimization variables are the parameters θ and the intermediate states $\{z_l\}_{l=1}^{L-1}$. In the traditional gradient-based training, the intermediate states are eliminated by successive substitution in the *forward pass* to compute the loss, and the loss gradient is computed in the *backward pass* (backpropagation) to update the parameters θ.

Recurrent neural network training as an optimal control problem

Since stability and backpropagation-related issues in FNN training are readily handled by regularization techniques [102] (see also Chapter 4), traditional backpropagation (gradient-based) training is often efficient and sufficient for FNNs. However, for RNNs, the training problem is more complex due to the presence of hidden states and the sequential nature of the optimization problem. In this case, the backpropagation through time (BPTT) algorithm, a variant of backpropagation, is used to compute the loss gradients. This involves first unrolling the RNN over time to form a *multilayer* (deep) feedforward neural network (that is, forward pass, which eliminates the hidden states), and then applying backpropagation to compute the loss gradients. This process is known to suffer from vanishing and exploding gradients [36, 116], which can lead to poor generalization and slow convergence. These issues can be addressed by incorporating second-order information into the optimization process, as we discuss in Chapter 5.

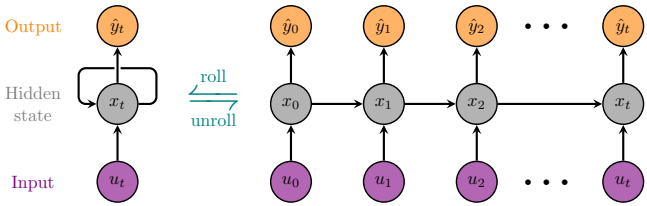


Figure 4: Schematic of a simple RNN unrolled over time.

Another approach to solving optimal control problems is the *simultaneous* approach, where all the decision variables are optimized simultaneously. When viewed as an optimal control problem, it is more natural and efficient to solve the training problem of RNNs using the simultaneous approach. Here, no elimination of intermediate states is done, and the optimization problem is solved directly as a large-scale nonlinear programming problem where subproblems are solved with structure-exploiting iterative algorithms.

Consider the RNN model in (1.5). Under the simultaneous approach, the training problem of the RNN is formulated as

$$\begin{aligned} \min_{\theta_x, \theta_y, \{x_t\}_{t=0}^{T}} \quad & \mathcal{L}_{\text{rnn}}(\theta_x, \theta_y, x_0) \\ \text{s.t.} \quad & \begin{bmatrix} x_1 - \mathcal{M}_x(x_0, u_0; \theta_x) \\ x_2 - \mathcal{M}_x(x_1, u_1; \theta_x) \\ \vdots \\ x_T - \mathcal{M}_x(x_{T-1}, u_{T-1}; \theta_x) \end{bmatrix} = 0, \end{aligned} \tag{1.10}$$

where

$$\mathcal{L}_{\text{rnn}}(\theta_x, \theta_y, x_0) \triangleq \sum_{t=0}^{T-1} \ell(y_t, \hat{y}_t) + \mathcal{R}(\theta_x, \theta_y, x_0),$$

is the regularized loss function, with $\mathcal{R}$ being separable in the model parameters and the initial state x_0. This formulation results in a large-scale optimization problem with $Tn_x + n_{\theta_x} + n_{\theta_y}$ decision variables and Tn_x equality constraints, where n_{θ_x} and n_{θ_y} are the number of parameters

in the state and output models, respectively. Several benefits of this approach for training RNNs are discussed in Chapter 5.

1.4 First- and second-order optimization methods

To simplify our presentation in this section, we consider the optimization problem (1.1) with $\mathcal{C} = \mathbb{R}^n$, and assume that f is twice continuously differentiable. The gradient of f at x is denoted by $\nabla f(x)$, and $\nabla^2 f(x)$ denotes its Hessian. In modern practice, these terms are estimated using automatic differentiation tools provided by software libraries of popular programming languages, and account for exact first- and second-order information, respectively, about the function f in the design of algorithms for solving smooth optimization problems. A technique for handling nonsmooth terms in composite optimization problems is discussed in Chapter 3.

1.4.1 First-order optimization methods

First-order optimization methods rely solely on gradient information to find the minimizer of the objective function f. The most common first-order method is the gradient descent method, which can be traced back to Augustin-Louis Cauchy's work in [62]. The gradient descent method updates the current iterate x_k of the optimization variable x using the iterative formula

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{1.11}$$

where $\alpha_k > 0$ is the step size or *learning rate*. There are now several variants of the gradient descent method (see [200] and the references in Chapter 2), including the stochastic gradient descent (SGD) method, which uses a noisy estimate of the gradient to update x. The SGD method is particularly useful for large-scale optimization problems, where computing the full gradient is computationally expensive [46]. While they are simple and easy to implement, with low per-iteration computational cost, first-order methods can suffer from slow convergence, especially when

the objective function has a complex landscape as in the case of neural network training [102]. A related issue arises when the step size is chosen poorly, often making the methods require a large number of iterations to converge to a solution [163, 46]. Additionally, first-order methods may struggle with saddle points, where the gradient is zero but the point is not a local minimum [77].

1.4.2 Second-order optimization methods

Second-order optimization methods use both gradient and Hessian information to achieve faster convergence, and improved accuracy and stability. Newton's method is a prominent second-order method, and can be traced back to Sir Isaac Newton's work in his unpublished 1669 manuscript titled *De analysi per aequationes numero terminorum infinitas* ("On the analysis by means of equations with an infinite number of terms" [104]). Newton's method uses the iterative formula

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k). \tag{1.12}$$

This method can achieve quadratic convergence near the optimal solution, making it significantly faster than first-order methods for well-behaved problems [163]. However, the computational overhead of Newton's method can be severe. Computing and inverting the Hessian matrix at each iteration can be highly prohibitive, especially for high-dimensional problems. This motivates the use of methods that replace $\nabla^2 f(x_k)$ (or its inverse) in (1.12) with an approximation of it to reduce the computational cost while maintaining fast convergence speeds comparable to Newton's method.

1.4.3 Quasi-Newton methods

The first *formal* quasi-Newton method was developed by William C. Davidon in [78]. The term "quasi-Newton" was first used by Charles G. Broyden in 1965 [48] to describe the class of optimization methods that replace the exact Hessian matrix inverse $[\nabla^2 f(x_k)]^{-1}$ in Newton's method with an approximation. The first set of quasi-Newton methods, developed in

a series of papers between 1959 and 1970 [78, 48, 49, 21, 50, 94, 101, 216], uses specified conditions to *iteratively* approximate $[\nabla^2 f(x_k)]^{-1}$ with a sequence of matrices $\{G_k\}$, such that

$$\lim_{k\to\infty} G_k = [\nabla^2 f(x^*)]^{-1},$$

where x^* is the optimal solution of the optimization problem. In other words, they compute G_k under the specified conditions, such that some of its properties approximate to those of $[\nabla^2 f(x_k)]^{-1}$ [49]. Quasi-Newton methods are widely used in practice due to their balance of efficiency and convergence properties [168]. They are particularly effective for medium to large-scale optimization problems where the computational cost of Newton's method is prohibitive.

The most popular quasi-Newton methods include the Broyden–Fletcher–Goldfarb–Shanno (BFGS), the Davidon–Fletcher–Powell (DFP), the limited-memory BFGS (L-BFGS), the generalized Gauss-Newton, and sequential quadratic programming (SQP) methods [95, 143, 168]. A common, distinct feature of these methods is the *low-rank* update of the approximation matrix G_k at each iteration which, in the case of generalized Gauss-Newton methods, is achieved through low-rank linear algebra techniques [4] (see Chapter 2). In the next chapters, we develop new efficient quasi-Newton methods for solving large-scale optimization problems in machine learning and control applications.

Part II

Solving nonsmooth convex composite optimization problems with generalized Gauss-Newton methods and self-concordant smoothing

Chapter 2

SCORE: Approximating curvature information under self-concordant regularization

2.1 Introduction

This chapter presents a pseudo-online optimization algorithm based on solving a regularized unconstrained minimization problem under the assumption of strong convexity and Lipschitz continuity. As we discussed in the previous chapter, first-order methods such as stochastic gradient descent (SGD) [194, 46] and its variants [86, 247, 128, 123] only make use of first-order information through the function gradients. Second-order methods [30, 143, 108, 11, 147, 177, 149], on the other hand, attempt to incorporate, in some way, second-order information in their approach, through the Hessian matrix or the Fisher information matrix (FIM). This generally provides second-order methods with better (quadratic) convergence than a typical first-order method which only converges linearly in the neighbourhood of the solution [163].

In most commonly used second-order methods, the natural gradient,

Gauss-Newton, and the sub-sampled Newton [56] – or its regularized version [91] – used for incorporating second-order information while maintaining desirable convergence properties, compute the Hessian matrix (or FIM) $\boldsymbol{H}$ by using the approximation $\boldsymbol{H} \approx \boldsymbol{J}^T\boldsymbol{J}$, where $\boldsymbol{J}$ is the Jacobian matrix. This approximation is still an $n_w \times n_w$ matrix, where n_w is the number of optimization variables, and remains computationally demanding to invert for large problems. In recent works [57, 249, 38, 124], second-order methods for overparameterized neural network models are made to bypass this difficulty by applying a matrix identity, and instead only compute the matrix $\boldsymbol{J}\boldsymbol{J}^T$ which is a $d \cdot N \times d \cdot N$ matrix, where d is the model output dimension and N is the number of data points. This approach significantly reduces computational overhead in the case $d \cdot N$ is much smaller than n_w (overparameterized models) and helps to accelerate convergence [57]. Nevertheless, for an objective function with a differentiable (up to two times) convex regularizer, this simplification requires a closer attention and special modifications for a general problem with a large number of variables.

The idea of exploiting desirable regularization properties for improving the convergence of the (Gauss-)Newton scheme has been around for decades and most of the published works on the topic combine in different ways the idea of Levenberg-Marquardt (LM) regularization, line-search, and trust-region methods [165]. For example, the recent work [155] combines the idea of cubic regularization (originally proposed by [165]) and a particular variant of the adaptive LM penalty that uses the *Euclidean* gradient norm of the output-fit loss function (see [152] for a comprehensive list and complexity bounds). Their proposed scheme achieves a global $\mathcal{O}(k^{-2})$ rate, where k is the iteration number. A similar idea is considered in [81] using the Bregman distances, extending the idea to develop an accelerated variant of the scheme that achieves a $\mathcal{O}(k^{-3})$ convergence rate.

We propose a new self-concordant regularization (SCORE) scheme for efficiently choosing optimal variables of the model involving smooth, strongly convex optimization objectives, where one of the objective functions regularizes the model's variable vector and hence avoids overfitting,

ultimately improving the model's ability to generalize well. By an *extra* assumption that the regularization function is self-concordant, we propose the GGN-SCORE algorithm (see Algorithm 1 below) that updates the minimization variables in the framework of the *local norm* $\|\cdot\|_x$ (a.k.a., Newton-decrement) of a self-concordant function $f(x)$ such as seen in [164]. Our proposed scheme does not require that the output-fit loss function is self-concordant, which in many applications does not hold [155]. Instead, we exploit the *greedy* descent provision of self-concordant functions, via regularization, to achieve a fast convergence rate while maintaining feasible assumptions on the combined objective function (from an application point of view). Although we assume a convex optimization problem in this chapter, we also provide experiments that show promising results for non-convex problems that arise when training neural networks. Our experimental results provide an interesting opportunity for future investigation and scaling of the proposed method for large-scale machine learning problems, as one of the non-convex problems considered in the experiments involves training an overparameterized neural network. We remark that overparameterization is an interesting and desirable property, and a topic of interest in the machine learning community [51, 157, 31, 9].

This chapter is organized as follows: First, we introduce some notations, formulate the optimization problem with basic assumptions, and present an initial motivation for the optimization method in Section 2.2. In Section 2.3, we derive a new generalized method for reducing the computational overhead associated with the mini-batch Newton-type updates. The idea of SCORE is introduced in Section 2.4 and our GGN-SCORE algorithm is presented thereafter. Experimental results that show the efficiency and fast convergence of the proposed method are presented in Section 4.4.

2.2 Preliminaries

2.2.1 Notation and basic assumptions

Let $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$ be a sequence of N input and output sample pairs, $\boldsymbol{x}_n \in \mathbb{R}^{n_p}, \boldsymbol{y}_n \in \mathbb{R}^d$, where n_p is the number of features and d is the number of targets. We assume a model $f(\boldsymbol{\theta}; \boldsymbol{x}_n)$, defined by $f\colon \mathbb{R}^{n_w} \times \mathbb{R}^{n_p} \to \mathbb{Y}$ and parameterized by the vector of variables $\boldsymbol{\theta} \in \mathbb{R}^{n_w}$. We denote by $\partial_{\boldsymbol{a}} \boldsymbol{b} \equiv \partial \boldsymbol{b}$ the gradient (or first derivative) of $\boldsymbol{b}$ (with respect to $\boldsymbol{a}$) and $\partial^2_{\boldsymbol{aa}} \boldsymbol{b} \equiv \partial^2 \boldsymbol{b}$ the second derivative of $\boldsymbol{b}$ with respect to $\boldsymbol{a}$. We write $\|\cdot\|$ to denote the 2-norm. The set $\{\mathrm{diag}(\boldsymbol{v})\colon \boldsymbol{v} \in \mathbb{R}^n\}$, where $\mathrm{diag}\colon \mathbb{R}^n \to \mathbb{R}^{n\times n}$, denotes the set of all diagonal matrices in $\mathbb{R}^{n\times n}$. Throughout this chapter, bold-face letters denote vectors and matrices.

Suppose that $f(\boldsymbol{\theta}; \boldsymbol{x}_n)$ outputs the value $\hat{\boldsymbol{y}}_n \in \mathbb{R}^d$. The regularized minimization problem we want to solve is

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \triangleq \underbrace{\sum_{n=1}^{N} \ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)}_{g(\boldsymbol{\theta})} + \lambda \underbrace{\sum_{j=1}^{n_w} r_j(\boldsymbol{\theta}_j)}_{h(\boldsymbol{\theta})}, \tag{2.1}$$

where $\ell : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a (strongly) convex twice-differentiable output-fit loss function, $r_j : \mathbb{R} \to \mathbb{R}$, $j = 1, \ldots, n_w$, define a separable regularization term on $\boldsymbol{\theta}$, $g(\boldsymbol{\theta})\colon \mathbb{R}^{n_w} \to \mathbb{R}$, $h(\boldsymbol{\theta})\colon \mathbb{R}^{n_w} \to \mathbb{R}$. We assume that the regularization function $h(\boldsymbol{\theta})$, scaled by the parameter $\lambda > 0$, is twice differentiable and strongly convex. The following preliminary conditions define the regularity of the Hessian of $\mathcal{L}(\boldsymbol{\theta})$ and are assumed to hold only locally in this chapter:

Assumption 1. *The functions g and h are twice-differentiable with respect to $\boldsymbol{\theta}$ and are respectively γ_l- and γ_a-strongly convex.*

Assumption 2. *$\exists \gamma_u, \gamma_b$ with $0 \le \gamma_l \le \gamma_u < \infty$, $0 < \gamma_a \le \gamma_b < \infty$, such that the gradient of $\mathcal{L}(\boldsymbol{\theta})$ is $(\gamma_u + \lambda\gamma_b)$-Lipschitz continuous $\forall \boldsymbol{x} \in \mathbb{R}^{n_p}, \boldsymbol{y} \in \mathbb{R}^d$. That is, $\forall \boldsymbol{x} \in \mathbb{R}^{n_p}, \forall \boldsymbol{y} \in \mathbb{R}^d$, the gradient $\partial_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) + \lambda \partial_{\boldsymbol{\theta}} h(\boldsymbol{\theta})$ satisfies*

$$\left\| \partial_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{\theta_1}; \boldsymbol{x})) - \partial_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{\theta_2}; \boldsymbol{x})) \right\| \le (\gamma_u + \lambda\gamma_b) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|, \tag{2.2}$$

for any $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathbb{R}^{n_w}$.

Assumption 3. *$\exists \gamma_g, \gamma_h$ with $0 < \gamma_g, \gamma_h < \infty$ such that $\forall \boldsymbol{x} \in \mathbb{R}^{n_p}, \forall \boldsymbol{y} \in \mathbb{R}^d$, the second derivatives of $g(\boldsymbol{\theta})$ and $h(\boldsymbol{\theta})$ respectively satisfy*

$$\left\| \partial^2 g(\boldsymbol{y}, f(\boldsymbol{\theta_1}; \boldsymbol{x})) - \partial^2 g(\boldsymbol{y}, f(\boldsymbol{\theta_2}; \boldsymbol{x})) \right\| \leq \gamma_g \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|, \tag{2.3a}$$

$$\left\| \partial^2 h(\boldsymbol{y}, f(\boldsymbol{\theta_1}; \boldsymbol{x})) - \partial^2 h(\boldsymbol{y}, f(\boldsymbol{\theta_2}; \boldsymbol{x})) \right\| \leq \gamma_h \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|, \tag{2.3b}$$

for any $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathbb{R}^{n_w}$.

Commonly used loss functions such as the squared loss, and the sigmoidal cross-entropy loss are twice differentiable and (strongly) convex in the model variables. Certain smoothed sparsity-inducing penalties such as the (pseudo-)Huber function – presented later – constitute the class of functions that may be well-suited for $h(\boldsymbol{\theta})$ defined above.

The assumptions of strong convexity and smoothness about the objective $\mathcal{L}(\boldsymbol{\theta})$ are standard conventions in many optimization problems as they help to characterize the convergence properties of the underlying solution method [153, 245]. However, the smoothness assumption about the objective $\mathcal{L}(\boldsymbol{\theta})$ is sometimes not feasible for some multi-objective (or regularized) problems where a non-smooth (penalty-inducing) function $h(\boldsymbol{\theta})$ is used (see the recent work [42]). In such a case, and when the need to incorporate second-order information arise, a well-known approach in the optimization literature is generally either to approximate the non-smooth objectives by a smooth quadratic function (when such an approximation is available) or use a "proximal splitting" method and replace the 2-norm in this setting with the Q-norm, where Q is the Hessian matrix or its approximation [182]. In [182], the authors propose two techniques that help to avoid the complexity that is often introduced in subproblems when the latter approach is used. While proposing new approaches, [211] highlights some popular techniques to handle non-differentiability. Each of these works highlight the importance of incorporating second-order information in the solution techniques of optimization problems. By conveniently solving the optimization problem (2.1) where the assumptions made above are satisfied, our method ensures the full curvature information is captured while reducing computational overhead.

2.2.2 Approximate Newton Scheme

Given the current value of $\boldsymbol{\theta}$, the (Gauss-)Newton method computes an update to $\boldsymbol{\theta}$ via

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \rho \boldsymbol{G}, \tag{2.4}$$

where ρ is the step size, $\boldsymbol{G} = \boldsymbol{H}^{-1} \partial \mathcal{L}(\boldsymbol{\theta})$ and $\boldsymbol{H}$ is the Hessian of $\mathcal{L}$ or its approximation. In this chapter, we consider the generalized Gauss-Newton (GGN) approximation of $\boldsymbol{H}$ which we now define in terms of the function $g(\boldsymbol{\theta})$. This approximation and its detailed expression motivates the modified version introduced in the next section to include the regularization function $h(\boldsymbol{\theta})$.

Definition 1 (Generalized Gauss-Newton Hessian)**.** *Let* $\operatorname{diag}(\boldsymbol{q}_n) \in \mathbb{R}^{d \times d}$ *be the second derivative of the loss function* $\ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$ *with respect to the predictor* $\hat{\boldsymbol{y}}_n$*,* $\boldsymbol{q}_n = \partial^2_{\hat{\boldsymbol{y}}_n \hat{\boldsymbol{y}}_n} \ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$ *for* $n = 1, 2, \ldots, N$*, and let* $\boldsymbol{Q}_g \in \mathbb{R}^{dN \times dN}$ *be a block diagonal matrix with* $\boldsymbol{q}_n$ *being the* n*-th diagonal block. Let* $\boldsymbol{J}_n \in \mathbb{R}^{d \times n_w}$ *denote the Jacobian of* $\hat{\boldsymbol{y}}_n$ *with respect to* $\boldsymbol{\theta}$ *for* $n = 1, 2, \ldots, N$*, and let* $\boldsymbol{J}_g \in \mathbb{R}^{dN \times n_w}$ *be the vertical concatenation of all* $\boldsymbol{J}_n$*'s. Then, the generalized Gauss-Newton (GGN) approximation of the Hessian matrix* $\boldsymbol{H}_g \in \mathbb{R}^{n_w \times n_w}$ *associated with the fit loss* $\ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$ *with respect to* $\boldsymbol{\theta}$ *is defined by*

$$\boldsymbol{H}_g \approx \boldsymbol{J}_g^T \boldsymbol{Q}_g \boldsymbol{J}_g = \sum_{n=1}^{N} \boldsymbol{J}_n^T \operatorname{diag}(\boldsymbol{q}_n) \boldsymbol{J}_n. \tag{2.5}$$

Let $\boldsymbol{e}_n \in \mathbb{R}^d$ be the Jacobian of the fit loss defined by $\boldsymbol{e}_n = \partial_{\hat{\boldsymbol{y}}_n} \ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$ for $n = 1, 2, \ldots, N$. For example, in case of squared loss $\ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n) = \frac{1}{2}(\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n)^2$ we get that $\boldsymbol{e}_n$ is the residual $\boldsymbol{e}_n = \hat{\boldsymbol{y}}_n - \boldsymbol{y}_n$. Let $\boldsymbol{e}_g \in \mathbb{R}^{dN}$ be

the vertical concatenation of all $\boldsymbol{e}_n$'s. Then, using the chain rule, we write

$$\begin{aligned}
\boldsymbol{J}_n^T \boldsymbol{e}_n^T &= \begin{bmatrix} \partial_{\theta_1}\hat{y}^{(1)} & \partial_{\theta_1}\hat{y}^{(2))} & \cdots & \partial_{\theta_1}\hat{y}^{(d)} \\ \partial_{\theta_2}\hat{y}^{(1)} & \partial_{\theta_2}\hat{y}^{(2)} & \cdots & \partial_{\theta_2}\hat{y}^{(d)} \\ \vdots & \vdots & & \vdots \\ \partial_{\theta_{n_w}}\hat{y}^{(1)} & \partial_{\theta_{n_w}}\hat{y}^{(2)} & \cdots & \partial_{\theta_{n_w}}\hat{y}^{(d)} \end{bmatrix} \begin{bmatrix} \partial_{\hat{y}}\ell^{(1)} \\ \partial_{\hat{y}}\ell^{(2)} \\ \vdots \\ \partial_{\hat{y}}\ell^{(d)} \end{bmatrix} \\
&= \begin{bmatrix} \partial_{\theta_1}\ell^{(1)} + \partial_{\theta_1}\ell^{(2)} + \cdots + \partial_{\theta_1}\ell^{(d)} \\ \partial_{\theta_2}\ell^{(1)} + \partial_{\theta_2}\ell^{(2)} + \cdots + \partial_{\theta_2}\ell^{(d)} \\ \vdots \\ \partial_{\theta_{n_w}}\ell^{(1)} + \partial_{\theta_{n_w}}\ell^{(2)} + \cdots + \partial_{\theta_{n_w}}\ell^{(d)} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{d} \partial_{\theta_1}\ell^{(i)} & \sum_{i=1}^{d} \partial_{\theta_2}\ell^{(i)} & \cdots & \sum_{i=1}^{d} \partial_{\theta_{n_w}}\ell^{(i)} \end{bmatrix}^T, \qquad (2.6)
\end{aligned}$$

and

$$\boldsymbol{g}_g(\boldsymbol{\theta}) = \boldsymbol{J}_g^T \boldsymbol{e}_g = \sum_{n=1}^{N} \boldsymbol{J}_n^T \boldsymbol{e}_n. \qquad (2.7)$$

As noted in [212], the GGN approximation has the advantage of capturing the curvature information of ℓ in $g(\boldsymbol{\theta})$ through the term $\boldsymbol{Q}_g$ as opposed to the FIM, for example, which ignores the full second-order interactions. While it may become obvious, say when training a deep neural network with certain loss functions, how the GGN approximation can be exploited to simplify expressions for $\boldsymbol{H}_g$ (see e.g. in [47]), a modification is required to take account of a twice-differentiable convex regularization function to achieve some degree of simplicity and elegance. We derive a modification to the above for the mini-batch scheme presented in the next section that includes the derivatives of $h(\boldsymbol{\theta})$ in the GGN approximation of $\boldsymbol{H}_g$. This modification leads to our GGN-SCORE algorithm in Section 2.4.

2.3 Second-order (pseudo-online) optimization

Suppose that at each mini-batch step k we uniformly select a random index set $\mathcal{I}_k \subseteq \{1, 2, \ldots, N\}, |\mathcal{I}_k| = m \leq N$ (usually $m \ll N$) to access

a mini-batch of m samples from the training set. The loss derivatives used for the recursive update of the variables $\boldsymbol{\theta}$ in this way is computed at each step k, and are estimated as running averages over the batch-wise computations. This leads to a stochastic approximation of the true derivatives at each iteration for which we assume unbiased estimations.

The problem of finding the optimal adjustment $\delta\boldsymbol{\theta}_{mk} \triangleq \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$ that solves (2.1) results in solving either an *overdetermined* or an *underdetermined* linear system depending on whether $dm \geq n_w$ or $dm < n_w$, respectively. Consider, for example, the squared fit loss and the penalty-inducing square norm as the scalar-valued functions $g(\boldsymbol{\theta})$ and $h(\boldsymbol{\theta})$, respectively in (2.1). Then, $\boldsymbol{Q}_g$ will be the identity matrix, and the LM solution $\delta\boldsymbol{\theta}$ [134, 146] is estimated at each iteration k according to the rule[1]:

$$\delta\boldsymbol{\theta} = -(\boldsymbol{H}_g + \lambda\boldsymbol{I})^{-1}\boldsymbol{g}_g = -(\boldsymbol{J}_g^T\boldsymbol{J}_g + \lambda\boldsymbol{I})^{-1}\boldsymbol{J}_g^T\boldsymbol{e}_g. \tag{2.8}$$

If $dm < n_w$ (possibly $dm \ll n_w$), then by using the Searle identity $(\boldsymbol{A}\boldsymbol{B} + \lambda\boldsymbol{I})\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{A} + \lambda\boldsymbol{I})$ [213], we can conveniently update the adjustment $\delta\boldsymbol{\theta}$ by

$$\delta\boldsymbol{\theta} = -\boldsymbol{J}_g^T(\boldsymbol{J}_g\boldsymbol{J}_g^T + \lambda\boldsymbol{I})^{-1}\boldsymbol{e}_g. \tag{2.9}$$

Clearly, this provides a more computationally efficient way of solving for $\delta\boldsymbol{\theta}$. In what follows, we formulate a generalized solution method for the regularized problem (2.1) which similarly exploits the Hessian matrix structure when solving the given optimization problem, thereby conveniently and efficiently computing the adjustment $\delta\boldsymbol{\theta}$.

Taking the second-order approximation of $\mathcal{L}(\boldsymbol{\theta})$, we have

$$\mathcal{L}(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}) + \boldsymbol{g}^T\delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T\boldsymbol{H}\delta\boldsymbol{\theta}, \tag{2.10}$$

where $\boldsymbol{H} \in \mathbb{R}^{n_w \times n_w}$ is the Hessian of $\mathcal{L}(\boldsymbol{\theta})$ and $\boldsymbol{g} \in \mathbb{R}^{n_w}$ is its gradient.

[1]For simplicity of notation, and unless where the full notations are explicitly required, we shall subsequently drop the subscripts m and k, and assume that each expression represents stochastic approximations performed at step k using randomly selected data batches each of size m.

Let $M = dm + 1$ and define the Jacobian $\boldsymbol{J} \in \mathbb{R}^{M \times n_w}$:

$$\boldsymbol{J}^T = \begin{bmatrix} \partial_{\theta_1} \hat{\boldsymbol{y}}_1 & \partial_{\theta_1} \hat{\boldsymbol{y}}_2 & \cdots & \partial_{\theta_1} \hat{\boldsymbol{y}}_m & \lambda \partial_{\theta_1} r_1 \\ \partial_{\theta_2} \hat{\boldsymbol{y}}_1 & \partial_{\theta_2} \hat{\boldsymbol{y}}_2 & \cdots & \partial_{\theta_2} \hat{\boldsymbol{y}}_m & \lambda \partial_{\theta_2} r_2 \\ \vdots & \vdots & & \vdots & \vdots \\ \partial_{\theta_{n_w}} \hat{\boldsymbol{y}}_1 & \partial_{\theta_{n_w}} \hat{\boldsymbol{y}}_2 & \cdots & \partial_{\theta_{n_w}} \hat{\boldsymbol{y}}_m & \lambda \partial_{\theta_{n_w}} r_{n_w} \end{bmatrix}. \tag{2.11}$$

Let $\boldsymbol{e}_M = 1$ and denote by $\boldsymbol{e} \in \mathbb{R}^M$ the vertical concatenation of all $\boldsymbol{e}_n$'s, $\boldsymbol{e}_n \in \mathbb{R}^d$, $n = 1, 2, \ldots, m$ and $\boldsymbol{e}_M$. Then by using the chain rule as in (2.6) and (2.7), we obtain

$$\boldsymbol{g}(\boldsymbol{\theta}) = \partial_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{J}^T \boldsymbol{e}. \tag{2.12}$$

Let $\boldsymbol{q}_n = \partial^2_{\hat{\boldsymbol{y}}_n \hat{\boldsymbol{y}}_n} \ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n)$, $\boldsymbol{q}_n \in \mathbb{R}^d$ for $n = 1, 2, \ldots, m$ (clearly $\boldsymbol{q}_n = \boldsymbol{1}$ in case of squared fit loss terms) and let $\boldsymbol{q}_M = 0$. Define $\boldsymbol{Q} \in \mathbb{R}^{M \times M}$ as the diagonal matrix with diagonal elements $\boldsymbol{q}_n$, $n = 1, 2, \ldots, m$ and $\boldsymbol{q}_M$, where $\boldsymbol{Q} = \boldsymbol{Q}^T \succeq \boldsymbol{0}$ by convexity of ℓ. Consider the following slightly modified GGN approximation of the Hessian $\boldsymbol{H} \in \mathbb{R}^{n_w \times n_w}$ associated with $\mathcal{L}(\boldsymbol{\theta})$:

$$\boldsymbol{H} \approx \boldsymbol{J}^T \boldsymbol{Q} \boldsymbol{J} + \lambda \boldsymbol{H}_h, \tag{2.13}$$

where $\boldsymbol{H}_h$ is the Hessian of the regularization term $h(\boldsymbol{\theta})$, $\boldsymbol{H}_h \in \mathbb{R}^{n_w \times n_w}$, and is a diagonal matrix whose diagonal terms are

$$\boldsymbol{H}_{h_{jj}} = \frac{d^2 r_j(\theta_j)}{d\theta_j^2},\ j = 1, \ldots, n_w.$$

We hold on to the notation $\boldsymbol{H}$ to represent the modified GGN approximation of the full Hessian matrix $\partial^2 \mathcal{L}$. By differentiating (2.10) with respect to $\delta\boldsymbol{\theta}$ and equating to zero, we obtain the optimal adjustment

$$\delta\boldsymbol{\theta} = -(\boldsymbol{J}^T \boldsymbol{Q} \boldsymbol{J} + \lambda \boldsymbol{H}_h)^{-1} \boldsymbol{J}^T \boldsymbol{e}. \tag{2.14}$$

Remark 1. *The inverse matrix in (2.14) exists due to the strong convexity assumption on the regularization function h which makes $\boldsymbol{H}_{\boldsymbol{r}} \succeq (\min_j \boldsymbol{H}_{h_{jj}})\boldsymbol{I}$ and therefore matrix $\boldsymbol{J}^T \boldsymbol{Q} \boldsymbol{J} + \lambda \boldsymbol{H}_h$ is symmetric positive definite and hence invertible.*

Let $\boldsymbol{U} = \boldsymbol{J}^T\boldsymbol{Q}$. Using the identity [87, 106]

$$(\boldsymbol{D} - \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1} = \boldsymbol{D}^{-1}\boldsymbol{V}(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{V})^{-1} \tag{2.15}$$

with $\boldsymbol{D} = \lambda\boldsymbol{H}_{\boldsymbol{r}}$, $\boldsymbol{V} = -\boldsymbol{J}^T$, $\boldsymbol{A} = \boldsymbol{I}_M$, and $\boldsymbol{B} = \boldsymbol{U}^T$, and recalling that $\boldsymbol{Q}$ is symmetric, from (2.14) we get

$$\begin{aligned}
\delta\boldsymbol{\theta} &= \left(\lambda\boldsymbol{H}_h - (-\boldsymbol{J}^T)\boldsymbol{I}_M\boldsymbol{U}^T\right)^{-1}(-\boldsymbol{J}^T)\boldsymbol{I}_M^{-1}\boldsymbol{e} \\
&= -\frac{1}{\lambda}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\left(\boldsymbol{I}_M + \boldsymbol{U}^T\frac{1}{\lambda}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{e} \\
&= -\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\left(\lambda\boldsymbol{I}_M + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{e}.
\end{aligned} \tag{2.16}$$

Remark 2. *When combined with a second identity, namely* $\boldsymbol{V}\boldsymbol{A}^{-1}(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{V}) = (\boldsymbol{D} - \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{B})\boldsymbol{D}^{-1}\boldsymbol{V}$*, one can directly derive from* (2.15) *Woodbury identity defined as [114]* $(\boldsymbol{A} + \boldsymbol{U}\boldsymbol{B}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{B} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}$*, or in the special case* $\boldsymbol{B} = -\boldsymbol{D}^{-1}$*, as* $(\boldsymbol{A} - \boldsymbol{U}\boldsymbol{D}^{-1}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{U}(\boldsymbol{D} - \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}$*. Using Woodbury identity would, in fact,* structurally *not result into the closed-form update step* (2.16) *in an exact sense. Our construction involves a more general regularization function than the commonly used square norm, where the Woodbury identity can be equally useful, as its Hessian yields a multiple of the identity matrix.*

Compared to (2.14), the clear advantage of the form (2.16) is that it requires the factorization of an $M \times M$ matrix rather than an $n_w \times n_w$ matrix, where the term $\boldsymbol{H}_h^{-1}$ can be conveniently obtained by exploiting its diagonal structure. Given these modifications, we proceed by making an assumption that defines the residual $\boldsymbol{e}$ and the Jacobian $\boldsymbol{J}$ in the region of convergence where we assume the starting point $\boldsymbol{\theta}_0$ of the process (2.16) lies.

Let $\boldsymbol{\theta}^*$ be a nondegenerate minimizer of $\mathcal{L}$, and define $\mathcal{B}_\epsilon(\boldsymbol{\theta}^*) \triangleq \{\boldsymbol{\theta}_k \in \mathbb{R}^{n_w} : \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \leq \epsilon\}$, a closed ball of a sufficiently small radius $\epsilon \geq 0$ about $\boldsymbol{\theta}^*$. We denote by $\mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$ an open neighbourhood of the sublevel set $\Gamma(\mathcal{L}) \triangleq \{\boldsymbol{\theta}_k : \mathcal{L}(\boldsymbol{\theta}_k) \leq \mathcal{L}(\boldsymbol{\theta}_0)\}$, so that $\mathcal{B}_\epsilon(\boldsymbol{\theta}^*) = \mathrm{cl}(\mathcal{N}_\epsilon(\boldsymbol{\theta}^*))$. We then have $\mathcal{N}_\epsilon(\boldsymbol{\theta}^*) \triangleq \{\boldsymbol{\theta}_k \in \mathbb{R}^{n_w} : \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| < \epsilon\}$.

Assumption 4. *(i) Each* $\boldsymbol{e}_n(\boldsymbol{\theta}_k)$ *and each* $\boldsymbol{q}_n(\boldsymbol{\theta}_k)$ *is Lipschitz smooth, and* $\forall\boldsymbol{\theta}_k \in \mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$ *there exists* $\nu > 0$ *such that* $\|\boldsymbol{J}(\boldsymbol{\theta}_k)\boldsymbol{z}\| \geq \nu\|\boldsymbol{z}\|$.

(ii) $\lim_{k\to\infty} \left\|\mathbb{E}_m[\boldsymbol{H}(\boldsymbol{\theta}_k)] - \partial^2\mathcal{L}(\boldsymbol{\theta}_k)\right\| = 0$ *almost surely whenever* $\lim_{k\to\infty} \|\boldsymbol{g}_{\mathcal{L}}(\boldsymbol{\theta}_k)\| = 0, \forall \boldsymbol{\theta}_k \in \mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$, *where* $\mathbb{E}_m[\cdot]$ *denotes*[2] *expectation with respect to* m.

Remark 3. *Assumption 4(i) implies that the singular values of* $\boldsymbol{J}$ *are uniformly bounded away from zero and* $\exists \beta, \tilde{\beta} > 0$ *such that* $\|\boldsymbol{e}\| \leq \beta, \|\boldsymbol{J}\| = \|\boldsymbol{J}^T\| \leq \tilde{\beta}$, *then as* $\boldsymbol{Q} \succeq \boldsymbol{0}$, *we have* $\exists K_1$ *such that* $\boldsymbol{Q} \leq K_1\boldsymbol{I}$, *and hence* $\|\lambda\boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\| \leq \lambda + (K/\gamma_a)$, *where* $K = K_1\tilde{\beta}^2$. *Note that although we use limits in Assumption 4(ii), the assumption similarly holds in expectation by unbiasedness. Also, a sufficient sample size* m *may be required for Assumption 4(ii) to hold, by law of large numbers, see e.g. [198, Lemma 1, Lemma 2].*

Remark 4. $\boldsymbol{H}_g(\boldsymbol{\theta})$ *and* $\boldsymbol{H}_h(\boldsymbol{\theta})$ *satisfy*

$$\begin{aligned}
&\gamma_l\boldsymbol{I}_{n_w} \preceq \boldsymbol{H}_g(\boldsymbol{\theta}_k) \preceq \gamma_u\boldsymbol{I}_{n_w},\\
&\gamma_a\boldsymbol{I}_{n_w} \preceq \boldsymbol{H}_h(\boldsymbol{\theta}_k) \preceq \gamma_b\boldsymbol{I}_{n_w},\\
&\left\|\boldsymbol{H}_g(\boldsymbol{y}, f(\boldsymbol{\theta_1};\boldsymbol{x})) - \boldsymbol{H}_g(\boldsymbol{y}, f(\boldsymbol{\theta_2};\boldsymbol{x}))\right\| \leq \gamma_g \left\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\right\|,\\
&\left\|\boldsymbol{H}_h(\boldsymbol{y}, f(\boldsymbol{\theta_1};\boldsymbol{x})) - \boldsymbol{H}_h(\boldsymbol{y}, f(\boldsymbol{\theta_2};\boldsymbol{x}))\right\| \leq \gamma_h \left\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\right\|,
\end{aligned}$$

at any point $\boldsymbol{\theta}_k \in \mathbb{R}^{n_w}$, *for any* $\boldsymbol{x} \in \mathbb{R}^{n_p}, \boldsymbol{y} \in \mathbb{R}^d$, *and for any* $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$ *where Assumption 4 holds.*

We now state a convergence result for the update type (2.14) (and hence (2.16)). First, we define the second-order optimality condition and state two useful lemmas.

Definition 2 (Second-order sufficiency condition (SOSC))**.** *Let* $\boldsymbol{\theta}^*$ *be a local minimum of a twice-differentiable function* $\mathcal{L}(\cdot)$. *The second-order sufficiency condition (SOSC) holds if*

$$\partial\mathcal{L}(\boldsymbol{\theta}^*) = 0, \quad \partial^2\mathcal{L}(\boldsymbol{\theta}^*) \succ 0. \tag{SOSC}$$

Lemma 1 ([163, Theorem 1.2.3])**.** *Suppose that Assumption 3 holds. Let* $\mathbb{W} \subseteq \mathbb{R}^{n_w}$ *be a closed and convex set on which* $\mathcal{L}(\boldsymbol{\theta})$ *is twice-continuously differentiable. Let* $S \subset \mathbb{W}$ *be an open set containing some* $\boldsymbol{\theta}^*$, *and suppose that* $\mathcal{L}(\boldsymbol{\theta}^*)$ *satisfies (SOSC). Then, there exists* $\mathcal{L}^* = \mathcal{L}(\boldsymbol{\theta}^*)$ *satisfying*

$$\mathcal{L}(\boldsymbol{\theta}_k) > \mathcal{L}^* \quad \forall \boldsymbol{\theta}_k \in S. \tag{2.18}$$

[2]Subsequently, we shall omit the notation for the Hessian and gradient estimates as we assume unbiasedness.

Lemma 2. *The adjustment $\delta\boldsymbol{\theta}$ given by (2.14) (and hence, (2.16)) provides a descent direction for the total loss $\mathcal{L}(\boldsymbol{\theta}_k)$ in (2.1) at the k^{th} oracle call.*

Remark 5. *Lemma 1, Lemma 2 and the first part of (SOSC) ensure that the second part of Assumption 4(ii) always holds. In essence, it holds at every point $\boldsymbol{\theta}_k$ of the sequence $\{\boldsymbol{\theta}_k\}$ generated by the process (2.16) as long as we choose a starting point $\boldsymbol{\theta_0} \in \mathcal{B}_\epsilon(\boldsymbol{\theta}^*)$.*

Theorem 1. *Suppose that Assumptions 1, 2, 3 and 4 hold, and that $\boldsymbol{\theta}^*$ is a local minimizer of $\mathcal{L}(\boldsymbol{\theta})$ for which the assumptions in Lemma 1 hold. Let $\{\boldsymbol{\theta}_k\}$ be the sequence generated by the process (2.16). Then starting from a point $\boldsymbol{\theta}_0 \in \mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$, $\{\boldsymbol{\theta}_k\}$ converges at a Q-quadratic rate. Namely:*

$$\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\| \leq \xi_k \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2,$$

where

$$\xi_k = \frac{\gamma_g + b\gamma_h}{2\left(\gamma_l + a\gamma_a - (\gamma_g + b\gamma_h)\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|\right)}.$$

The proofs of these results are reported in Appendix 2.A and Appendix 2.B.

2.4 Self-concordant regularization (SCORE)

In the case $dm < n_w$, one could deduce by mere visual inspection that the matrix $\boldsymbol{H}_h^{-1}$ in (2.16) plays an important role in the perturbation of $\boldsymbol{\theta}$ within the region of convergence due to its "dominating" structure in the equation. This may be true as it appears. But beyond this "naive" view, let us derive a more technical intuition about the update step (2.16) by using a similar analogy as that given in [163, Chapter 5]. We have

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{H}_h^{-1}\boldsymbol{J}^T\left(\lambda\boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\right)^{-1}\boldsymbol{e}, \tag{2.19}$$

where $\boldsymbol{I}$ is the identity matrix of suitable dimension. By some simple algebra (see e.g., [163, Lemma 5.1.1]), one can show that indeed the update method (2.19) is affine-invariant. In other words, the region of convergence $\mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$ does not depend on the problem scaling but on the

local topological structure of the minimization functions $g(\boldsymbol{\theta})$ and $h(\boldsymbol{\theta})$ [163].

Consider the non-augmented form of (2.19): $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{Q}_g \boldsymbol{J}_g^T \boldsymbol{G}_g^{-1} \boldsymbol{e}_g$, where $\boldsymbol{G}_g = \boldsymbol{J}_g \boldsymbol{J}_g^T$ is the so-called *Gram matrix*. It has been shown that indeed when learning an overparameterized model (sample size smaller than number of variables), and as long as we start from an initial point $\boldsymbol{\theta}_0$ close enough to the minimizer $\boldsymbol{\theta}^*$ of $\mathcal{L}(\boldsymbol{\theta})$ (assuming that such a minimizer exists), both $\boldsymbol{J}_g$ and $\boldsymbol{G}_g$ remain stable throughout the learning process (see e.g., [57, 83, 84]). The term $\boldsymbol{Q}_g$ may have little to no effect on this notion, for example, in case of the squared fit loss, $\boldsymbol{Q}_g$ is just an identity term. However, the original Equation (2.19) involves the Hessian matrix $\boldsymbol{H}_h$ which, together with its bounds (namely, γ_a, γ_b characterizing the region of convergence), changes rapidly when we scale the problem [163]. It therefore suffices to impose an additional assumption on the function $h(\boldsymbol{\theta})$ that will help to control the rate at which its second-derivative $\boldsymbol{H}_h$ changes, thereby enabling us to characterize an affine-invariant structure for the region of convergence. Namely:

Assumption 5 (SCORE)**.** *The scaled regularization function $\lambda h(\boldsymbol{\theta})$ has a third derivative and is M_h-self-concordant. That is, the inequality*

$$\left| \left\langle \boldsymbol{u}, \left(\partial^3 h(\boldsymbol{\theta})[\boldsymbol{u}] \right) \boldsymbol{u} \right\rangle \right| \leq 2 M_h \left\langle \boldsymbol{u}, \partial^2 h(\boldsymbol{\theta}) \boldsymbol{u} \right\rangle^{3/2}, \tag{2.20}$$

where $M_h \geq 0$, holds for any $\boldsymbol{\theta}$ in the closed convex set $\mathbb{W} \subseteq \mathbb{R}^{n_w}$ and $\boldsymbol{u} \in \mathbb{R}^{n_w}$.

Here, $\partial^3 h(\boldsymbol{\theta})[\boldsymbol{u}] \in \mathbb{R}^{n_w \times n_w}$ denotes the limit

$$\partial^3 h(\boldsymbol{\theta})[\boldsymbol{u}] \triangleq \lim_{t \to 0} \frac{1}{t} \left(\partial^2(\boldsymbol{\theta} + t\boldsymbol{u}) - \partial^2(\boldsymbol{\theta}) \right), \quad t \in \mathbb{R}.$$

For a detailed analysis of self-concordant functions, we refer the interested reader to [164, 163].

Given this *extra* assumption, and following the idea of Newton-decrement in [164], we propose to update $\boldsymbol{\theta}$ by

$$\delta\boldsymbol{\theta} = -\frac{\alpha}{1 + M_h \eta_k} \boldsymbol{H}_h^{-1} \boldsymbol{J}^T \left(\lambda \boldsymbol{I} + \boldsymbol{Q} \boldsymbol{J} \boldsymbol{H}_h^{-1} \boldsymbol{J}^T \right)^{-1} \boldsymbol{e}, \tag{2.21}$$

where $\alpha > 0$, $\eta_k = \left\langle \boldsymbol{g}_h, \boldsymbol{H}_h^{-1}\boldsymbol{g}_h \right\rangle^{1/2}$ and $\boldsymbol{g}_h$ is the gradient of $h(\boldsymbol{\theta})$. The proposed method which we call GGN-SCORE is summarized, for one oracle call, in Algorithm 1. There is a wide class of self-concordant functions

Algorithm 1 GGN-SCORE

1: **Input:** variables vector $\boldsymbol{\theta}_k$, data $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^m$, $\boldsymbol{H}_h$ (see (2.13)), $\boldsymbol{Q}$ (see (2.13)), $\boldsymbol{J}$ (see (2.13)), $\boldsymbol{e}$ (see (2.12)), parameters α, M_h, λ
2: **Output:** variables vector $\boldsymbol{\theta}_{k+1}$
3: Compute $\boldsymbol{g}_h = \partial_{\boldsymbol{\theta}_k} h(\boldsymbol{\theta}_k)$
4: Choose $\eta_k = \left\langle \boldsymbol{g}_h, \boldsymbol{H}_h^{-1}\boldsymbol{g}_h \right\rangle^{1/2}$
5: Set $\rho_k = \frac{\alpha}{1+M_h\eta_k}$
6: Solve $\left(\lambda \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\right)\boldsymbol{b} = \boldsymbol{e}$ for $\boldsymbol{b}$
7: Set $\boldsymbol{G} = \boldsymbol{H}_h^{-1}\boldsymbol{J}^T\boldsymbol{b}$
8: Compute $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \rho_k \boldsymbol{G}$

that meet practical requirements, either in their scaled form [163, Corollary 5.1.3] or in their original form (see e.g., in [223] for a comprehensive list). Two of them are used in our experiments in Section 4.4.

In the following, we state a local convergence result for the process (2.21). We introduce the notation $\|\Delta\|_{\boldsymbol{\theta}}$ to represent the local norm of a direction Δ taken with respect to the local Hessian of a self-concordant function h, $\|\Delta\|_{\boldsymbol{\theta}} \triangleq \left\langle \Delta, \partial^2_{\boldsymbol{\theta}\boldsymbol{\theta}} h(\boldsymbol{\theta})\Delta \right\rangle^{1/2} = \left\| \left[\partial^2_{\boldsymbol{\theta}\boldsymbol{\theta}} h(\boldsymbol{\theta})\right]^{1/2} \Delta \right\|$. Hence, without loss of generality, a ball $\mathcal{B}_\epsilon(\boldsymbol{\theta}^*)$ of radius ϵ about $\boldsymbol{\theta}^*$ is taken with respect to the local norm for the function h in the result that follows.

Theorem 2. *Let Assumptions 1, 2, 3, 4 and 5 hold, and let $\boldsymbol{\theta}^*$ be a local minimizer of $\mathcal{L}(\boldsymbol{\theta})$ for which the assumptions in Lemma 1 hold. Let $\{\boldsymbol{\theta}_k\}$ be the sequence generated by Algorithm 1 with $\alpha = \frac{\sqrt{\gamma_a}}{\beta_1}(K + \lambda\gamma_a)$, $\beta_1 = \beta\tilde{\beta}$. Then starting from a point $\boldsymbol{\theta}_0 \in \mathcal{N}_{M_h^{-1}}(\boldsymbol{\theta}^*)$, $\{\boldsymbol{\theta}_k\}$ converges to $\boldsymbol{\theta}^*$ according to the following descent properties:*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{k+1})] \leq \mathcal{L}(\boldsymbol{\theta}_k) - \left(\frac{\lambda}{M_h^2}\omega(\zeta_k) + \frac{\gamma_l}{2\gamma_a}\omega''(\tilde{\zeta}_k) - \xi\right),$$

$$\mathbb{E}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\|_{\boldsymbol{\theta}_{k+1}} \leq \vartheta\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|_{\boldsymbol{\theta}_k} + \frac{\gamma_u}{\beta_1}\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| + \frac{\gamma_g}{2}\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2,$$

where $\omega(\cdot)$ is an auxiliary univariate function defined by $\omega(t) \triangleq t - \ln(1+t)$ and has a second derivative $\omega''(t) = 1/(1+t)^2$, and

$$\zeta_k \triangleq \frac{M_h}{1 + M_h \eta_k}, \quad \tilde{\zeta}_k \triangleq M_h \eta_k, \quad \xi \triangleq \frac{2(\gamma_u + \lambda \gamma_b)}{\sqrt{\gamma_a}},$$
$$\vartheta \triangleq 1 + \frac{\lambda}{\sqrt{\gamma_a}\beta_1(1 - M_h \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|_{\boldsymbol{\theta}_k})}.$$

The proof is provided in Appendix 2.B. The results of Theorem 2 combine strong convexity and smoothness properties of both $g(\boldsymbol{\theta})$ and $h(\boldsymbol{\theta})$, and requires that only $h(\boldsymbol{\theta})$ is self-concordant.

2.5 Experiments

In this section, we validate the efficiency of GGN-SCORE (Algorithm 1) in solving the problem of minimizing a regularized strongly convex quadratic function and in solving binary classification tasks. For the binary classification tasks, we use real datasets: `ijcnn1` and `w2a` from the LIBSVM repository [65] and `dis`, `hypothyroid` and `coil2000` from the PMLB repository [197], with an 80:20 train:test split each. The datasets are summarized in Table **1**. In each classification task, the model with a sigmoidal output $\hat{\boldsymbol{y}}_n$ is trained using the cross-entropy fit loss function $\ell(\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n) = \frac{1}{2}\sum_{n=1}^{N} \boldsymbol{y}_n \log\left(\frac{1}{\hat{\boldsymbol{y}}_n}\right) + (\mathbf{1} - \boldsymbol{y}_n)\log\left(\frac{1}{\mathbf{1}-\hat{\boldsymbol{y}}_n}\right)$, and the "deviance" residual [88] $\boldsymbol{e}_n = (-1)^{\boldsymbol{y}_n+1}\sqrt{-2\left[\boldsymbol{y}_n \log(\hat{\boldsymbol{y}}_n) + (\mathbf{1} - \boldsymbol{y}_n)\log(\mathbf{1} - \hat{\boldsymbol{y}}_n)\right]}$, $\boldsymbol{y}_n, \hat{\boldsymbol{y}}_n \in \{\mathbf{0}, \mathbf{1}\}$. We map the input space to higher dimensional feature space using the radial basis function (RBF) kernel $K(\boldsymbol{x}_n, \boldsymbol{x}'_n) = \exp(-\gamma\|\boldsymbol{x}_n - \boldsymbol{x}'_n\|^2)$ with $\gamma = 0.1$. In each experiment, we use the penalty value $\lambda = 0.1$ with both pseudo-Huber regularization $h_\mu(\boldsymbol{\theta})$ [172] parameterized by $\mu > 0$ [67, 110] and ℓ^2 regularization $h_2(\boldsymbol{\theta})$ defined respectively as

$$h_\mu(\boldsymbol{\theta}) \triangleq \sqrt{\mu^2 + \boldsymbol{\theta}^2} - \mu, \qquad h_2(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2 \triangleq \sum_{i=1}^{n_w} |\theta_i|^2,$$

with coefficient $\mu = 1.0$. Throughout, we choose a batch size, m of 512 for `w2a`, `dis` and `hypothyroid`, 2048 for `coil2000` and 4096 for

Table 1: Real datasets: N = number of data samples, n_p = number of features, d = number of targets.

dataset	N	n_p before & after feature mapping	d
`dis`	3772	29 3017	1
`hypothyroid`	3163	25 2530	1
`w2a`	3470	300 2776	1
`ijcnn1`	35000	22 28000	1
`coil2000`	9822	85 7857	1

`ijcnn1`, unless otherwise stated. We assume a scaled self-concordant regularization so that $M_h = 1$ [163, Corollary 5.1.3].

2.5.1 GGN-SCORE for different values of α

To illustrate the behaviour of GGN-SCORE for different values of α in Algorithm 1 versus its value indicated in Theorem 2, we consider the problem of minimizing a regularized strongly convex quadratic function:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{2}\boldsymbol{\theta}^\top \hat{\boldsymbol{Q}}\boldsymbol{\theta} - \boldsymbol{p}^\top\boldsymbol{\theta} + \lambda h(\boldsymbol{\theta}) \equiv g(\boldsymbol{\theta}) + \lambda h(\boldsymbol{\theta}), \tag{2.22}$$

where $\hat{\boldsymbol{Q}} \in \mathbb{R}^{n_w \times n_w}$ is symmetric positive definite, $\boldsymbol{p} \in \mathbb{R}^{n_w}$, g is γ_a-strongly convex and has γ_u-Lipschitz gradient, with the smallest and largest eigenvalues of $\hat{\boldsymbol{Q}}$ corresponding to γ_a and γ_u, respectively. For this function, suppose the gradient and Hessian of $h(\boldsymbol{\theta})$ is known, for example when we choose $h = h_\mu$ or $h = h_2$, we have $\partial\mathcal{L}(\boldsymbol{\theta}) = \hat{\boldsymbol{Q}}\boldsymbol{\theta} - \boldsymbol{p} + \lambda\partial h(\boldsymbol{\theta})$ and $\partial^2\mathcal{L}(\boldsymbol{\theta}) = \hat{\boldsymbol{Q}} + \lambda\partial^2 h(\boldsymbol{\theta})$. The coefficients $\hat{\boldsymbol{Q}}$ and $\boldsymbol{p}$ form our data, and with

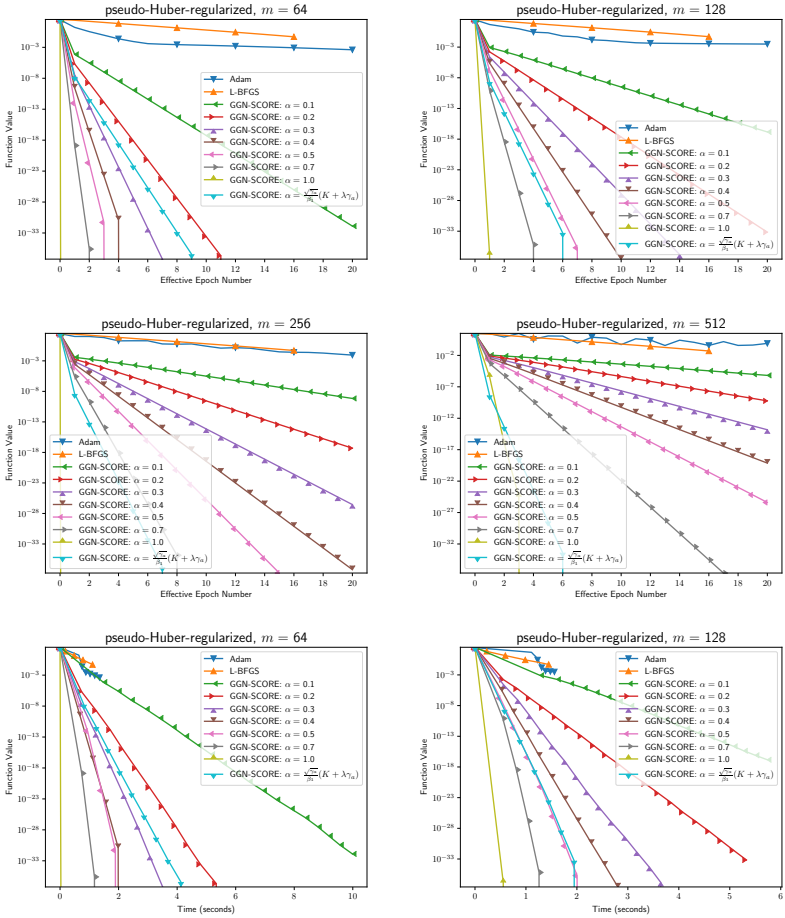


Figure 5: Numerical behaviour of GGN-SCORE for different values of α in the strongly convex quadratic test problem (2.22).

$\hat{\boldsymbol{Q}} \equiv 0.1 \times \boldsymbol{M}^\top \boldsymbol{M}$, $N = n_w = 1000$, we generate the data $\boldsymbol{M} \in \mathbb{R}^{N \times N}$ randomly from a uniform distribution on $[0, 1]$ and consider the case $\mathcal{L}^* = \mathbf{0}$ in which $\boldsymbol{p}$ is the zero vector. The optimization variable $\boldsymbol{\theta}$ is initialized to a random value generated from a normal distribution with mean 0 and standard deviation 0.01. Figure 5, Figure 6 and Figure 7 show the behaviour

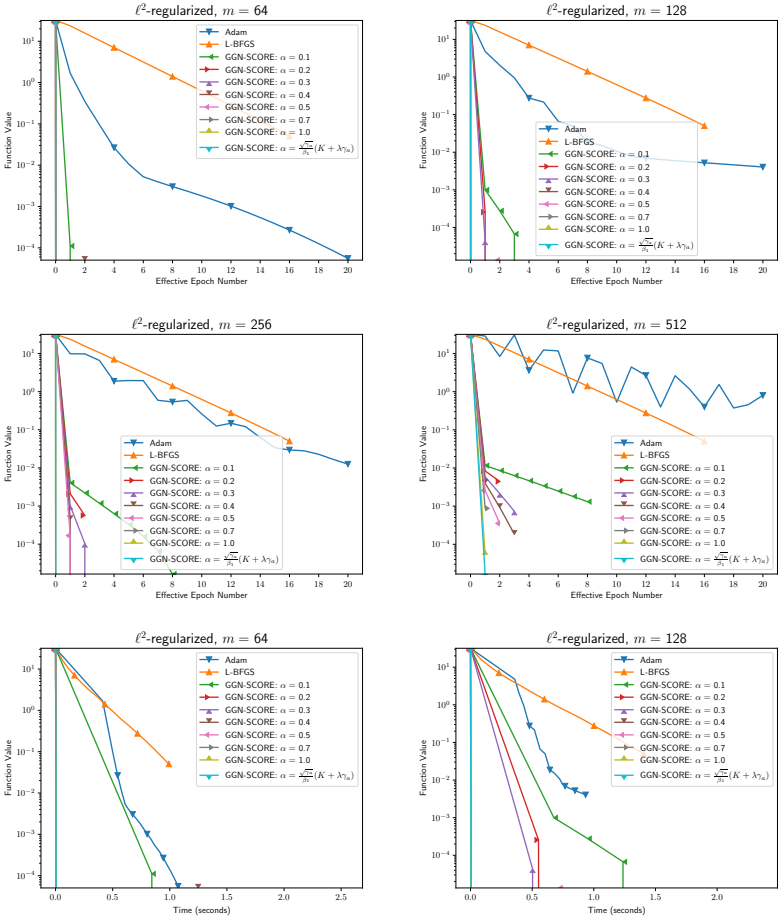


Figure 6: Numerical behaviour of GGN-SCORE for different values of α in the strongly convex quadratic test problem (2.22).

of GGN-SCORE for this problem with different values of α in $(0, 1]$ and $\alpha = \frac{\sqrt{\gamma_a}}{\beta_1}(K + \lambda\gamma_a)$ indicated in Theorem 2. We experiment with different batch sizes $m \in \{64, 128, 256, 512\}$. One observes from Figure 5, Figure 6 and Figure 7 that larger batch size yields better convergence speed when we choose $\alpha = \frac{\sqrt{\gamma_a}}{\beta_1}(K+\lambda\gamma_a)$, validating the recommendation in Remark 3.

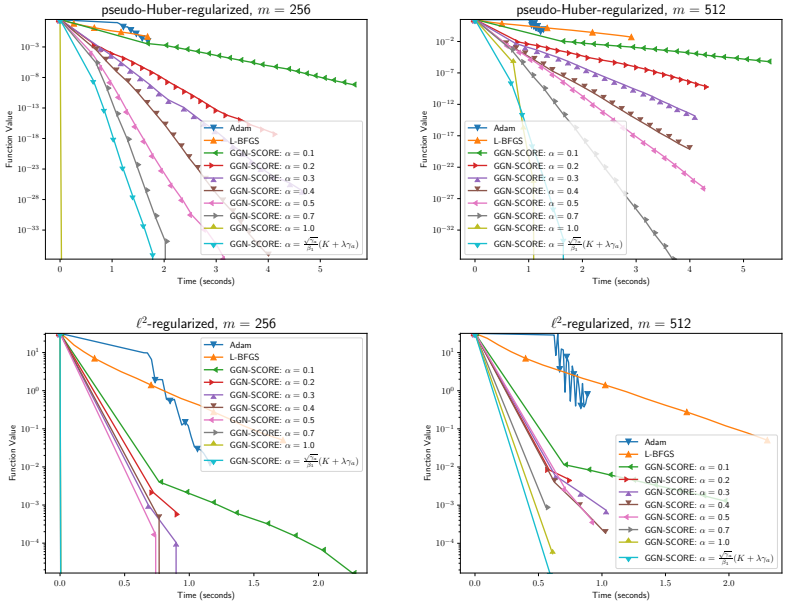


Figure 7: Numerical behaviour of GGN-SCORE for different values of α in the strongly convex quadratic test problem (2.22).

Figure 5, Figure 6 and Figure 7 also show the comparison of GNN-SCORE with the first-order Adam [128] algorithm, and the quasi-Newton Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [167] method using optimally tuned learning rates. While choosing $\alpha = \frac{\sqrt{\gamma_a}}{\beta_1}(K + \lambda\gamma_a)$ yields the kind of convergence shown in Theorem 2, Figure 5, Figure 6 and Figure 7 show that by choosing α in $(0, 1]$, we can similarly achieve a great convergence that scale well with the problem.

Strictly speaking, the value of α indicated in Theorem 2 is not of practical interest, as it contains terms that may not be straightforward to retrieve in practice. In practice, we treat α as a hyperparameter that takes a *fixed* positive value in $(0, 1]$. For an adaptive step-size selection rule, such as that in Line 5 of Algorithm 1, choosing a suitable scaling constant such as α is often straightforward, as the main step-size selection

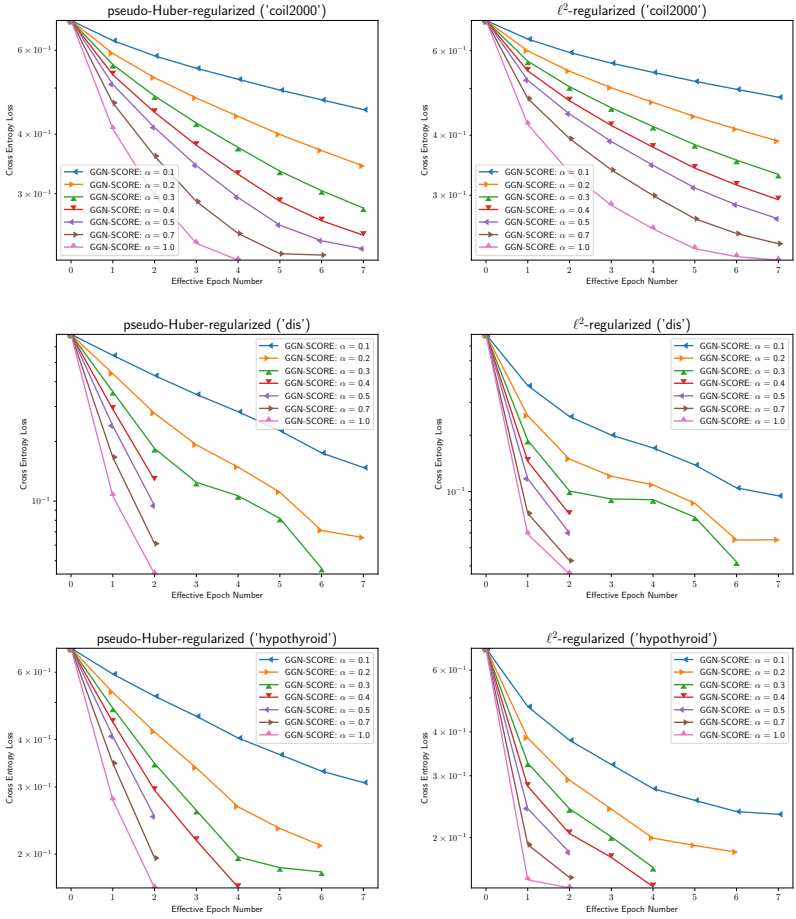


Figure 8: Numerical behaviour of GGN-SCORE for different values of α using real datasets. $m = 512$ for `dis` and `hypothyroid`, and $m = 2048$ for `coil2000`.

task is accomplished by the defined rule. We show the behaviour of GGN-SCORE on the real datasets for different values of α in $(0, 1]$ in Figure 8 and Figure 9. In general, a suitable scaling factor α should be selected based on the application demands.

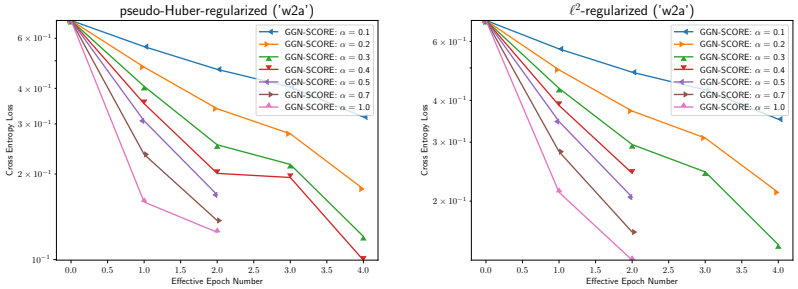


Figure 9: Numerical behaviour of GGN-SCORE for different values of α using the `w2a` dataset with $m = 512$.

2.5.2 Comparison with SGD, Adam, and L-BFGS methods on real datasets

Using the real datasets, we compare GGN-SCORE for solving (2.1) with results from the SGD, Adam, and the L-BFGS algorithms using optimally tuned learning rates. We also consider the training problem of a neural network with two hidden layers of dimensions $(2, 128)$, respectively for the `coil2000` dataset, one hidden layer with dimension 1 for the `ijcnn1` dataset, and two hidden layers of dimensions $(4, 128)$, respectively for the remaining datasets. We use ReLU activation functions in the hidden layers of the networks, and the network is overparameterized for `dis`, `hypothyroid`, `w2a` and `coil2000` with 25425, 21529, 23497 and 16229 trainable parameters, respectively. We choose $\alpha \in \{0.2, 0.5\}$ for GGN-SCORE. Minimization variables are initialized to the zero vector for all the methods. The neural network training problems are solved under the same settings. The results are respectively displayed in Figure **10**–Figure **16** for the convex and non-convex cases.

To investigate how well the learned model generalizes, we use the binary accuracy metric which measures how often the model predictions match the true labels when presented with new, previously unseen data: $Accuracy = \frac{1}{N}\sum_{n=1}^{N}(2\boldsymbol{y}_n\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n - \hat{\boldsymbol{y}}_n + 1)$. While GGN-SCORE converges faster than SGD, Adam and L-BFGS methods, it generalizes

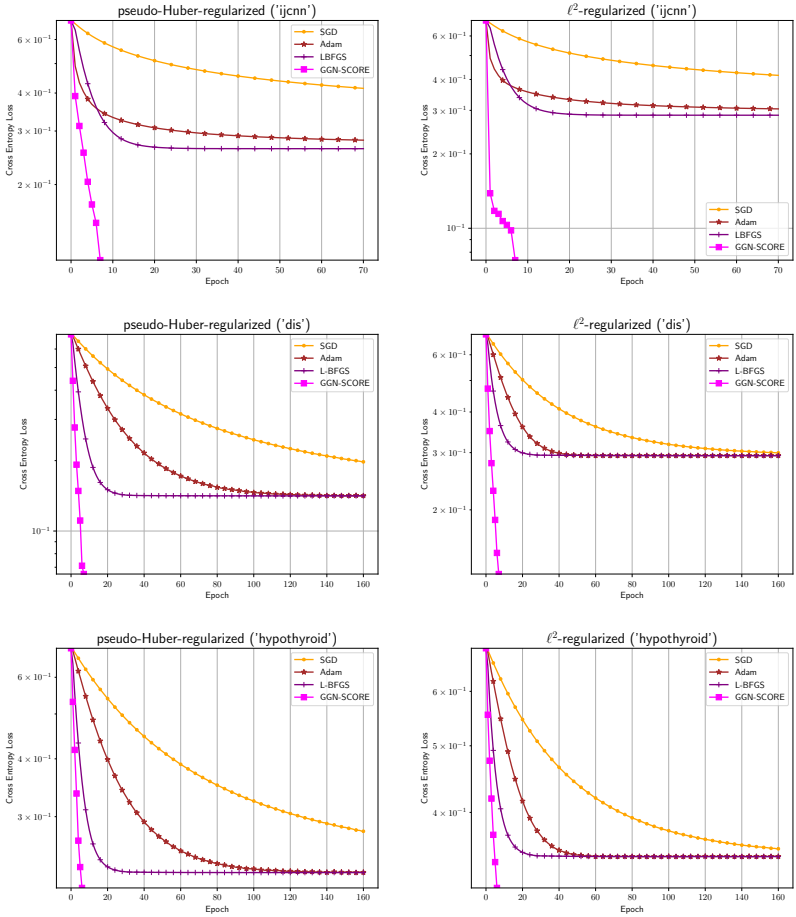


Figure 10: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the proposed convex problem. $m = 512$ for `dis` and `hypothyroid`, and $m = 4096$ for `ijcnn1`.

comparatively well. The results are further compared with other known binary classification techniques to measure the quality of our solutions. The accuracy scores for `dis`, `hypothyroid`, `coil2000` and `w2a` datasets, with a 60:40 train:test split each, are computed on the test set. The mean

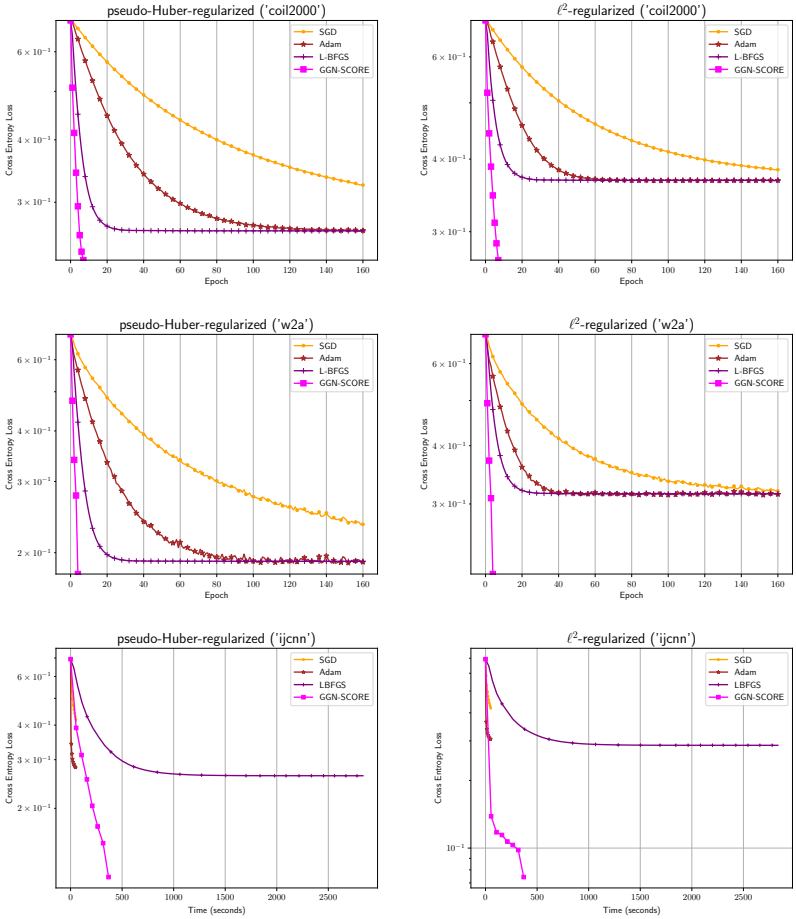


Figure 11: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the proposed convex problem. $m = 512$ for `w2a`, $m = 2048$ for `coil2000`, and $m = 4096$ for `ijcnn1`.

scores are compared with those from the different classification techniques, and are shown in Figure **17**. The CPU runtimes are also compared where it is indicated that on average GGN-SCORE solves each of the problems within one second. This scales well with the other techniques,

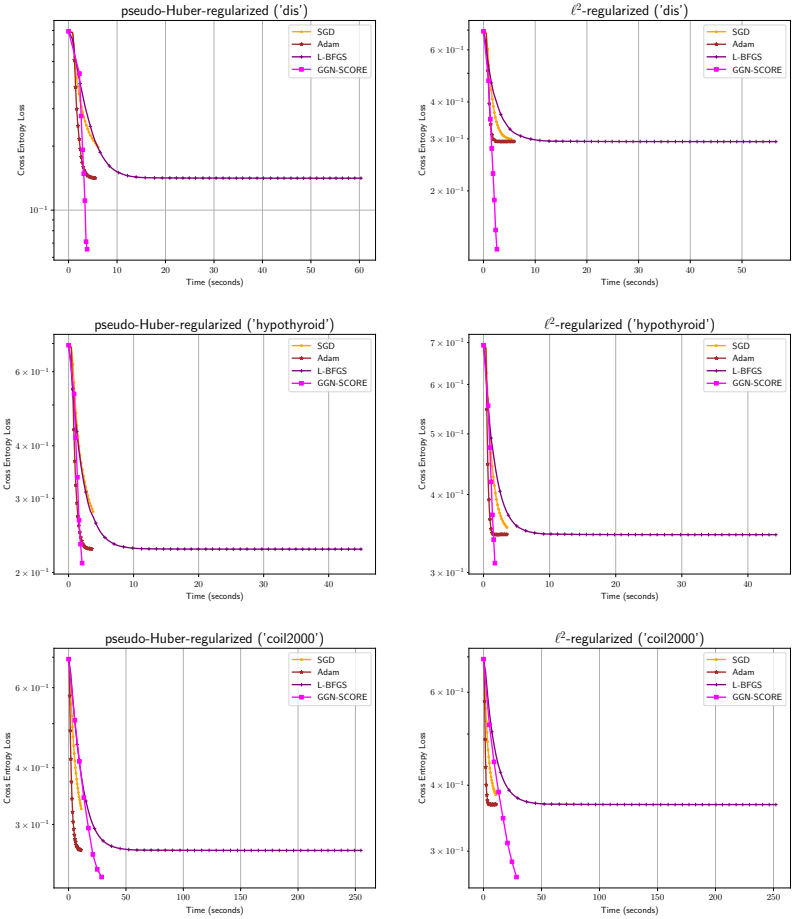


Figure 12: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the proposed convex problem. $m = 512$ for `dis` and `hypothyroid` and $m = 2048$ for `coil2000`.

as we note that while GNN-SCORE solves each of the problems in high dimensions, the success of most of the other techniques are limited to relatively smaller dimensions of the problems. The obtained results from the classification techniques used for comparison are computed directly

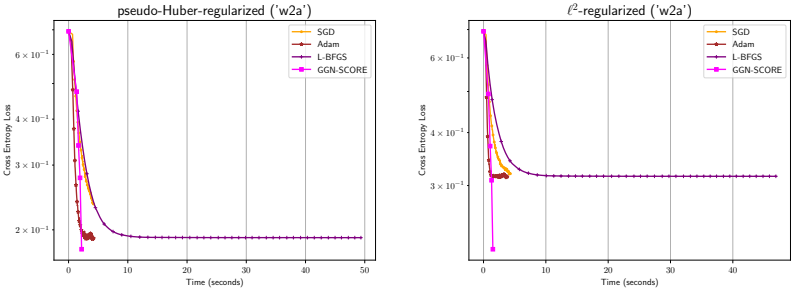


Figure 13: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the proposed convex problem with $m = 512$.

from the respective scikit-learn [183] functions.

The L-BFGS experiments are implemented with PyTorch [178] (v. 1.10.1+cu102) in full-batch mode. The GGN-SCORE, Adam and SGD methods are implemented using the open-source Keras API with TensorFlow [151] backend (v. 2.7.0). All experiments are performed on a laptop with dual (2.30GHz + 2.30GHz) Intel Core i7-11800H CPU and 16GB RAM.

In summary, GGN-SCORE converges way faster (in terms of number of epochs) than SGD, Adam, and L-BFGS, and generalizes comparatively well. Experimental results show the computational convenience and elegance achieved by our "augmented" approach for including regularization functions in the GGN approximation. Although GGN-SCORE comes with a higher computational cost (in terms of wall-clock time per iteration) than first-order methods on average, if per-iteration learning time is not provided as a bottleneck, this may not become an obvious issue as we need to pass the proposed optimizer on the dataset only a few times (epochs) to obtain superior function approximation and relatively high-quality solutions in our experiments.

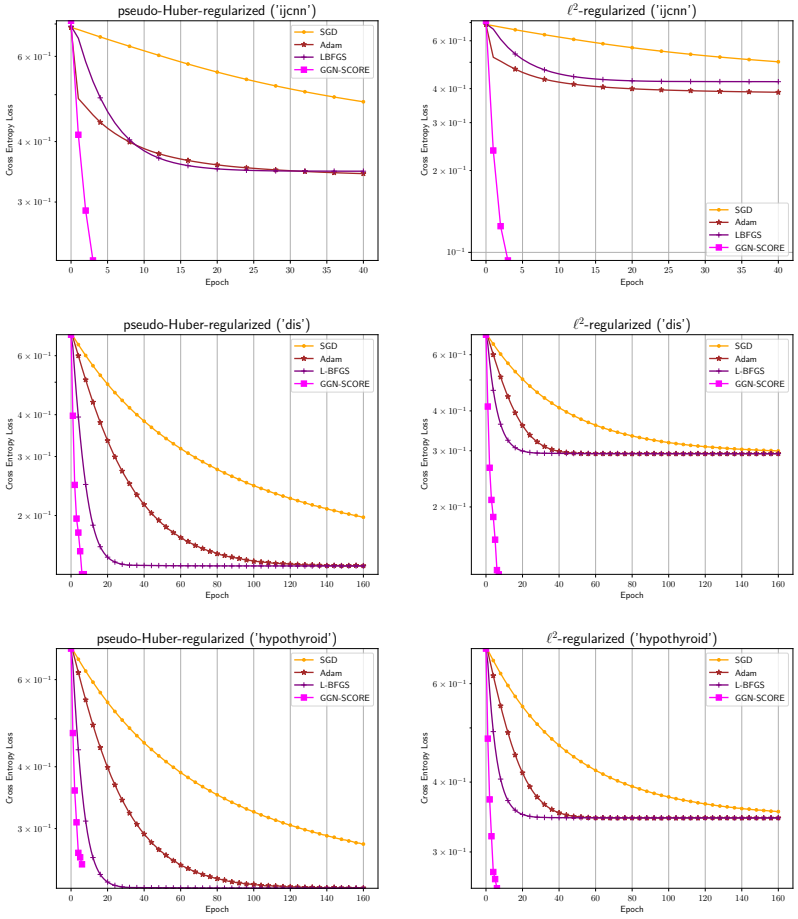


Figure 14: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the non-convex neural network training problem: overparameterized for `dis` and `hypothyroid`. $m = 512$ for `dis` and `hypothyroid`, and $m = 4096$ for `ijcnn1`.

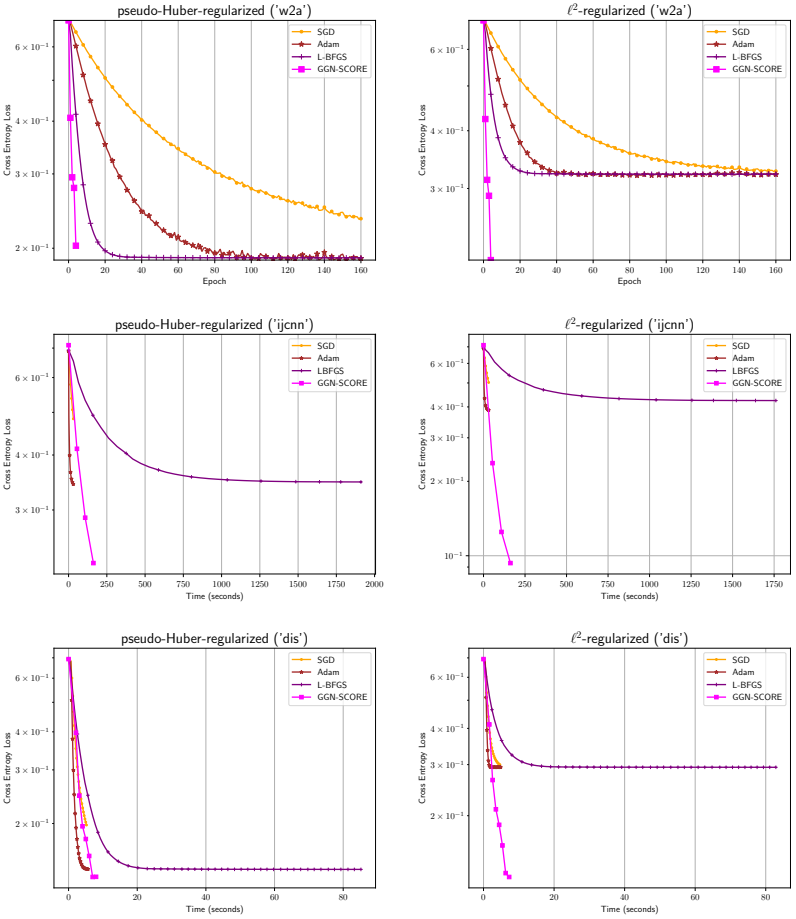


Figure 15: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the non-convex neural network training problem: overparameterized for `dis` and `w2a`. $m = 512$ for `w2a` and `dis`, and $m = 4096$ for `ijcnn1`.

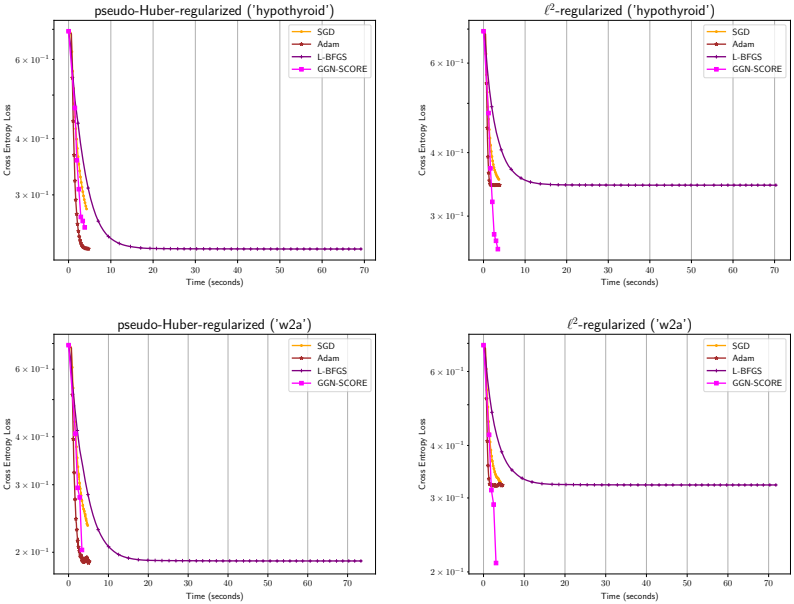


Figure 16: Convergence curves for GGN-SCORE, SGD, Adam and L-BFGS in the non-convex neural network training problem: overparameterized for `hypothyroid` and `w2a` with $m = 512$.

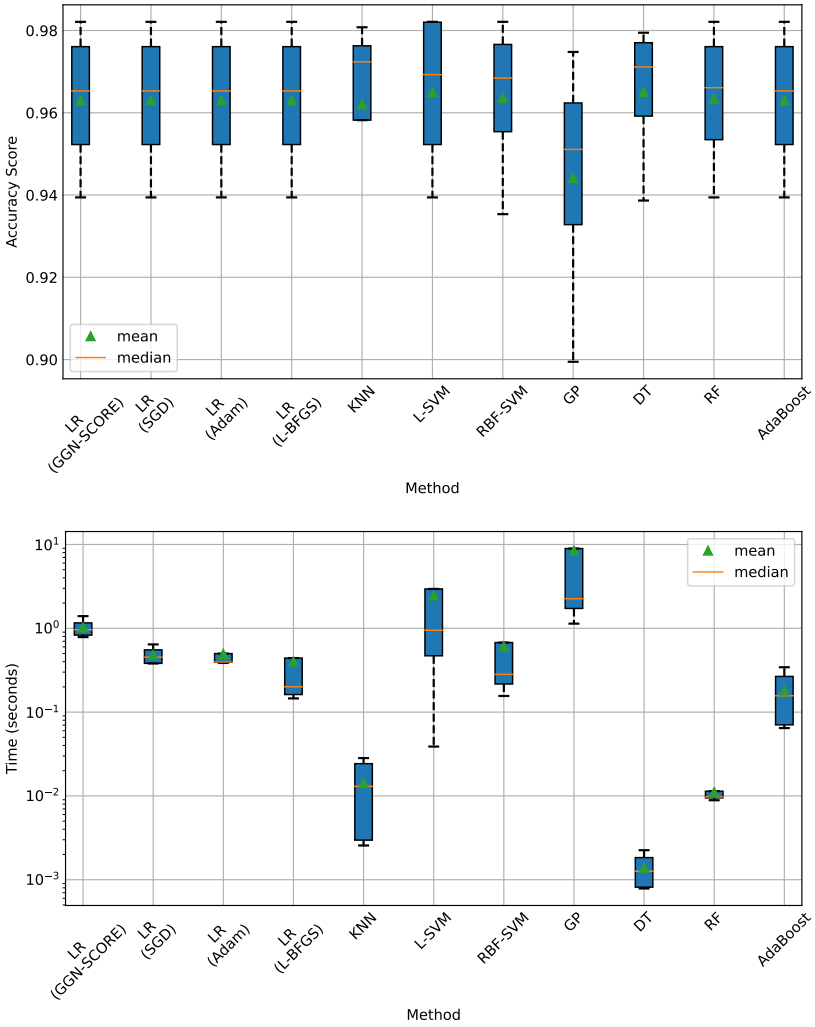


Figure 17: Classification task results using the proposed method – logistic regression (LR), k-nearest neighbours (KNN), linear support vector machine (L-SVM), RBF-SVM, Gaussian process (GP), decision tree (DT), random forest (RF) and adaptive boosting (AdaBoost) techniques for `dis`, `hypothyroid`, `coil2000` and `w2a` datasets. LR and SVM methods use the ℓ_2 penalty function with parameter $\lambda = 1.0$. Solutions with GGN-SCORE, SGD, Adam, and L-BFGS are obtained over one data pass with $m = 128$.

Appendix for Chapter 2

2.A Useful results

Following Assumptions 1 – 3, we get that [163, Theorem 2.1.6]

$$\gamma_l \boldsymbol{I}_{n_w} \preceq \partial^2 g(\boldsymbol{\theta}) \preceq \gamma_u \boldsymbol{I}_{n_w}, \quad \gamma_a \boldsymbol{I}_{n_w} \preceq \partial^2 h(\boldsymbol{\theta}) \preceq \gamma_b \boldsymbol{I}_{n_w}, \tag{2.23}$$

where $\boldsymbol{I}_{n_w} \in \mathbb{R}^{n_w \times n_w}$ is an identity matrix. Consequently,

$$(\gamma_l + \lambda\gamma_a)\boldsymbol{I}_{n_w} \preceq \partial^2 \mathcal{L}(\boldsymbol{\theta}) \preceq (\gamma_u + \lambda\gamma_b)\boldsymbol{I}_{n_w}. \tag{2.24}$$

In addition, the second derivative of $\mathcal{L}(\boldsymbol{\theta})$ is $(\gamma_g + \lambda\gamma_h)$-Lipschitz continuous $\forall \boldsymbol{x} \in \mathbb{R}^{n_p}, \boldsymbol{y} \in \mathbb{R}^d$, that is,

$$\left\| \partial^2 \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{\theta_1}; \boldsymbol{x})) - \partial^2 \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{\theta_2}; \boldsymbol{x})) \right\| \leq (\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|. \tag{2.25}$$

Lemma 3. *Let Assumptions 1, 2 and 3 hold. Let $\boldsymbol{\theta_1}, \boldsymbol{\theta_2}$ be any two points in $\mathbb{R}^{n_w}$. Then we have*

$$\left\| \boldsymbol{g}(\boldsymbol{\theta_1}) - \boldsymbol{g}(\boldsymbol{\theta_2}) - \left\langle \boldsymbol{H}(\boldsymbol{\theta_2}), \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\rangle \right\| \leq \frac{\gamma_g + \lambda\gamma_h}{2} \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|^2, \tag{2.26}$$

$$\left| \mathcal{L}(\boldsymbol{\theta_1}) - \mathcal{L}(\boldsymbol{\theta_2}) - \left\langle \boldsymbol{g}(\boldsymbol{\theta_2}), \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\rangle - \frac{1}{2} \left\langle \boldsymbol{H}(\boldsymbol{\theta_2})(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}), \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\rangle \right| \leq \frac{\gamma_g + \lambda\gamma_h}{6} \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|^3. \tag{2.27}$$

Proof. Fix $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathbb{R}^{n_w}$. Then

$$\boldsymbol{g}(\boldsymbol{\theta_1}) - \boldsymbol{g}(\boldsymbol{\theta_2}) = \int_0^1 \partial^2 \mathcal{L}(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}))(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) d\tau.$$

As $\tau \in [0,1]$, we have $\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) \in \mathbb{R}^{n_w}$. Hence writing $\partial^2 \mathcal{L} = \boldsymbol{H}$, we have

$$\begin{aligned}
&\boldsymbol{g}(\boldsymbol{\theta_1}) - \boldsymbol{g}(\boldsymbol{\theta_2}) = \int_0^1 \boldsymbol{H}(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}))(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) d\tau, \\
&\boldsymbol{g}(\boldsymbol{\theta_1}) - \boldsymbol{g}(\boldsymbol{\theta_2}) - \boldsymbol{H}(\boldsymbol{\theta_2})(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) = \\
&\int_0^1 \left(\boldsymbol{H}(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2})) - \boldsymbol{H}(\boldsymbol{\theta_2})\right)(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) d\tau
\end{aligned}$$

$$\begin{aligned}
&\left\|\boldsymbol{g}(\boldsymbol{\theta_1}) - \boldsymbol{g}(\boldsymbol{\theta_2}) - \boldsymbol{H}(\boldsymbol{\theta_2})(\boldsymbol{\theta_1} - \boldsymbol{\theta_2})\right\| = \\
&\qquad \left\| \int_0^1 \left(\boldsymbol{H}(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2})) - \boldsymbol{H}(\boldsymbol{\theta_2})\right)(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) d\tau \right\| \\
&\leq \int_0^1 \left\| \left(\boldsymbol{H}(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2})) - \boldsymbol{H}(\boldsymbol{\theta_2})\right)(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}) \right\| d\tau \\
&\leq \int_0^1 \left\| \boldsymbol{H}(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2})) - \boldsymbol{H}(\boldsymbol{\theta_2}) \right\| \cdot \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\| d\tau \\
&\leq \int_0^1 \tau(\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|^2 d\tau \\
&= \frac{\gamma_g + \lambda\gamma_h}{2} \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|^2 .
\end{aligned}$$

The proof of (2.27) follows immediately using a similar procedure (see e.g., [96]). ■

Corollary 1. *Let Assumptions 1, 2 and 3 hold. Let $\boldsymbol{\theta_1}, \boldsymbol{\theta_2}$ be any two points in $\mathbb{R}^{n_w}$. Then by writing $\partial^2 \mathcal{L} = \boldsymbol{H}$,*

$$\begin{aligned}
\boldsymbol{H}(\boldsymbol{\theta_2}) - (\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\| &\leq \boldsymbol{H}(\boldsymbol{\theta_1}) \\
&\leq \boldsymbol{H}(\boldsymbol{\theta_2}) + (\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\| . \quad (2.28)
\end{aligned}$$

Proof. The proof follows immediately by recalling for any $\boldsymbol{\theta} \in \mathbb{R}^{n_w}$, $\boldsymbol{H}(\boldsymbol{\theta})$ is positive definite, and hence the eigenvalues s_j of the difference $\boldsymbol{H}(\boldsymbol{\theta_1}) - \boldsymbol{H}(\boldsymbol{\theta_2})$ satisfy

$$\left| s_j \right| \leq (\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\|, \quad j = 1, 2, \ldots, n_w,$$

so that we have

$$-(\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\| \leq \boldsymbol{H}(\boldsymbol{\theta_1}) - \boldsymbol{H}(\boldsymbol{\theta_2}) \leq (\gamma_g + \lambda\gamma_h) \left\| \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\| .$$

■

Lemma 4 ([163, Theorem 2.1.5]). *Let the first derivative of a function $\Phi(\cdot)$ be L-Lipschitz on $\mathrm{dom}(\Phi)$. Then for any $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathrm{dom}(\Phi)$, we have*

$$0 \leq \Phi(\boldsymbol{\theta_1}) - \Phi(\boldsymbol{\theta_2}) - \left\langle \partial\Phi(\boldsymbol{\theta_2}), \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\rangle \leq \frac{L}{2}\|\boldsymbol{\theta_2} - \boldsymbol{\theta_1}\|^2.$$

Definition 3 ([163, Definition 2.1.3]). *A continuously differentiable function $\psi(\cdot)$ is γ_{sc}-strongly convex on $\mathrm{dom}(\psi)$ if for any $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathrm{dom}(\psi)$, we have*

$$\psi(\boldsymbol{\theta_1}) \geq \psi(\boldsymbol{\theta_2}) + \left\langle \partial\psi(\boldsymbol{\theta_2}), \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\rangle + \frac{\gamma_{sc}}{2}\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|^2, \quad \gamma_{sc} > 0.$$

We remark that if the first derivative of the function $\psi(\cdot)$ in the above definition is L-Lipschitz continuous, then by construction, for any $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathrm{dom}(\psi)$, ψ satisfies the Lipschitz constraint

$$|\psi(\boldsymbol{\theta_1}) - \psi(\boldsymbol{\theta_2})| \leq L\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|. \tag{2.29}$$

Lemma 5 ([163, Theorem 5.1.8]). *Let the function $\phi(\cdot)$ be M_ϕ-self-concordant. Then, for any $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathrm{dom}(\phi)$, we have*

$$\phi(\boldsymbol{\theta_1}) \geq \phi(\boldsymbol{\theta_2}) + \left\langle \partial\phi(\boldsymbol{\theta_2}), \boldsymbol{\theta_1} - \boldsymbol{\theta_2} \right\rangle + \frac{1}{M_\phi^2}\omega(M_\phi\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_{\boldsymbol{\theta_2}}),$$

where $\omega(\cdot)$ is an auxiliary univariate function defined by $\omega(t) \triangleq t - \ln(1+t)$.

Lemma 6 ([163, Corollary 5.1.5]). *Let the function $\phi(\cdot)$ be M_ϕ-self-concordant. Let $\boldsymbol{\theta_1}, \boldsymbol{\theta_2} \in \mathrm{dom}(\phi)$ and $r = \|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\|_{\boldsymbol{\theta_1}} < \frac{1}{M_\phi}$. Then*

$$\begin{aligned}\left(1 - M_\phi r + \frac{1}{3}M_\phi^2 r^2\right)\partial^2\phi(\boldsymbol{\theta_1}) &\preceq \int_0^1 \partial^2\phi(\boldsymbol{\theta_2} + \tau(\boldsymbol{\theta_1} - \boldsymbol{\theta_2}))d\tau \\ &\preceq \frac{1}{1 - M_\phi r}\partial^2\phi(\boldsymbol{\theta_1}).\end{aligned}$$

2.B Missing proofs

Proof of Lemma 2. By the arguments of Remark 1, we have that the matrix $(\boldsymbol{J}(\boldsymbol{\theta}_k)^T\boldsymbol{Q}\boldsymbol{J}(\boldsymbol{\theta}_k) + \lambda\boldsymbol{H}_h(\boldsymbol{\theta}_k))^{-1}$ is positive definite. Hence, with $\boldsymbol{g}(\boldsymbol{\theta}_k) \neq \mathbf{0}$, we have $\boldsymbol{g}\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\boldsymbol{g}(\boldsymbol{\theta}_k) = -\delta\boldsymbol{\theta}^T\boldsymbol{g}(\boldsymbol{\theta}_k) > 0$ and $\delta\boldsymbol{\theta}^T\boldsymbol{g}(\boldsymbol{\theta}_k) < \mathbf{0}$. ■

Proof of Theorem 1. The process formulated in (2.16) performs the update

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\boldsymbol{g}(\boldsymbol{\theta}_k).$$

As $\boldsymbol{g}(\boldsymbol{\theta}^*) = 0$ by mean value theorem and the first part of (SOSC), we have

$$\begin{aligned}
\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^* &= \boldsymbol{\theta}_k - \boldsymbol{\theta}^* - \boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\int_0^1 \partial^2\mathcal{L}(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)d\tau \\
&= \left[\boldsymbol{I} - \boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\int_0^1 \partial^2\mathcal{L}(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))d\tau\right](\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) \\
&= \boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\int_0^1 \left(\boldsymbol{H}(\boldsymbol{\theta}_k) - \partial^2\mathcal{L}(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))\right)d\tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*), \\
\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\| &= \left\|\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\int_0^1 \left(\boldsymbol{H}(\boldsymbol{\theta}_k) - \partial^2\mathcal{L}(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))\right)d\tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)\right\| \\
&\leq \left\|\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\right\|\int_0^1 \left\|\boldsymbol{H}(\boldsymbol{\theta}_k) - \partial^2\mathcal{L}(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))\right\|d\tau\,\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|.
\end{aligned}$$

Also, as $\tau \in [0,1]$ we have that $\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) \in \mathcal{B}_\epsilon(\boldsymbol{\theta}^*) \subseteq \mathbb{R}^{n_w}$. By taking the limit $\lim_{k\to\infty}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\|$, we have by the first part of (SOSC) and Assumption 4(ii)

$$\begin{aligned}
\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\| &\leq \left\|\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\right\|\int_0^1 \left\|\boldsymbol{H}(\boldsymbol{\theta}_k) - \boldsymbol{H}(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))\right\|d\tau\,\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \\
&\leq \left\|\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\right\|\int_0^1 (1-\tau)(\gamma_g + \lambda\gamma_h)\,\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|\,d\tau\,\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \\
&= \frac{\gamma_g + \lambda\gamma_h}{2}\left\|\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\right\|\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2.
\end{aligned}$$

By combining the claims in Corollary 1 and the bounds of $\boldsymbol{H}(\boldsymbol{\theta}_k)$ in (2.17), we obtain the relation

$$\gamma_l + a\gamma_a - (\gamma_g + \lambda\gamma_h)\,\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \preceq \boldsymbol{H}(\boldsymbol{\theta}^*) - (\gamma_g + \lambda\gamma_h)\,\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \preceq \boldsymbol{H}(\boldsymbol{\theta}_k).$$

Recall that $\boldsymbol{H}(\boldsymbol{\theta}_k)$ is positive definite, and hence invertible. We deduce that, indeed for all $\boldsymbol{\theta}_k$ satisfying $\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \leq \epsilon$, ϵ small enough, we have

$$\left\|\boldsymbol{H}(\boldsymbol{\theta}_k)^{-1}\right\| \leq \left(\gamma_l - \gamma_g - \lambda(\gamma_h - \gamma_a)\left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|\right)^{-1}.$$

Therefore,

$$\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\| \leq \xi_k \left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|^2,$$

where

$$\xi_k = \frac{1}{2}\frac{\gamma_g + \lambda\gamma_h}{\left(\gamma_l - \gamma_g - \lambda(\gamma_h - \gamma_a)\left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|\right)}.$$

■

Proof of Theorem 2. First, we upper bound the norm $\|\delta\boldsymbol{\theta}\|_{\boldsymbol{\theta}_k} \triangleq \|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_{\boldsymbol{\theta}_k} = \|\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)\|$. From Remark 3, we have

$$\|\boldsymbol{J}(\boldsymbol{\theta}_k)^T(\lambda\boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}_k))^{-1}\| \leq \frac{\gamma_a\|\boldsymbol{J}(\boldsymbol{\theta}_k)^T\|}{K + \lambda\gamma_a} \leq \frac{\tilde{\beta}\gamma_a}{K + \lambda\gamma_a}.$$

Hence,

$$\begin{aligned}
&\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_{\boldsymbol{\theta}_k} \\
&\quad\leq \frac{\alpha}{1 + M_h\eta_k}\left\|\boldsymbol{H}_h^{1/2}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}_k)(\lambda\boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}_k))^{-1}\boldsymbol{e}(\boldsymbol{\theta}_k)\right\| \\
&\quad\leq \frac{\alpha a_k}{1 + M_h\eta_k}\left\|\boldsymbol{H}_h(\boldsymbol{\theta}_k)^{1/2}\right\|\left\|\boldsymbol{H}_h(\boldsymbol{\theta}_k)^{-1}\right\|\left\|\boldsymbol{e}(\boldsymbol{\theta}_k)\right\| \\
&\quad\leq \frac{\alpha\beta\tilde{\beta}\gamma_a^{-1/2}}{(K + \lambda\gamma_a)(1 + M_h\eta_k)},
\end{aligned} \tag{2.30}$$

where $a_k = \left\|\boldsymbol{J}(\boldsymbol{\theta}_k)^T(\lambda\boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}_k))^{-1}\right\|$. Similarly, we have

$$\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\| \leq \frac{\alpha\beta\tilde{\beta}}{(K + \lambda\gamma_a)(1 + M_h\eta_k)}. \tag{2.31}$$

By Lemma 4, the function $\mathcal{L}(\boldsymbol{\theta})$ satisfies

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}_{k+1}) &\leq \mathcal{L}(\boldsymbol{\theta}_k) + \left\langle\partial\mathcal{L}(\boldsymbol{\theta}_k), \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\right\rangle + \frac{\gamma_u + \lambda\gamma_b}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 \\
&= \mathcal{L}(\boldsymbol{\theta}_k) + \left\langle\boldsymbol{g}_g(\boldsymbol{\theta}_k), \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\right\rangle \\
&\quad + \lambda\left\langle\boldsymbol{g}_h(\boldsymbol{\theta}_k), \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\right\rangle\frac{\gamma_u + \lambda\gamma_b}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2.
\end{aligned}$$

By convexity of g and h, and self-concordance of h, we have (using Definition 3, Lemma 5 and (2.29))

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_{k+1}) &\leq \mathcal{L}(\boldsymbol{\theta}_k) + (\gamma_u + \lambda\gamma_b)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\| + \frac{\gamma_u + \lambda\gamma_b - \gamma_l}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 \\ &\quad - \frac{\lambda}{M_h^2}\omega(M_h\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_{\boldsymbol{\theta}_k}) \\ &\overset{(2.30)}{\leq} \mathcal{L}(\boldsymbol{\theta}_k) + (\gamma_u + \lambda\gamma_b)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\| + \frac{\gamma_u + \lambda\gamma_b - \gamma_l}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 \\ &\quad - \frac{\lambda}{M_h^2}\omega\left(\frac{\alpha\beta\tilde{\beta}\gamma_a^{-1/2}M_h}{(K + \lambda\gamma_a)(1 + M_h\eta_k)}\right).\end{aligned}$$

Substituting the choice $\alpha = (\beta\tilde{\beta})^{-1}(\gamma_a)^{1/2}(K + \lambda\gamma_a)$, we have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}_{k+1}) &\leq \mathcal{L}(\boldsymbol{\theta}_k) + (\gamma_u + \lambda\gamma_b)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\| + \frac{\gamma_u + \lambda\gamma_b - \gamma_l}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 \\ &\quad - \frac{\lambda}{M_h^2}\omega\left(\frac{M_h}{1 + M_h\eta_k}\right).\end{aligned}$$

Taking expectation on both sides with respect to m conditioned on $\boldsymbol{\theta}_k$, we get

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{k+1})] &\leq \mathcal{L}(\boldsymbol{\theta}_k) + (\gamma_u + \lambda\gamma_b)\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\| \\ &\quad + \frac{\gamma_u + \lambda\gamma_b - \gamma_l}{2}\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|^2 - \mathbb{E}\left[\frac{\lambda}{M_h^2}\omega\left(\frac{M_h}{1 + M_h\eta_k}\right)\right].\end{aligned}$$

Note the second derivative ω'' of ω: $\omega''(t) = 1/(1+t)^2$. By the convexity of ω and using Jensen's inequality, also recalling unbiasedness of the derivatives,

$$\begin{aligned}\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}_{k+1})] &\leq \mathcal{L}(\boldsymbol{\theta}_k) + \frac{\gamma_u + \lambda\gamma_b}{\sqrt{\gamma_a}(1 + M_h\eta_k)} + \frac{\gamma_u + \lambda\gamma_b}{2\gamma_a(1 + M_h\eta_k)^2} \\ &\quad - \frac{\gamma_l}{2\gamma_a}\omega''(M_h\eta_k) - \frac{\lambda}{M_h^2}\omega\left(\frac{M_h}{1 + M_h\eta_k}\right) \\ &\leq \mathcal{L}(\boldsymbol{\theta}_k) - \left[\frac{\lambda}{M_h^2}\omega\left(\frac{M_h}{1 + M_h\eta_k}\right) + \frac{\gamma_l}{2\gamma_a}\omega''(M_h\eta_k) - \frac{2(\gamma_u + \lambda\gamma_b)}{\sqrt{\gamma_a}}\right].\end{aligned}$$

In the above, we used the bounds of the norm in (2.31), and again used the choice $\alpha = (\beta\tilde{\beta})^{-1}(\gamma_a)^{1/2}(K + \lambda\gamma_a)$.

To proceed, let us make a simple remark that is not explicitly stated in Remark 4: For any $\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1} \in \mathcal{N}_\epsilon(\boldsymbol{\theta}^*)$, we have

$$\begin{aligned}
\left\|\boldsymbol{H}_g(\boldsymbol{\theta}_{k+1}) - \boldsymbol{H}_g(\boldsymbol{\theta}_k)\right\| &\leq \gamma_g \left\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\right\| \to 0 \text{ as } k \to \infty, \\
\left\|\boldsymbol{H}_h(\boldsymbol{\theta}_{k+1}) - \boldsymbol{H}_h(\boldsymbol{\theta}_k)\right\| &\leq \gamma_h \left\|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}\right\| \to 0 \text{ as } k \to \infty.
\end{aligned}$$

Now, recall the proposed update step (2.21):

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\alpha}{1 + M_h \eta_k} \boldsymbol{H}_h^{-1} \boldsymbol{J}^T \left(\lambda \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\right)^{-1} \boldsymbol{e}.$$

Then the above remark allows us to perform the following operation: Subtract $\boldsymbol{\theta}^*$ from both sides and pre-multiply by $\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_{k+1}) \approx \boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_k) = \boldsymbol{H}_h^{1/2}$, we get the recursion

$$\begin{aligned}
\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*) &= \boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) \\
&\quad - \frac{\alpha}{1 + M_h \eta_k} \boldsymbol{H}_h^{1/2} \boldsymbol{H}_h^{-1} \boldsymbol{J}^T(\boldsymbol{\theta}_k) \left(\lambda \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}_k)\right)^{-1} \boldsymbol{e}(\boldsymbol{\theta}_k), \\
\left\|\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*)\right\| &\leq \left\|\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)\right\| \\
&\quad + \frac{\alpha}{1 + M_h \eta_k} \left\|\boldsymbol{H}_h^{1/2}\right\| \left\|\boldsymbol{H}_h^{-1} \boldsymbol{J}^T \left(\lambda \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\right)^{-1} \boldsymbol{e}(\boldsymbol{\theta}_k)\right\|.
\end{aligned}$$

Take expectation with respect to m on both sides conditioned on $\boldsymbol{\theta}_k$ and again consider unbiasedness of the derivatives. Further, recall the definition of the *local norm* $\|\cdot\|_{\boldsymbol{\theta}}$, and the bounds of $\boldsymbol{H}_h$, then

$$\begin{aligned}
\mathbb{E}\left\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_{k+1}} &\leq \left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k} \\
&\quad + \frac{\alpha}{\sqrt{\gamma_a}(1 + M_h \eta_k)} \left\|\left(\lambda \boldsymbol{I} + \boldsymbol{Q}\boldsymbol{J}\boldsymbol{H}_h^{-1}\boldsymbol{J}^T(\boldsymbol{\theta}_k)\right)^{-1}\right\| \left\|\boldsymbol{H}_h^{-1}\boldsymbol{J}^T\boldsymbol{e}(\boldsymbol{\theta}_k)\right\| \\
&\leq \left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k} + \frac{\alpha\gamma_a}{\sqrt{\gamma_a}(1 + M_h \eta_k)(K + \lambda\gamma_a)} \left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}(\boldsymbol{\theta}_k)\right\| \\
&= \left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k} + \frac{\gamma_a}{\beta\tilde{\beta}(1 + M_h \eta_k)} \left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_g(\boldsymbol{\theta}_k) + \lambda \boldsymbol{H}_h^{-1}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\| \\
&\leq \left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k} + \frac{\gamma_a}{\beta\tilde{\beta}} \left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_g(\boldsymbol{\theta}_k) + \lambda \boldsymbol{H}_h^{-1}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\| \\
&\leq \left\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k} + \frac{\gamma_a}{\beta\tilde{\beta}} \left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_g(\boldsymbol{\theta}_k)\right\| + \frac{\lambda\gamma_a}{\beta\tilde{\beta}} \left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\|.
\end{aligned}$$

By the mean value theorem and the first part of (SOSC),

$$\begin{aligned}
\boldsymbol{g}_g(\boldsymbol{\theta}_k) &= \int_0^1 \boldsymbol{H}_g(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*))(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)d\tau \\
&= \boldsymbol{H}_g(\boldsymbol{\theta}^*)(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) + \int_0^1 \left(\boldsymbol{H}_g(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)) - \boldsymbol{H}_g(\boldsymbol{\theta}^*)\right)(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)d\tau,
\end{aligned}$$

where $\boldsymbol{H}_g$ is the second derivative of g.

$$\begin{aligned}
&\left\|\boldsymbol{g}_g(\boldsymbol{\theta}_k)\right\| = \\
&\left\|\boldsymbol{H}_g(\boldsymbol{\theta}^*)(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*) + \int_0^1 \left(\boldsymbol{H}_g(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)) - \boldsymbol{H}_g(\boldsymbol{\theta}^*)\right)(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)d\tau\right\| \\
&\leq \left\|\boldsymbol{H}_g(\boldsymbol{\theta}^*)\right\| \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \\
&\qquad + \int_0^1 \left\|\boldsymbol{H}_g(\boldsymbol{\theta}^* + \tau(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*)) - \boldsymbol{H}_g(\boldsymbol{\theta}^*)\right\| \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| \, d\tau \\
&= \left\|\boldsymbol{H}_g(\boldsymbol{\theta}^*)\right\| \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| + \int_0^1 \tau\gamma_g \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2 \, d\tau \\
&= \left\|\boldsymbol{H}_g(\boldsymbol{\theta}^*)\right\| \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| + \frac{\gamma_g}{2} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2 \, d\tau \\
&\leq \gamma_u \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| + \frac{\gamma_g}{2} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2 .
\end{aligned}$$

In the above steps, we have used Assumption 4 and the remarks that follow it. Further,

$$\begin{aligned}
\left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_g(\boldsymbol{\theta}_k)\right\| &= \left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_g(\boldsymbol{\theta}_k)\right\| \\
&\leq \left\|\boldsymbol{H}_h^{-1}\right\| \left\|\boldsymbol{g}_g(\boldsymbol{\theta}_k)\right\| \\
&\leq \frac{1}{\gamma_a}\left(\gamma_u \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| + \frac{\gamma_g}{2} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2\right) \\
&= \frac{\gamma_u}{\gamma_a} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| + \frac{\gamma_g}{2\gamma_a} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2 .
\end{aligned}$$

Next, we analyze $\left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\|$. We have

$$\begin{aligned}\left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\| &= \left\|\boldsymbol{H}_h^{-3/2}\boldsymbol{H}_h^{1/2}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\| \\ &\leq \frac{1}{\gamma_a^{3/2}}\left\|\boldsymbol{H}_h^{1/2}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\| \\ &\overset{\text{(SOSC)}}{=} \frac{1}{\gamma_a^{3/2}}\left\|\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_k)\left(\boldsymbol{g}_h(\boldsymbol{\theta}_k)-\boldsymbol{g}_h(\boldsymbol{\theta}^*)\right)\right\|,\end{aligned}$$

and by the mean value theorem,

$$\begin{aligned}&\left\|\boldsymbol{H}_h^{-1}\boldsymbol{g}_h(\boldsymbol{\theta}_k)\right\| = \frac{1}{\gamma_a^{3/2}}\left\|\int_0^1 \boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_k)\boldsymbol{H}_h(\boldsymbol{\theta}^*+\tau(\boldsymbol{\theta}_k-\boldsymbol{\theta}^*))(\boldsymbol{\theta}_k-\boldsymbol{\theta}^*)d\tau\right\| \\ &\leq \frac{1}{\gamma_a^{3/2}}\left\|\boldsymbol{H}_h^{1/2}(\boldsymbol{\theta}_k)(\boldsymbol{\theta}_k-\boldsymbol{\theta}^*)\right\|\left\|\int_0^1 \boldsymbol{H}_h(\boldsymbol{\theta}^*+\tau(\boldsymbol{\theta}_k-\boldsymbol{\theta}^*))d\tau\right\| \\ &= \frac{1}{\gamma_a^{3/2}}\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k}\left\|\int_0^1 \boldsymbol{H}_h(\boldsymbol{\theta}^*+\tau(\boldsymbol{\theta}_k-\boldsymbol{\theta}^*))d\tau\right\| \\ &\leq \frac{1}{\gamma_a^{3/2}}\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k}\left\|\int_0^1 \boldsymbol{H}_h^{-1/2}(\boldsymbol{\theta}_k)\boldsymbol{H}_h(\boldsymbol{\theta}^*+\tau(\boldsymbol{\theta}_k-\boldsymbol{\theta}^*))\boldsymbol{H}_h^{-1/2}(\boldsymbol{\theta}_k)d\tau\right\| \\ &\overset{\text{Lemma 6}}{\leq} \frac{\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k}}{\gamma_a^{3/2}\left(1-M_h\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k}\right)}.\end{aligned}$$

In the above, we have used the fact $\boldsymbol{\theta}_0 \in \mathcal{N}_{M_h^{-1}}(\boldsymbol{\theta}^*) \implies \boldsymbol{\theta}_k \in \mathcal{N}_{M_h^{-1}}(\boldsymbol{\theta}^*)$ for all $\boldsymbol{\theta}_k$ generated by the process (2.21).

Combining the above results, we have

$$\begin{aligned}\mathbb{E}\left\|\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_{k+1}} \leq &\left[1+\frac{\lambda\gamma_a^{-1/2}}{\beta\tilde{\beta}\left(1-M_h\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k}\right)}\right]\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|_{\boldsymbol{\theta}_k} \\ &+\frac{\gamma_u}{\beta\tilde{\beta}}\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|+\frac{\gamma_g}{2}\left\|\boldsymbol{\theta}_k-\boldsymbol{\theta}^*\right\|^2.\end{aligned}$$

■

Chapter 3

Self-concordant smoothing in proximal quasi-Newton methods

3.1 Introduction

We consider the composite optimization problem

$$\min_{x \in \mathbb{R}^n} \mathcal{L}(x) \triangleq f(x) + g(x), \tag{3.1}$$

where f is a smooth, convex loss function and g is a closed, proper, convex (nonsmooth) regularization function. A common smoothing framework for solving (3.1) involves replacing the nonsmooth function g sequentially by its smooth approximation such that with an efficient algorithm for solving the resulting smooth optimization problem, we may approach the solution of the original problem. However, as noted in [27], the nonsmooth function g in (3.1) often plays a key role in describing some desirable properties specific to the application in which it appears, such as sparsifying the solution of the problem or enforcing some penalties or constraints on x, e.g., in sparse signal recovery and image processing [63, 26], compressed sensing [82, 59], model predictive control of constrained dynamical systems [35, 193, 220], neural network training [33], as well

as various classification and regression problems in machine learning. In order to retain such properties about the optimization vector x in these applications, [27] proposes to keep a part of g unchanged and hence considers a *partial smoothing* where g is only partially smoothed. The particular class of problems considered in [27] are those in which the nonsmooth function g takes on the form $g(x) = \mathcal{R}(x) + \Omega(x)$, and there, it is proposed to smooth a part of g, say, $\mathcal{R}$, leaving the other part, say, Ω unchanged. Nevertheless, in many of the applications where this class of problems appears, each of $\mathcal{R}$ and Ω is used to promote particular structures in the solution estimates, and hence smoothing one of them potentially destroys the overall desired structure. Prominent examples are found in lasso and multi-task regression problems with structured sparsity-inducing penalties. More specifically, $\mathcal{R}(x)$ is the scaled ℓ_1-norm penalty $\beta\|x\|_1$ that encourages sparse estimates of x for $\beta > 0$, and $\Omega(x)$ additionally enforces a more specific structure on these estimates, such as groups and fused structures.

One of the main motivations in [27] for the partial smoothing technique is the possibility to derive "fast" proximal gradient methods [166, 161, 230, 25] for the resulting problem. While the fast proximal methods prove to be more efficient than methods such as the subgradient and bundle-type methods, they are first-order methods which often fall back to weak solution estimates and accuracies [181]. It is evident, from their performance on unconstrained smooth optimization problems, that incorporating second-order information into a gradient scheme often yields superior performance and better solution quality. A line of work (see, e.g., [28, 133, 182, 228, 219]) has made efforts to incorporate (approximate) second-order information into proximal gradient schemes to emulate the performance of relative second-order methods for unconstrained smooth problems. The main drawback here is the computational overhead associated with second-order methods. This drawback is often largely mitigated by choosing a special structure for the matrix of the second-order terms of f. To deal with globalization issues, some of these approaches assume specific structures and regularity of the function f. For example, the authors in [228] assume a self-concordant structure of f allowing for

efficient step-size and correction techniques for proximal Newton-type and proximal quasi-Newton algorithms. However, because f oftentimes define a loss or data-misfit in real-world applications, the self-concordant assumption is not easy to check for many of these applications, and also restricts the applicability of the approach. Our self-concordant smoothing framework in this chapter provides a remedy to this limitation. We propose a new step-size selection technique that is suitable for Newton-type and quasi-Newton methods. This also exploits a self-concordant structure albeit not imposed on any of functions f and g that define the original problem.

In particular, we *regularize*[1] problem (3.1) by a second smooth function g_s and propose to keep all parts of g unchanged, but instead solve the following problem:

$$\min_{x\in\mathbb{R}^n} \mathcal{L}_s(x) \triangleq f(x) + g_s(x;\mu) + g(x), \tag{3.2}$$

where[2] g_s is a self-concordant, epi-smoothing function for g with a positive smoothing parameter μ (see Definition 6 below). By construction, g and g_s do not conflict, and therefore an algorithm exists that provides a solution to (3.2), which also (approximately) solves (3.1) (see Section 3.3). The smooth regularization g_s serves two main algorithmic purposes in this chapter. First, it provides an adaptive step-size selection method similar to the Newton decrement framework but without requiring knowledge of the self-concordance of f. Second, it provides a simple diagonal-structured variable-metric for efficiently scaling the proximal operator of g. As a result, the regularization enhances both the solvability of the smooth part of the original problem and the efficient handling of the nonsmooth part.

We do not give a special attention to the particular structure induced by g in the development of our technique. Yet in Section 3.4, we propose an approach to incorporate certain known structures into our framework,

[1] In this chapter, we use "regularization" and "smoothing" interchangeably but use "regularization" to emphasize explicit addition of a smooth function (a smooth approximation of the *nonsmooth part* of the problem) to the *smooth part* of the problem.

[2] We occasionally write $g_s(x)$ instead of $g_s(x;\mu)$ to refer to the same function.

thereby making it amenable to more general structured penalty functions. In particular, for the lasso and multi-task regression problems with structured sparsity-inducing penalties, we highlight the relation between Nesterov's smoothing [161] for a class of structured problems and the smoothing framework of this chapter, and then synthesize the so-called "prox-decomposition" property of g with the smoothness property of g_s for easily handling the structures promoted by g in the solution.

Most notably, the following points are vital to the development of our algorithmic framework in this chapter:

1. The first is to notice that for many practical problems, specifically those arising from modern machine learning systems, we often deal with overparameterized models (that is, in which number of data points is much less than the size of the optimization vector x). In this case, the pure proximal Newton method is not computationally ideal. This necessitates the use of generalized Gauss-Newton (GGN) approximations which, by our stylized "augmentation" technique, can be found to provide a practically efficient proximal algorithm for overparameterized models in which f can be expressed as a finite sum.

2. Secondly, we observe that the infimal convolution smoothing technique that we will introduce to construct g_s reveals a structure that is characterized by the self-concordant regularization (SCORE) framework of [4]. This provides a way to devise efficient adaptive step-size selection rule for proximal Newton-type algorithms without imposing a self-concordant structure on the original problem.

3. Lastly, via the notion of *epi-smoothing functions* established in [53] (a weaker notion than the *smoothable functions* of [27]), we can guarantee certain convergence notions on the epigraph of g allowing to combine the proposed smooth regularization technique with the Moreau-infimal-based (proximal) algorithms to handle the nonsmooth function g. As is customary, this assumes we can find an efficient method to compute a closed-form solution to the minimization of the sum of g and an auxiliary function ψ_α. However, unless the variable-metric associated with the proximal Newton-type method has a specific structure

that can be exploited for computational efficiency, the scaled proximal operator can be very difficult to compute and potentially poses a serious numerical issue. For this, the "simple" structure of the Hessian of g_s naturally provides a good candidate for the variable-metric, which allows for an efficient computation of the scaled proximal operator.

Burke and Hoheisel [53, 52] developed the notion of *epi-smoothing* for studying several epigraphical convergence (*epi-convergence*) properties for convex composite functions by combining the infimal convolution smoothing framework due to Beck and Teboulle [27] with the idea of *gradient consistency* due to Chen [71]. The key variational analysis tool used throughout their development is the *coercivity* of the class of regularization kernels studied in [27]. In particular, they establish the close connection between epi-convergence of the regularization functions and supercoercivity of the regularization kernel. Then, based on the above observations, we synthesize this idea with the notion of *self-concordant regularization* [4] to propose two proximal-type algorithms, viz., `Prox-N-SCORE` (Algorithm 2) and `Prox-GGN-SCORE` (Algorithm 3), for convex composite minimization.

The rest of this chapter is organized as follows: In Section 3.1.1, we present some notations and background on convex analysis. In Section 3.2, we establish our self-concordant smoothing notion with some properties and results. We describe our proximal Newton-type scheme in Section 3.3, and present the `Prox-N-SCORE` and `Prox-GGN-SCORE` algorithms. In Section 3.4, we describe an approach for handling specific structures promoted by the nonsmooth function g in problem (3.1), and propose a practical extension of the so-called *prox-decomposition* property of g for the self-concordant smoothing framework, which has certain in-built smoothness properties. Convergence properties of the `Prox-N-SCORE` and `Prox-GGN-SCORE` algorithms are studied in Section 3.5. In Section 3.6, we present some numerical simulation results for our proposed framework, and compare the results with other state-of-the-art approaches.

3.1.1 Notation and preliminaries

We denote by $\bar{\mathbb{R}} \triangleq \mathbb{R} \cup \{-\infty, +\infty\}$ the set of extended real numbers. The sets $\mathbb{R}_{\geq 0} \triangleq [0, +\infty[$ and $\mathbb{R}_{>0} \triangleq \mathbb{R}_{\geq 0} \backslash \{0\}$, respectively, denote the set of nonnegative and positive real numbers. Let $g \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ be an extended real-valued function. The *(effective) domain* of g is given by $\operatorname{dom} g \triangleq \{x \in \mathbb{R}^n \mid g(x) < +\infty\}$ and its *epigraph* (resp., *strict epigraph*) is given by $\operatorname{epi} g \triangleq \{(x, \gamma) \in \mathbb{R}^n \times \mathbb{R} \mid g(x) \leq \gamma\}$ (resp., $\operatorname{epi}_{\mathrm{s}} g \triangleq \{(x, \gamma) \in \mathbb{R}^n \times \mathbb{R} \mid g(x) < \gamma\}$). Given $\gamma \in \mathbb{R}_{>0}$, the γ-sublevel set of g is $\Gamma_\gamma(g) \triangleq \{x \in \mathbb{R}^n \colon g(x) \leq \gamma\}$. The standard inner product between two vectors $x, y \in \mathbb{R}^n$ is denoted by $\langle \cdot, \cdot \rangle$, that is, $\langle x, y \rangle \triangleq x^\top y$, where $x^\top$ is the transpose of x. The operation $x \odot y$ denotes the Hadamard product between two vectors $x, y \in \mathbb{R}^n$; we also denote by x^2 the product $x \odot x$.

For an $n \times n$ matrix H, we write $H \succ 0$ (resp., $H \succeq 0$) to say H is positive definite (resp., positive semidefinite). The sets $\mathcal{S}_+^n$ and $\mathcal{S}_{++}^n$, respectively, denote the set of $n \times n$ symmetric positive semidefinite and symmetric positive definite matrices. The set $\{\operatorname{diag}(v) \mid v \in \mathbb{R}^n\}$, where $\operatorname{diag} \colon \mathbb{R}^n \to \mathbb{R}^{n \times n}$, defines the set of all diagonal matrices in $\mathbb{R}^{n \times n}$. Matrix I_d denotes the $d \times d$ identity matrix. We denote by $\operatorname{card}(\mathcal{G})$, the cardinality of a set $\mathcal{G}$. For any two functions f and g, we define $(f \circ g)(\cdot) \triangleq f(g(\cdot))$. We denote by $\mathcal{C}^k(\mathbb{R}^n)$, the class of k-times continuously-differentiable functions on $\mathbb{R}^n$, $k \geq 0$. If the p-th derivatives of a function $f \in \mathcal{C}^k(\mathbb{R}^n)$ is L_f-Lipschitz continuous on $\mathbb{R}^n$ with $p \leq k$, $L_f \geq 0$, we write $f \in \mathcal{C}_{L_f}^{k,p}(\mathbb{R}^n)$. The notation $\|\cdot\|$ stands for the standard Euclidean (or 2-) norm $\|\cdot\|_2$. We define the weighted norm induced by $H \in \mathcal{S}_{++}^n$ by $\|x\|_H \triangleq \langle Hx, x \rangle^{\frac{1}{2}}$, for $x \in \mathbb{R}^n$. The associated *dual* norm is $\|x\|_H^* \triangleq \langle H^{-1}x, x \rangle^{\frac{1}{2}}$. An Euclidean ball of radius r centered at $\bar{x}$ is denoted by $\mathcal{B}_r(\bar{x}) \triangleq \{x \in \mathbb{R}^n \mid \|x - \bar{x}\| \leq r\}$. Associated with a given $H \in \mathcal{S}_{++}^n$, the (Dikin) ellipsoid of radius r centered at $\bar{x}$ is defined by $\mathcal{E}_r(\bar{x}) \triangleq \{x \in \mathbb{R}^n \mid \|x - \bar{x}\|_H \leq r\}$. We define the spectral norm $\|A\| \equiv \|A\|_2$ of a matrix $A \in \mathbb{R}^{m \times n}$ as the square root of the maximum eigenvalue of $A^\top A$, where $A^\top$ is the transpose of A.

A convex function $g \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is said to be *proper* if $\operatorname{dom} g \neq \emptyset$. The function g is said to be lower semicontinuous (lsc) at y if $g(y) \leq \liminf_{x \to y} g(x)$; if it is lsc at every $y \in \operatorname{dom} g$, then it is said to be lsc on

$\operatorname{dom} g$. We denote by $\Gamma_0(D)$ the set of proper convex lsc functions from $D \subseteq \mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$. Given $g \in \mathcal{C}^3(\operatorname{dom} g)$, we respectively denote by $g'(t)$, $g''(t)$ and $g'''(t)$ the first, second and third derivatives of g, at $t \in \mathbb{R}$, and by $\nabla_x g(x)$, $\nabla^2{}_x g(x)$, and $\nabla^3{}_x g(x)$ the gradient, Hessian and third-order derivative tensor of g, respectively, at $x \in \mathbb{R}^n$; if the variables with respect to which the derivatives are taken are clear from context, the subscripts are omitted. If $\nabla^2 g(x) \in \mathcal{S}^n_{++}$ for a given $x \in \mathbb{R}^n$, then the *local* norm $\|\cdot\|_x$ with respect to g at x is defined by $\|d\|_x \triangleq \langle \nabla^2 g(x) d, d \rangle^{1/2}$, the weighted norm of d induced by $\nabla^2 g(x)$. The associated dual norm is $\|v\|_x^* \triangleq \langle \nabla^2 g(x)^{-1} v, v \rangle^{1/2}$, for $v \in \mathbb{R}^n$. The subdifferential $\partial g \colon \mathbb{R}^n \to 2^{\mathbb{R}^n}$ of a proper function $g \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is defined by $x \mapsto \left\{ u \in \mathbb{R}^n \mid (\forall y \in \mathbb{R}^n)\ \langle y - x, u \rangle + g(x) \leq g(y) \right\}$, where $2^{\mathbb{R}^n}$ denotes the set of all subsets of $\mathbb{R}^n$. The function g is said to be subdifferentiable at $x \in \mathbb{R}^n$ if $\partial g(x) \neq \emptyset$; the subgradients of g at x are the members of $\partial g(x)$.

We define set convergence in the sense of Painlevé-Kuratowski. Let $\mathbb{N}$ denote the set of natural numbers. Let $\{C_k\}_{k \in \mathbb{N}}$ be a sequence of subsets of $\mathbb{R}^n$. The outer limit of $\{C_k\}_{k \in \mathbb{N}}$ is the set

$$\limsup_{k \to \infty} C_k \triangleq \left\{ x \in \mathbb{R}^n \mid \exists \{k_j\}_{j \in \mathbb{N}}, \exists \{x_j\}_{j \in \mathbb{N}}\, \forall j, x_k \in C_k, \{x_k\} \to x \right\},$$

and its inner limit is

$$\liminf_{k \to \infty} C_k \triangleq \left\{ x \in \mathbb{R}^n \mid \exists x_k \in C_k \colon \{x_k\} \to x, \forall k \in \mathbb{N} \right\}.$$

The limit C of $\{C_k\}_{k \in \mathbb{N}}$ exists if its outer and inner limits coincide, and we write

$$C = \lim_{k \to \infty} C_k \triangleq \limsup_{k \to \infty} C_k = \liminf_{k \to \infty} C_k.$$

We say that a function $g \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is coercive if $\liminf_{\|x\| \to \infty} g(x) = +\infty$, and supercoercive if $\liminf_{\|x\| \to \infty} \frac{g(x)}{\|x\|} = +\infty$. The sequence $\{g_k\}$ of functions $g_k \colon \mathbb{R}^n \to \bar{\mathbb{R}}$ is said to epi-converge to the function $g \colon \mathbb{R}^n \to \bar{\mathbb{R}}$ if $\lim_{k \to \infty} \operatorname{epi} g_k = \operatorname{epi} g$; it is said to continuously converge to g if for all $x \in \mathbb{R}^n$

and $\{x_k\} \to x$, we have $\lim_{k\to\infty} g_k(x_k) = g(x)$; and it converges pointwise to g if for all $x \in \mathbb{R}^n$, $\lim_{k\to\infty} g_k(x) = g(x)$. Epi-convergence, continuous convergence, and pointwise convergence of $\{g_k\}$ to g are respectively denoted by e$-\lim g_k = g$ (or $g_k \xrightarrow{e} g$), c$-\lim g_k = g$ (or $g_k \xrightarrow{c} g$), and p$-\lim g_k = g$ (or $g_k \xrightarrow{p} g$).

The conjugate (or Fenchel conjugate, or Legendre transform, or Legendre-Fenchel transform) $g^* \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ of a function $g \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ is the mapping $y \mapsto \sup_{x\in\mathbb{R}^n} \{\langle x, y\rangle - g(x)\}$, and its biconjugate is $g^{**} = (g^*)^*$.

3.2 Self-concordant regularization

This section introduces the concept of self-concordant smoothing, which provides structures that can be exploited in composite optimization problems. We begin by presenting the definition of generalized self-concordant functions, as given in [223].

Definition 4 (Generalized self-concordant function on $\mathbb{R}$)**.** *A univariate convex function $g \in \mathcal{C}^3(\operatorname{dom} g)$, with $\operatorname{dom} g$ open, is said to be (M_g, ν)-generalized self-concordant, with $M_g \in \mathbb{R}_{\geq 0}$ and $\nu \in \mathbb{R}_{>0}$, if*

$$\left|g'''(t)\right| \leq M_g\, g''(t)^{\frac{\nu}{2}}, \qquad \forall t \in \operatorname{dom} g.$$

Definition 5 (Generalized self-concordant function on $\mathbb{R}^n$ of order ν)**.** *A convex function $g \in \mathcal{C}^3(\operatorname{dom} g)$, with $\operatorname{dom} g$ open, is said to be (M_g, ν)-generalized self-concordant of order $\nu \in \mathbb{R}_{>0}$, with $M_g \in \mathbb{R}_{\geq 0}$, if $\forall x \in \operatorname{dom} g$*

$$\left|\left\langle \nabla^3 g(x)[v]u, u \right\rangle\right| \leq M_g \|u\|_x^2 \|v\|_x^{\nu-2} \|v\|^{3-\nu}, \qquad \forall u, v \in \mathbb{R}^n,$$

where $\nabla^3 g(x)[v] \triangleq \lim_{t\to 0} \left\{ \left(\nabla^2 g(x+tv) - \nabla^2 g(x)\right)/t \right\}$ is the third directional derivative of g.

Note that for an (M_g, ν)-generalized self-concordant function g defined on $\mathbb{R}^n$, the univariate function $\varphi \colon \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ defined by $\varphi(t) \triangleq g(x+tv)$ is (M_g, ν)-generalized self-concordant for every $x, v \in \operatorname{dom} g$ and

$x + tv \in \operatorname{dom} g$. This provides an alternative definition for the generalized self-concordant function on $\mathbb{R}^n$.

A key observation from the above definition is the possibility to extend the theory beyond the case $\nu = 3$ and $u = v$ originally presented in [164]. This observation, for instance, allowed the authors in [19] to introduce a *pseudo* self-concordant framework, in which $\nu = 2$, for the analysis of logistic regression. In a recent development, the authors in [172] identified a new class of pseudo self-concordant functions and showed how these functions may be slightly modified to make them *standard* self-concordant (i.e., where $M_g = 2, \nu = 3, u = v$), while preserving desirable structures. With such generalizations, and stemming from the idea of *Newton decrement* in [164], new analytic step-size selection and correction techniques for a number of proximal algorithms were developed in [228]. It is in the same spirit that we propose new step-size selection techniques from the self-concordant smoothing framework developed in this chapter. We denote by $\mathcal{F}_{M_g,\nu}$ the class of (M_g, ν)-generalized self-concordant functions, with generalized self-concordant parameters $M_g \in \mathbb{R}_{\geq 0}$ and $\nu \in \mathbb{R}_{>0}$.

Definition 6 (Self-concordant smoothing function)**.** *We say that the parameterized function $g_s \colon \mathbb{R}^n \times \mathbb{R}_{>0} \to \mathbb{R}$ is a self-concordant smoothing function for $g \in \Gamma_0(\mathbb{R}^n)$ if the following two conditions are satisfied:*

SC.1 $e\text{–}\lim_{\mu \downarrow 0} g_s(x;\mu) = g(x)$.

SC.2 $g_s(x;\mu) \in \mathcal{F}_{M_g,\nu}$.

We denote by $\mathcal{S}^{\mu}_{M_g,\nu}$ the set of self-concordant smoothing functions for a function $g \in \Gamma_0(\mathbb{R}^n)$, that is, $\mathcal{S}^{\mu}_{M_g,\nu} \triangleq \left\{ g_s \colon \mathbb{R}^n \times \mathbb{R}_{>0} \to \mathbb{R} \mid g_s \xrightarrow{e} g,\ g_s \in \mathcal{F}_{M_g,\nu} \right\}$.

3.2.1 Self-concordant regularization via infimal convolution

Next, we present key elements of smoothing through infimal convolution, which includes the Moreau-Yosida regularization process as a special case in defining the (scaled) proximal operator.

Definition 7 (Infimal convolution). *Let g and h be two functions from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$. The infimal convolution (or "inf-convolution" or "inf-conv")[3] of g and h is the function $g\square h \colon \mathbb{R}^n \to \bar{\mathbb{R}}$ defined by*

$$(g\square h)(x) = \inf_{w \in \mathbb{R}^n} \left\{ g(w) + h(x - w) \right\}. \tag{3.3}$$

The infimal convolution of g with h is said to be *exact at* $x \in \operatorname{dom} g$ if the infimum (3.3) is attained. It is *exact* if it is exact at each $x \in \operatorname{dom} g$, in which case we write $g \boxdot h$. Of utmost importance about the inf-conv operation in this chapter is its use in the approximation of a function $g \in \Gamma_0(\mathbb{R}^n)$; that is, the approximation of g by its infimal convolution with a member $h_\mu(\cdot)$ of a parameterized family $\mathcal{H} \triangleq \{h_\mu \mid \mu \in \mathbb{R}_{>0}\}$ of (regularization) kernels. In more formal terms, we recall the notion of inf-conv regularization in Definition 8 below. For $h \in \Gamma_0(\mathbb{R}^n)$ and $\mu \in \mathbb{R}_{>0}$, we define the function $h_\mu \colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ by the *epi-multiplication* operation[4]

$$h_\mu(\cdot) \triangleq \mu h\left(\frac{\cdot}{\mu}\right), \quad \mu \in \mathbb{R}_{>0}. \tag{3.4}$$

Definition 8 (Inf-conv regularization). *Let g be a function in $\Gamma_0(\mathbb{R}^n)$. Define*

$$\mathcal{H} \triangleq \left\{ (x, w) \mapsto h_\mu(x - w) \mid \mu \in \mathbb{R}_{>0} \right\}$$

a parameterized family of regularization kernels. The inf-conv regularization process of g with $h_\mu \in \mathcal{H}$ is given by $(g\square h_\mu)(x)$, for any $x \in \mathbb{R}^n$.

The operation of the inf-conv regularization generalizes the Moreau-Yosida regularization process in which case, $h_\mu(\cdot) = \|\cdot\|^2/(2\mu)$ or, with a scaled norm, $h_\mu(\cdot) = \|\cdot\|_Q^2/(2\mu)$ for some $Q \in \mathcal{S}^n_{++}$. The Moreau-Yosida regularization process provides the value function of the proximal operator associated with a function $g \in \Gamma_0(\mathbb{R}^n)$. This leads us to the definition of the scaled proximal operator.

Definition 9 (Scaled proximal operator). *The scaled proximal operator of a function $g \in \Gamma_0(\mathbb{R}^n)$, written $\operatorname{prox}^Q_{\alpha g}(\cdot)$, for $\alpha \in \mathbb{R}_{>0}$ and $Q \in \mathcal{S}^n_{++}$, is defined as the unique point in $\operatorname{dom} g$ that satisfies*

$$(g\square\psi_\alpha)(x) = g(\operatorname{prox}^Q_{\alpha g}(x)) + \psi_\alpha(x - \operatorname{prox}^Q_{\alpha g}(x)),$$

[3] Also sometimes called "epigraphic sum" or "epi-sum", as its operation yields the (strict) *epigraphic sum* $\operatorname{epi} g + \operatorname{epi} h$ [115, p. 93].

[4] It is easy to show that $h^*_\mu = \mu h^*$.

where $\psi_\alpha(\cdot) \triangleq \|\cdot\|_Q^2/(2\alpha)$. *That is,* $\operatorname{prox}_{\alpha g}^Q(x) \triangleq \operatorname{argmin}_{w\in\mathbb{R}^n}\{g(w)+\psi_\alpha(x-w)\}$.

A key property of the scaled proximal operator is its *nonexpansiveness*; that is, the property that (see, e.g., [195, 228])

$$\left\|\operatorname{prox}_{\alpha g}^Q(x)-\operatorname{prox}_{\alpha g}^Q(y)\right\|_Q \leq \|x-y\|_Q^*, \tag{3.5}$$

for all $x, y \in \mathbb{R}^n$.

In the sequel, we assume that the regularization kernel function h is of the form

$$h(x)=\sum_{i=1}^n \phi(x_i), \tag{3.6}$$

where ϕ is a univariate *potential function*. We are now left with the question of what properties we need to hold for ϕ such that $g\square h_\mu$ produces g_s satisfying the self-concordant smoothing conditions SC.1 – SC.2. To this end, we impose the following conditions on ϕ:

K.1 ϕ is supercoercive.

K.2 $\phi \in \mathcal{F}_{M_\phi,\nu}$.

Many functions that appear in different settings naturally exhibit the structures in conditions K.1 – K.2. For example, the ones belonging to the class of *Bregman/Legendre functions* introduced by Bauschke and Borwein [23] (see also [79] for a related characterization of the class of *Bregman functions*). In the context of proximal gradient algorithms for solving (3.1), the recent paper [22] enlists these functions as satisfying the new descent lemma (a.k.a *descent lemma without Lipschitz gradient continuity*) which the paper introduced. We summarize examples of these regularization kernel functions on different domains in Table **2**. We extract practical examples on $\mathbb{R}$ for the smoothing of the 1-norm and the indicator functions below.

Remark 6. *Suppose that* $\operatorname{dom} h$ *is a nonempty bounded subset of* $\mathbb{R}^n$*, for example, if* $\phi \in \Gamma_0(\operatorname{dom}\phi)$*, then since we have that* $g \in \Gamma_0(\operatorname{dom} g)$ *is bounded below as it possesses a continuous affine minorant (in view of [24, Theorem*

9.20]), the less restrictive condition that ϕ is coercive sufficiently replaces the condition K.1. In other words, the key convergence notion presented below holds similarly for the resulting function $g\square h_\mu$ in this case. Particularly, we get that $g\square h_\mu$ in this case is exact, finite-valued and locally Lipschitz continuous (see, e.g., [52, Proposition 3.6]) making it fit into our algorithmic framework.

Remark 7. *Following Remark 6, it is possible to relax the supercoercivity condition provided that g is bounded below. However, this condition requires that $\operatorname{dom} g$ is bounded, which restricts the practical application of our results in such cases. Nevertheless, whenever the supercoercivity condition is difficult to check (and the condition in Remark 6 does not hold), two possibilities exist for the epi-convergence of $g\square h_\mu$ to g according to [52, Proposition 3.9]: (1) If $h \in \Gamma_0(\mathbb{R}^n)$ is such that $g\square h_\mu \xrightarrow{e} g$, and $g \in \Gamma_0(\mathbb{R}^n)$ is supercoercive, then h is necessarily supercoercive. (2) If, however, $g \in \Gamma_0(\mathbb{R}^n)$ is not supercoercive, then we can find some $h \in \Gamma_0(\mathbb{R}^n)$ that is not supercoercive but for which $g\square h_\mu \xrightarrow{e} g$.*

In light of Remark 6 and Remark 7, our examples in Table **2** include both coercive and supercoercive functions. In either case, we have $\phi \in \mathcal{F}_{M_\phi,\nu}$. We keep the supercoercivity condition to emphasize other realizable properties of $g\square h_\mu$ highlighted below.

Table 2: Examples of regularization kernel functions for self-concordant smoothing, and their generalized self-concordant parameters M_ϕ and ν (see Definition 4).

$\phi(t)$	$\operatorname{dom}\phi$	M_ϕ	ν	Remark
$\frac{1}{p}\sqrt{1+p^2\lvert t\rvert^2}-1,\ p\in\mathbb{R}_{>0}$	$\mathbb{R}$	2	2.6	$p=1$
$\frac{1}{2}\left[\sqrt{1+4t^2}-1+\log\left(\frac{\sqrt{1+4t^2}-1}{2t^2}\right)\right]$	$\mathbb{R}$	$2\sqrt{2}$	3	Ostrovskii & Bach [172]
$\frac{1}{2}t^2$	$\mathbb{R}$	0	3	"Energy"
$\frac{1}{p}\lvert t\rvert^p,\ p\in(1,2)$	$\mathbb{R}_{\geq 0}$	4	6	$p=1.5$
$\log(1+\exp(t))$	$\mathbb{R}$	1	2	"Logistic"
$t\log t - t$	$[0,+\infty]$	1	4	"Boltzmann-Shannon"
$\begin{cases}\frac{1}{2}(t^2-4t+3), & \text{if } t\leq 1\\ -\log t, & \text{otherwise}\end{cases}$	$\mathbb{R}$	4	3	De Pierro & Iusem [79]

Examples. For some functions g and h_μ, there exists a closed form solution to $g\square h_\mu$. On the other hand, if one gets that $g\square h_\mu = g \boxdot h_\mu \in \Gamma_0(\mathbb{R}^n)$,

e.g., as a result of Proposition 2(i) below, then knowing in this case that

$$g\square h_\mu = (g^* + h_\mu^*)^*, \tag{3.7}$$

we can efficiently estimate $g\square h_\mu$ using *fast* numerical schemes (see, e.g., [145]). The structure of h implies g_s can be expressed in terms of a corresponding univariate function $\varphi\colon \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ by defining $\varphi_s(t;\mu) \triangleq (\varphi\square\phi_\mu)(t)$, and then

$$g_s(x;\mu) = \sum_{i=1}^{n} \varphi_s(x_i;\mu).$$

In the following, we provide examples of such φ_s for some $\phi \in \mathcal{F}_{M_\phi,\nu}$.

Infimal convolution of $\|\cdot\|_1$ with h_μ. In the first two examples, we consider $g(x) = \|x\|_1$.

Example 1. *Let $p = 1$ in $\phi(t) = \frac{1}{p}\sqrt{1 + p^2|t|^2} - 1$, with $\operatorname{dom}\phi = \mathbb{R}$. Then,*

$$\varphi_s(t;\mu) = \frac{\mu^2 - \mu\sqrt{\mu^2 + t^2} + t^2}{\sqrt{\mu^2 + t^2}}.$$

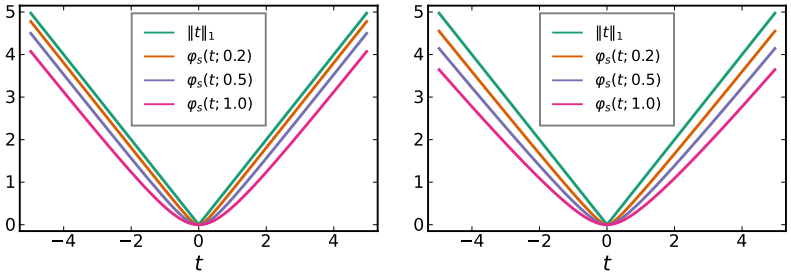


Figure 18: Generalized self-concordant smoothing of $\|\cdot\|_1$ with $\phi(t) = \sqrt{1+|t|^2} - 1$ (left) and $\phi(t) = \frac{1}{2}\left[\sqrt{1+4t^2} - 1 + \log\left(\frac{\sqrt{1+4t^2}-1}{2t^2}\right)\right]$ (right). The smooth approximation is shown for $\mu = 0.2, 0.5, 1.0$.

Example 2. $\phi(t) = \frac{1}{2}\left[\sqrt{1+4t^2} - 1 + \log\left(\frac{\sqrt{1+4t^2}-1}{2t^2}\right)\right]$, *with* $\operatorname{dom}\phi = \mathbb{R}$*:*

$$\varphi_s(t;\mu) = \frac{\sqrt{\mu^2+4t^2}}{2} - \frac{\mu}{2}\left[1 + \log(2) - \log\left(\frac{2t - \sqrt{\mu^2+4t^2}+\mu}{t}\right)\right.$$
$$\left. - \log\left(\frac{2t + \sqrt{\mu^2+4t^2}-\mu}{t}\right)\right].$$

Infimal convolution of $\delta_C(x)$ with h_μ. In the next example, we consider $g(x) = \delta_C(x)$, where $C \triangleq \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$ and

$$\delta_C(x) \triangleq \begin{cases} 0, & \text{if } x \in C, \\ +\infty, & \text{otherwise.} \end{cases}$$

Example 3. *Let* $g(x) = \delta_C(x)$*, and consider*

$$\phi(t) = \begin{cases} \frac{1}{2}(t^2 - 4t + 3), & \textit{if } t \leq 1 \\ -\log t, & \textit{otherwise}, \end{cases}$$

with $\operatorname{dom}\phi = \mathbb{R}$*. We have*

$$\varphi_s(t;\mu) = \begin{cases} \frac{1}{2\mu}(l - t + 3\mu)(l - t + \mu), & \textit{if } l \geq t - \mu \\ \mu\log(\mu) - \mu\log(t - l), & \textit{otherwise}. \end{cases}$$

The next two results characterize the functions h and h_μ defined by supercoercive and generalized self-concordant kernel functions.

Lemma 7. *Let* $\phi \in \Gamma_0(\mathbb{R})$ *be a function from* $\mathbb{R}$ *to* $\mathbb{R} \cup \{+\infty\}$*, and let the function* $h\colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ *be defined by* $h(x) \triangleq \sum_{i=1}^n \lambda_i \phi(x_i)$ *with* $x_i \in \operatorname{dom}\phi$*,* $\lambda_i > 0$*,* $i = 1, 2, \ldots, n$*. Then the following properties hold:*

(i) $h \in \Gamma_0(\mathbb{R}^n)$*.*

(ii) h *is supercoercive if and only if* ϕ *is supercoercive on its domain.*

(iii) If $\phi \in \mathcal{F}_{M_\phi,\nu}$*, where* $M_\phi \in \mathbb{R}_{\geq 0}$ *and* $\nu \geq 2$*, then* $h(x)$ *is well-defined on* $\operatorname{dom} h = \{\operatorname{dom}\phi\}^n$*, and* $h(x) \in \mathcal{F}_{M_h,\nu}$*, with* $M_h \triangleq \max\{\lambda_i^{1-\frac{\nu}{2}} M_\phi \mid 1 \leq i \leq n\} \in \mathbb{R}_{\geq 0}$*.*

Proof. (i) This statement is a direct consequence of [24, Corollary 9.4, Lemma 1.27 and Proposition 8.17].

(ii) Follows directly from the definition of supercoercivity.

(iii) $h(\cdot) \in \mathcal{F}_{M_h,\nu}$ with $M_h \triangleq \max\{\lambda_i^{1-\frac{\nu}{2}} M_\phi \mid 1 \leq i \leq n\} \in \mathbb{R}_{\geq 0}$ follows from [223, Proposition 1].

■

Proposition 1 (Self-concordance of h_μ)**.** *Suppose the conditions of Lemma 7 hold such that the function $h\colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ defined by (3.6) is (M_h, ν)-generalized self-concordant. Let $A \in \mathbb{R}^{n\times n}$ be a diagonal matrix defined by $A \triangleq \operatorname{diag}(\frac{1}{\mu})$ such that $h(\frac{x}{\mu}) \equiv h(Ax)$ is an affine transformation of $h(x)$. Then the following properties hold:*

(i) If $\nu \in (0, 3]$, then $h_\mu \in \mathcal{F}_{M,\nu}$ with $M = n^{\frac{3-\nu}{2}} \mu^{\frac{\nu}{2}-2} M_h$.

(ii) If $\nu > 3$, then $h_\mu \in \mathcal{F}_{M,\nu}$ with $M = \mu^{4-\frac{3\nu}{2}} M_h$.

Proof. (i) We have $\|A\| = \frac{\sqrt{n}}{\mu}$. By [223, Proposition 2(a)], $h(\frac{x}{\mu}) \in \mathcal{F}_{M,\nu}$ with $M = \|A\|^{3-\nu} M_h$. In view of Lemma 7(iii), the scaling $h(\frac{\cdot}{\mu}) \mapsto \mu h(\frac{\cdot}{\mu})$ gives $M \mapsto \mu^{1-\frac{\nu}{2}} M$. The result follows.

(ii) The value $\mu^2 > 0$ corresponds to the unique eigenvalue of $A^\top A$. By [223, Proposition 2(b)], $h(\frac{x}{\mu}) \in \mathcal{F}_{M,\nu}$ with $M = \mu^{3-\nu} M_h$. The result follows as in Item (i) above.

■

The next result concerns the epi-convergence of smoothing via infimal convolution under the condition of supercoercive regularization kernels in $\Gamma_0(\mathbb{R}^n)$.

Lemma 8. *[52, Theorem 3.8] Let $g, h \in \Gamma_0(\mathbb{R}^n)$ with h supercoercive and $0 \in \operatorname{dom} h$. Let h_μ be defined as in (3.4). Then the following hold:*

(i) $\operatorname{e-lim}_{\mu\downarrow 0}\{g^* + \mu h^*\} = g^*$.

(ii) $\operatorname{e-lim}_{\mu\downarrow 0}\{g \square h_\mu\} = g$.

(iii) If $h(0) \leq 0$, we have $\operatorname{p-lim}_{\mu\downarrow 0}\{g \square h_\mu\} = g$.

The main argument for the notion of epi-convergence in optimization problems is that when working with functions that may take infinite values, it is necessary to extend traditional convergence notions by applying the theory of *set convergence* to epigraphs in order to adequately capture local properties of the function (through a resulting calculus of smoothing functions), which on the other hand may be challenging due to the *curse of differentiation* associated with nonsmoothness. We refer the interested reader to [196, Chapter 7] for further details on the notion of epi-convergence, and to [221, 53, 52] for extended results on epi-convergent smoothing via infimal convolution.

In the following, we highlight key properties of the infimal convolution of $g \in \Gamma_0(\mathbb{R}^n)$ with h_μ satisfying $h \in \mathcal{F}_{M_h,\nu}$.

Proposition 2. *Let $g, h \in \Gamma_0(\mathbb{R}^n)$. Suppose further that h is (M_h, ν)-generalized self-concordant and supercoercive, and define $g_s \triangleq g \square h_\mu$ for all $\mu > 0$. Then the following hold:*

(i) $g \square h_\mu = g \boxdot h_\mu \in \Gamma_0(\mathbb{R}^n)$.

(ii) $g_s \in \mathcal{S}^\mu_{M_g,\nu}$ *with*

$$M_g = \begin{cases} n^{\frac{3-\nu}{2}} \mu^{\frac{\nu}{2}-2} M_h, & \text{if } \nu \in (0, 3], \\ \mu^{4-\frac{3\nu}{2}} M_h, & \text{if } \nu > 3. \end{cases}$$

(iii) g_s *is locally Lipschitz continuous.*

Proof. First, as an immediate consequence of [24, Lemma 1.28, Lemma 1.27 and Proposition 8.17], we have $h_\mu \in \Gamma_0(\mathbb{R}^n)$.

(i) Follows immediately from [24, Proposition 12.14].

(ii) By Item (i), $g_s = g \boxdot h_\mu \in \Gamma_0(\mathbb{R}^n)$. As a consequence of [24, Proposition 12.14], we have

$$g_s(x, \mu) = \min_{w \in \mathbb{R}^n} \left\{ g(w) + h_\mu(x - w) \right\},$$

and $g_s \xrightarrow{e} g$ (by [196, Theorem 11.34]). In view of [196, Proposition 7.2], for $x \in \operatorname{dom} g$ and

$$w_\mu(x) \in \operatorname*{argmin}_{w \in \mathbb{R}^n} \left\{ g(w) + h_\mu(x - w) \right\} \neq \emptyset,$$

$g_s \xrightarrow{e} g$ implies that $g_s(x,\mu) \to g(x)$ for at least one sequence $w_\mu(x) \to x$. Hence, we have

$$(g \square h_\mu)(x) = g(w_\mu(x)) + h_\mu(x - w_\mu(x)).$$

And, given $h \in \mathcal{F}_{M_h,\nu}$, we have by Proposition 1 that h_μ is (M_g, ν)-generalized self-concordant, where M_g is given by

$$M_g = \begin{cases} n^{\frac{3-\nu}{2}} \mu^{\frac{\nu}{2}-2} M_h, & \text{if } \nu \in (0,3], \\ \mu^{4-\frac{3\nu}{2}} M_h, & \text{if } \nu > 3. \end{cases}$$

Hence, $h_\mu \in \mathcal{C}^3(\operatorname{dom} g)$, and by [24, Proposition 18.7/Corollary 18.8], noting that higher-order derivatives are defined inductively in this sense [24, Definition 2.54, Remark 2.55], we deduce

$$\left| \left\langle \nabla^3 (g \square h_\mu)(x)[v]u, u \right\rangle \right| = \left| \left\langle \nabla^3 h_\mu(x - w_\mu(x))[v]u, u \right\rangle \right|,$$

$\forall u, v \in \operatorname{dom} g$, and similarly for the second-order derivatives. By definition, the univariate function

$$\varphi(t) \triangleq h_\mu(u_1 + t v_1), \tag{3.8}$$

is (M_g, ν)-generalized self-concordant, for every $u_1, v_1 \in \operatorname{dom} g$. That is, $\forall t \in \mathbb{R}$,

$$\left|\varphi'''(t)\right| \leq M_g \, \varphi''(t)^{\frac{\nu}{2}},$$

which concludes the proof with $u_1 \equiv x$, $v_1 \equiv w(\frac{x}{\mu})$ and $t \equiv -\mu$ in (3.8).

(iii) Following the arguments in Items (i) and (ii) above, w_μ (and hence g_s) is finite-valued (see also [53, Lemma 4.2]). Then the Lipschitz continuity of g_s near some $\bar{x} \in \operatorname{dom} g$ follows from the convexity of g_s (see [196, Example 9.14]; see also [52, Proposition 3.6]).

■

3.3 A proximal Newton-type scheme

Our notion of self-concordant smoothing developed in the previous section is motivated by algorithmic purposes. Notably, we have established the epi-convergence of $g_s \in \mathcal{F}_{M_g,\nu}$ to $g \in \Gamma_0(\mathbb{R}^n)$ under suitable conditions, which plays a critical role in the optimization problem

(3.2) in a global sense. We next characterize the optimal solution set of (3.2) using the notion of ε-optimality with respect to (3.1). We define ε-$\operatorname{argmin} g \triangleq \{x \mid g(x) \leq \inf g + \varepsilon\}$ to be the set of points that minimize the function g up to a tolerance $\varepsilon \in \mathbb{R}_{\geq 0}$. For our approach, it suffices to state the following about the set of minimizers of g_s.

Proposition 3. *Fix any $\mu \in \mathbb{R}_{>0}$. Suppose $g \in \Gamma_0(\mathbb{R}^n)$ and $g_s \in \mathcal{S}^{\mu}_{M_g,\nu}$. Then a minimizer of g_s is ε^μ-optimal for g with $\varepsilon^\mu \in \mathbb{R}_{\geq 0}$.*

Proof. From Proposition 2(iii), we have that, for any $\bar{x} \in \operatorname{dom} g$, $g_s \xrightarrow{e} g$ implies there is at least one sequence $w_\mu(\bar{x}) \to \bar{x}$. By the (super)coercivity of g_s, the level set $\{x \in \mathbb{R}^n \mid g_s(x;\mu) \leq \hat{\alpha}\}$ at $\hat{\alpha} \in \mathbb{R}$ is bounded and contained in a compact set C such that $w_\mu(\bar{x}) \in C$. Let $w_\mu(\bar{x}) \in \varepsilon^\mu$-$\operatorname{argmin} g_s \subseteq C$ (with $\mu \in \mathbb{R}_{>0}$ fixed). Then, since $g_s \xrightarrow{e} g$, we get from [196, Theorem 7.31(b)] that $g(\bar{x}) \leq \inf g + \varepsilon^\mu$. Hence, $\bar{x} \in \varepsilon^\mu$-$\operatorname{argmin} g$. Finally, $w_\mu(\bar{x}) \in \varepsilon^\mu$-$\operatorname{argmin} g$ necessarily follows from [196, Theorem 7.33]. ■

Proposition 3, along with the observation in [196, Theorem 7.37], suggests that a proximal (Newton-type) algorithm can provide a highly accurate solution to (3.2), which also solves (3.1). Hence, the proximal method effectively handles the nonsmooth part of the problem, while our regularization approach enhances both the solvability of the smooth part of the original problem and improves the handling of the nonsmooth part through the choice of the variable-metric. For the optimization problem (3.2), we assume the following:

P.1 f is convex and $f \in \mathcal{C}^{2,2}_{L_f}(\mathbb{R}^n)$.

P.2 $\rho_1 I_n \leq \nabla^2 f(x^*) \leq L_1 I_n$, $\rho_2 I_n \leq \nabla^2 g_s(x^*) \leq L_2 I_n$ at a locally optimal solution x^* of (3.2) with $L_1 \geq \rho_1 > 0$ and $L_2 \geq \rho_2 > 0$.

P.3 $g \in \Gamma_0(\mathbb{R}^n)$.

P.4 $g_s \in \mathcal{S}^{\mu}_{M_g,\nu}$.

In particular, we consider $g_s(x;\mu) \triangleq g \square h_\mu$, where h is a suitable regularization kernel for self-concordant smoothing of g in the sense of Section 3.2. We are interested in practically efficient composite minimization algorithms that utilize the idea of the *Newton decrement* framework but

without imposing the self-concordant structure on the objective functions of the problem. In this section, we present proximal Newton-type algorithms that exploit the structure of self-concordant smoothing functions developed in § 3.2 for variable-metric selection and the computation of their step-lengths.

Proximal Newton-type algorithms for solving (3.2) consist in minimizing a sequence of *upper approximation* of $\mathcal{L}_s$ obtained by summing the nonsmooth part $g(x_k)$ and a local quadratic model of the smooth part $q(x_k) \triangleq f(x_k) + g_s(x_k)$ near x_k. That is, for $x \in \operatorname{dom} \mathcal{L} \equiv \operatorname{dom} f \cap \operatorname{dom} g$, we iteratively define

$$\hat{q}_k(x) \triangleq q(x_k) + \langle \nabla q(x_k), x - x_k \rangle + \frac{1}{2}\|x - x_k\|_Q^2, \tag{3.9a}$$

$$\hat{m}_k(x) \triangleq \hat{q}_k(x) + g(x), \tag{3.9b}$$

where $Q \in \mathcal{S}_{++}^n$, and then solve the subproblem

$$\delta_k \in \operatorname*{argmin}_{d \in \mathbb{R}^n} \hat{m}_k(x_k + d), \tag{3.10}$$

for a proximal Newton-type search direction δ_k. Proximal Newton-type algorithms encompass several specific cases. Our characterization of the optimality conditions for (3.2), particularly the flexibility in the choice of the variable metric Q, is well-motivated by the class of *cost approximation (CA) methods* [180]. This leads to a novel approach for selecting $\{x_k\}$ from the sequence of iterates $\{\delta_k\}$. The necessary optimality conditions for (3.2) are defined by

$$0 \in \nabla q(x^*) + \partial g(x^*), \tag{3.11}$$

for $x^* \in \operatorname{dom} \mathcal{L}$. To find points x^* satisfying (3.11), CA methods, as the name implies, iteratively approximate $\nabla q(x_k)$ by a *cost approximating mapping* $\Phi \colon \mathbb{R}^n \to \mathbb{R}^n$, taking into account the fixed approximation error term $\Phi(x_k) - \nabla q(x_k)$. That is, a point d is sought satisfying

$$0 \in \Phi(d) + \partial g(d) + \nabla q(x_k) - \Phi(x_k). \tag{3.12}$$

Let Φ be the gradient mapping of a continuously differentiable convex function $\psi\colon \mathbb{R}^n \to \mathbb{R}$. A CA method iteratively solves the subproblem

$$\min_{d\in\mathbb{R}^n} \left\{ \psi(d) + q(x_k) + g(d) - \psi(x_k) + \left\langle \nabla\, q(x_k) - \nabla\, \psi(x_k), d - x_k \right\rangle \right\}. \tag{3.13}$$

A step is then taken in the direction $\delta_k - x_k$, namely

$$x_{k+1} = x_k + \alpha_k(\delta_k - x_k), \tag{3.14}$$

where δ_k solves (3.13) and $\alpha_k > 0$ is a step-length typically computed via a line search such that an appropriately selected *merit function* is sufficiently decreased along the direction $\delta_k - x_k$.

Remark 8. *Evaluating the merit function too many times can be impractical. One way to mitigate this issue for large-scale problems is to incorporate "predetermined step-lengths" into the solution scheme of* (3.13)*. This allows us to update x_k as $x_{k+1} \equiv \delta_k$. However, methods that use this approach do not generally yield a monotonically decreasing sequence of objective values. Instead, convergence is characterized by a metric that measures the distance from iteration points to the set of optimal solutions [179].*

In view of Remark 8, we discuss next a new proximal Newton-type scheme that compromises between minimizing the objective values and decreasing the distance from iteration points to the set of optimal solutions as specified by a curvature-exploiting variable-metric.

3.3.1 Variable-metric and adaptive step-length selection

A very nice feature of the CA framework is that it can help, for instance, through the specific choice of Φ, to efficiently utilize the original problem's structure—a practice which is particularly useful when solving medium- to large-scale problems. This feature fits directly into our self-concordant smoothing framework. We notice that (3.13) gives (3.10) with the following choice of ψ:

$$\psi(\cdot) = \frac{1}{2}\|\cdot\|_Q^2\,, \qquad Q \in \mathcal{S}_{++}^n. \tag{3.15}$$

In this case, the optimality conditions and our assumptions give

$$(Q - \nabla q)(x_k) \in (Q + \partial g)(d), \tag{3.16}$$

which leads to

$$\delta_k = \operatorname{prox}_g^Q(x_k - Q^{-1} \nabla q(x_k)). \tag{3.17}$$

In the proximal Newton-type scheme, Q may be the Hessian of $q(x_k)$ or its approximation[5]. Although a diagonal structure of Q is often desired due to its ease of implementation in the proximal framework, we most likely throw away relevant curvature information by performing a diagonal or scalar approximation of $\nabla^2 q(x_k)$, especially when q is not assumed to be separable. Our consideration in this chapter entails the following characterization of the optimality conditions:

$$(H_k - \nabla q)(x_k) \in (Q_k + \partial g)(d), \tag{3.18}$$

where H_k may be the Hessian, $\nabla^2 q(x_k) \equiv H_k^f + H_k^g$, of q or its approximation, where $H_k^f \equiv \nabla^2 f(x_k)$, $H_k^g \equiv \nabla^2 g_s(x_k; \mu)$, and $Q_k \in \mathcal{S}_{++}^n$. Specifically, we set $Q_k = H_k^g$ in (3.18) and propose the following step update formula:

$$x_{k+1} = \operatorname{prox}_{\alpha_k g}^{H_k^g}(x_k - \bar{\alpha}_k H_k^{-1} \nabla q(x_k)), \tag{3.19}$$

where $\bar{\alpha}_k \in \mathbb{R}_{>0}$ results from *damping* the Newton-type steps.

The validity of this procedure in the present scheme may be seen in the interpretation of the proximal operator $\operatorname{prox}_g(x^+)$ for some $x^+ \in \operatorname{dom} g$ as compromising between minimizing the function g and staying close to x^+ (see [175, Chapter 1]). When scaled by, say, H_k^g, "closeness" is quantified in terms of the metric induced by H_k^g, and we want the proximal steps to stay close (as much as possible) to the Newton iterates relative to, say, $\|\cdot\|_{H_k^g}$. To see this, we note that in view of the fixed-point characterization (3.13) via CA methods, we may interpret proximal Newton-type algorithms as

[5] If Q is the scaled identity matrix, then we have the *proximal gradient method*, if $Q = \nabla^2 q$, we have the *proximal Newton method*, and if Q is a quasi-Newton, say BFGS, approximation of the Hessian, we have a *proximal quasi-Newton method*.

Algorithm 2 `Prox-N-SCORE` (A proximal Newton algorithm)

Require: $x_0 \in \mathbb{R}^n$, problem functions f, g, self-concordant smoothing function $g_s \in \mathcal{S}^{\mu}_{M_g,\nu}$, $\alpha \in (0, 1]$

1: **for** $k = 0, \ldots$ **do**
2: $\quad \operatorname{grad}_k \leftarrow \nabla f(x_k) + \nabla g_s(x_k)$
3: $\quad H_k^g \leftarrow \nabla^2 g_s(x_k)$; $\eta_k \leftarrow \left\|\nabla g_s(x_k)\right\|^*_{H_k^g}$ ▷ Note: H_k^g is diagonal
4: $\quad \bar{\alpha}_k = \frac{\alpha}{1+M_g\eta_k}$
5: $\quad H_k \leftarrow \nabla^2 f(x_k) + H_k^g$; Solve for Δ_k: $H_k\Delta_k = \operatorname{grad}_k$
6: $\quad x_{k+1} \leftarrow \operatorname{prox}^{H_k^g}_{\alpha g}(x_k - \bar{\alpha}_k\Delta_k)$
7: **end for**

a fixation of the error term $\nabla\psi - \nabla q$ at some point in $\operatorname{dom} q \cap \operatorname{dom} g$. Let us fix some $\bar{x} \in \operatorname{dom} q \cap \operatorname{dom} g$ and introduce the operator $E_{\bar{x}}$ defined by

$$E_{\bar{x}}(z) \triangleq \nabla^2 q(\bar{x})z - \bar{\alpha}\nabla q(z), \tag{3.20}$$

where $0 < \bar{\alpha} \leq \alpha \leq 1$. Set $Q = Q_k \in \mathcal{S}^n_{++}$ arbitrary in (3.15). We aim to exploit the structure in g_s (and $\nabla^2 g_s$), so we define an operator $\xi_{\bar{x}}(Q_k, \cdot)$ to quantify the error between $\nabla^2 g_s$ and Q_k as follows:

$$\xi_{\bar{x}}(Q_k, z) \triangleq (\nabla^2 g_s(\bar{x}) - Q_k)(z - x_k). \tag{3.21}$$

We provide a local characterization of the optimality conditions for (3.13) in terms of $E_{\bar{x}}$ and $\xi_{\bar{x}}$ in the next result.

Proposition 4. *Let the operators $E_{\bar{x}}$ and $\xi_{\bar{x}}(Q_k, \cdot)$ be defined by (3.20) and (3.21), respectively. Then the optimality conditions for (3.13) with $\psi(\cdot) = \frac{1}{2}\|\cdot\|^2_{Q_k}$ are locally characterized in terms of $E_{\bar{x}}$ and $\xi_{\bar{x}}(Q_k, \cdot)$ by*

$$E_{\bar{x}}(x_k) + \xi_{\bar{x}}(Q_k, d) \in \nabla^2 g_s(\bar{x})d + \alpha\partial g(d). \tag{3.22}$$

More precisely, (3.18) holds with $Q_k = \nabla^2 g_s(\bar{x})$ whenever $\bar{x}$ is the unique optimizer satisfying (3.22) at a local solution d of (3.13).

Proof. As g_s satisfies the property in SC.1, it holds that [53, Lemma 3.4]

$$\limsup_{\substack{x\to\bar{x}\\ \mu\downarrow 0}} \nabla g_s(x;\mu) = \partial g(\bar{x}). \tag{3.23}$$

Hence, by Lemma 8 and [196, Theorem 13.2], there exists $v_g \in \mathbb{R}^n$, in the *extended sense* of differentiability (see [196, Definition 13.1]), such that

$$\limsup_{\substack{x \to \bar{x} \\ \mu \downarrow 0}} \nabla g_s(x) = \partial g(\bar{x}) = \{v_g\}, \tag{3.24a}$$

$$\emptyset \neq \partial g(d) \subset v_g + \nabla^2 g_s(\bar{x})(d - \bar{x}) + o(\|d - \bar{x}\|)\mathcal{E}_r(\bar{x}). \tag{3.24b}$$

Let x_k be in some neighbourhood of $\bar{x}$ and let $\{x_k\} \to \bar{x}$ be generated by an iterative process. By assumption, the differentiable terms in (3.24b) are convex and the differential operators are monotone. It then holds that

$$\partial g(d) \subset v_g + \nabla^2 g_s(\bar{x})(d - x_k) + o(\|d - \bar{x}\|)\mathcal{E}_r(\bar{x}), \tag{3.25}$$

for all x_k in the neighbourhood of $\bar{x}$. Since differentiability in the extended sense is necessary and sufficient for differentiability in the *classical sense* (see [196, Definition 13.1 and Theorem 13.2]), it holds for some $\mu \in \mathbb{R}_{>0}$ that $v_g \equiv \nabla g_s(\bar{x})$ which is defined through:

$$\nabla g_s(d) = \nabla g_s(\bar{x}) + \nabla^2 g_s(\bar{x})(d - \bar{x}) + o(\|d - \bar{x}\|). \tag{3.26}$$

Consequently, using (3.12) (with $\Phi = \nabla \psi$), and defining the Dikin ellipsoid $\mathcal{E}_r(\bar{x})$ in terms of g_s for r small enough, we deduce from (3.25), (3.26) that $Q_k(x_k - d) + \nabla^2 g_s(\bar{x})(x_k - d) - \bar{\alpha}\nabla q(x_k) \in \bar{\alpha}\nabla g_s(\bar{x})$ for $0 < \bar{\alpha} \leq 1$. We assert $\nabla^2 f(\bar{x})(d - \bar{x}) \in \mathcal{E}_r(\bar{x})$ at a local solution d of (3.13), and then deduce again from (3.25), (3.26) that $\bar{\alpha}\nabla g_s(\bar{x}) + \nabla^2 g_s(\bar{x})(d - x_k) + \nabla^2 f(\bar{x})x_k \in \alpha\partial g(d)$ holds for $0 < \bar{\alpha} \leq \alpha \leq 1$ near $\bar{x}$, whenever $\bar{x}$ is the unique solution x^* of (3.2). As a result, using $q \triangleq f + g_s$, we get

$$(\nabla^2 q(\bar{x}) - \bar{\alpha}\nabla q)x_k - \nabla^2 g_s(\bar{x})x_k \in Q_k(d - x_k) + \alpha\partial g(d). \tag{3.27}$$

In terms of $E_{\bar{x}}$ and $\xi_{\bar{x}}(Q_k, \cdot)$, (3.27) may be written as (3.22), which exactly gives (3.18) with the choice $Q_k = \nabla^2 g_s(\bar{x})$. ■

As we shall see in the GGN approximation discussed below, we may exploit the properties of the function g_s in ensuring stability of the Newton-type steps via the notion of *Newton decrement*. In essence, we consider damping the Newton-type steps such that

$$\bar{\alpha}_k = \frac{\alpha_k}{1 + M_g \eta_k}, \tag{3.28}$$

where by P.4, M_g is a generalized self-concordant parameter for g_s, and $\eta_k \triangleq \left\|\nabla g_s(x_k)\right\|_{x_k}^*$ is the dual norm of $\nabla g_s(x_k)$ with respect to $g_s(x_k)$. Note that the above choice for $\bar{\alpha}_k$, in the context of minimizing generalized self-concordant functions, assumes $\nu \geq 2$ (see e.g. [223, Equation 12]). Suppose for example $\alpha_k = 1$ is fixed and $\nu = 3$, then (3.27) leads to the standard damped-step proximal Newton-type method for minimizing the sum of a function $g \in \mathcal{F}_{M_g,\nu}$ and a nonsmooth function in $\Gamma_0(\mathbb{R}^n)$ (cf. [228, 223]) in the framework of Newton decrement.

In view of (3.6), H_k^g has a desirable diagonal structure and hence can be cheaply updated from iteration to iteration. This structure provides an efficient way to compute the scaled proximal operator $\operatorname{prox}_g^{H_k^g}$, for example via a special case of the proximal calculus derived in [29] (see Section 3.6 for two practical examples). Overall, by exploiting the structure of the problem, precisely

(i) taking adaptive steps that properly capture the curvature of the objective functions, and

(ii) scaling the proximal operator of g by a variable-metric H_k^g which has a simple, diagonal structure,

we can adapt to an affine-invariant structure due to the algorithm and ensure we remain close to the Newton-type iterates towards convergence.

If we choose $H_k \equiv \nabla^2 q(x_k)$ in (3.19), we obtain a proximal Newton step (see Algorithm 2):

$$x_{k+1} = \operatorname{prox}_{\alpha_k g}^{H_k^g}(x_k - \bar{\alpha}_k \nabla^2 q(x_k)^{-1} \nabla q(x_k)). \tag{3.29}$$

However, H_k may be any approximation of the Hessian of q at x_k. In view of (3.22), this corresponds to replacing the Hessian term $\nabla^2 q(\bar{x})$ in (3.20) by the approximating matrix evaluated at $\bar{x}$.

3.3.2 A proximal generalized Gauss-Newton algorithm

In describing the proximal GGN algorithm, consider first the simple case $g \equiv 0$. Then (3.19) with $\bar{\alpha}_k = 1$ gives exactly the pure Newton-type

Algorithm 3 `Prox-GGN-SCORE` (A proximal generalized Gauss-Newton algorithm)

Require: $x_0 \in \mathbb{R}^n$, problem functions f, g, self-concordant smoothing function $g_s \in \mathcal{S}^{\mu}_{M_g,\nu}$, model $\mathcal{M}$, input-output pairs $\{u_i, y_i\}_{i=1}^m$ with $y_i \in \mathbb{R}^{n_y}$, $\alpha \in (0, 1]$

for $k = 0, \ldots$ **do**
  $H_k^g \leftarrow \nabla^2 g_s(x_k)$; $\eta_k \leftarrow \left\|\nabla g_s(x_k)\right\|_{H_k^g}^*$ ▷ Note: H_k^g is diagonal
  $\bar{\alpha}_k \leftarrow \frac{\alpha}{1+M_g\eta_k}$
  if $m + n_y \leq n$ **then**
    Compute δ_k^{ggn} via (4.9)
  else
    Compute δ_k^{ggn} via (3.33)
  end if
  $x_{k+1} \leftarrow \operatorname{prox}_{\alpha g}^{H_k^g}(x_k + \bar{\alpha}_k \delta_k^{\text{ggn}})$
end for

direction

$$\delta_k^{\text{ggn}} = -H_k^{-1} \nabla q(x_k). \tag{3.30}$$

Now suppose that the function f quantifies a data-misfit or loss between the outputs[6] $\hat{y}_i$ of a model $\mathcal{M}(\cdot; x)$ and the expected outputs y_i, for $i = 1, 2, \ldots, m$, as in a typical machine learning problem, and that $g \neq 0$. Precisely, let $\hat{y}_i \equiv \mathcal{M}(u_i; x)$, and suppose that f can be written as

$$f(x) = \sum_{i=1}^{m} \ell(y_i, \hat{y}_i), \tag{3.31}$$

where $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function. Define an "augmented" Jacobian matrix $J_k \in \mathbb{R}^{(m+1)\times n}$ by [4]

$$J_k^T \triangleq \begin{bmatrix} \hat{y}_1'(x^{(1)}) & \hat{y}_2'(x^{(1)}) & \cdots & \hat{y}_m'(x^{(1)}) & g^{(1)\prime}(x^{(1)}) \\ \hat{y}_1'(x^{(2)}) & \hat{y}_2'(x^{(2)}) & \cdots & \hat{y}_m'(x^{(2)}) & g^{(2)\prime}(x^{(2)}) \\ \vdots & \vdots & & \vdots & \vdots \\ \hat{y}_1'(x^{(n)}) & \hat{y}_2'(x^{(n)}) & \cdots & \hat{y}_m'(x^{(n)}) & g^{(n)\prime}(x^{(n)}) \end{bmatrix}, \tag{3.32}$$

[6] Note that for the sake of simplicity, we assume here $y_i \in \mathbb{R}$, but it is straightforward to extend the approach that follows to cases where $y_i \in \mathbb{R}^{n_y}$, $n_y > 1$.

where $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ are components of x_k, and $g^{(1)}, g^{(2)}, \ldots, g^{(n)}$ are the components of $g_s(x_k; \mu)$. Then GGN approximation of the Newton direction (3.30) gives

$$\delta_k^{\text{ggn}} = -(H_k^f + H_k^g)^{-1} \nabla q \approx -(J_k^\top V_k J_k + H_k^g)^{-1} J_k^\top u_k, \tag{3.33}$$

where $V_k \equiv \text{diag}(v_k)$, $v_k \triangleq [l''_{\hat{y}_1}(y_1, \hat{y}_1; x_k), \ldots, l''_{\hat{y}_m}(y_m, \hat{y}_m; x_k), 0]^\top \in \mathbb{R}^{(m+1)}$, and the vector $u_k \triangleq [l'_{\hat{y}_1}(y_1, \hat{y}_1; x_k), \ldots, l'_{\hat{y}_m}(y_m, \hat{y}_m; x_k), 1]^\top \in \mathbb{R}^{m+1}$ defines an augmented "residual" term. If $m + 1 < n$ (possibly $m \ll n$), that is, when the model is overparameterized, the following equivalent formulation of (3.33) provides a convenient way to compute the GGN search direction [4]:

$$\delta_k^{\text{ggn}} = -H_k^{g^{-1}} J_k^\top (I_m + V_k J_k H_k^{g^{-1}} J_k^\top)^{-1} u_k. \tag{3.34}$$

Note that in case the function g (and hence g_s) is scaled by some (non-negative) constant, only the identity matrix I_m may be scaled accordingly. Following [4, Section 4], it suffices to assume stability of the GGN iterates by ensuring the stability of H_k^g. This is achieved, for instance, through the generalized self-concordant structure of g_s.

Now if we choose $H_k \equiv J_k^\top V_k J_k + H_k^g$ in the proximal Newton-type scheme of (3.19), we have the proximal GGN update (see Algorithm 3):

$$x_{k+1} = \text{prox}_{\alpha_k g}^{H_k^g}(x_k + \bar{\alpha}_k \delta_k^{\text{ggn}}), \tag{3.35}$$

where δ_k is computed via (3.33), or by (4.9) in case $m + 1$ is less than n, and $\bar{\alpha}_k$ is as defined in (3.28).

3.4 Structured penalties

As we have noted, more general nonsmooth regularized problems impose certain structures on the variables that must be handled explicitly by the algorithm. Such situations can be seen in some lasso and multi-task regression problems in which problem (3.1) takes on the form

$$\min_{x \in \mathbb{R}^n} f(x) + \underbrace{\mathcal{R}(x) + \Omega(Cx)}_{g(x)}, \tag{3.36}$$

where, in addition to $\mathcal{R}(x)$, the function (cf. [69, 70])

$$\Omega(Cx) \triangleq \max_{u \in \mathcal{Q}} \langle u, Cx \rangle, \tag{3.37}$$

characterizes a specific desired structure of the solution estimates and, for $\mathbb{V}$ a finite-dimensional vector space such that $C \colon \mathbb{R}^n \to \mathbb{V}$ is a linear map, $\mathcal{Q} \subseteq \mathbb{V}^*$ is closed and convex, where $\mathbb{V}^*$ is the dual space to $\mathbb{V}$.

For example, in the sparse group lasso problem [98, 217], $\Omega(Cx) = \gamma \sum_{j \in \mathcal{G}} \omega_j \left\| x_j \right\|$ induces group level sparsity on the solution estimates and $\mathcal{R}(x) = \beta \|x\|_1$ promotes the overall sparsity of the solution, so that the optimization problem is written as

$$\min_{x \in \mathbb{R}^n} f(x) + \beta \|x\|_1 + \beta_{\mathcal{G}} \sum_{j \in \mathcal{G}} \omega_j \left\| x_j \right\|, \tag{3.38}$$

where $\beta \in \mathbb{R}_{>0}$, $\beta_{\mathcal{G}} \in \mathbb{R}_{>0}$, $\mathcal{G} = \{j_k, \ldots, j_{n_g}\}$ is the set of variables groups with $n_g = \operatorname{card}(\mathcal{G})$, $x_j \in \mathbb{R}^{n_j}$ is the subvector of x corresponding to variables in group j and $\omega_j \in \mathbb{R}_{>0}$ is the group penalty parameter. Another example is the graph-guided fused lasso for multi-task regression problems [127], where the function $\Omega(Cx) = \beta_{\mathcal{G}} \sum_{e=(r,s) \in E, r<s} \tau(\omega_{rs}) \left| x_r - \operatorname{sign}(\omega_{rs}) x_s \right|$ encourages a fusion effect over variables x_r and x_s shared across tasks through a graph $G \equiv (V, E)$ of relatedness, where $V = \{1, \ldots, n\}$ denotes the set of nodes and E the edges; $\beta_{\mathcal{G}} \in \mathbb{R}_{>0}$, $\tau(\omega_{rs})$ is a fusion penalty function, and $\omega_{rs} \in \mathbb{R}$ is the weight of the edge $e = (r, s) \in E$. Here, with $\mathcal{R}(x) = \beta \|x\|_1$, $\beta \in \mathbb{R}_{>0}$, the optimization problem is written as

$$\min_{x \in \mathbb{R}^n} f(x) + \beta \|x\|_1 + \beta_{\mathcal{G}} \sum_{e=(r,s) \in E, r<s} \tau(\omega_{rs}) \left| x_r - \operatorname{sign}(\omega_{rs}) x_s \right|. \tag{3.39}$$

3.4.1 Structure reformulation for self-concordant smoothing

The key observation in problems of the form (3.36) is that the function $\Omega(Cx)$ belongs to the class of nonsmooth convex functions that is well-structured for Nesterov's smoothing [161] in which a smooth approxima-

tion Ω_s of Ω has the form[7]

$$\Omega_s(Cx;\mu) = \max_{u\in\mathcal{Q}} \left\{ \langle u, Cx\rangle - \mu d(u) \right\}, \quad \mu \in \mathbb{R}_{>0}, \tag{3.40}$$

where d is a *prox-function*[8] of the set $\mathcal{Q}$. Note that Nesterov's smoothing approach assumes the knowledge of the exact structure of C. In the sequel, we shall write $\Omega^C(x) \equiv \Omega(Cx)$ or $\Omega_s^C(x;\mu) \equiv \Omega_s(Cx)$, with the superscript "$C$" to indicate the function is *structure-aware* via C.

Proposition 5. *Let $C\colon \mathbb{R}^n \to \mathbb{R}^n$ be a linear map and let ω be a continuous convex function defined on a closed and convex set $Q \subseteq \operatorname{dom}\omega \subseteq \mathbb{R}^n$. Further, define*

$$\tilde{\Omega}(x) \triangleq \max_{u\in\mathcal{Q}} \left\{ \langle u, Cx\rangle - \omega(u) \right\},$$

and let $d \triangleq h^*$, where $h\colon \mathbb{R}^n \to \mathbb{R}$ satisfies $\nabla^2 h \in \mathcal{S}^n_{++}$ and is of the form (3.6) with ϕ satisfying K.1 – K.2 so that $h \in \mathcal{F}_{M_h,\nu}$ with $\nu \in [3,6)$ if $n > 1$ and with $\nu \in (0,6)$ if $n = 1$. Then the function

$$\Omega_s(x;\mu) = \max_{u\in\mathcal{Q}} \left\{ \langle u, Cx\rangle - \omega(u) - \mu d(u) \right\}, \quad \mu \in \mathbb{R}_{>0}, \tag{3.41}$$

is a self-concordant smoothing function for $\tilde{\Omega}(x)$.

Proof. We follow the approach in [27, Section 4]. First note that we can write $\tilde{\Omega}(x) = \Omega(Cx)$, where

$$\Omega \triangleq (\omega + \delta_Q)^*.$$

Now, let $\tilde{d} \triangleq d + \delta_Q$. In view of [223, Proposition 6], we have $d, \tilde{d} \in \mathcal{F}_{M_d,\nu_d}$ where $M_d = M_h$ and $\nu_d = 6 - \nu$. Next, define $\tilde{h} \triangleq (\tilde{d})^*$. We have

$$\begin{aligned}(\Omega^* + \tilde{h}^*_\mu)^*(x) &= (\omega + \delta_Q + \mu\tilde{d})^*(x) \\ &= \max_{u\in\mathcal{Q}} \left\{ \langle u, x\rangle - \omega(u) - \mu d(u) \right\},\end{aligned}$$

which is precisely $(\tilde{\Omega}\square h^*_\mu)(x)$ according to [27, Theorem 4.1(a)] (cf. (3.7)). Now, since $d \triangleq h^* \in \mathcal{F}_{M_d,\nu_d}$, the result follows from Proposition 1 and Proposition 2(ii). ∎

[7] The reader should not confuse the barrier smoothing technique of, say, [162, 229], with the self-concordant smoothing framework of this chapter. The self-concordant barrier smoothing techniques, just like Nesterov's smoothing, realize first-order and subgradient algorithms that solve problems of this exact form.

[8] A function d_1 is called a *prox-function* of a closed and convex set $\mathcal{Q}_1$ if $\mathcal{Q}_1 \subseteq \operatorname{dom} d_1$, and d_1 is continuous and strongly convex on $\mathcal{Q}_1$ with convexity parameter $\rho_1 > 0$.

Under the assumptions of Proposition 5, $\Omega_s^C(x;\mu)$ provides a self-concordant smooth approximation of $\Omega(x)$ with $\mathbb{V} \equiv \mathbb{R}^n$. In this case, $\omega = 0$ in Proposition 5 and the prox-function d in (3.40) is given by h^*, the dual of $h \in \mathcal{F}_{M_h,\nu}$.

3.4.2 Prox-decomposition and smoothness properties

An important property of the function $g = \mathcal{R} + \Omega^C$ we want to infer here is its prox-decomposition property [246] in which the (unscaled) proximal operator of g satisfies

$$\operatorname{prox}_g = \operatorname{prox}_{\Omega^C} \circ \operatorname{prox}_{\mathcal{R}} . \tag{3.42}$$

Under our assumptions on g and h, this property extends for the inf-conv regularization (and hence the self-concordant smoothing framework)[9]. To see this, let $\mathbb{V} \equiv \mathbb{R}^n$, and note the following equivalent expression for the definition of inf-convolution (3.3):

$$(\mathcal{R}\square h_\mu)(x) = \inf_{\substack{(u,v)\in\mathbb{R}^n\times\mathbb{R}^n \\ u+v=x}} \left\{\mathcal{R}(u) + h_\mu(v)\right\}.$$

Define also the function $r_s : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that

$$(\mathcal{R}\square h_\mu)(x) \equiv \sum_{i=1}^{n} r_s(x_i;\mu).$$

The next result follows, highlighting what we propose as the *inf-decomposition* property.

Proposition 6. *Let $g \in \Gamma_0(\mathbb{R}^n)$ be given as the sum $g(x) = \mathcal{R}(x) + \Omega^C(x)$. Suppose that the function $h \in \Gamma_0(\mathbb{R}^n)$ is supercoercive and define $z \triangleq [r_s(x_1;\mu),\ldots,r_s(x_n;\mu)]^\top$. Then the regularization process $(g\square h_\mu)(x)$, for all $\mu > 0$, is given by the composition*

$$(g\square h_\mu)(x) = (\Omega^C\square h_\mu)(z). \tag{3.43}$$

[9] Additional assumptions may be required to hold in order to *correctly* define this property in our framework, e.g., nonoverlapping groups in case of the sparse group lasso problem, in which case, $\mathbb{V}$ is the space $\mathbb{R}^n$.

Proof. The exactness of the inf-conv regularization process by Proposition 2(i) allows to infer

$$\begin{aligned}(\Omega^C \square h_\mu)(z) &= \inf_{\substack{(u,v)\in\mathbb{R}^n\times\mathbb{R}^n \\ u+v=z}} \Big\{\Omega^C(u) + h_\mu(v)\Big\} \\ &= \inf_{\substack{(u,v)\in\mathbb{R}^n\times\mathbb{R}^n \\ 2u+v=x}} \Big\{\mathcal{R}(u) + \Omega^C(u) + h_\mu(v)\Big\} \\ &= ((\mathcal{R}+\Omega^C)\square h_\mu)(x) = (g\square h_\mu)(x).\end{aligned}$$

■

Given the smoothness properties of $\Omega^C \square h_\mu$ and $\mathcal{R}\square h_\mu$, we can apply the chain rule to obtain the derivatives of their composition $g\square h_\mu$. Precisely, [222, Lemma 2.1] provides sufficient conditions for the validity of the derivatives obtained via the chain rule for composite functions, which are indeed satisfied for $g\square h_\mu$ by our assumptions.

3.5 Convergence analysis

We analyze the convergence of Algorithms 2 and 3 under the proposed smoothing framework. The local behaviour of the algorithms are discussed in Appendix 3.A. In view of the numerical examples considered in Section 3.6, we restrict our analyses to the case $2 \leq \nu \leq 3$. However, similar convergence properties are expected to hold for the general case $\nu > 0$, as the key bounds describing generalized self-concordant functions hold similarly for all of these cases (see, e.g., Section 2 and the concluding remark of [223]). We define the following metric term, taking the local norm $\|\cdot\|_x$ with respect to g_s:

$$d_\nu(x,y) \triangleq \begin{cases} M_g\|y-x\| & \text{if } \nu = 2, \\ \left(\frac{\nu}{2}-1\right) M_g \|y-x\|_2^{3-\nu}\|y-x\|_x^{\nu-2} & \text{if } \nu > 2. \end{cases} \tag{3.44}$$

We introduce the notations $H_k^{g^*} \equiv \nabla^2 g_s(x^*)$, $H_k^{f^*} \equiv \nabla^2 f(x^*)$ and $H^* \equiv \nabla^2 q(x^*)$. Recall also the notations $H_k^g \equiv \nabla^2 g_s(x_k)$, $H_k^f \equiv \nabla^2 f(x_k)$ and $H_k \equiv \nabla^2 q(x_k)$ at x_k. Furthermore, we define the following matrices

associated with any given twice differentiable function f:

$$\Sigma_f^{x,y} \triangleq \int_0^1 \left(\nabla^2 f(x+\tau(y-x)) - \nabla^2 f(x)\right) d\tau, \tag{3.45a}$$

$$\Upsilon_f^{x,y} \triangleq \nabla^2 f(x)^{-1/2} \Sigma_f^{x,y} \nabla^2 f(x)^{-1/2}. \tag{3.45b}$$

We begin by stating some useful preliminary results. The following result provides bounds on the function g_s in (3.2).

Lemma 9. *[223, Proposition 10] Suppose that P.3–P.4 hold. Then, given any $x, y \in \operatorname{dom} g$, we have*

$$\begin{aligned}\omega_\nu(-d_\nu(x,y))\|y-x\|_x^2 &\leq g_s(y) - g_s(x) - \langle \nabla g_s(x), y-x\rangle \\ &\leq \omega_\nu(d_\nu(x,y))\|y-x\|_x^2,\end{aligned} \tag{3.46}$$

in which, if $\nu > 2$, the right-hand side inequality holds if $d_\nu(x,y) < 1$, and

$$\omega_\nu(\tau) \triangleq \begin{cases} \frac{\exp(\tau)-\tau-1}{\tau^2} & \text{if } \nu = 2, \\ \frac{-\tau-\ln(1-\tau)}{\tau^2} & \text{if } \nu = 3, \\ \frac{(1-\tau)\ln(1-\tau)+\tau}{\tau^2} & \text{if } \nu = 4, \\ \left(\frac{\nu-2}{4-\nu}\right)\frac{1}{\tau}\left[\frac{\nu-2}{2(3-\nu)\tau}\left((1-\tau)^{\frac{2(3-\nu)}{2-\nu}}-1\right)-1\right] & \text{otherwise.}\end{cases} \tag{3.47}$$

The next two lemmas are instrumental in our convergence analysis, and are immediate consequences of the (local) Hessian regularity of the smooth functions f and g_s in (3.2).

Lemma 10. *[163, Lemma 1.2.4] For any given $x, y \in \operatorname{dom} f$, we have*

$$\left\|\nabla f(y) - \nabla f(x) - \nabla^2 f(x)(y-x)\right\| \leq \frac{L_f}{2}\|y-x\|^2, \tag{3.48}$$

$$\left|f(y) - f(x) - \langle \nabla f(x), y-x\rangle - \frac{1}{2}\langle \nabla^2 f(x)(y-x), y-x\rangle\right| \leq \frac{L_f}{6}\|y-x\|^3. \tag{3.49}$$

Lemma 11. *[223, Lemma 2] For any given $x, y \in \operatorname{dom} g$, $\Upsilon_{g_s}^{x,y}$ satisfies*

$$\|\Upsilon_{g_s}^{x,y}\| \leq R_\nu(d_\nu(x,y))d_\nu(x,y),$$

where, for $\tau \in [0,1)$, $R_\nu(\tau)$ *is defined by*

$$R_\nu(\tau) \triangleq \begin{cases} \left(\frac{3}{2} + \frac{\tau}{3}\right)\exp(\tau) & \text{if } \nu = 2, \\ \frac{1-(1-\tau)^{\frac{4-\nu}{\nu-2}} - \left(\frac{4-\nu}{\nu-2}\right)\tau(1-\tau)^{\frac{4-\nu}{\nu-2}}}{\left(\frac{4-\nu}{\nu-2}\right)\tau^2(1-\tau)^{\frac{4-\nu}{\nu-2}}} & \text{if } \nu \in (2,3]. \end{cases} \tag{3.50}$$

Global convergence. We prove a first global result for the proximal Newton-type scheme (3.19). We show that the iterates of this scheme decrease the objective function values with the step-lengths specified by (3.28) and $\alpha_k \in (0,1]$, and converge to an optimal solution of (3.1).

Let us define the following mapping:

$$G_{\alpha_k g}(x_k) \triangleq \frac{1}{\bar{\alpha}_k} H_k \left(x_k - \operatorname{prox}^{H_k^g}_{\alpha_k g}(x_k - \bar{\alpha}_k H_k^{-1} \nabla\, q(x_k)) \right). \tag{3.51}$$

Clearly, (3.19) is equivalent to

$$x_{k+1} = x_k - \bar{\alpha}_k H_k^{-1}\, G_{\alpha_k g}(x_k)\,. \tag{3.52}$$

Using (3.18) with $Q_k = H_k^g$ and the definition of the (scaled) proximal operator, $G_{\alpha_k g}(x_k)$ satisfies

$$G_{\alpha_k g}(x_k) \in \nabla\, q(x_k) + \partial g(x_k - \bar{\alpha}_k H_k^{-1}\, G_{\alpha_k g}(x_k)). \tag{3.53}$$

Moreover, $G_{\alpha_k g}(\bar{x}) = 0$ if and only if $\bar{x}$ solves problem (3.2).

Proposition 7. *Suppose that P.1, P.3 and P.4 hold for* (3.2). *Let* $\{x_k\}$ *be the sequence generated by scheme* (3.19) *for problem* (3.2) *and satisfying* $\omega_\nu(d_\nu(x_{k+1}, x_k)) \leq \frac{1}{2}$, *where* ω_ν *and* d_ν *are respectively defined by* (3.47) *and* (3.44). *Define* $\varepsilon_k^\mu(y) \triangleq (L_f/6)\|y - x_k\|^3$, *and let* $\bar{\alpha}_k$ *be specified by* (3.28) *with* $\alpha_k \in (0,1]$. *Then* $\{x_k\}$ *satisfies*

$$\mathcal{L}(x_{k+1}) \leq \mathcal{L}(x_k) - \varepsilon_k^\mu(x_{k+1}). \tag{3.54}$$

Proof. Letting $y = x_k - \bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k)$ and $x = x_k$ in Lemma 10, where $G_{\alpha_k g}$ is defined by (3.51), we have

$$\begin{aligned} f(x_{k+1}) \leq f(x_k) - \bar{\alpha}_k (H_k^{-1} \nabla\, f(x_k))^\top G_{\alpha_k g}(x_k) + \frac{\bar{\alpha}_k^2}{2} \left\| H_k^{-1}\, G_{\alpha_k g}(x_k) \right\|^2_{H_k^f} \\ + \quad \frac{\bar{\alpha}_k^3 L_f}{6} \left\| H_k^{-1}\, G_{\alpha_k g}(x_k) \right\|^3. \end{aligned} \tag{3.55}$$

Using $\mathcal{L}(x_{k+1}) \triangleq f(x_{k+1}) + g(x_{k+1})$ and (3.55), we get

$$\begin{aligned}
\mathcal{L}(x_{k+1}) &\leq f(x_k) - \bar{\alpha}_k (H_k^{-1} \nabla f(x_k))^\top G_{\alpha_k g}(x_k) + \frac{\bar{\alpha}_k^2}{2} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|_{H_k^f}^2 \\
&\quad + \frac{\bar{\alpha}_k^3 L_f}{6} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|^3 + g(x_k - \bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k)) \\
&\overset{Lemma\ 10}{\leq} f(z) - \langle \nabla f(x_k), z - x_k \rangle - \frac{1}{2} \|z - x_k\|_{H_k^f}^2 + \frac{L_f}{6} \|z - x_k\|^3 \\
&\quad - \bar{\alpha}_k (H_k^{-1} \nabla f(x_k))^\top G_{\alpha_k g}(x_k) + \frac{\bar{\alpha}_k^2}{2} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|_{H_k^f}^2 \\
&\quad + \frac{\bar{\alpha}_k^3 L_f}{6} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|^3 + g(x_k - \bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k)). \qquad (3.56)
\end{aligned}$$

In the above, we used the lower bound in Lemma 10 on $f(z)$. By the convexity of g, we have $g(z) - g(x_{x_{k+1}}) \geq v^\top (z - x_{k+1})$ for all $v \in \partial g(x_{k+1})$. Now since from (3.53), we have $G_{\alpha_k g}(x_k) - \nabla q(x_k) \in \partial g(x_k - \bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k))$, and noting that $\nabla q - \nabla f = \nabla g_s$, (3.56) gives

$$\begin{aligned}
\mathcal{L}(x_{k+1}) &\leq f(z) + g(z) - \langle \nabla f(x_k), z - x_k \rangle - \frac{1}{2} \|z - x_k\|_{H_k^f}^2 \\
&\quad - \bar{\alpha}_k (H_k^{-1} \nabla f(x_k))^\top G_{\alpha_k g}(x_k) + \frac{\bar{\alpha}_k^2}{2} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|_{H_k^f}^2 \\
&\quad - (G_{\alpha_k g}(x_k) - \nabla q(x_k))^\top (z - x_k + \bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k)) \\
&\quad + \frac{L_f}{6} \|z - x_k\|^3 + \frac{\bar{\alpha}_k^3 L_f}{6} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|^3 \\
&\leq \mathcal{L}(z) - \langle \nabla f(x_k), z - x_k \rangle - \frac{1}{2} \|z - x_k\|_{H_k^f}^2 - G_{\alpha_k g}(x_k)^\top (z - x_k) \\
&\quad + \frac{\bar{\alpha}_k^2}{2} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|_{H_k^f}^2 + \frac{L_f}{6} \|z - x_k\|^3 + \frac{\bar{\alpha}_k^3 L_f}{6} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|^3 \\
&\quad - \bar{\alpha}_k (H_k^{-1} \nabla f(x_k))^\top G_{\alpha_k g}(x_k) - \frac{\bar{\alpha}_k^2}{2} \langle H_k^{-1} G_{\alpha_k g}(x_k), G_{\alpha_k g}(x_k) \rangle \\
&\quad - \nabla q(x_k)^\top (z - x_k + \bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k)) \\
&= \mathcal{L}(z) + G_{\alpha_k g}(x_k)^\top (x_k - z) + \frac{L_f}{6} \|z - x_k\|^3 + \frac{\bar{\alpha}_k^3 L_f}{6} \left\| H_k^{-1} G_{\alpha_k g}(x_k) \right\|^3 \\
&\quad + \nabla g_s(x_k)^\top (z - x_k) + \bar{\alpha}_k (H_k^{-1} \nabla g_s(x_k))^\top G_{\alpha_k g}(x_k) - \frac{1}{2} \|z - x_k\|_{H_k^f}^2 \\
&\quad + \frac{\bar{\alpha}_k^2}{2} \langle H_k^{-1} (H_k^f H_k^{-1} - I_n) G_{\alpha_k g}(x_k), G_{\alpha_k g}(x_k) \rangle, \qquad (3.57)
\end{aligned}$$

where the second inequality results from the fact that $\langle H_k^{-1} G_{\alpha_k g}(x_k), G_{\alpha_k g}(x_k)\rangle \geq 0$ and $\bar{\alpha}_k \geq \bar{\alpha}_k^2$ for $0 < \bar{\alpha}_k \leq 1$. Now set $z = x_k$ in (3.57) and use the following relations from (3.52):

$$\bar{\alpha}_k H_k^{-1} G_{\alpha_k g}(x_k) = x_k - x_{k+1}, \quad G_{\alpha_k g}(x_k) = \frac{1}{\bar{\alpha}_k} H_k(x_k - x_{k+1}).$$

We get

$$\begin{aligned}
\mathcal{L}(x_{k+1}) &\leq \mathcal{L}(x_k) + \frac{\bar{\alpha}_k^2}{2}\langle H_k^{-1}(H_k^f H_k^{-1} - I_n)\, G_{\alpha_k g}(x_k), G_{\alpha_k g}(x_k)\rangle \\
&\quad + \bar{\alpha}_k (H_k^{-1} \nabla g_s(x_k))^\top G_{\alpha_k g}(x_k) + \frac{\bar{\alpha}_k^3 L_f}{6}\left\| H_k^{-1} G_{\alpha_k g}(x_k)\right\|^3 \\
&= \mathcal{L}(x_k) - \Big[\langle \nabla g_s(x_k), x_{k+1} - x_k\rangle + \frac{L_f}{6}\|x_{k+1} - x_k\|^3 \\
&\quad + \frac{1}{2}\langle H_k^g (x_{k+1} - x_k), x_{k+1} - x_k\rangle\Big]. \qquad (3.58)
\end{aligned}$$

Now, let us define the following cubic-regularized upper quadratic model of g_s near x_k (cf. [165]):

$$\begin{aligned}
\hat{g}_s(y) &\triangleq g_s(x_k) + \langle \nabla g_s(x_k), y - x_k\rangle + \frac{1}{2}\langle H_k^g (y - x_k), y - x_k\rangle \\
&\quad + \frac{L_f}{6}\|y - x_k\|^3,
\end{aligned}$$

for $y \in \mathbb{R}^n$ and L_f given by P.1. Then, using Lemma 9 with $x = x_k$, we have

$$\begin{aligned}
g_s(y) - \hat{g}_s(y) &\leq \omega_\nu(d_\nu(y, x_k))\|y - x_k\|_x^2 - \frac{1}{2}\langle H_k^g (y - x_k), y - x_k\rangle \\
&\quad - \frac{L_f}{6}\|y - x_k\|^3. \qquad (3.59)
\end{aligned}$$

Next, using (3.59) with $y = x_{k+1}$, (3.58) gives

$$\begin{aligned}
\mathcal{L}(x_{k+1}) &\leq \mathcal{L}(x_k) + g_s(x_{k+1}) - \hat{g}_s(x_{k+1}) \\
&\leq \mathcal{L}(x_k) + \left(\omega_\nu(d_\nu(x_{k+1}, x_k)) - \frac{1}{2}\right)\|x_{k+1} - x_k\|_x^2 \\
&\quad - \frac{L_f}{6}\|x_{k+1} - x_k\|^3,
\end{aligned}$$

which proves the result. ■

A straightforward implication of Proposition 7 is that the sequence $\{\mathcal{L}(x_k)\}$ is monotonically decreasing if $\bar{\delta}_k \triangleq x_{k+1} - x_k \neq 0$. Consider the set of indices

$$\mathcal{K}_S \triangleq \left\{k \text{ such that } x_k \in S \text{ and } S \text{ is a subsequence of } \{x_k\}\right\}. \tag{3.60}$$

Then, for all $k_j \in \mathcal{K}_S$, $\{x_{k_j}\}$ converges to some x^*.

Lemma 12. *Let an iterate x_k be generated by the scheme (3.19) for problem (3.2). Then, x_k is a stationary point of $\mathcal{L}$ if and only if $\bar{\delta}_k = 0$.*

Proof. The statement holds true by our characterization of the optimiality conditions in (3.18) with $Q_k = H_k^g$. ■

Theorem 3. *Let $\{x_k\} \subset \mathbb{R}^n$ in Proposition 7. Then every limit point x^* of $\{x_k\}$ at which (3.18) holds with $Q_k = H_k^g$ is a stationary point of the objective function $\mathcal{L}$ in problem (3.1).*

Proof. Proposition 7 implies $\{\mathcal{L}(x_k)\}$ is non-increasing and bounded below. Hence, it converges to a finite value $\mathcal{L}^*$. Consequently (and from the proof of Proposition 3), the sequence of iterates $\{x_k\}$ generated from (3.19) is bounded, and every limit point exists. Let x^* be a limit point of $\{x_k\}$, and now consider all $k_j \in \mathcal{K}_S$ with $\{x_{k_j}\} \to x^*$, where $\mathcal{K}_S$ is defined by (3.60). The relation in (3.23) implies inclusion in both directions, and hence since $g_s \xrightarrow{e} g$, if $\{x_{k_j}\}$ is such that

$$\limsup_{\substack{x_{k_j} \to x^* \\ \mu \downarrow 0}} \nabla g_s(x_{k_j}; \mu) \to 0, \tag{3.61}$$

one finds x^* is a stationary point of g [53]. For any suitably chosen fixed $\mu \in \mathbb{R}_{>0}$, it suffices that both properties (3.23) and (3.61) hold only approximately with respect to Proposition 3 as they pertain only to the smooth part of the problem. Taking the limit of (3.18) as $k_j \to \infty$ with $Q_k = H_k^g$, the result follows from Lemma 12. Precisely, $\bar{\delta}_{k_j} \to 0$, and hence all the limit points of $\{x_k\}$ are stationary points of $\mathcal{L}$. ■

How to choose α_k. In previous results, we did not specify a particular way to choose α_k. The results hold for any value of $\alpha_k \in (0, 1]$. Compared to the step-length selection rule proposed in [223], for instance, our approach and analysis do not directly rely on the actual value of ν in the

choice of both $\bar{\alpha}_k$ and α_k. Indeed, in the context of minimizing a function $g_s \in \mathcal{F}_{M_g,\nu}$, an optimal choice for $\bar{\alpha}_k$, in view of [223], corresponds to setting

$$\alpha_k = \begin{cases} \frac{\ln(1+d_k)(1+M\eta_k)}{d_k} & \text{if } \nu = 2, \\ \frac{2(1+M_g\eta_k)}{2+M_g\eta_k} & \text{if } \nu = 3, \end{cases}$$

where $d_k \triangleq M_g \| \nabla^2 H_k^{g^{-1}} \nabla g_s(x_k) \|$ and in each case, it can be shown that $\bar{\alpha}_k \in (0,1)$. However, choosing α_k this way does not guarantee certain theoretical bounds in the context of the framework studied in this chapter, especially for $\nu = 2$. We therefore propose to leave α_k as a hyperparameter that must satisfy $0 < \alpha_k \equiv \alpha \leq 1$. This however provides the freedom to exploit specific properties about the function f, when they are known to hold. One of such properties is the global Lipschitz continuity of ∇f, where supposing the Lipschitz constant L is known, one may set

$$\alpha_k = \min\{1/L, 1\}.$$

3.6 Numerical experiments

In this section, we validate the efficiency of the technique introduced in this chapter in numerical examples using both synthetic and real datasets from the LIBSVM repository [65]. The approach and algorithms proposed in this chapter are implemented in the Julia programming language and are available in the SCSO package (see Section 1.1.1). We test the performance of Algorithms 2 and 3 for various fixed values of $\alpha_k \equiv \alpha \in (0,1]$ (see Figure **19**). Clearly, convergence is fastest with $\alpha_k = 1$, so we fix this value for the two algorithms in the remainder of our experiments. We compare our technique with other algorithms, namely `PANOC` [220], `ZeroFPR` [226], `OWL-QN` [12], proximal gradient [142], and fast proximal gradient [25] algorithms[10]. In the sparse group lasso experiments, we compare our

[10]We use the open-source package `ProximalAlgorithms.jl` for the `PANOC`, `ZeroFPR`, and fast proximal gradient algorithms, while we use our own implementation of the `OWL-QN` (modification of `https://gist.github.com/yegortk/ce18975200e7dffd1759125972cd54f4`) and proximal gradient methods which can also be found in our package `SelfConcordantSmoothOptimization.jl`.

approach with the proximal gradient, block coordinate descent (BCD)[11] algorithm, and the semismooth Newton augmented Lagrangian (SSNAL) method [136] which was extended[12] in [250] to solve sparse group lasso problems. BCD is known to be an efficient algorithm for general regularized problems [97], and is used as a standard approach for the sparse group lasso problem [118, 98, 217]. Since the problems considered in our experiments use the ℓ_1 and ℓ_2 regularizers, we use $\phi(t) = \frac{1}{p}\sqrt{1+p^2|t|^2} - 1$ from Example 1, with $p = 1$ and derive g_s in problem (3.2) accordingly (see also Figure **18**). This provides a good (smooth) approximation for the 1- and 2-norms with an appropriate value of μ. Its gradient and Hessian are respectively

$$\nabla\, g_s(x) = x/\sqrt{\mu^2 + x^2}, \qquad \nabla^2\, g_s(x) = \operatorname{diag}\left(\mu^2/(\mu^2+x^2)^{\frac{3}{2}}\right).$$

For a diagonal matrix $H_k^g \in \mathbb{R}^{n\times n}$, the scaled proximal operator for the 1- and 2-norms are obtained using the proximal calculus derived in [29]. Let the vector $\hat{d} \in \mathbb{R}^n$ contain the diagonal entries of H_k^g, and let $\beta \in \mathbb{R}_{>0}$:

(i) $\operatorname{prox}^{H_k^g}_{\beta\|x\|_1} = \operatorname{sign}(x)\cdot\max\{|x| - \beta\hat{d}, 0\}$, and

(ii) $\operatorname{prox}^{H_k^g}_{\beta\|x\|} = x\cdot\max\{1 - \beta\hat{d}/\|x\|, 0\}$.

We terminate most of the algorithms with the default stopping criterion or when $\frac{\|x_k - x_{k-1}\|}{\max\{\|x_{k-1}\|,1\}} < \varepsilon_{tol}$ with $\varepsilon_{tol} \in \{10^{-6}, 10^{-9}\}$.

All experiments are performed on a laptop with dual (2.30GHz + 2.30GHz) Intel Core i7-11800 H CPU and 32GB RAM.

[11] We use the BCD method of [160] which is efficiently implemented with a gap safe screening rule. The open-source implementation can be found in https://github.com/EugeneNdiaye/Gap_Safe_Rules.

[12] We use the freely available implementation provided by the authors in https://github.com/YangjingZhang/SparseGroupLasso.

3.6.1 Sparse logistic regression

We consider the problem of finding a sparse solution x to the following logistic regression problem

$$\min_{x\in\mathbb{R}^n} \mathcal{L}(x) \triangleq \underbrace{\sum_{i=1}^{m} \log\left(1+\exp(-y_i\langle a_i, x\rangle)\right)}_{\triangleq f(x)} + \beta\|x\|_1, \tag{3.62}$$

where, in view of (3.1), $g(x) \triangleq \beta\|x\|_1$, $\beta \in \mathbb{R}_{>0}$, and $a_i \in \mathbb{R}^n, y_i \in \{-1, 1\}$

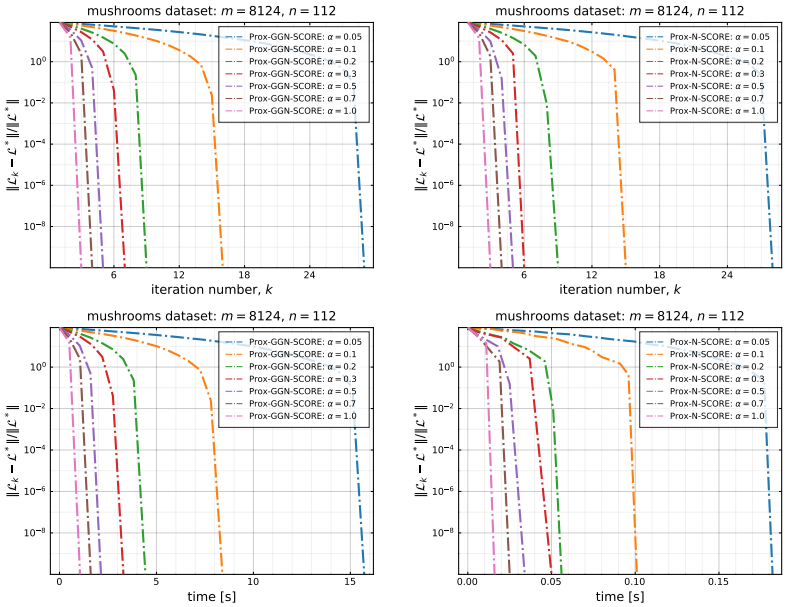


Figure 19: Behaviour of `Prox-N-SCORE` and `Prox-GGN-SCORE` for different fixed values of α_k in problem (3.62).

form the data. We perform experiments on both randomly generated data and real datasets summarized in Table 3. For the synthetic data, we set $\beta = 0.2$, while for the real datasets, we set $\beta = 1$. We fix $\mu = 1$ in both Algorithms 2 and 3, and set $\alpha_k = 1/L$ for the proximal gradient algorithm, where L is estimated as $L = \lambda_{max}(A^\top A)$, the columns of $A \in \mathbb{R}^{n\times m}$ are

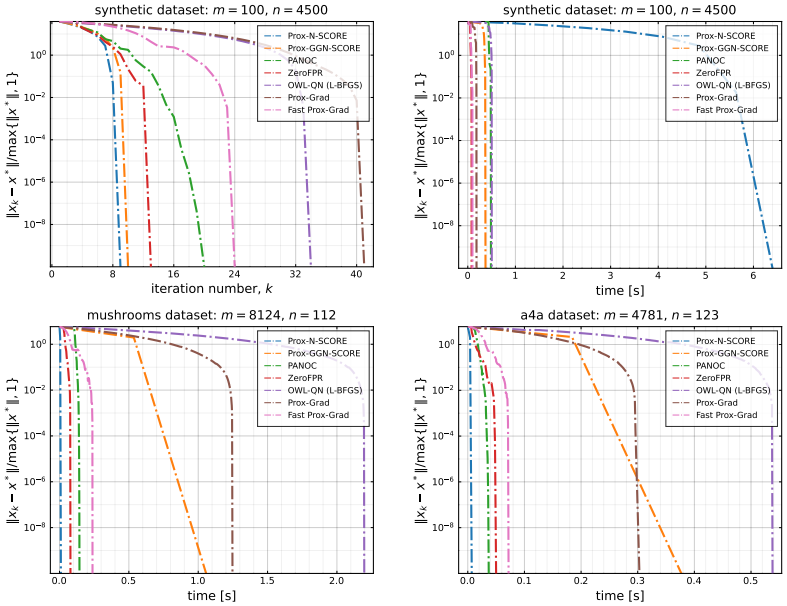


Figure 20: Test case for overparameterization (top) versus non-overparameterization (bottom) in problem (3.62).

the vectors a_i and λ_{max} denotes the largest eigenvalue. For the sake of fairness, we provide this value of L to each of `PANOC`, `ZeroFPR`, and fast proximal gradient algorithms for computing their step-lengths in our comparison.

The results are shown in Figure **19**, Figure **20** and Figure **21**. In Figure **20**, we observe that `Prox-GGN-SCORE` reduces most of computational burden of the Newton-type method when $m + n_y < n$ and makes the method competitive with the first-order methods considered. However, as shown in both Figure **19** and Figure **20**, `Prox-GGN-SCORE` is no longer preferred when $n < m + n_y$ and, by our experiments, the algorithm can run into computational issues when $n \ll m$. In this case (particularly for all of the real datasets that we use in this example), `Prox-N-SCORE` would be preferred and, as shown in the performance profile of Figure **21**, outperforms other tested algorithms in most cases, especially with $\alpha = 1$.

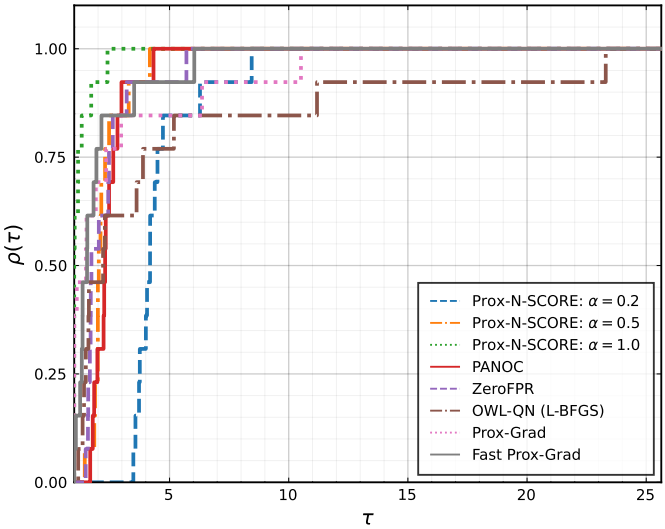


Figure 21: Performance profile (CPU time) for the sparse logistic regression problem (3.62) using the LIBSVM datasets summarized in Table **3**. Here, τ denotes the performance ratio (CPU times in seconds) averaged over 20 independent runs with different random initializations, and $\rho(\tau)$ is the corresponding frequency.

3.6.2 Sparse group lasso

In this example, we consider the sparse group lasso problem:

$$\min_{x\in\mathbb{R}^n} \mathcal{L}(x) \triangleq \underbrace{\frac{1}{2}\|Ax-y\|^2}_{\triangleq f(x)} + \underbrace{\beta\|x\|_1 + \beta_{\mathcal{G}}\sum_{j\in\mathcal{G}}\omega_j\|x_j\|}_{\triangleq g(x)}, \tag{3.63}$$

as described in Section 3.4. We use the common example used in the literature [235, 227], which is based on the model $y = Ax^* + 0.01\epsilon \in \mathbb{R}^{m\times 1}$, $\epsilon \sim \mathcal{N}(0,1)$. The entries of the data matrix $A \in \mathbb{R}^{m\times n}$ are drawn from the normal distribution with pairwise correlation $\mathrm{corr}(A_i, A_j) = 0.5^{|i-j|}$, $\forall (i,j) \in [n]^2$. We generate datasets for different values of m and n with n satisfying $(n \mod n_g) = 0$. In this problem, we want to further highlight the faster computational time achieved by the approximation in

Table 3: Summary of the real datasets used for sparse logistic regression.

Data	m	n	Density
mushrooms	8124	112	0.19
phishing	11055	68	0.44
w1a	2477	300	0.04
w2a	3470	300	0.04
w3a	4912	300	0.04
w4a	7366	300	0.04
w5a	9888	300	0.04
w8a	49749	300	0.04
a1a	1605	123	0.11
a2a	2265	123	0.11
a3a	3185	123	0.11
a4a	4781	123	0.11
a5a	6414	123	0.11

Prox-GGN-SCORE, so we consider only overparameterized models (i.e., with $m + n_y \leq n$).

In this problem, the matrix C in the reformulation (3.36) is a diagonal matrix with row indices given by all pairs $(i,j) \in \{(i,j)|i \in j, i \in \{1,\ldots,n_g\}\}$, and column indices given by $k \in \{1,\ldots,n_g\}$. That is,

$$C_{(i,j),k} = \begin{cases} \beta_{\mathcal{G}}\omega_j & \text{if } i = k, \\ 0 & \text{otherwise.} \end{cases}$$

We construct x^* in a similar way as [159]: We fix $n_g = 100$ and break n randomly into groups of equal sizes with 0.1 percent of the groups selected to be *active*. The entries of the subvectors in the *nonactive* groups are set to zero, while for the active groups, $\lceil \frac{n}{n_g} \rceil \times 0.1$ of the subvector entries are drawn randomly and set to $\text{sign}(\xi) \times U$ where ξ and U are uniformly distributed in $[0.5, 10]$ and $[-1, 1]$, respectively; the remaining entries are set to zero. For the sake of fair comparison, each data and the associated initial vector x_0 are generated in Julia, and exported for the BCD implementation in Python and also for the SSNAL implementation in MATLAB. For Prox-GGN-SCORE, Prox-Grad and BCD, we set $\beta =$

Table 4: Performance of `Prox-GGN-SCORE` (`alg.A`), `SSNAL` (`alg.B`), `Prox-Grad` (`alg.C`) and `BCD` (`alg.D`) on the sparse group lasso problem (3.63) for different values of m and n. nnz stands for the number of nonzero entries of x^* and of the solutions found by the algorithms. MSE stands for the mean squared error between the true solution x^* and the estimated solutions.

$(m, n; nnz)$	nnz				Iteration[a]				Time [s]				MSE			
	alg.A	alg.B	alg.C	alg.D	alg.A	alg.B	alg.C	alg.D	alg.A	alg.B	alg.C	alg.D	alg.A	alg.B	alg.C	alg.D
$(500, 2000; 19)$	19	198	19	19	161	62	5904	9690	**2.81**	4.65	13.39	3.55	2.9305E-09	5.3188E-08	2.5189E-07	8.0350E-06
$(500, 4000; 36)$	36	39	36	36	253	140	10991	16790	**8.44**	39.51	51.91	11.60	1.4291E-08	4.3952E-08	1.1653E-06	1.7127E-05
$(500, 5000; 45)$	45	45	45	45	530	111	13919	20830	**16.60**	35.57	90.05	18.53	2.6339E-07	6.0898E-08	2.0121E-06	2.1641E-05
$(1000, 5000; 45)$	45	82	45	45	112	35	3051	9100	**8.71**	15.73	11.57	22.14	2.4667E-07	1.9757E-06	2.1747E-06	5.0779E-06
$(1000, 7000; 65)$	65	65	65	65	185	82	7012	20870	**30.26**	148.07	42.45	70.08	4.5689E-07	2.2847E-08	4.0172E-06	1.8038E-05
$(1000, 10000; 94)$	93	94	94	94	497	102	9879	29330	**53.26**	252.05	90.25	126.17	3.8421E-06	2.8441E-08	3.6320E-06	3.5855E-05
$(1000, 12000; 112)$	112	113	113	164	663	68	21178	59360	**166.15**	194.40	221.26	373.50	1.5750E-05	4.6965E-08	7.3285E-06	5.9521E-05

[a] Number of "outer" iterations for `SSNAL` (`alg.B`).

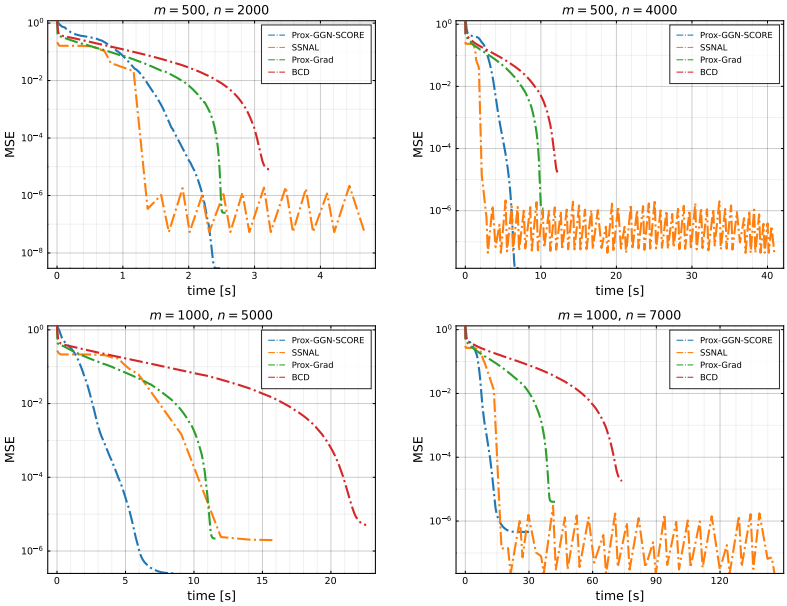


Figure 22: Mean squared error (MSE) between the estimates x_k and the true coefficient x^* for `Prox-GGN-SCORE`, `SSNAL`, `Prox-Grad` and `BCD` on the sparse group lasso problem (3.63).

$\tau_1\gamma\|A^Ty\|_\infty$, $\beta_{\mathcal{G}} = (10-\tau_1)\gamma\|A^Ty\|_\infty$ with $\tau_1 = 0.9$ and $\gamma \in \{10^{-7}, 10^{-8}\}$. `SSNAL` can be made to return a solution estimate that has number of nonzero entries close to that of the true solution with a carefully tuned β and simply setting $\beta_{\mathcal{G}} = \beta$ (cf., [250, Table 1]). We set $\beta = \tau_1\gamma\|A^Ty\|_\infty$ and $\beta_{\mathcal{G}} = \|A^Ty\|_\infty$ with $\gamma = 10^{-5}$ and $\tau_1 \in \{4, 5, 10, 12\}$ for `SSNAL`. For each group j, the parameter ω_j is set to the standard value $\sqrt{n_j}$ [98, 217], where $n_j = \mathrm{card}(j)$. For fairness, the estimate $\alpha_k = 1/L$ with $L = \lambda_{max}(A^\top A)$ is used in the proximal gradient and `SSNAL` algorithms.

The smoothing parameter μ is set to 1.2 for the problem with $m = 500$, $n = 2000$, 2.0 for the problem with $m = 1000$, $n = 12000$, and to 1.6 for the remaining problems. In principle, μ is a hyperparameter that has to be chosen to scale with the size of the optimization vector x. Any $\mu > 0$ is suitable, but larger values of n may require to set $\mu \geq 1$. This is intuitive

in the sense that a rather small μ when n is large results into a "weak" smoothing, and in the algorithmic scope, we only require that g and g_s do not conflict, which holds by construction for any $\mu > 0$.

The simulation results are shown in Table 4 and Figure 22. As shown, `Prox-GGN-SCORE` solves the problem faster than `SSNAL`, `Prox-Grad` and `BCD` algorithms in most cases with the correct number of nonzero entries in its solution estimates. In the problems considered, `Prox-GGN-SCORE` benefits a lot from overparameterization which would have potentially posed a serious computational issue for a typical second-order method. This makes the algorithm competitive with first-order methods and other Newton-type methods in large-scale problems.

3.6.3 Sparse deconvolution

In this example, we consider the problem of estimating the unknown sparse input x to a linear system, given a noisy output signal and the system response. That is,

$$\min_{x\in\mathbb{R}^n} \mathcal{L}(x) \triangleq \underbrace{\frac{1}{2}\|Ax-y\|^2}_{\triangleq f(x)} + \beta\|x\|_p, \tag{3.64}$$

where $A \in \mathbb{R}^{n\times n}$ and $y \in \mathbb{R}^{n\times 1}$ are given data about the system which we randomly generate according to [214, Example F]. We solve with both ℓ_1 ($p = 1$) and ℓ_2 ($p = 2$) regularizers, and set $\beta = 10^{-3}$. We set $\mu = 5 \times 10^{-2}$ in the smooth approximation g_s of g. We estimate $L = \lambda_{max}(A^\top A)$ and set $\alpha_k = 1/L$ in the proximal gradient algorithm. Again, for fairness, we provide this value of L to each of `PANOC`, `ZeroFPR`, and fast proximal gradient procedures in our comparison. The results of the simulations are displayed in Figure 23 and Figure 24. While `Prox-GGN-SCORE` and `Prox-N-SCORE` sometimes have more running time in this problem, they provide better solution quality with smaller reconstruction error than the other tested algorithms, which we find appealing for such signal reconstruction problems.

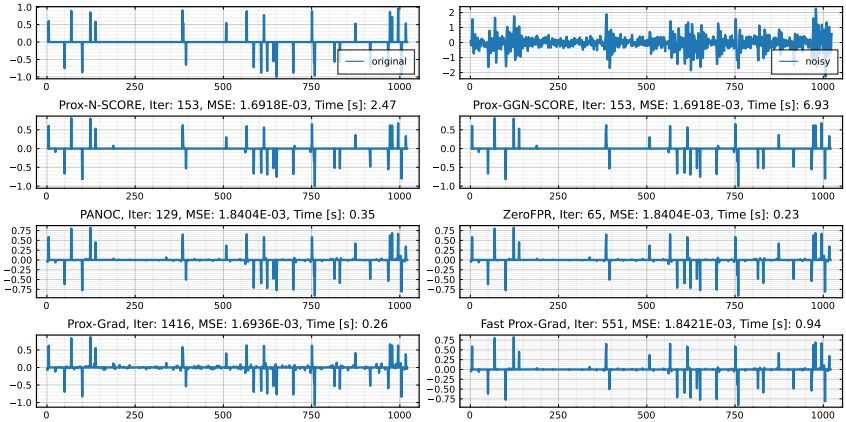


Figure 23: Sparse deconvolution via ℓ_1-regularized least squares (3.64) using different solvers with $n = 1024$.

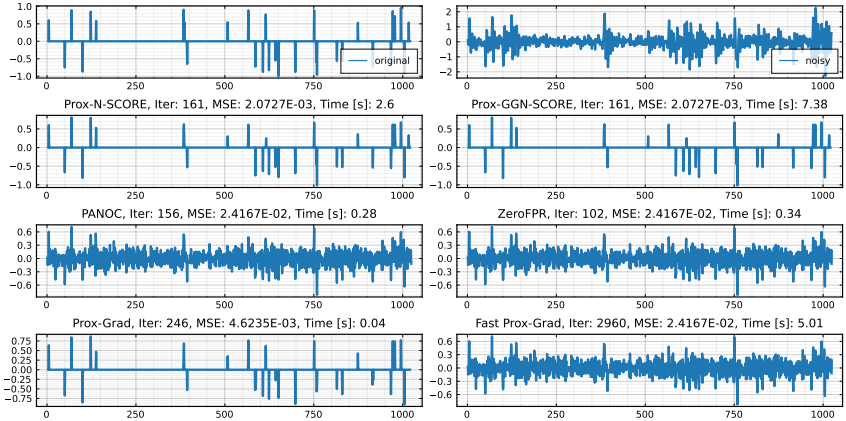


Figure 24: Sparse deconvolution via ℓ_2-regularized least squares (3.64) using different solvers with $n = 1024$.

Appendix for Chapter 3

3.A Local behaviours of Algorithms 2 and 3

In this section, we discuss the local properties of Algorithms 2 and 3. When nicely behaved, the local properties of the algorithms can be used to establish local convergence. For example, if we fix $\alpha_k = 1$ (or larger), and since $d_\nu < 1$ (noting also that ω_ν is a strictly increasing function), we get a (near) quadratic local convergence. Therefore, for an appropriate choice of some parameters of the algorithms, these properties provide very important information about the local behaviour of the algorithms. But we leave this discussion open. In the following, we take the local norm $\|\cdot\|_x$ (and its dual) with respect to g_s, and the standard Euclidean norm $\|\cdot\|$ with respect to the (local) Euclidean ball $\mathcal{B}_{r_0}(\cdot) \subset \mathcal{E}_r(\cdot)$.

Theorem 4. *Suppose that P.1–P.4 hold, and let x^* be an optimal solution of (3.2). Let $\{x_k\}$ be the sequence of iterates generated by Algorithm 2 and define $\lambda_k \triangleq 1 + M_g \omega_\nu(-d_\nu(x^*, x_k))\|x_k - x^*\|_{x_k}$, where ω_ν is defined by (3.47). Then starting from a point $x_0 \in \mathcal{E}_r(x^*)$, if $d_\nu(x^*, x_k) < 1$ with d_ν defined by (3.44), the sequence $\{x_k\}$ satisfies*

$$\|x_{k+1} - x^*\|_{x^*} \leq \vartheta_k \|x_k - x^*\| + R_k \|x_k - x^*\|_{x^*} + \frac{L_f}{2\sqrt{\rho_2}} \|x_k - x^*\|^2, \tag{3.65}$$

where $\vartheta_k \triangleq (L_1 + L_2)(\lambda_k - \alpha_k)/(\lambda_k \sqrt{\rho})$, $\alpha_k \in (0, 1]$, $R_k \triangleq R_\nu(d_\nu(x^*, x_k)) d_\nu(x^*, x_k)$ with R_ν defined by (3.50).

Proof. The iterative process of Algorithm 2 is given by

$$x_{k+1} = \operatorname{prox}^{H_k^g}_{\alpha_k g}(x_k - \bar{\alpha}_k \nabla^2 q(x_k)^{-1} \nabla q(x_k)).$$

In terms of $E_{\bar{x}}$ and $\xi_{\bar{x}}(Q_k, \cdot)$ with $Q_k \equiv H_k^g$, and using the definition of q, we have

$$
\begin{aligned}
&\|x_{k+1} - x^*\|_{x^*} \\
&= \left\| \operatorname{prox}_{\alpha_k g}^{H_k^{g^*}} \left(E_{x^*}(x_k) + \xi_{x^*}(Q_k, x_{k+1}) \right) - \operatorname{prox}_{\alpha_k g}^{H_k^{g^*}} \left(E_{x^*}(x^*) \right) \right\|_{x^*} \\
&\overset{(3.5)}{\leq} \left\| E_{x^*}(x_k) - E_{x^*}(x^*) + \xi_{x^*}(Q_k, x_{k+1}) \right\|_{x^*}^* \\
&= \left\| H^* x_k - \bar{\alpha}_k \nabla q(x_k) - H^* x^* + \bar{\alpha}_k q(x^*) \right\|_{x^*}^* \\
&= \left\| \nabla q(x^*) - \nabla q(x_k) + (1 - \bar{\alpha}_k)(\nabla q(x_k) - \nabla q(x^*)) + H^*(x_k - x^*) \right\|_{x^*}^* \\
&\leq \left\| \nabla q(x_k) - \nabla q(x^*) - H^*(x_k - x^*) \right\|_{x^*}^* + (1 - \bar{\alpha}_k) \left\| \nabla q(x_k) - \nabla q(x^*) \right\|_{x^*}^* \\
&\leq \left\| \nabla f(x_k) - \nabla f(x^*) - H_k^{f^*}(x_k - x^*) \right\|_{x^*}^* \\
&\quad + \left\| \nabla g_s(x_k) - \nabla g_s(x^*) - H_k^{g^*}(x_k - x^*) \right\|_{x^*}^* \\
&\quad + (1 - \bar{\alpha}_k) \left(\left\| \nabla f(x_k) - \nabla f(x^*) \right\|_{x^*}^* + \left\| \nabla g_s(x_k) - \nabla g_s(x^*) \right\|_{x^*}^* \right).
\end{aligned}
\tag{3.66}
$$

To estimate $\| \nabla f(x_k) - \nabla f(x^*) - H_k^{f^*}(x_k - x^*) \|_{x^*}^*$, we note that for $v \in \mathbb{R}^n$, $\|v\|_{x^*}^* \equiv \|H_k^{g^{*-\frac{1}{2}}} v\|$ since we take the dual norm with respect to g_s. Now, using P.2, we get that the matrix $H_k^{g^*}$ is positive definite and

$$
\|H_k^{g^{*-\frac{1}{2}}}\| \leq \frac{1}{\sqrt{\rho_2}}. \tag{3.67}
$$

Consequently, we have

$$
\begin{aligned}
&\left\| \nabla f(x_k) - \nabla f(x^*) - H_k^{f^*}(x_k - x^*) \right\|_{x^*}^* \\
&= \left\| H_k^{g^{*-\frac{1}{2}}} \left(\nabla f(x_k) - \nabla f(x^*) - H_k^{f^*}(x_k - x^*) \right) \right\| \\
&\leq \|H_k^{g^{*-\frac{1}{2}}}\| \left\| \nabla f(x_k) - \nabla f(x^*) - H_k^{f^*}(x_k - x^*) \right\| \\
&\overset{\text{Lemma 10}}{\leq} \frac{L_f \|x_k - x^*\|^2}{2\sqrt{\rho_2}}.
\end{aligned}
$$

To estimate $\| \nabla g_s(x_k) - \nabla g_s(x^*) - H_k^{g^*}(x_k - x^*) \|_{x^*}^*$, we can apply

Lemma 11 as in the proof of [223, Theorem 5], and get

$$\left\|\nabla g_s(x_k) - \nabla g_s(x^*) - H_k^{g^*}(x_k - x^*)\right\|_{x^*}^* \leq R_\nu(d_\nu(x^*, x_k))d_\nu(x^*, x_k)\|x_k - x^*\|_{x^*}.$$

Following [223, p. 195], we can derive the following inequality in a neighbourhood of the sublevel set of $\mathcal{L}_s$ in (3.2) using Lemma 9 and the convexity of g_s:

$$\|\nabla g_s(x_k)\|_{x_k}^* \geq \omega_\nu(-d_\nu(x^*, x_k))\|x_k - x^*\|_{x_k}. \tag{3.68}$$

In this regard, (3.28) gives

$$1 - \bar{\alpha}_k \leq \frac{\lambda_k - \alpha_k}{\lambda_k}. \tag{3.69}$$

Next, by P.2, we deduce

$$\left\|\nabla g_s(x_k) - \nabla g_s(x^*)\right\| \leq L_2\|x_k - x^*\|,$$

and

$$\left\|\nabla f(x_k) - \nabla f(x^*)\right\| \leq L_1\|x_k - x^*\|.$$

Then, using (3.67), we get

$$\begin{aligned}\left\|\nabla g_s(x_k) - \nabla g_s(x^*)\right\|_{x^*}^* &= \left\|H_k^{g^{*-\frac{1}{2}}}\left(\nabla g_s(x_k) - \nabla g_s(x^*)\right)\right\| \\ &\leq \frac{L_2}{\sqrt{\rho_2}}\|x_k - x^*\|.\end{aligned}$$

Similarly,

$$\left\|\nabla f(x_k) - \nabla f(x^*)\right\|_{x^*}^* \leq \frac{L_1}{\sqrt{\rho_2}}\|x_k - x^*\|.$$

Finally, putting the above estimates into (3.66), we obtain (3.65). ■

To prove a related property for Algorithm 3, we need an additional assumption about the behaviour of the Jacobian matrix J_k near x^*. As before, J_k denotes the Jacobian matrix evaluated at x_k; likewise, V_k and u_k. At x^*, we respectively write J^*, V^* and u^*. We assume the following:

G.1 $\|J_k v\| \geq \beta_1 \|v\|$, $\beta_1 > 0$, for all x_k near x^*, and for any $v \in \mathbb{R}^n$.

For f defined by (3.31), condition G.1 implies that the singular values of J_k are uniformly bounded away from zero, at least locally. Let the unaugmented version of the residual vector u_k be denoted $\tilde{u}_k$ at x_k, that is,

$$\tilde{u}_k \triangleq [l'_{\hat{y}_1}(y_1, \hat{y}_1; x_k), \ldots, l'_{\hat{y}_m}(y_m, \hat{y}_m; x_k)]^\top \in \mathbb{R}^m.$$

Define the following matrix:

$$W_k^T \triangleq \begin{bmatrix} \hat{y}_1''(x^{(1)}) & \hat{y}_2''(x^{(1)}) & \cdots & \hat{y}_m''(x^{(1)}) \\ \hat{y}_1''(x^{(2)}) & \hat{y}_2''(x^{(2)}) & \cdots & \hat{y}_m''(x^{(2)}) \\ \vdots & \vdots & & \vdots \\ \hat{y}_1''(x^{(n)}) & \hat{y}_2''(x^{(n)}) & \cdots & \hat{y}_m''(x^{(n)}) \end{bmatrix} \in \mathbb{R}^{n \times m}. \tag{3.70}$$

We note that the "full" Hessian matrix H_k can be expressed as

$$H_k \equiv J_k^\top V_k J_k + (\mathbf{1} \otimes (W_k^\top \tilde{u}_k))^\top + H_k^g, \tag{3.71}$$

where $\mathbf{1} \in \mathbb{R}^{n \times 1}$ is the $n \times 1$ matrix of ones and $\otimes$ denotes the outer product. By P.1, P.2 and the Lipschitz continuity of g_s around x^* in Proposition 2(iii), we have: for r small enough, there exists a constant $\beta_2 > 0$ such that $\|\tilde{u}_k\| \leq \beta_2$ near x^*. Furthermore by our assumptions (see, e.g., [169, Theorem 10.1]), we deduce that there exists $\beta_3 > 0$ such that $\|W_k\| \leq \beta_3$ near x^*.

The next result follows. Note that for Algorithm 3, we consider the case where f in problem (3.2) may, in general, be expressed in the form (3.31).

Theorem 5. *Suppose that P.1–P.4 hold, and let x^* be an optimal solution of (3.2) where f is defined by (3.31). Additionally, let G.1 hold for the Jacobian matrix J_k defined by (3.32). Let $\{x_k\}$ be the sequence of iterates generated by Algorithm 3, and define $\lambda_k \triangleq 1 + M_g \omega_\nu(-d_\nu(x^*, x_k))\|x_k - x^*\|_{x_k}$, where ω_ν is defined by (3.47). Then starting from a point $x_0 \in \mathcal{E}_r(x^*)$, if $d_\nu(x^*, x_k) < 1$ with d_ν defined by (3.44), the sequence $\{x_k\}$ satisfies*

$$\|x_{k+1} - x^*\|_{x^*} \leq \vartheta_k \|x_k - x^*\| + R_k \|x_k - x^*\|_{x^*} + \frac{L_f}{2\sqrt{\rho_2}} \|x_k - x^*\|^2, \tag{3.72}$$

where R_k is as defined in Theorem 4, $\vartheta_k \triangleq (\lambda_k(L_1+L_2)(\lambda_k-\alpha_k)+\tilde{\beta})/\sqrt{\rho_2}$, $\alpha_k \in (0,1]$, and $\tilde{\beta} \triangleq \beta_2\beta_3 > 0$.

Proof. Let $\hat{H}_k \triangleq J_k^\top V_k J_k + H_k^g$, and consider the iterative process of Algorithm 3 given by

$$x_{k+1} = \operatorname{prox}_{\alpha_k g}^{H_k^g}(x_k - \bar{\alpha}_k \hat{H}_k^{-1} J_k^\top u_k).$$

We first note that $J_k^\top u_k$ is a compact way of writing $\nabla f(x_k) + \nabla g_s(x_k) \triangleq \nabla q(x_k)$, where f is given by (3.31). Following the proof of Theorem 4, we have

$$\begin{aligned}
&\|x_{k+1} - x^*\|_{x^*} \\
&= \left\| \operatorname{prox}_{\alpha_k g}^{H_k^{g^*}}(E_{x^*}(x_k) + \xi_{x^*}(Q_k, x_{k+1})) - \operatorname{prox}_{\alpha_k g}^{H_k^{g^*}}(E_{x^*}(x^*)) \right\|_{x^*} \\
&\leq \left\| \nabla q(x_k) - \nabla q(x^*) - \hat{H}_k^*(x_k - x^*) \right\|_{x^*}^* \\
&\quad + (1-\bar{\alpha}_k) \left\| \nabla q(x_k) - \nabla q(x^*) \right\|_{x^*}^* .
\end{aligned} \tag{3.73}$$

Let W^* and $\tilde{u}^*$ respectively denote expressions for W_k and $\tilde{u}$ evaluated at x^*. Substituting (3.71) into (3.73) and using (3.67) in the estimate

$$\left\| (\mathbf{1} \otimes ({W^*}^\top \tilde{u}_k))^\top (x_k - x^*) \right\|_{x^*}^* \leq \left\| H_k^{g^{*-\frac{1}{2}}} (\mathbf{1} \otimes ({W^*}^\top \tilde{u}^*))^\top \right\| \|x_k - x^*\| ,$$

where W_k is defined by (3.70), we get

$$\begin{aligned}
\|x_{k+1} - x^*\|_{x^*} &\leq \left\| \nabla q(x_k) - \nabla q(x^*) - H^*(x_k - x^*) \right\|_{x^*}^* \\
&\quad + \left\| (\mathbf{1} \otimes ({W^*}^\top \tilde{u}^*))^\top (x_k - x^*) \right\|_{x^*}^* + (1-\bar{\alpha}_k) \left\| \nabla q(x_k) - \nabla q(x^*) \right\|_{x^*}^* \\
&\leq \left\| \nabla f(x_k) - \nabla f(x^*) - H_k^{f^*}(x_k - x^*) \right\|_{x^*}^* \\
&\quad + \left\| \nabla g_s(x_k) - \nabla g_s(x^*) - H_k^{g^*}(x_k - x^*) \right\|_{x^*}^* \\
&\quad + (1-\bar{\alpha}_k) \left(\left\| \nabla f(x_k) - \nabla f(x^*) \right\|_{x^*}^* + \left\| \nabla g_s(x_k) - \nabla g_s(x^*) \right\|_{x^*}^* \right) \\
&\quad + \frac{\tilde{\beta}\|x_k - x^*\|}{\sqrt{\rho_2}},
\end{aligned} \tag{3.74}$$

where $\tilde{\beta} = \beta_2\beta_3$. Now, using the estimates derived in the proof of Theorem 4 in (3.74) above, we obtain (3.72). ∎

Part III

Theoretical guarantees and practical frameworks for solving supervised learning and control problems in neural networks using quasi-Newton methods

Chapter 4

Regularized Gauss-Newton for optimizing overparameterized neural networks

4.1 Introduction

Despite their superior convergence rates compared to first-order methods, (approximate) second-order methods are still rarely used — and as such, underexplored — for training large-scale machine learning and neural network (NN) models. This is due to their highly prohibitive computations and memory footprints at each iteration. Some past and recent works have, however, made efforts to reduce this overhead by proposing different approximations to the Hessian of the loss function, which the methods ultimately exploit to achieve their impressive convergence properties (see e.g., Roux, Manzagol, and Bengio [199], Martens et al. [147], Vinyals and Povey [234], Martens and Grosse [149], Botev, Ritter, and Barber [45], Arbel et al. [14], Ren and Goldfarb [191], Cai et al. [57], Yao et al. [244], and Adeoye and Bemporad [4]).

One of the most appealing approximations to the Hessian matrix

within the context of practical deep learning and nonlinear optimization in general is the generalized Gauss-Newton (GGN) approximation of Schraudolph [212], which uses a positive semi-definite (PSD) matrix to model the curvature about an arbitrary convex loss function. In fact, the Fisher information matrix (FIM) — a curvature approximating matrix which certain approximate second-order methods seek to estimate — is shown to have direct connections with the GGN matrix in many practical cases [149, 177]. In addition to the desirable property of maintaining positive-definiteness throughout the training procedure, other nice properties of the GGN matrix, in comparison with the Hessian matrix, are discussed in Martens [148, Section 8.1]; see also Chen [68] for discussions in the context of nonlinear least-squares estimation and Bemporad [34] for efficient training of (deep) recurrent neural networks with a GGN approach.

Towards understanding the theoretical working of deep neural networks, a line of work [75, 137, 84, 83, 10, 253, 121, 16, 73, 242] attributes their optimization and generalization success in many applications to their immense overparameterization, that is, the property of having way more parameters than the number of data points they are being trained on. These generalization properties of the NN are known to have connections with the implicit regularization of the overparameterized NN by the gradient descent (GD) method [248, 138, 105, 122, 9, 72, 140, 225, 39]. For the (generalized) Gauss-Newton and its related FIM (or *natural gradient*), some recent works [57, 249, 124, 126, 100, 13] have shown similar approximation properties and global convergence in the overparameterized regime, also mostly attributing generalization to the implicit regularization effect of the Gauss-Newton via the NTK [57, 249, 124, 126, 100] and the mean-field [13].

In many of the works showing the implicit regularization effect of gradient-based optimizers, it is suggested that explicit regularizers are not needed at all in order to see the impressive generalization results. However, recent works, such as Wei et al. [238], Raj and Bach [190], and Orvieto et al. [171], argue that explicit regularization of the network indeed matters and should at least be given as much attention, both from a generalization

and an optimization point of view. In particular, Wei et al. [238] proved an approximation bound (in number of samples), via the lens of *margin theory*, for an infinite-width one-hidden-layer NN *weakly-regularized* by the ℓ_2-norm, which significantly improves upon other results that rely on the NTK and/or implicit regularization formalism. In addition, they proved a global polynomial convergence rate for the noisy gradient descent, an improvement over related works that similarly study NN optimization in the infinite-width limit. In Cai et al. [57], the interpretation of the GGN updates as an explicit solution of the NTK regression is used to prove a global linear convergence in the mini-batch setting. Apart from the explicit addition of a regularization term to the objective function, explicit regularization is also induced in other forms [109, 243, 218, 201, 190, 171].

In this chapter, we study the optimization of a two-layer NN by the GGN method, and by drawing inspiration from their performance in convex optimization, we consider explicit self-concordant regularization of the GGN. To the best of our knowledge, our convergence result is the first in this kind of setting: optimization of an explicitly regularized NN by the GGN method in the overparameterized regime. The structure of the class of regularization functions considered not only helps to control the local rate of change of their second derivatives [173], but can also be used in the selection of adaptive learning rates. Given bounded GGN matrices and a Lipschitz smooth hidden-layer activation, we provide a non-asymptotic guarantee for the last-iterate convergence of neural network predictions to a target function. We also discuss global convergence of the GGN despite the explicit regularization. Unlike Wei et al. [238], we do not assume an arbitrarily *weak* regularization under our setting; instead the smoothing framework covered by our study allows to choose a regularization strength which may depend only on the initialization of the NN, and is characterized by a smoothing parameter. However, for a proper choice of the regularization strength, the final trained NN model can be made to be reasonably small and "simple" in spite of the overparameterization, and we can have a tradeoff between test error and training error. Numerical simulations show how generalization and stability of the optimized neural network are established, and how they relate with

the structural properties of the class of regularizers.

4.1.1 Notation and preliminaries

We reuse notations from Chapter 3. For ease of reference, we recall the ones relevant to this chapter. The standard Euclidean norm is denoted by $\|\cdot\|$ or $\|\cdot\|_2$ and the 1-norm by $\|\cdot\|_1$. We denote the standard inner product between two vectors by $\langle\cdot,\cdot\rangle$, i.e., $\langle x, y\rangle \triangleq x^\top y$ for $x, y \in \mathbb{R}^p$. For a positive integer m, we define $[m] \triangleq \{1, 2, \ldots, m\}$. We let $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$ denote the set of nonnegative and positive real numbers, respectively. For an extended real-valued function $g\colon \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$, we denote by $\operatorname{dom} g \triangleq \{x \in \mathbb{R}^p \mid g(x) < +\infty\}$ the *(effective) domain* of g. $\Gamma_0(\mathcal{X})$ denotes the set of proper convex lower-semicontinuous (lsc) functions from $\mathcal{X} \subseteq \mathbb{R}^p$ to $\mathbb{R} \cup \{+\infty\}$. We denote by $\mathcal{C}^k(\mathbb{R}^p)$, the class of k-times continuously-differentiable functions on $\mathbb{R}^p$, $k \in \mathbb{R}_{\geq 0}$. For $g \in \mathcal{C}^3(\operatorname{dom} g)$, we let $g'(t)$, $g''(t)$ and $g'''(t)$ denote the first, second, and third derivatives of g, at $t \in \mathbb{R}$, respectively. The gradient, Hessian, and third-order derivative tensor of g at $x \in \mathbb{R}^p$ are respectively written as $\nabla_x g(x)$, $\nabla^2{}_x g(x)$, and $\nabla^3{}_x g(x)$. We omit the subscripts if the variables with respect to which the derivatives are taken are clear from the context. For a symmetric matrix $H \in \mathbb{R}^{p\times p}$, we write $H \succ 0$ (resp. $H \succeq 0$) to say H is positive definite (resp., positive semidefinite). We let $\lambda_1(H)$ denote the maximum eigenvalue of a matrix $H \in \mathbb{R}^{p\times p}$, and $\lambda_p(H)$ its minimum eigenvalue; $\operatorname{tr}(H)$ denotes the trace of H. The scalars $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ respectively denote the maximum and minimum singular values of an $m \times p$ matrix A. Given that $\nabla^2 g(x) \succ 0$, the *local* norm $\|\cdot\|_x$ with respect to g at x is the weighted norm induced by $\nabla^2 g(x)$, i.e., $\|d\|_x \triangleq \langle \nabla^2 g(x) d, d\rangle^{1/2}$. The dual norm is $\|v\|_x^* \triangleq \langle \nabla^2 g(x)^{-1} v, v\rangle^{1/2}$. We also define the notations $\|x\|_H \triangleq \langle Hx, x\rangle^{\frac{1}{2}}$, $\|x\|_H^* \triangleq \langle H^{-1}x, x\rangle^{\frac{1}{2}}$, for $H \succ 0, x \in \mathbb{R}^p$. An Euclidean ball of radius r centered at $\bar{x}$ is denoted by $\mathcal{B}_r(\bar{x}) \triangleq \{x \in \mathbb{R}^p \mid \|x - \bar{x}\| \leq r\}$. The (Dikin) ellipsoid of radius r centered at $\bar{x}$ is defined by $\mathcal{E}_r(\bar{x}) \triangleq \{x \in \mathbb{R}^p \mid \|x - \bar{x}\|_H < r\}$, for $H \succ 0$. We define set convergence in the sense of Painlevé-Kuratowski [196, Chapter 4]. Given $\{g_t\}_{k\in\mathbb{R}_{\geq 0}}$ with $g_t\colon \mathbb{R}^p \to \mathbb{R} \cup \{-\infty, +\infty\}$, e$-\lim g_t = g$ denotes

the epigraphic convergence (*epi-convergence*) of $\{g_t\}_{k\in\mathbb{R}_{\geq 0}}$ to a function $g\colon \mathbb{R}^p \to \mathbb{R}\cup\{-\infty,+\infty\}$.

4.2 Learning neural networks with GGN

Given the sequence of data points $S \triangleq \{(x_i, y_i)\}_{i\in[m]}$ with $x_i \in \mathbb{R}^{n_0}, y_i \in \mathbb{R}^{n_L}$, an L-layer fully-connected feedforward NN is defined as follows. Starting with an input $z_0 \in \mathbb{R}^{n_0\times m}$, and for $l = 1,\dots,L$,

$$a_l = W^{(l)} z_{l-1} + b^{(l)}, \quad z_l = \varrho_l(a_l), \tag{4.1}$$

where $W^{(l)} \in \mathbb{R}^{n_l\times n_{l-1}}$ and $b^{(l)} \in \mathbb{R}^{n_l}$ are the l-th layer weights and biases of the network, respectively; each $\varrho_l\colon \mathbb{R}\to\mathbb{R}$ is an element-wise activation function. Let $\Phi(\cdot;\theta) \triangleq z_L$ be the output of the NN, where $\theta = [\theta_1,\theta_2,\dots,\theta_L]^\top \in \mathbb{R}^p$ with $\theta_l \triangleq vec([W^{(l)}\ b^{(l)}])$, the stacked vectorization of $W^{(l)}$ and $b^{(l)}$. In the *supervised learning* task, we look for the parameter vector θ minimizing the regularized empirical risk

$$\min_{\theta\in\mathbb{R}^p} \mathcal{L}(\theta) \triangleq \hat{R}_s(\Phi) + g(\theta), \tag{4.2}$$

where $\hat{R}_s(\Phi) \triangleq \frac{1}{m}\sum_{i=1}^{m} \ell(\Phi(x_i;\theta), y_i)$ is the empirical risk associated with the NN learning task, $\ell\colon \mathbb{R}^{n_L}\times\mathbb{R}^{n_L}\to\mathbb{R}$ is a loss function, and $g\colon \mathbb{R}^p\to\mathbb{R}$ is a regularization function. We denote by Φ^* an output function that best interpolates the data set S.

Let $n_1 \equiv n$ (number of hidden neurons), $W^{(1)} \equiv u = [u_1,u_2,\dots,u_n]^\top \in \mathbb{R}^{n\times n_0}$ and $W^{(2)} \equiv v = [v_1,v_2,\dots,v_n] \in \mathbb{R}^n$. Without loss of generality, we consider a *biasless* one-hidden layer NN:

$$\mathbb{R}^{n_0} \ni x \mapsto \Phi(x;\theta) \triangleq \kappa(n)\sum_{i=1}^{n} v_i \varrho(u_i x), \tag{4.3}$$

where $\kappa(n)$ is some scaling that depends on n, e.g., $\kappa(n) = 1/\sqrt{n}$ as in Du et al. [84]. We remark that for an $(L-1)$-hidden layer NN written in the biasless form, the bias vectors can always be recovered by redefining

$$x \leftarrow \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad \theta_l \leftarrow vec\left(\begin{bmatrix} W^{(l)} & b^{(l)} \\ 0 & 1 \end{bmatrix}\right),$$

for $l \in [L-1]$, and $\theta_L \leftarrow vec([W^{(L)}\ b^{(L)}])$. We assume the following about the activation function ϱ, which is satisfied by most activation functions but piecewise linear ones.

A.1 The activation function ϱ is twice differentiable, Lipschitz, and smooth.

Below, we briefly describe the NTK regression and its connection with GGN for overparameterized networks by first considering the case $g = 0$ in (4.2).

The NTK and gradient descent. In the infinite-width limit, there is an established [121] relation between the steps obtained via a gradient-based method for NNs and the so-called *kernel gradient descent* in function space. In particular, it is shown that, as $n \to \infty$, $\forall i, j \in [m]$, $\langle \nabla_\theta \Phi(x_i, \theta_0), \nabla_\theta \Phi(x_j, \theta_0) \rangle$ converges to some positive definite deterministic kernel $k(x_i, x_j) = k(x_i, x_j)^\top \in \mathbb{R}^{n_L \times n_L}$ (the limiting NTK), and remains unchanged during training. Consider the case $g = 0$ in (4.2). In the infinite-width limit, the gradient descent for solving the resulting problem reduces to the kernel gradient descent:

$$\Phi_{t+1} = \Phi_t - \alpha_t G_t \nabla_{\Phi_t} \hat{R}_s(\Phi_t), \tag{4.4}$$

where $\Phi_t \triangleq (\Phi(x_i; \theta_t))_{i \in [m]} \in \mathbb{R}^m$ denotes the network outputs on x_i's at iteration t, $\alpha_t \in \mathbb{R}_{>0}$ is a step-size (or *learning rate*), and G_t is an $m \times m$ matrix whose (i, j)-th entry is given by $\langle \nabla_\theta \Phi(x_i, \theta_t), \nabla_\theta \Phi(x_j, \theta_t) \rangle$; see, e.g., Arora et al. [17, Lemma 3.1] which considers the continuous-time evaluation of Φ_t, $t \in \mathbb{R}_{\geq 0}$.

GGN and the NTK regression. An important feature of GGN for infinite-width NNs is its direct relation with the NTK regression solution in the overparameterized regime. We introduce the notations $Q_t \equiv \nabla^2_{\Phi_t} \hat{R}_s(\Phi_t)$, $e_t \equiv \nabla_{\Phi_t} \hat{R}_s(\Phi_t)$, and consider again the case $g = 0$ in problem (4.2). The GGN for the resulting problem is given by the following iterative process:

$$\theta_{t+1} = \theta_t - \alpha_t (J_t^\top Q_t J_t)^{-1} J_t^\top e_t, \tag{4.5}$$

where $J_t = (\nabla_\theta \Phi(x_1, \theta_t), \ldots, \nabla_\theta \Phi(x_m, \theta_t))^\top \in \mathbb{R}^{m \times p}$ is the Jacobian matrix (of features) at iteration t. For overparameterized networks, if ℓ is the squared loss, Q_t becomes the identity matrix and we can conveniently rewrite the GGN updates with respect to the NTK matrix G_t as

$$\theta_{t+1} = \theta_t - \alpha_t J_t^\top G_t^{-1} e_t. \tag{4.6}$$

Note that the GGN in (5.44) (and (4.6)) corresponds to taking the zero damping limit ($\tau \to 0$ in Remark 9 below) for FIM, where it is natural to use the Moore-Penrose inverse due to the rank-deficiency of the matrices in the overparameterized regime [124]. On the other hand, corresponding to (4.4), the updates to the parameters θ via θ_t can be obtained by solving the regression problem (cf. [57])

$$\theta_{t+1} = \operatorname*{argmin}_\theta \frac{1}{2} \left\| \langle J_t, \theta - \theta_t \rangle + \nabla_{\Phi_t} \hat{R}_s(\Phi_t) \right\|^2, \tag{4.7}$$

which results from the linearization of Φ around θ_t. The Hessian (with respect to θ) of the resulting empirical risk by replacing Φ by this linearization, gives the GGN approximation (see, e.g., Martens and Sutskever [150]). For an overparameterized NN, when θ_t is close θ_0, the linearization (resp., the GGN) provides a good approximation to Φ (resp., the actual Hessian). In terms of kernel "ridgeless" regression solution to problems of the form (4.2) (with $g = 0$) and ℓ taken as the squared loss, the updates in (4.6) converge to a *minimum-norm* interpolating solution in the so-called Reproducing Kernel Hilbert Space (RKHS) [141, 156]. Hence, the GGN, in this case, provides a closed-form solution to the NTK regression, which efficiently replaces gradient descent in the NTK formalism.

4.2.1 Regularized GGN for overparameterized neural networks

If $g \neq 0$, the relation between gradient descent and NTK will probably break [238]. Apart from the NTK parameterization, a commonly studied parameterization in the context of overparameterized NNs is the random feature (RF) model [189, 20] which, due to its close connection with a one-hidden layer NN, often provides a prototype for studying realistic

NNs. Whether the NTK or the RF parameterization is used, one of the key properties we desire about the dynamics of the optimizer is *stability* which, when established, can help to still benefit a lot from overparameterization in the optimization scope. In essence, we seek properties of g which not only help to improve generalization of the trained model, but which also introduces stability in the optimizer's dynamics. To this end, it is important to recall the definition of generalized self-concordant (GSC) functions on $\mathbb{R}^p$ from Sun and Tran-Dinh [223] (see Definition 5).

We now make the following assumption about g.

G.1 The regularization function g is convex and (M_g, ν)-GSC.

The class of regularization functions satisfying condition G.1 includes the self-concordant smoothing functions for commonly used regularizers such as the ℓ_1- and ℓ_2-norms (see Definition 6). The resulting smooth approximation has the key property that it *epi-converges* to the original regularizer, providing useful features that can be exploited on the epigraph of g for optimization.

For the regularized problem (4.2) (with g satisfying G.1), the corresponding GGN update is obtained by augmenting the terms Q_t, e_t and J_t, respectively by 0, 1 and $\nabla g(\theta_t)$ in the appropriate dimensions [4]. Let us denote these augmented counterparts by $\hat{Q}_t$, $\hat{e}_t$ and $\hat{J}_t$. We then write for the GGN

$$\theta_{t+1} = \theta_t - \alpha_t(\hat{J}_t^\top \hat{Q}_t \hat{J}_t + H_t)^{-1} \hat{J}_t^\top \hat{e}_t, \tag{4.8}$$

or in its convenient form for overparameterized models as [4]

$$\theta_{t+1} = \theta_t - \alpha_t H_t^{-1} \hat{J}_t^\top (I + \hat{Q}_t \hat{J}_t H_t^{-1} \hat{J}_t^\top)^{-1} \hat{e}_t, \tag{4.9}$$

where $H_t \equiv \nabla^2 g(\theta_t)$. Relative to the minimal assumptions required to control the dynamics of the network outputs for the unregularized case, e.g., positive definiteness of G_t (for overparameterized NNs with a specific random initialization), we need the following standard regularity assumptions on $\hat{R}_s$, $\hat{Q}_t$ and $\hat{e}_t$ (see Appendix 4.B for details on the regularity terms):

R.1 $\hat{R}_s$ is γ_R-strongly convex, and has upper-bounded gradients and Hessian; g, $\hat{Q}_t$ and $\hat{e}_t$ are locally bounded.

An important consequence of condition A.1 is that, in addition to g admitting a Lipschitz continuous gradient (see Lemma 2 in Appendix 4.A.2), we get that J_t is (locally) Lipschitz continuous (see Appendix 4.A.1). Then, together with R.1 and the stability of H_t, we can control the key terms $H_t^{-1}\hat{J}_t^\top(I + \hat{Q}_t\hat{J}_t H_t^{-1}\hat{J}_t^\top)^{-1}$ and $\hat{e}_t$ appearing in (4.9).

In the same spirit as (4.4), which results from infinite overparameterization, the NN trained according (4.9) evolves in discrete-time as

$$\Phi_{t+1} = \Phi_t - \alpha_t \hat{G}_t \hat{e}_t, \tag{4.10}$$

where $\hat{G}_t \triangleq J_t H_t^{-1}\hat{J}_t^\top(I + \hat{Q}_t\hat{J}_t H_t^{-1}\hat{J}_t^\top)^{-1} \in \mathbb{R}^{m\times(m+1)}$. Empirically, the behaviour of the NN in (4.10) can be simulated with the *hidden learning* phenomenon which GGN exhibits for small step-sizes [13]. One major observation about the behaviour of the dynamics of G_t in (4.4) in the overparameterized setting is its stability throughout the training process, which characterizes the optimizer's global optimality [83]. In the analysis of gradient descent, most stability and convergence results in the literature heavily rely on the (strictly positive) minimum eigenvalue of G_t. These kinds of results are not immediate with $\hat{G}_t$ or, in general, with explicit regularization. However, the self-concordant condition on g ensures that its Hessian is at least locally stable[1], and hence for an appropriate parameterization of the NN, we can ensure the stability of the dynamics of $\hat{G}_t$. In addition to these, also noteworthy is an immediate deduction from the Lipschitzness of ϱ and ∇g: the boundedness of the singular values of $\hat{J}_t$ away from zero.

[1] As noted in the introduction, self-concordance helps to control the rate at which the Hessian of g changes locally, and this property has been recently formalized and studied for the notion of local and global *Hessian stability* in convex optimization (see, e.g., Karimireddy, Stich, and Jaggi [125], Gower et al. [103], and Carmon et al. [61]).

4.3 Theoretical result

In line with the settings of Section 4.2, we choose the learning rate for GGN as

$$\alpha_t = \frac{\bar{\alpha}_t}{1 + M_g \eta_t}, \tag{4.11}$$

where $0 < \bar{\alpha}_t \leq 1$ and $\eta_t = \|\nabla g(\theta_t)\|^*_{\theta_t}$. Without any emphasis on the particular choice of the target function Φ^*, we assume it is given by any *universally consistent*[2] algorithm as a minimum requirement. An example, in the case $\ell(\Phi, y) \triangleq \frac{1}{2}(\Phi - y)^2$ in (4.2), is the regularized least squares algorithm which, for a kernel prediction function $\Phi_{\mathrm{RF}} \equiv \Phi^*$ is defined by

$$\mathbb{R}^{n_0} \ni x \mapsto \Phi_{\mathrm{RF}}(x; (u^\infty, v^*)) \triangleq \sum_{i=1}^{n} v_i^* \varrho(u_i^\infty x). \tag{4.12}$$

This yields the estimator $v^* = [v_1^*, v_2^*, \ldots, v_n^*] \in \mathbb{R}^n$, the unique minimizer of the ℓ_2-regularized empirical loss $\frac{1}{2m}\sum_{i=1}^{m}(\Phi_{\mathrm{RF}}(x_i; v) - y_i)^2 + \frac{\lambda}{2}\|v\|^2$, for some $\lambda \in \mathbb{R}_{\geq 0}$, where the entries of $u^\infty = [u_1^\infty, u_2^\infty, \ldots, u_n^\infty]^\top \in \mathbb{R}^{n \times n_0}$ remain fixed iid random variables.

We state our main result of this section in Theorem 6 below. The detailed proof is given in Appendix 4.B. We let $\tilde{\Phi}_t \in \mathbb{R}^{m+1}$ denote the vector obtained by augmenting Φ_t by 1. This $\tilde{\Phi}_t$ corresponds to augmenting the rows of J_t in the definition of $\hat{G}_t$ by the vector whose entries are all zeros except the last entry which has the value $\tilde{\phi}_t = 1/\phi_{t-1}^{m+1} \in \mathbb{R}$, where ϕ_t^{m+1} denotes the last entry of $H_t^{-1}\hat{J}_t^\top(I + \hat{Q}_t \hat{J}_t H_t^{-1} \hat{J}_t^\top)^{-1}\hat{e}_t \in \mathbb{R}^{n \times 1}$ at a time t. Let this augmented version of J_t be denoted by $\tilde{J}_t$. Then, we define $\tilde{G}_t \triangleq \tilde{J}_t H_t^{-1} \hat{J}_t^\top (I + \hat{Q}_t \hat{J}_t H_t^{-1} \hat{J}_t^\top)^{-1} \in \mathbb{R}^{(m+1)\times(m+1)}$, and

$$\tilde{\Phi}_{t+1} = \tilde{\Phi}_t - \alpha_t \tilde{G}_t \hat{e}_t. \tag{4.13}$$

We also let $\tilde{\Phi}^* \in \mathbb{R}^{m+1}$ denote the vector obtained by augmenting Φ_t^* by 0.

[2]Informally speaking, a learning algorithm is said to be universally consistent if the error of its estimate tends to zero as the sample size tends to infinity, for all distributions of the sample space such that the second moment of the output variable is finite. For a more precise context, see, e.g., Caponnetto and De Vito [60] and the references therein.

Theorem 6. *Suppose that A.1, G.1 and R.1 hold, and let Φ^* be a universally consistent target function that best interpolates the training data set S. Then, for an initialization $\theta_0 \in \mathbb{R}^p$, the following convergence properties hold:*

P.1 $\|\Phi_T - \Phi^*\|^2 \leq \epsilon$ *after* $T \triangleq \dfrac{1}{\bar{\alpha}_t} \log(\|\tilde{\Phi}_0 - \tilde{\Phi}^*\|^2/\epsilon)$ *iterations for any* $\epsilon \in (0,1)$ *and* $\bar{\alpha}_t \in (0,1)$*, if* $\|\tilde{G}_t\|_F = \Omega(1)$*.*

P.2 $\mathcal{L}(\theta_{t+1}) \leq \mathcal{L}(\theta_t) - \xi_t$*, for all* $t \in \mathbb{R}_{\geq 0}$*, where* $\xi_t = \mathcal{O}\left(\frac{\hat{\beta}_1^2(\omega_\nu(d_\nu(\theta_t,\theta_{t+1})) - \omega_\nu(-d_\nu(\theta_t,\theta_{t+1})))}{\hat{\beta}_m^4}\right) \in \mathbb{R}_{>0}$*,* $\hat{\beta}_m \triangleq \sigma_{\min}(J)$*,* ω_ν *is an increasing univariate function defined by (3.47), and* $d_\nu(\bar{\theta}, \tilde{\theta}) < 1$ *is a scaled metric term defined by (3.44).*

Remark 9. *The asymptotic lower bound* $\|\tilde{G}_t\|_F = \Omega(1)$ *in P.1 inherently dictates how well we can control the regularization strength* $\tau \in \mathbb{R}_{>0}$*, that is, when g takes the form* $g(\theta) = \tau\bar{g}(\theta)$*. Here, we only require that $\bar{g}$ is GSC so that g satisfies G.1. Recall the norm property* $\|\tilde{G}_t\|_F \leq \sqrt{(m+1)\lambda_1(\tilde{G}_t^\top \tilde{G}_t)}$ *for all t. Then since η_t can become arbitrarily small, and without loss of generality, if* $1 + M_g\eta_t \leq \|\tilde{G}_t\|_F$*, one can choose τ to satisfy* $1 + \tau M_g \eta_0 \leq \sqrt{(m+1)\lambda_1(\tilde{G}_0^\top \tilde{G}_0)}$ *such that P.1 holds. In this setting, we essentially rely on the local stability of $\tilde{G}_t$ via overparameterization and the self-concordance of g.*

Since the direct relation between GGN and NTK probably breaks with an explicit regularization, our main proof step in Theorem 6 involves analyzing a partitioning of the matrix $\hat{G}_t$. In this way, we determine what conditions on the separate blocks help to combine certain spectral properties of $\tilde{G}_t$ with our regularity conditions and the self-concordance of g. The second result becomes almost immediate in the optimization scope under the regularity conditions.

Under the strong convexity assumption on $\hat{R}_s$, the global convergence of GGN can be guaranteed in the case of no regularization, for example, by training only the last layer of the NN given the property of *no blow-up* of the GGN dynamics [13, Proposition 1, Proposition 3]. Consider the matrix G^∞ whose (i,j)-th entry is given by $\langle \nabla_v \Phi(x_i; (v, u^\infty)), \nabla_v \Phi(x_j; (v, u^\infty)) \rangle$, where u^∞ is as defined in (4.12). The main observation here is that the function $v \mapsto r(v) = \hat{R}_s(\Phi(\cdot; (v, u^\infty)))$ can be shown to satisfy a certain Polyak-Łojasiewicz (PL)

inequality, that is [13, Proposition 4] $\frac{1}{2}\left\|\nabla_v l(v)\right\|^2 \geq \gamma_R \sigma^2_{\infty_n}(l(v) - \hat{R}_s(\Phi^*))$, where σ_{∞_n} is the minimum singular value of G^∞. With a self-concordant regularization function g, this kind of global property is retained, provided that g and $\hat{R}_s$ do not conflict. Consider, for example, the sublevel set $\mathcal{S}_{g(\theta)}(g) \triangleq \left\{\bar{\theta} \in \operatorname{dom} g \mid g(\bar{\theta}) \leq g(\theta)\right\}$ of g. Then, following Sun and Tran-Dinh [223, Theorem 4], we get that $\mathcal{S}_{g(\theta)}(g)$ is bounded for $\nu \in [2, 3]$, and hence g attains its minimum.

4.4 Simulations

We perform numerical simulations on GGN with self-concordant regularization (GGN-SCORE) for overparameterized NNs using synthetic datasets as well as the MNIST dataset. Results of additional simulations on the FashionMNIST and three UCI datasets are reported in Appendix 4.C.

Simulation setup. We consider the *teacher-student* setting in which Φ defined by (4.3) with $\kappa(n) = 1/\sqrt{n}$ is the student NN, while the teacher NN is the target function Φ^*, a one-hidden layer NN given as

$$\mathbb{R}^{n_0} \ni x \mapsto \Phi^*(x; \theta^*) \triangleq \sum_{i=1}^{n^*} v_i^* \varrho(u_i^* x), \tag{4.14}$$

where $\theta^* \equiv (u^*, v^*)$. In both teacher and student networks, we use the SiLU activation function [90] $\varrho(x) \triangleq x/(1 + \exp(-x))$. In each simulation, we generate m training data points $(x_i, y_i)_{i \in [m]}$, where the inputs x_i are uniformly sampled on the unit sphere $\mathbb{S}^{n_0 - 1} \triangleq \{x \mid \|x\| = 1\}$ and the corresponding target outputs are given by $y_i = \Phi^*(x_i; \theta^*)$. The weights of the teacher NN are randomly generated as in [73]: they are normalized random weights satisfying $\|v_i^* u_i^*\| = 1$ for $i = 1, \ldots, n^*$. The student NN is initialized with randomly generated weights from the Gaussian distribution. In all the simulations, we fix $n = 500$ and $n^* = 5$. The student NN is trained by minimizing the regularized empirical risk in (4.2) with the squared loss ℓ (the empirical risk is unregularized for GD), and we consider regularization of the form $g(\theta) = \tau \bar{g}(\theta)$, where $\tau \in \mathbb{R}_{>0}$ and $\bar{g}$

is given by [5, Example 1]: $\bar{g}(\theta) = (\|\cdot\|_1 \square h_\mu)(\theta) = \sum_{i=1}^p \frac{\mu^2 - \mu\sqrt{\mu^2+\theta_i^2}+\theta_i^2}{\sqrt{\mu^2+\theta_i^2}}$, $h_\mu(\cdot) \triangleq \mu h(\cdot/\mu)$, $h(\theta) = \sum_{i=1}^p ((1+|\theta_i|^2)^{1/2} - 1)$, which gives the (M_g, ν)-GSC function

$$g(\theta) = \tau \sum_{i=1}^p \frac{\mu^2 - \mu\sqrt{\mu^2+\theta_i^2}+\theta_i^2}{\sqrt{\mu^2+\theta_i^2}}, \tag{4.15}$$

with $M_g = 2\mu^{-0.7}p^{0.2}$, $\nu = 2.6$ (see Lemma 2). We choose $\mu = 1/\kappa(n)$ and $\tau = 10^{-4}$, except for where we consider different values for comparison. We set $\bar{\alpha}_t \equiv \bar{\alpha} = 0.95$ in (4.11) for GGN and use a learning rate of 1 for GD. All experiments are performed on a laptop with 16×2.30GHz Intel Core i7-11800H CPU and 32GB RAM.

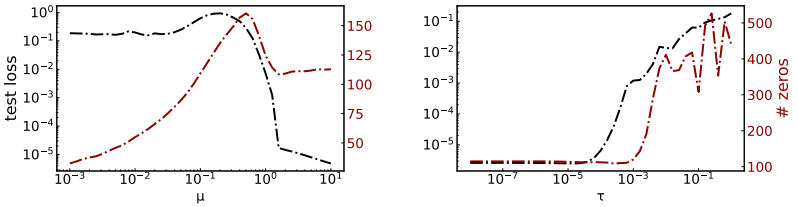


Figure 25: Performance of GGN-SCORE with $g(\theta)$ as in (4.15). Left: Results for different values of μ, with $\tau = 10^{-4}$. Right: Results for different values of τ, with $\mu = 1/\sqrt{n}$. Results are averaged over 10 independent runs for each value of μ and τ, resp.; the total computation time is ~ 21 hours, 2 minutes on CPU.

4.4.1 Results and discussion

Test loss vs. smoothing parameter. We compare the performance of GGN-SCORE for different values of the regularization smoothing parameter μ evenly spaced in the range $[10^{-3}, 10]$, giving 41 different values in total. We use a reasonable amount of training samples, 500, which allows to perform several independent runs for each value of μ considered. We use a test size of 1000 to measure generalization of the student NN for each μ. We perform 10 independent runs for each value of μ considered and took the average value of the results. These are shown in

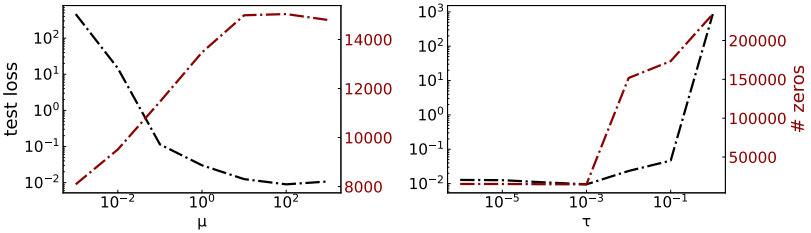


Figure 26: Test loss evaluation of the GGN-SCORE-trained NN on MNIST dataset for different values of the regularization smoothing parameter μ fixing $\tau = 10^{-4}$ (left) and different values of the regularization strength τ fixing $\mu = 1/\sqrt{n}$ (right). The regularization function $g(\theta)$ is given by (4.15).

Figure **25**. We observe that larger values of μ yields better performance in the optimization scope and also better generalization. This result is quite intuitive, since by definition of the regularization function, the size of μ should scale with the size of the variable θ in order to have an adequate smooth approximation of the original nonsmooth function. For this reason, it is recommended to choose $\mu = c/\kappa(n)$ for any $c > 0$ when scaling with $\kappa(n) = 1/\sqrt{n}$.

Test loss vs. regularization strength. We study the influence of the regularization strength on the evolution of the test loss within the optimization loop of GGN-SCORE. We consider different values of τ evenly spaced in the range $[10^{-8}, 1]$, giving 41 different values in total, and set $\mu = 1/\sqrt{n}$. Here, we also use a training size of 500 and a test size 1000 which allows to perform several independent runs for each value of τ considered. We perform 10 independent runs for each value of τ and compute the average value of the results. These average values are shown in Figure **25**. We observe that smaller values of τ yield smaller training and test errors for the overparameterized NN. This observation corroborates with the analysis of [238] for GD. However, contrarily to [238], what we observe for the GGN-SCORE is not an arbitrarily small regularization strength to achieve a good generalization performance. In fact, any value of τ slightly smaller than 10^{-4} in our experiment gives a similar generalization error

as the choice $\tau = 10^{-6}$ (and smaller). Figure 25 also displays the average number of zero entries in the value of θ at the end of training. As observed, larger values of τ yields a sparser/simpler model. In principle, a desirable value of τ is one which helps to avoid overfitting of the NN model such that a simpler model implies better generalization.

Loss decay in the optimization loop. We generate training and test datasets of sizes 1000 and 2000, respectively, and compare the training and test losses per iteration and time in seconds between GD and GGN-SCORE for training the student NN. The results are displayed in Figure 27. The dimension of the input data in this experiment is 20, and the number of hidden neurons for the student network is 500. Choosing a much smaller number of hidden neurons $n^* = 5$ for the teacher network keeps the optimization loop in the overparameterized regime. For the GD, we use a learning rate of 1 which yields a much better performance than smaller values. While larger learning rates could yield faster learning at the beginning of training, we notice traces of divergence later on; a learning rate of 1 gives a reasonably good descent and good performance of GD. We run GD for a total of 10000 steps and GGN-SCORE for a total of 4000 steps. GGN is well-known for its faster convergence in terms of number iterations while sometimes we may have to train for a longer time. Results here show that we do not trade total training time for better performance with our GGN-SCORE setup.

4.4.2 Experiments on real datasets

The computations involved in full-batch GD and/or GGN are intractable on real-world datasets. We study the properties of GGN-SCORE in the mini-batch setting on the standard MNIST dataset [131] with $n_0 = 784$, $m = 60000 : 10000$ (training:test splits). Experimental results in the teacher-student setup according to [13, Appendix C.4] with the SiLU activation function are reported in Appendix 4.C. Additional experiments on three UCI datasets, as well as those on the FashionMNIST dataset, are also considered in the appendix. Here, we consider a NN of the form (4.3) with a hidden size $n = 512$, a scaling $\kappa(n) = 1/\sqrt{n}$ and the ReLU

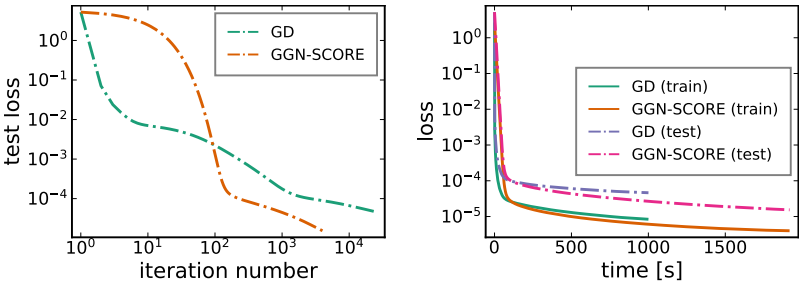


Figure 27: Loss decay with GGN-SCORE and GD per iteration number (left) and time in seconds (right) with $g(\theta)$ as in (4.15) for GGN-SCORE, $\tau = 10^{-4}$, $\mu = 1/\sqrt{n}$.

[158] activation function. The NN is initialized with randomly generated weights from the Gaussian distribution, and is trained with the squared loss. The regularization function g used in GGN-SCORE is given by (4.15). All results shown for GGN-SCORE are for a training batch size of 16 (i.e., 3750 training steps) and a single epoch. In addition to the test loss and prediction accuracy of the trained model, we adopt a time-invariance "T-I" measure, representing the average proportion (in percentage) of the entries of the *pre-activation* $a_l \in \mathbb{R}^{n \times n_0}$ that satisfy $\text{sign}(a_{ij}^{\text{start}}) = \text{sign}(a_{ij}^{\text{final}})$, where sign is the *signum function*, a_{ij}^{start} are positional entries of a_l at initialization and a_{ij}^{final} are its entries at the end of training. This metric was used in [73] to measure the "stability of activations" where high values indicate an effective linearization of the NN model. See additional details and remark in Appendix 4.C.1.

Influence of the regularization parameters. We investigate the performance of the GGN-SCORE-trained model for different values of the regularization smoothing parameter μ and the regularization strength τ on MNIST dataset. First, we fix $\mu = 1/\sqrt{n}$ and measure the performance of the trained model for different values of τ. Similarly, we fix $\tau = 10^{-4}$ and measure the trained model's performance for varying values of μ. In each of the two cases, we use 7 different values of τ and μ as is respectively shown in Figure **26** and Figure **28**. As in the case with synthetic

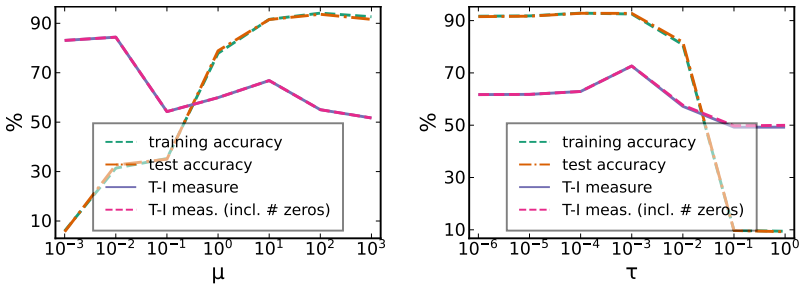


Figure 28: Evaluation of GGN-SCORE on MNIST dataset for different values of the regularization smoothing parameter μ, fixing $\tau = 10^{-4}$ (left), and different values of the regularization strength τ, fixing $\mu = 1/\sqrt{n}$ (right). The regularization function $g(\theta)$ is given by (4.15).

datasets, optimal choices for τ and μ are seen to necessarily yield good generalization of the model, and are such that give a relatively simple model and stable dynamics (as indicated by the number of zeros in the parameters of the final optimized model and the T-I measure). The total computation time to generate the results in Figure **26** and Figure **28** is ~ 23 hours, 5 minutes on CPU.

Acknowledgment

We acknowledge financial support from the Erasmus+ Programme of the European Union for the traineeship that contributed to the research presented in this chapter, facilitated by IMT Lucca and the University of Vienna.

Appendix for Chapter 4

4.A Preliminary results

4.A.1 Lipschitz continuity of J

Here, we look at the Lipschitz property of the Jacobian matrix J. For this, we need the following additional standard assumptions:

J.1 $\|x\| = \|x^\top\| \leq 1$.

J.2 $\exists L_\varrho$ such that $\|\varrho(\bar{x}) - \varrho(\tilde{x})\| \leq L_\varrho \|\bar{x} - \tilde{x}\|$ and $\|\varrho'(\bar{x}) - \varrho'(\tilde{x})\| \leq L_\varrho \|\bar{x} - \tilde{x}\|$ for all $\bar{x}, \tilde{x} \in \mathbb{R}$.

J.3 $\exists L_v$ such that $\|v\| \leq L_v$ at all time t until the training is stopped.

Condition J.2 simply restates the Lipschitzness and smoothness assumptions in A.1 explicitly. This kind of condition has been used, for example in [83, Condition 3.1], to show the stability of the training process of NNs via gradient descent. A consequence of the condition is that it also provides an upper bound on the gradients of ϱ, that is, $\|\varrho'(\bar{x})\| \leq L_\varrho$ for all $\bar{x} \in \mathbb{R}$ (see, e.g., Lemma 13 below).

Proposition 8 (Lipschitz constant of J; training both layers)**.** *Under assumptions J.1, J.2 and J.3, J is L_J-Lipschitz continuous, where* $L_J \triangleq m\kappa(n)(1 + L_v)L_\varrho\sqrt{2}$.

Proof. Let $(\bar{u}, \bar{v}) \equiv \bar{\theta}, (\tilde{u}, \tilde{v}) \equiv \tilde{\theta}$ for any $\bar{\theta}, \tilde{\theta} \in \mathbb{R}^p$. We have

$$
\begin{aligned}
&\left\|J(\bar{\theta}) - J(\tilde{\theta})\right\| \\
&\leq \kappa(n) \left(\sum_{i=1}^{n} \left| \varrho'(\bar{u}_i x) \bar{v}_i x^\top - \varrho'(\tilde{u}_i x) \tilde{v}_i x^\top \right| + \sum_{i=1}^{n} \left| \varrho(\bar{u}_i x) - \varrho(\tilde{u}_i x) \right| \right) \\
&\leq \kappa(n) \left(\sum_{i=1}^{n} \left(\left| \varrho'(\bar{u}_i x) \bar{v}_i x^\top - \varrho'(\tilde{u}_i x) \bar{v}_i x^\top \right| + \left| \varrho'(\tilde{u}_i x) \bar{v}_i x^\top - \varrho'(\tilde{u}_i x) \tilde{v}_i x^\top \right| \right) \right. \\
&\qquad \left. + \sum_{i=1}^{n} \left| \varrho(\bar{u}_i x) - \varrho(\tilde{u}_i x) \right| \right) \\
&\leq \kappa(n) \left(\sum_{i=1}^{n} \left(\left| \varrho'(\bar{u}_i x) - \varrho'(\tilde{u}_i x) \right| |\bar{v}_i| \|x\| + |\bar{v}_i - \tilde{v}_i| \left| \varrho'(\tilde{u}_i x) \right| \|x\| \right) \right. \\
&\qquad \left. + \sum_{i=1}^{n} \left| \varrho(\bar{u}_i x) - \varrho(\tilde{u}_i x) \right| \right) \\
&\leq m\kappa(n) \left((1 + L_v) L_\varrho \|\bar{u} - \tilde{u}\| + L_\varrho \|\bar{v} - \tilde{v}\| \right) \\
&\leq m\kappa(n)(1 + L_v) L_\varrho \left(\|\bar{u} - \tilde{u}\| + \|\bar{v} - \tilde{v}\| \right) \\
&= m\kappa(n)(1 + L_v) L_\varrho \left(\left\|(\bar{u}, 0) - (\tilde{u}, 0)\right\| + \left\|(0, \bar{v}) - (0, \tilde{v})\right\| \right). \qquad (4.16)
\end{aligned}
$$

Next, we recall Peter-Paul inequality for two quantities $a, b \in \mathbb{R}_{\geq 0}$ which reads $2ab \leq a^2 + b^2$, from which we obtain

$$
(a + b)^2 = a^2 + 2ab + b^2 \leq a^2 + a^2 + b^2 + b^2 = 2(a^2 + b^2). \qquad (4.17)
$$

We also derive the expression

$$
\begin{aligned}
&\left\|(\bar{u}, 0) - (\tilde{u}, 0)\right\|^2 + \left\|(0, \bar{v}) - (0, \tilde{v})\right\|^2 \\
&= \left\|(\bar{u}, 0)\right\|^2 - 2\langle (\bar{u}, 0), (\tilde{u}, 0) \rangle + \left\|(\tilde{u}, 0)\right\|^2 + \left\|(0, \bar{v})\right\|^2 - 2\langle (0, \bar{v}), (0, \tilde{v}) \rangle \\
&\quad + \left\|(0, \tilde{v})\right\|^2 \\
&= \left\|(\bar{u}, \bar{v})\right\|^2 - 2\langle (\bar{u}, \bar{v}), (\tilde{u}, \tilde{v}) \rangle + \left\|(\tilde{u}, \tilde{v})\right\|^2 \\
&= \left\|(\bar{u}, \bar{v}) - (\tilde{u}, \tilde{v})\right\|^2. \qquad (4.18)
\end{aligned}
$$

Now, using (4.17) and (4.18) in (4.16) with $a = \left\|(\bar{u}, 0) - (\tilde{u}, 0)\right\|$ and $b = \left\|(0, \bar{v}) - (0, \tilde{v})\right\|$, we get

$$\left\|J(\bar{\theta}) - J(\tilde{\theta})\right\| \leq m\kappa(n)(1 + L_v)L_\varrho\sqrt{2}\left\|(\bar{u}, \bar{v}) - (\tilde{u}, \tilde{v})\right\|,$$

which proves the result. ■

4.A.2 Lipschitz continuity of g

As is customary, our results are restricted to the case $\nu \in [2, 3]$, but they extend to other cases [223].

Lemma 13. *Let g be a convex and (locally) L-Lipschitz function. Then,*

$$\left\|\nabla g(\bar{\theta})\right\| \leq L,$$

for some $\bar{\theta}$ in a set $\mathcal{X} \subset \mathbb{R}^p$.

Proof. Take some $\tilde{\theta} = \bar{\theta} + \alpha \nabla g(\bar{\theta})$ for $\alpha \in \mathbb{R}_{>0}$ small enough. By the convexity and Lipschitzness of g, we have

$$\begin{aligned}
\left\|\alpha \nabla g(\bar{\theta})\right\|^2 &= \alpha^2 \left|\langle \nabla g(\bar{\theta}), \nabla g(\bar{\theta})\rangle\right| \\
&= \alpha \left|\langle \tilde{\theta} - \bar{\theta}, \nabla g(\bar{\theta})\rangle\right| \\
&\leq \alpha \left|g(\tilde{\theta}) - g(\bar{\theta})\right| \\
&\leq \alpha L \left|\tilde{\theta} - \bar{\theta}\right| \\
&= \alpha^2 L \left\|\nabla g(\bar{\theta})\right\|,
\end{aligned}$$

which completes the proof. ■

4.B Proof of the main result

Detailed regularity assumptions in R.1. We detail missing regularity terms in condition R.1 as follows. For this, we define the ball $\mathcal{B}_{r_0}(\theta_0) \subset \mathcal{E}_r(\theta_0)$ for some initialization θ_0, where $\mathcal{E}_r(\theta_0)$ is an ellipsoid defined for some $r \in \mathbb{R}_{>0}$. Note that, corresponding to $\mathcal{E}_r(\theta_0)$, we compute local norms with respect to g.

RR.1 $\hat{R}_s(\Phi(\cdot,\bar{\theta})) \geq \hat{R}_s(\Phi(\cdot,\tilde{\theta})) + \langle \nabla_\Phi \hat{R}_s(\Phi(\cdot,\tilde{\theta})), \Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\rangle + \frac{\gamma_R}{2}\left\|\Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\right\|^2 \ \forall \bar{\theta},\tilde{\theta} \in \mathbb{R}^p$, and $\exists B_R, D_R$ such that $\left\|\nabla_\Phi \hat{R}_s(\Phi(\cdot,\bar{\theta}))\right\| \leq B_R, \left\|\nabla^2_\Phi \hat{R}_s(\Phi(\cdot,\bar{\theta}))\right\|_{op} \leq D_R \ \forall \bar{\theta} \in \mathcal{B}_{r_0}(\theta_0)$.

RR.2 $d_g I \leq H_t \leq D_g I$ with $D_g \geq d_g \in \mathbb{R}_{>0}$, $d_q I \leq \hat{Q}_t \leq D_q I$ with $D_q \geq d_q \in \mathbb{R}_{\geq 0}$, and $\|\hat{e}_t\| \leq \beta \ \forall \theta_t \in \mathcal{B}_{r_0}(\theta_0)$.

Using the Lipschitness of ϱ, one can easily find some B_Φ satisfying $\left\|\Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\right\| \leq B_\Phi \left\|\bar{\theta} - \tilde{\theta}\right\|$ for some $\bar{\theta},\tilde{\theta} \in \mathbb{R}^p$ at least near the initialization. Hence, we do not impose this regularity property as an additional assumption. Subsequently, we recall the following notations: $L_{D_t} \triangleq \frac{\alpha_t \beta \hat{\beta}_1 D_g}{d_g(D_g + d_q \hat{\beta}_m^2)}$, $\xi \triangleq B_R B_\Phi + B_g$, $\vartheta \triangleq B_\Phi^2(\gamma_R - D_R)$, $\varpi_t \triangleq \omega_\nu(d_\nu(\theta_t,\theta_{t+1})) - \omega_\nu(-d_\nu(\theta_t,\theta_{t+1}))$, $\hat{\beta}_m \triangleq \sigma_{\min}(\hat{J}_t)$. We also introduce the notations $\hat{\beta}_1 \triangleq \sigma_{\max}(\hat{J}_t)$ and $\delta_t \triangleq \theta_{t+1} - \theta_t$.

Lemma 14. *Under assumption RR.2, we have*

$$\|\delta_t\| \leq L_{D_t}, \qquad \|\delta_t\|_{\theta_t} \leq \sqrt{D_g} L_{D_t}.$$

Proof. We obtain the following estimate

$$\left\|H_t^{-1}\hat{J}_t^\top\right\| \leq \left\|H_t^{-1}\right\|\left\|\hat{J}_t^\top\right\| \leq \frac{\beta_1}{d_g}. \tag{4.19}$$

We have also

$$\left\|I + \hat{Q}_t \hat{J}_t H_t^{-1} \hat{J}_t^\top\right\| \geq 1 + \frac{d_q \hat{\beta}_m^2}{D_g}. \tag{4.20}$$

From (4.9), we have

$$\|\delta_t\| \leq \alpha_t \left\|H_t^{-1}\hat{J}_t^\top (I + \hat{Q}_t \hat{J}_t H_t^{-1} \hat{J}_t^\top)^{-1} \hat{e}_t\right\|. \tag{4.21}$$

Using (4.19) and (4.20) in (4.21), we obtain

$$\|\delta_t\| \leq L_{D_t}.$$

The result follows, noting that $\|\delta_t\|_{\theta_t} = \left\|H_t^{1/2}\delta_t\right\|$ by definition. ∎

We obtain the following slightly loose estimate of the Lipschitz constant of the objective function $\mathcal{L}$ in problem (4.2).

Lemma 15. *Let g be constructed under the settings of Lemma 2 such that condition G.1 holds. Then, under the additional condition RR.1, the objective function $\mathcal{L}$ is $(B_R B_\Phi + B_g)$-Lipschitz continuous, where B_g is the local Lipschitz constant of g in Lemma 2(iii).*

Proof. By the convexity of $\hat{R}_s$ and g, we have

$$\hat{R}_s(\Phi(\cdot,\tilde{\theta})) \geq \hat{R}_s(\Phi(\cdot,\bar{\theta})) + \langle \nabla \hat{R}_s(\Phi(\cdot,\tilde{\theta})), \Phi(\cdot,\tilde{\theta}) - \Phi(\cdot,\bar{\theta})\rangle,$$
$$g(\tilde{\theta}) \geq g(\bar{\theta}) + \langle \nabla g(\tilde{\theta}), \tilde{\theta} - \bar{\theta}\rangle,$$

for some $\tilde{\theta}, \bar{\theta}$ in the vicinity of θ_0. Then, using $\mathcal{L} \triangleq \hat{R}_s + g$ and the Cauchy-Schwarz inequality, we get

$$\left|\mathcal{L}(\bar{\theta}) - \mathcal{L}(\tilde{\theta})\right| \leq \left\|\nabla_\Phi \hat{R}_s(\Phi(\cdot,\bar{\theta}))\right\| \left\|\Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\right\| + \left\|\nabla g(\bar{\theta})\right\| \left\|\bar{\theta} - \tilde{\theta}\right\|.$$

By assumption RR.1, we have $\left\|\nabla_\Phi \hat{R}_s(\Phi(\cdot,\bar{\theta}))\right\| \leq B_R$. By Lemma 2(iii), g is locally Lipschitz, and hence we have $\left\|\nabla g(\bar{\theta})\right\| \leq B_g$ for some B_g, according to Lemma 13. Then, using the local Lipschitz property of Φ, we obtain

$$\begin{aligned}\left|\mathcal{L}(\bar{\theta}) - \mathcal{L}(\tilde{\theta})\right| &\leq B_R B_\Phi \left\|\bar{\theta} - \tilde{\theta}\right\| + B_g \left\|\bar{\theta} - \tilde{\theta}\right\| \\ &= \xi \left\|\bar{\theta} - \tilde{\theta}\right\|.\end{aligned}$$

■

The following result from [236, Lemma 1] provides a useful inequality for the trace of the product of two symmetric matrices, one of which is positive semidefinite.

Lemma 16. *Let $P, Q \in \mathbb{R}^{n\times n}$. If $P = P^\top \succeq 0$ and Q is symmetric, then*

$$\operatorname{tr}(P)\lambda_n(Q) \leq \operatorname{tr}(PQ) \leq \operatorname{tr}(P)\lambda_1(Q).$$

The next result concerns the 2×2 block partitioning of $\tilde{G}_t$, and characterizes the positive-definiteness of its leading principal blocks. For this, we require that the function g is such that $H_t \succ 0$. This, indeed, is a

property of many functions constructed from the ℓ_1-norm in the sense of Definition 6. An example is the pseudo-Huber function or the function $\bar{g}$ considered in Section 4.4.

Lemma 17. *Consider a 2×2 block partitioning of $\tilde{G}_t$, and let $\tilde{G}_{11} \in \mathbb{R}^{m \times m}$, $\tilde{G}_{22} \in \mathbb{R}^{1 \times 1}$ respectively denote the upper left and lower right blocks. If $H_t \succ 0$, then it holds that $\tilde{G}_{22} \in \mathbb{R}_{>0}$ and $\tilde{G}_{11} \succ 0$.*

Proof. By the definition of $\tilde{G}_t$ and using (4.8), we have $\tilde{G}_t = \tilde{J}_t(\hat{J}_t^\top \hat{Q}_t \hat{J}_t + H_t)^{-1} \hat{J}_t^\top$. We note that for the squared loss that we consider, Q_t is the identity matrix and that we can write $\tilde{G}_t = \tilde{J}_t(J_t^\top Q_t J_t + H_t)^{-1} \hat{J}_t^\top = \tilde{J}_t(J_t^\top J_t + H_t)^{-1} \hat{J}_t^\top$. Notice the removal of the augmentations, as the last diagonal entry of $\hat{Q}_t$ is zero. We have $\langle \hat{v}, J_t^\top J_t \hat{v} \rangle = \|J\hat{v}\|^2 \in \mathbb{R}_{\geq 0}$ for all non-zero $\hat{v} \in \mathbb{R}^p$, and hence $B_t \triangleq J_t^\top J_t + H_t \succ 0$. Next, observe that $\tilde{G}_{11}$ results from removing the augmentations on $\tilde{J}$ and $\hat{Q}_t$ in $\tilde{G}_t$, that is, $\tilde{G}_{11} \equiv J_t B_t^{-1} J_t^\top$. Let $\hat{u} \triangleq B_t \hat{v}$; we have $\langle \hat{u}, B_t^{-1} \hat{u} \rangle = \langle B_t \hat{v}, B_t^{-1} B_t \hat{v} \rangle = \langle \hat{v}, B_t^\top \hat{v} \rangle \succ 0$. Then, in a similar way, if J_t does not have all its entries equal to zero, we get that $\tilde{G}_{11} \succ 0$.

To show $\tilde{G}_{22} \in \mathbb{R}_{>0}$, we note that since $B_t \succ 0$, it has a non-zero determinant, and hence by Sylvester's criterion, we have $\frac{1}{\det(B_t)}((-1)^{2p} M_{p,p}) > 0$, where $\det(B_t)$ denotes the determinant of B_t and $M_{p,p}$ denotes the (p,p)-th minor of B_t. Then, $\tilde{G}_{22} \in \mathbb{R}_{>0}$ follows from the definition of $\tilde{J}$. ■

We are now ready to prove our main result.

Proof of Theorem 6.

Proof. Consider the time evolution of the regularized NN given by (4.10). Using the augmentation specified by (4.13), we have

$$\begin{aligned}\left\|\tilde{\Phi}_{t+1} - \tilde{\Phi}^*\right\|^2 &= \left\|\tilde{\Phi}_t - \alpha_t \tilde{G}_t \hat{e}_t - \tilde{\Phi}^*\right\|^2 \\ &= \left\|\tilde{\Phi}_t - \tilde{\Phi}^*\right\|^2 - 2\alpha_t \langle \tilde{\Phi}_t - \tilde{\Phi}^*, \tilde{G}_t(\tilde{\Phi}_t - \tilde{\Phi}^*) \rangle + \alpha_t^2 \left\|\tilde{G}_t(\tilde{\Phi}_t - \tilde{\Phi}^*)\right\|^2 .\end{aligned} \tag{4.22}$$

Let us partition $\tilde{\Phi}_t - \tilde{\Phi}^*$ and $\tilde{G}_t$ as follows (omitting dependence on t in the blocks for brevity):

$$\tilde{\Phi}_t - \tilde{\Phi}^* \equiv \left[\begin{array}{c} \tilde{\Phi}_1 \\ \hdashline \tilde{\Phi}_2 \end{array}\right], \quad \tilde{G}_t \equiv \left[\begin{array}{c:c} \tilde{G}_{11} & \tilde{G}_{12} \\ \hdashline \tilde{G}_{21} & \tilde{G}_{22} \end{array}\right], \tag{4.23}$$

where $\tilde{\Phi}_1 = \Phi_t - \Phi^* \in \mathbb{R}^m, \tilde{\Phi}_2 = 1$ and hence $\tilde{G}_{11} \in \mathbb{R}^{m\times m}$. Then, we have

$$\begin{aligned}
&\langle \tilde{\Phi}_t - \tilde{\Phi}^*, \tilde{G}_t(\tilde{\Phi}_t - \tilde{\Phi}^*)\rangle \\
&\quad = \langle \tilde{\Phi}_1, \tilde{G}_{11}\tilde{\Phi}_1\rangle + \langle \tilde{\Phi}_2, \tilde{G}_{21}\tilde{\Phi}_1\rangle + \langle \tilde{\Phi}_1, \tilde{G}_{12}\tilde{\Phi}_2\rangle + \langle \tilde{\Phi}_2, \tilde{G}_{22}\tilde{\Phi}_2\rangle \\
&\quad = \langle \tilde{\Phi}_1, \tilde{G}_{11}\tilde{\Phi}_1\rangle + \langle \tilde{G}_{21} + \tilde{G}_{12}^\top, \tilde{\Phi}_1\rangle + \tilde{G}_{22},
\end{aligned} \tag{4.24}$$

where we have used $\tilde{\Phi}_2 = 1$. Recall that by Lemma 17, we get $\tilde{G}_{22} \in \mathbb{R}_{>0}$ and $\tilde{G}_{11} \succ 0$.

Using the block partitioning of $\tilde{G}_t$ in (4.23), the product $\tilde{G}_t^\top \tilde{G}_t$ gives the following block structure

$$\begin{aligned}
\tilde{G}_t^\top \tilde{G}_t &= \left[\begin{array}{c:c} (\tilde{G}_t^\top \tilde{G}_t)_{11} & (\tilde{G}_t^\top \tilde{G}_t)_{12} \\ \hdashline (\tilde{G}_t^\top \tilde{G}_t)_{21} & (\tilde{G}_t^\top \tilde{G}_t)_{22} \end{array}\right] \\
&\triangleq \left[\begin{array}{c:c} \tilde{G}_{11}^\top \tilde{G}_{11} + \tilde{G}_{21}^\top \tilde{G}_{21} & \tilde{G}_{11}^\top \tilde{G}_{12} + \tilde{G}_{21}^\top \tilde{G}_{22} \\ \hdashline \tilde{G}_{12}^\top \tilde{G}_{11} + \tilde{G}_{22}^\top \tilde{G}_{21} & \tilde{G}_{12}^\top \tilde{G}_{12} + \tilde{G}_{22}^\top \tilde{G}_{22} \end{array}\right].
\end{aligned}$$

Consider the congruence

$$\begin{aligned}
\left[\begin{array}{c:c} (\tilde{G}_t^\top \tilde{G}_t)_{11} & (\tilde{G}_t^\top \tilde{G}_t)_{12} \\ \hdashline (\tilde{G}_t^\top \tilde{G}_t)_{21} & (\tilde{G}_t^\top \tilde{G}_t)_{22} \end{array}\right] &\sim \left[\begin{array}{c:c} (\tilde{G}_t^\top \tilde{G}_t)_{11}^{-1/2} & 0 \\ \hdashline 0 & (\tilde{G}_t^\top \tilde{G}_t)_{22}^{-1/2} \end{array}\right] \\
&\quad \times \left[\begin{array}{c:c} (\tilde{G}_t^\top \tilde{G}_t)_{11} & (\tilde{G}_t^\top \tilde{G}_t)_{12} \\ \hdashline (\tilde{G}_t^\top \tilde{G}_t)_{21} & (\tilde{G}_t^\top \tilde{G}_t)_{22} \end{array}\right] \times \left[\begin{array}{c:c} (\tilde{G}_t^\top \tilde{G}_t)_{11}^{-1/2} & 0 \\ \hdashline 0 & (\tilde{G}_t^\top \tilde{G}_t)_{22}^{-1/2} \end{array}\right] \\
&\quad = \left[\begin{array}{c:c} I & \boldsymbol{A} \\ \hdashline \boldsymbol{B} & I \end{array}\right],
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{A} &= (\tilde{G}_t^\top \tilde{G}_t)_{11}^{-1/2} (\tilde{G}_t^\top \tilde{G}_t)_{12} (\tilde{G}_t^\top \tilde{G}_t)_{22}^{-1/2}, \\
\boldsymbol{B} &= (\tilde{G}_t^\top \tilde{G}_t)_{22}^{-1/2} (\tilde{G}_t^\top \tilde{G}_t)_{21} (\tilde{G}_t^\top \tilde{G}_t)_{11}^{-1/2}.
\end{aligned}$$

Using this relation, one can show that $\tilde{G}_t^\top \tilde{G}_t \succ 0$; since $(\tilde{G}_t^\top \tilde{G}_t)_{21} = (\tilde{G}_t^\top \tilde{G}_t)_{12}^\top$, we only require that $\|\boldsymbol{A}\| \leq 1$. Now, let $1 + M_g \eta_t \leq \|\tilde{G}_t\|_F$ and $|\langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{v}_t \rangle| \leq |\tilde{G}_{22}|$ for some $\tilde{v}_t$ where, given a 2×2 block partitioning of $\tilde{G}_t$, $\tilde{G}_{22}$, $\tilde{G}_{21}$ and $\tilde{G}_{12}$, respectively denote the lower right, lower left and upper right blocks. We now invoke Lemma 16 and obtain

$$\begin{aligned}\left\|\tilde{G}_t(\tilde{\Phi}_t - \tilde{\Phi}^*)\right\|^2 &= \operatorname{tr}(\tilde{G}_t^\top \tilde{G}_t(\tilde{\Phi}_t - \tilde{\Phi}^*)(\tilde{\Phi}_t - \tilde{\Phi}^*)^\top) \\ &\leq \operatorname{tr}(\tilde{G}_t^\top \tilde{G}_t)\lambda_1((\tilde{\Phi}_t - \tilde{\Phi}^*)(\tilde{\Phi}_t - \tilde{\Phi}^*)^\top) \\ &= \operatorname{tr}(\tilde{G}_t^\top \tilde{G}_t)\left\|\tilde{\Phi}_t - \tilde{\Phi}^*\right\|^2. \end{aligned} \tag{4.25}$$

Using (4.24) and (4.25) in (4.22), we have

$$\begin{aligned}\left\|\tilde{\Phi}_{t+1} - \tilde{\Phi}^*\right\|^2 \leq \left\|\tilde{\Phi}_t - \tilde{\Phi}^*\right\|^2 - 2\alpha_t \left\|\tilde{G}_{11}^{1/2}\tilde{\Phi}_1\right\|^2 - 2\alpha_t \langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle \\ - 2\alpha_t \tilde{G}_{22} + \alpha_t^2 \operatorname{tr}(\tilde{G}_t^\top \tilde{G}_t)\left\|\tilde{\Phi}_t - \tilde{\Phi}^*\right\|^2.\end{aligned}$$

Now, using the inequality $-\left|\langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle\right| \leq \langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle \leq \left|\langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle\right|$, we get that

$$-\langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle \leq \left|\langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle\right|,$$

and then,

$$\begin{aligned}\left\|\tilde{\Phi}_{t+1} - \tilde{\Phi}^*\right\|^2 \leq \left\|\tilde{\Phi}_t - \tilde{\Phi}^*\right\|^2 - 2\alpha_t \left\|\tilde{G}_{11}^{1/2}\tilde{\Phi}_1\right\|^2 + 2\alpha_t \left|\langle \tilde{G}_{21}^\top + \tilde{G}_{12}, \tilde{\Phi}_1 \rangle\right| \\ - 2\alpha_t \tilde{G}_{22} + \alpha_t^2 \operatorname{tr}(\tilde{G}_t^\top \tilde{G}_t)\left\|\tilde{\Phi}_t - \tilde{\Phi}^*\right\|^2.\end{aligned}$$

Setting $\tilde{v} = \tilde{\Phi}_1$ in the condition $|\tilde{G}_{22}| \geq |\langle \tilde{G}_{21} + \tilde{G}_{12}^\top, \tilde{v} \rangle|$, it holds that $|\tilde{G}_{22}| > |\langle \tilde{G}_{21} + \tilde{G}_{12}^\top, \tilde{\Phi}_1 \rangle| - C_1$ for any arbitrary constant $C_1 > 0$. Set $C_1 = \frac{1}{\alpha_t C_2}$ for some constant $C_2 > 0$, noting that $\alpha_t > 0$ for all t, then we get

$$-\tilde{G}_{22} \leq -\left|\langle \tilde{G}_{21} + \tilde{G}_{12}^\top, \tilde{\Phi}_1 \rangle\right| + \frac{1}{\alpha_t C_2}.$$

Consequently,

$$\left\|\tilde{\Phi}_{t+1}-\tilde{\Phi}^*\right\|^2 \leq \left\|\tilde{\Phi}_t-\tilde{\Phi}^*\right\|^2 - 2\alpha_t\left\|\tilde{G}_{11}^{1/2}\tilde{\Phi}_1\right\|^2 + \frac{2}{C_2} + \alpha_t^2\,\mathrm{tr}(\tilde{G}_t^\top\tilde{G}_t)\left\|\tilde{\Phi}_t-\tilde{\Phi}^*\right\|^2.$$

Now, if the condition $1+M_g\eta_t \leq \|\tilde{G}_t\|_F$ is such that

$$\frac{\left\|\tilde{G}_{11}^{1/2}\tilde{\Phi}_1\right\|^2}{\mathrm{tr}(\tilde{G}_t^\top\tilde{G}_t)\left\|\tilde{\Phi}_t-\tilde{\Phi}^*\right\|^2} \geq \alpha_t \triangleq \frac{\bar{\alpha}_t}{1+M_g\eta_t} \geq \frac{\bar{\alpha}_t}{\left\|\tilde{G}_t\right\|_F} \equiv \frac{\bar{\alpha}_t}{\sqrt{\mathrm{tr}(\tilde{G}_t^\top\tilde{G}_t)}},$$

by fixing $0<\bar{\alpha}_t\equiv\bar{\alpha}<1$, then

$$\left\|\tilde{\Phi}_{t+1}-\tilde{\Phi}^*\right\|^2 \leq (1-\bar{\alpha})\left\|\tilde{\Phi}_t-\tilde{\Phi}^*\right\|^2 + \frac{2}{C_2}. \tag{4.26}$$

The recurrence in (4.26) can be expanded as follows:

$$\begin{aligned}
&\left\|\tilde{\Phi}_{t+1}-\tilde{\Phi}^*\right\|^2 \leq (1-\bar{\alpha})\left\|\tilde{\Phi}_t-\tilde{\Phi}^*\right\|^2 + \frac{2}{C_2}\\
&\leq (1-\bar{\alpha})\left((1-\bar{\alpha})\left\|\tilde{\Phi}_{t-1}-\tilde{\Phi}^*\right\|^2 + \frac{2}{C_2}\right) + \frac{2}{C_2}\\
&\leq (1-\bar{\alpha})\left((1-\bar{\alpha})\left((1-\bar{\alpha})\left\|\tilde{\Phi}_{t-2}-\tilde{\Phi}^*\right\|^2 + \frac{2}{C_2}\right) + \frac{2}{C_2}\right) + \frac{2}{C_2}\\
&= (1-\bar{\alpha})^3\left\|\tilde{\Phi}_{t-2}-\tilde{\Phi}^*\right\|^2 + (1-\bar{\alpha})^2\frac{2}{C_2} + (1-\bar{\alpha})\frac{2}{C_2} + \frac{2}{C_2},
\end{aligned}$$

and so on. This gives, for any $T\geq 1$,

$$\left\|\tilde{\Phi}_T-\tilde{\Phi}^*\right\|^2 \leq (1-\bar{\alpha})^T\left\|\tilde{\Phi}_0-\tilde{\Phi}^*\right\|^2 + \frac{2}{C_2}\sum_{j=0}^{T-1}(1-\bar{\alpha})^{T-j-1}. \tag{4.27}$$

Since $C_2>0$ is arbitrary, we set $C_2 = 2\sum_{j=0}^{T-1}(1-\bar{\alpha})^{T-j-1}$. We also have that since $\bar{\alpha}>0$, it satisfies the inequality $1-\bar{\alpha}\leq\exp(-\bar{\alpha})$. Then (4.27) gives

$$\left\|\tilde{\Phi}_T-\tilde{\Phi}^*\right\|^2 \leq \exp(-\bar{\alpha}T)\left\|\tilde{\Phi}_0-\tilde{\Phi}^*\right\|^2 + 1. \tag{4.28}$$

Substituting our choice of T into (4.28) gives

$$\left\|\tilde{\Phi}_T - \tilde{\Phi}^*\right\|^2 \leq \epsilon + 1,$$

which is result P.1.

To prove P.2, we first notice that the local condition $\left\|\nabla^2{}_\Phi \hat{R}_s(\Phi)\right\|_{op} \leq D_R$ in RR.1 implies local D_R-Lipschitz continuity of $\nabla_\Phi \hat{R}_s(\Phi)$ with respect to Φ, that is, for $\bar{\theta}, \tilde{\theta}$ around the initialization, we have

$$\left\|\nabla \hat{R}_s(\Phi(\cdot,\bar{\theta})) - \nabla \hat{R}_s(\Phi(\cdot,\tilde{\theta}))\right\| \leq D_R \left\|\Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\right\|,$$

or equivalently,

$$\begin{aligned}\hat{R}_s(\Phi(\cdot,\bar{\theta})) \leq \hat{R}_s(\Phi(\cdot,\tilde{\theta})) &+ \langle \nabla \hat{R}_s(\Phi(\cdot,\tilde{\theta})), \Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\rangle \\ &+ \frac{D_R}{2}\left\|\Phi(\cdot,\bar{\theta}) - \Phi(\cdot,\tilde{\theta})\right\|^2. \end{aligned} \tag{4.29}$$

We recall the notation $\Phi_t \triangleq \Phi(\cdot,\theta_t)$ for all $t \in \mathbb{R}_{\geq 0}$. Then, using $\mathcal{L} \triangleq \hat{R}_s + g$, Lemma 9, and (4.29), we get

$$\begin{aligned}\mathcal{L}(\theta_{t+1}) &\leq \mathcal{L}(\theta_t) + \langle \nabla \hat{R}_s(\Phi_t), \Phi_{t+1} - \Phi_t\rangle + \frac{D_R}{2}\|\Phi_{t+1} - \Phi_t\|^2 \\ &\quad + \langle \nabla g(\theta_t), \theta_{t+1} - \theta_t\rangle + \omega_\nu(d_\nu(\theta_t,\theta_{t+1}))\|\theta_{t+1} - \theta_t\|_{\theta_t}^2 \\ &\leq \mathcal{L}(\theta_t) - \frac{\gamma_R}{2}\|\Phi_{t+1} - \Phi_t\|^2 - \mathcal{L}(\theta_t) + \mathcal{L}(\theta_{t+1}) \\ &\quad - \omega_\nu(-d_\nu(\theta_t,\theta_{t+1}))\|\theta_{t+1} - \theta_t\|_{\theta_t}^2 + \frac{D_R}{2}\|\Phi_{t+1} - \Phi_t\|^2 \\ &\quad + \omega_\nu(d_\nu(\theta_t,\theta_{t+1}))\|\theta_{t+1} - \theta_t\|_{\theta_t}^2.\end{aligned}$$

Using the γ_R-strong convexity assumption on $\hat{R}$ in RR.1 and the Lipschitz property of $\mathcal{L}$ in Lemma 15, this gives

$$\begin{aligned}\mathcal{L}(\theta_{t+1}) &\leq \mathcal{L}(\theta_t) + \xi\|\theta_{t+1} - \theta_t\| + \frac{D_R - \gamma_R}{2}\|\Phi_{t+1} - \Phi_t\|^2 \\ &\quad + \big(\omega_\nu(d_\nu(\theta_t,\theta_{t+1})) - \omega_\nu(-d_\nu(\theta_t,\theta_{t+1}))\big)\|\theta_{t+1} - \theta_t\|_{\theta_t}^2 \\ &\leq \mathcal{L}(\theta_t) + \xi\|\theta_{t+1} - \theta_t\| + \frac{B_\Phi^2(D_R - \gamma_R)}{2}\|\theta_{t+1} - \theta_t\|^2 \\ &\quad + \big(\omega_\nu(d_\nu(\theta_t,\theta_{t+1})) - \omega_\nu(-d_\nu(\theta_t,\theta_{t+1}))\big)\|\theta_{t+1} - \theta_t\|_{\theta_t}^2. \end{aligned} \tag{4.30}$$

Recalling the notation $\delta_t \triangleq \theta_{t+1} - \theta_t$ and substituting the estimates on $\|\delta_t\|$ and $\|\delta_t\|_{\theta_t}$ from Lemma 14 into (4.30) yields result P.2. ■

4.C Additional experimental details and results

4.C.1 Remark on the T-I measure

The time-invariance measure provides a way to measure stability of the optimizer's dynamics from initialization. However, since the signum function does not account for indices (i, j) of a_l with $a_{ij} = 0$, i.e.,

$$\operatorname{sign}(a_{ij}) \triangleq \begin{cases} +1 & \text{if } a_{ij} > 0, \\ -1 & \text{if } a_{ij} < 0, \end{cases}$$

and, as we have seen, the GGN-SCORE framework potentially produces many of this instance (with $a_{ij} = 0$) to reduce the model's complexity and/or improve generalization, a natural question is what state should be assumed for neuron a_{ij} when it is exactly zero. For this, we follow the standard convention that if $a_{ij} = 0$, then the (i, j)-th neuron remains unchanged from its initial state [111, Section 13.7]. Under this convention, the proportion of the indices (i, j) of a_l^{final} satisfying $\operatorname{sign}(a_{ij}^{\text{start}}) \neq \operatorname{sign}(a_{ij}^{\text{final}})$ with $a_{ij}^{\text{final}} = 0$ contribute to the stability of activations, and hence should be accounted for in the T-I measure. However, this contribution appear to be insignificant for the values of τ and μ that give the best test accuracies. From what we observe in Figure **26** and Figure **28**, proper choices of μ and τ reliably produces stable dynamics of the optimizer as well as a good generalization of the final trained model.

4.C.2 MNIST teacher-student setting

In order to evaluate GGN-SCORE on the MNIST dataset such that we are close to the theoretical framework, we consider a teacher-student setup for the MNIST dataset in a similar way as Arbel [13, Appendix C.4]:

- We create a custom training dataset by combining the original MNIST test dataset (containing 10000 sample points) and a balanced subset of the original training dataset. This balanced subset is created by "undersampling" the first 3000 samples of the original training dataset to give 2610 sample points. In total, the custom training dataset contains 12610 sample points.

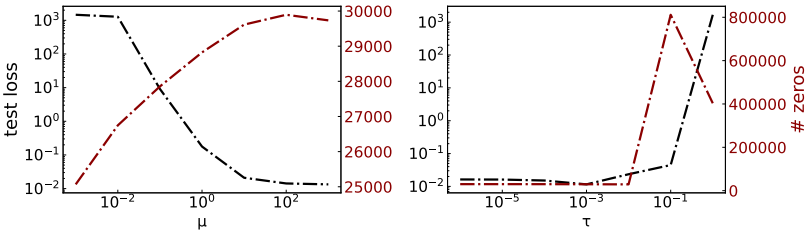


Figure 29: Test loss of the GGN-SCORE-trained NN on MNIST dataset (teacher-student) with $g(\theta)$ given by (4.15). Left: Results for different values of the regularization smoothing parameter μ with $\tau = 10^{-4}$ fixed. Right: Results for different values of the regularization strength τ with $\mu = 1/\sqrt{n}$ fixed.

- We then train a teacher NN Φ^* of the form (4.14) and hidden size $n^* = 16$ on this training dataset with the cross-entropy loss function and the SiLU activation function.

- A training "target" dataset is created from Φ^* (with the softmax function applied on each output of Φ^*).

- The student NN of the form (4.3) with the SiLU activation and hidden size $n = 1024$ is then trained on the custom training input samples and their corresponding target samples constructed from Φ^*. The trained student NN is tested on the original MNIST test dataset.

The training and test results are displayed Figure **29** and Figure **30**. We follow a similar evaluation procedure as in Section 4.4.2, i.e., the results are evaluated on the basis of the test loss, training and test accuracy, and T-I measure of the trained student NN. Interestingly, similar observations as in Section 4.4.2 are made from the displayed results. The total computation time to generate the results in Figure **29** and Figure **30** is ~ 8 hours, 39 minutes on CPU.

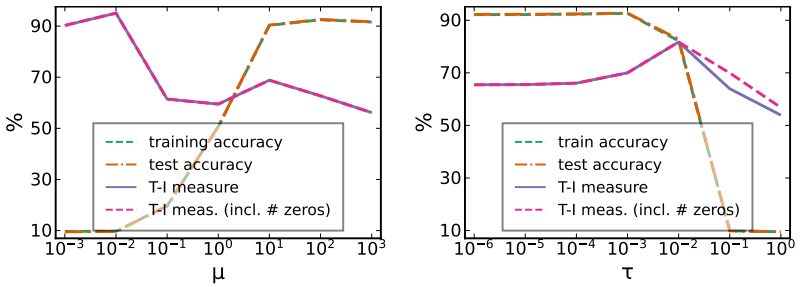


Figure 30: Accuracy and T-I measure of the GGN-SCORE-trained NN on MNIST dataset (teacher-student) for different values of μ (left) and different values of τ (right), with the regularization function $g(\theta)$ given by (4.15). In the left figure, $\tau = 10^{-4}$ is used. In the right figure, $\mu = 1/\sqrt{n}$ is used.

4.C.3 FashionMNIST experiments

We perform experiments on the FashionMNIST dataset [241] under the same setting as the MNIST experiments in Section 4.4.2. While the FashionMNIST classification tends to be a harder task than the MNIST, results shown in Figure **31** and Figure **32** indicate similar behaviours as those described in Section 4.4.2 regarding the influence of the regularization parameters.

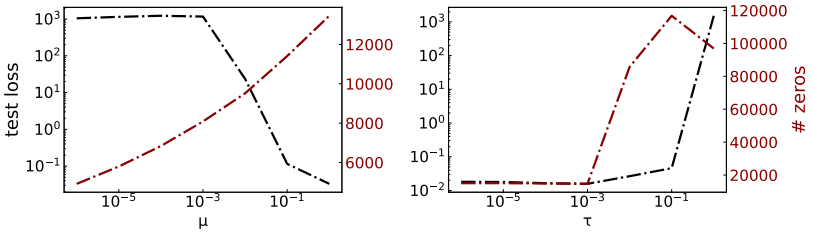


Figure 31: Test loss evaluation of the GGN-SCORE-trained NN on FashionMNIST dataset for different values of the regularization smoothing parameter μ fixing $\tau = 10^{-4}$ (left) and different values of the regularization strength τ fixing $\mu = 1/\sqrt{n}$ (right), where the regularization function $g(\theta)$ is given by (4.15).

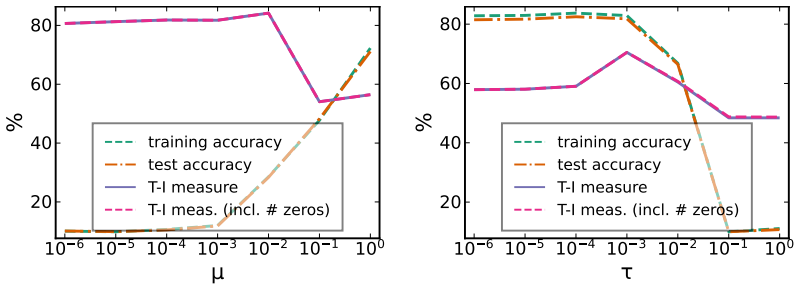


Figure 32: Evaluation of GGN-SCORE on FashionMNIST dataset for different values of the regularization strength μ (left) and different values of the regularization smoothing parameter τ (right). The regularization function $g(\theta)$ is given by (4.15). In the left figure, $\mu = 1/\sqrt{n}$ is used. In the right figure, $\tau = 10^{-4}$ is used.

	Num. of samples			
Dataset	Training	Test	Input dim.	Num. of classes
pendigits	7494	3498	16	10
letter	10500	5000	16	26
avila	10430	10437	11	12

Table 5: Summary of UCI datasets used for comparison.

			T-I measure (%)		T-I meas. incl. $a_{ij} = 0$ (%)	
Dataset	Batch-size	μ	GD	GGN-SCORE	GD	GGN-SCORE
pendigits	8	$0.001/\sqrt{n}$	50.066	52.7331	50.1041	52.7331
letter	64	$10/\sqrt{n}$	55.9237	55.2016	55.925	55.2112
avila	64	$10/\sqrt{n}$	73.318	71.4815	73.3189	71.4833

Table 6: Stability of activation measure.

4.C.4 Generalization and stability in comparison with GD

We now compare GGN-SCORE with GD on three UCI benchmark datasets[3]: pendigits, letter, and avila, summarized in Table 5. As in Section 4.4, we use a learning rate of 1 for GD, and set the hidden size

[3] https://archive.ics.uci.edu.

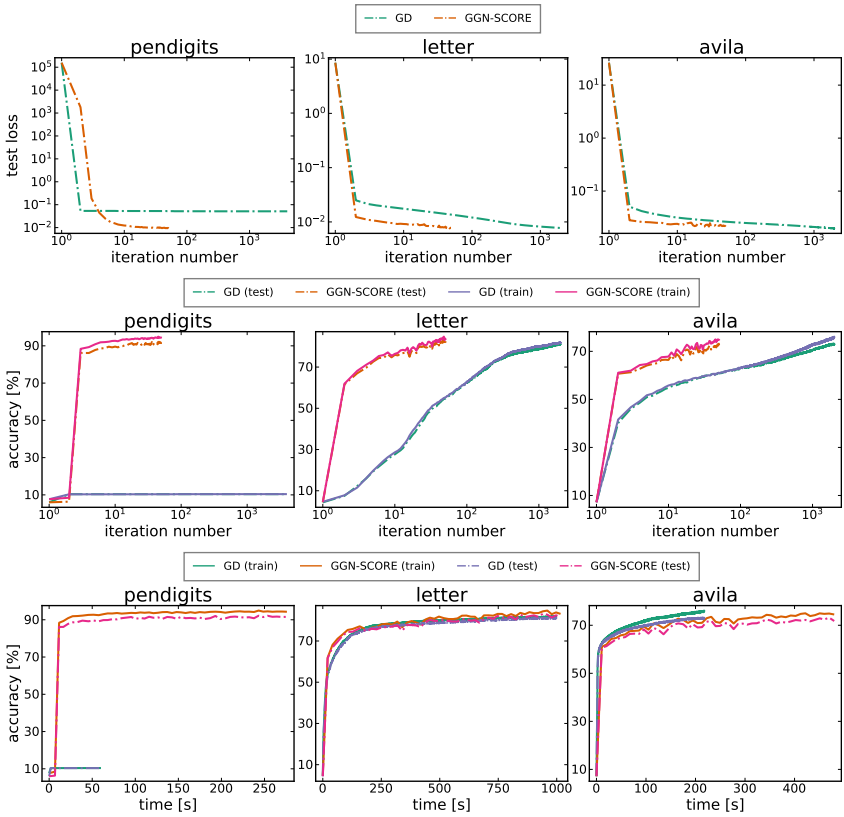


Figure 33: Test loss and accuracy evolution for GD and GGN-SCORE on pendigits, letter and avila datasets. The regularization function $g(\theta)$ in GGN-SCORE is given by (4.15) with $\tau = 10^{-4}$ and μ given in Table **6**.

$n = 128$ in all the experiments for a NN of the form (4.3), and a scaling $\kappa(n) = 1/\sqrt{n}$. The function g in GGN-SCORE is given by (4.15) with $\tau = 10^{-4}$. The results are shown Figure **33** and Table **6**. We observe faster convergence and better generalization in most cases for GGN-SCORE, and as in the case for the full-batch deterministic setting in Section 4.4 on synthetic datasets, we achieve this performance in faster time compared to GD. Note that much of the computational burden associated with the

regularized GGN is greatly reduced by using the stylized expression (4.9), since the mini-batch size is typically much smaller than p, the size of the optimization variable θ.

Chapter 5

An inexact sequential quadratic programming method for learning and control of recurrent neural networks

5.1 Introduction

Approximate dynamic programming or reinforcement learning (RL) involves an *agent* or *decision maker* that interacts with its *environment* based on the *states* of the environment observable by the agent [111, 224]. For each decision that the agent makes, it incurs a cost or, alternatively, gets a reward. Hence, the ultimate goal of the agent is to minimize the total costs, or maximize the sum of its rewards, received from the environment. In most cases, it is assumed that the operations of the agent are in accordance with a finite discrete-time *Markovian decision process* (MDP), which in turn assumes that the states of the environment are fully observable by the agent at any given time t, providing all necessary information that the agent can use to decide what action to take. This is not always the case

in reality, especially as the environment is mostly complex and its model is unknown. Hence, many practical RL problems are solved within the framework of *partially observable Markov decision process* (POMDP). In line with control problems, the environment is presented as the state-space model of a dynamical system that encodes the spatial and temporal information about the system which can be used to predict its future behaviour. With the popularity of deep learning methods [130], the paradigm of world model [107] has been recently used to describe RL methods in which a generative and/or predictive model is trained to represent the agent's own imagination of the environment. A control neural network which the agent uses to make decisions and take actions on-line is thereafter trained off-line on top of the representative model. This approach encompasses an idea that dates back as far as the 90s where mostly RNNs are trained to build a predictive model of the environment or complex system [208, 209, 210, 206], helping to possibly develop a meaningful representation of the *Markovian state-space* in the face of *partial observability* [206]. Data-efficiency becomes an added advantage in the design of the representative RNN model, which is a key element of model-based RL and control methods [224]. These so-called *recurrent control networks* [210] are mostly designed with much focus on their architecture that enable them to capture relevant information about the underlying system dynamics. In [206], the authors introduce the recurrent control neural network (RCNN) with an architectural design motivated by the two stages of a typical neural network based complex control method, namely: (1) a system identification stage where a RNN with dynamically consistent overshooting [252] is designed to train a state-space model of the system using process measurements, and (2) an optimal control policy selection stage where a FNN is concatenated with the RNN to learn an optimal control policy which is later used to control the real process.

This unified approach largely benefits from the approximation power of the individual neural networks when trained with an appropriate gradient-based algorithm. Besides their slow convergence, a drawback in learning RNNs for practical applications with gradient-based algorithms is the *possible* problem of *vanishing- and exploding-gradients* [36, 116], which

may lead to their difficulty in learning long-term dependencies, as their operations rely only upon the slope of the loss surface in the Jacobian. Second-order methods, however, can be used to mitigate these drawbacks by utilizing in their learning procedures the information contained in the curvature of the loss surface, which provably accelerates convergence. Second-order methods for learning RNNs have historically been used in two ways: (1) the use of second-order optimization algorithms such as generalized Gauss-Newton (GGN), Levenberg-Marquardt (LM) and conjugate gradient (CG) algorithms (see, e.g., [205, 233, 150, 64, 154, 99, 66, 34]), and (2) using nonlinear sequential state-estimation techniques such as extended Kalman filter (EKF) method (see, e.g., [144, 239, 187, 237, 92, 33]). In the first approach, the second-order information is captured through Hessian (or approximate Hessian) computations, while in the second approach the second-order information is computed recursively as a prediction-error covariance matrix. In any event, these approaches provide ways to capture relevant second-order information about the training loss function as well as help to curtail the possible vanishing/exploding-gradient problem. As the strength of data-based RL and control methods lies in their data-efficiency, it is further desirable to capture curvature information about the neural network training data for better approximation and representation capability within the allotted training time. In control applications, recovering a good representation of the underlying system is very important for robustness and reliability due to better generalization capabilities of the model to situations unseen before. For this reason, many recent works have focused on designing custom training algorithms for their application-specific RNN structures (see, e.g., the recent works [99, 170, 251, 139, 58]).

Although, we discuss the training problem of RCNNs in accordance with the two stages involved, our main focus is on the first stage of the training problem where training a RNN with dynamically consistent overshooting (DCRNN) is viewed as a constrained optimization problem. While this viewpoint helps us to develop alternative neural network training algorithms, it can also provide tools to derive useful convergence results for the algorithm [135]. In this chapter, we estab-

lish an inexact SQP framework for the DCRNN training problem where the quadratic programming (QP) subproblems are solved by a restarted Krylov-subspace iterative scheme that implicitly exploits the structure of the associated Karush–Kuhn–Tucker (KKT) subsystems. As the resulting algorithm is based on the restarted generalized minimal residual (GMRES) Krylov-subspace method, we call it GMRES recurrent learning (GRL) algorithm, or GRL($\hat{m}_r$) to include a given restart parameter $\hat{m}_r$. The iSQPRL framework allows us to use sparse iterative methods that exploit the structure of subproblems for fast convergence and to achieve a control- and data-efficient model of the DCRNN. As a back-propagation-through-time (BPTT) algorithm for training RNNs [202], GRL($\hat{m}_r$) also addresses one of the main shortcomings associated with the class of BPTT algorithms, viz. requiring an excessive number of iterations to converge to the true solution, largely increasing the complexity of the algorithm [18]. In the second stage – where we learn an optimal control law – we simply adapt the GGN with self-concordant regularization (GGN-SCORE) algorithm, established for convex optimization problems in Chapter 2. Because the GGN-SCORE algorithm was established for curvature exploiting, this also improves our overall model-based control decision approach for data-efficiency.

The rest of this chapter is organized as follows. In Section 5.2, we present the state-space RCNN model and the optimization problem involved in each of the two training stages. In Section 5.3, we introduce the SQP framework for the DCRNN learning problem of the first stage and present details of our proposed learning algorithm, which we call GRL($\hat{m}_r$), an algorithm that uses a nonmonotone line-search for its globalization (see Algorithm 4). In Section 5.4, we present the curvature-exploiting GGN-SCORE algorithm, with a slight modification, for training the control network of the second stage. We analyze the complexity of the proposed algorithm in Section 5.5. Numerical examples in which we demonstrate our approach is presented in Section 5.6 and a concluding remark is given in Section **??**.

5.2 Recurrent control neural networks

The RCNN training problem involves two stages: (1) identify a DCRNN model and (2) train a FNN on the DCRNN model for optimal control policy selection. In accordance with this, we first formulate the problem of training a RNN (and DCRNN). Then we present the RCNN and formulate its training problem on top of an extension of the DCRNN model.

5.2.1 RNN for state reconstruction in RL and control

Suppose that the state transition and observables of the environment or system process can be described by a nonlinear open-loop dynamical system, for which a model is assumed to be unknown. Let $\{u_0, u_1, \ldots, u_{N-1}\}$ be a sequence of inputs to the dynamical system and $\{y_0, y_1, \ldots, y_{N-1}\}$ the corresponding sequence of observable outputs, with $u_t \in \mathbb{R}^{n_u}, y_t \in \mathbb{R}^{n_y}$. System identification is performed to train a RNN model of the form (cf. (1.5))

$$x_{t+1} = f_x(x_t, y_t, u_t, \theta_x) \tag{5.1a}$$

$$\hat{y}_t = f_y(x_t, \theta_y) \tag{5.1b}$$

where f_x and f_y describe the Markovian state-space dynamics and are respectively parameterized by the vectorized weight/bias terms $\theta_x \in \mathbb{R}^{n_{\theta_x}}$ and $\theta_y \in \mathbb{R}^{n_{\theta_y}}$, $x_t \in \mathbb{R}^{n_x}$ is the RNN hidden state and $\hat{y}_t \in \mathbb{R}^{n_y}$ is its prediction of the system observable at time t. In the identification task, the RNN is trained to encode meaningful information about the true system in its hidden states x_t. In addition to passing the observables y_t as external inputs to the state-update function (5.1a), they are also given as the targets for the RNN predictions $\hat{y}_t$. In this chapter, we propose to learn x_t and the parameter vectors θ_x, θ_y in parallel in the context described in [187]; that is, in a way analogous to *direct multiple shooting* approach to optimal control (OC) problems. This is achieved by formulating the

learning task as the equality-constrained problem

$$\min_{z} \quad f(z) \triangleq \mathcal{R}(x_0, \theta_x, \theta_y) + \sum_{t=0}^{N-1} \ell(y_t, \hat{y}_t) \tag{5.2a}$$

$$\text{s.t.} \quad c(z) \triangleq \begin{bmatrix} x_1 - f_x(x_0, y_0, u_0, \theta_x) \\ \vdots \\ x_{N-1} - f_x(x_{N-2}, y_{N-2}, u_{N-2}, \theta_x) \\ c_N(x_0, f_x(x_{N-1}, y_{N-1}, u_{N-1}, \theta_x)) \end{bmatrix} = 0 \tag{5.2b}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$, $c\colon \mathbb{R}^n \to \mathbb{R}^m$, $z \triangleq [x_1^\top \cdots x_{N-1}^\top x_0^\top \theta_y^\top \theta_x^\top]^\top \equiv [z_1\, z_2 \cdots z_n]^\top$, $n = m + n_{\theta_y} + n_{\theta_x}, m = Nn_x$, $\ell\colon \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is a loss function, and $\mathcal{R}$ is a regularization term. Both ℓ and $\mathcal{R}$ are assumed to be twice continuously differentiable with respect to their arguments. The last constraint in (5.2b) represents possible constraints such as periodicity, $x_0 - f_x(x_{N-1}, y_{N-1}, u_{N-1}, \theta_x) = 0$ [231], or fixed terminal state constraint, where $c_N\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$.

5.2.2 Extension to DCRNNs

Since now on, we will consider the special structure of (5.1) given by

$$x_{t+1} = \sigma_x(W_{xu}u_t + W_{xy}y_t + W_{xx}x_t + b_x) \tag{5.3a}$$

$$\hat{y}_t = W_{yx}x_t + b_y \tag{5.3b}$$

where $\sigma_x(\cdot)$ is an element-wise activation function. Let $\theta_x \triangleq vec([W_{xu}\, W_{xy}\, W_{xx}\, b_x]), \theta_y \triangleq vec([W_{yx}\, b_y])$, where $W_{xu} \in \mathbb{R}^{n_u \times n_x}, W_{xy} \in \mathbb{R}^{n_x \times n_y}, W_{xx} \in \mathbb{R}^{n_x \times n_x}$ and $W_{yx} \in \mathbb{R}^{n_y \times n_x}$ are the RNN weight matrices, and $b_x \in \mathbb{R}^{n_x}, b_y \in \mathbb{R}^{n_y}$ its bias vectors associated with model (5.3). The formulation (5.2) does not minimize a pure open-loop simulation error, in that y_t enters the state-update equation rather than $\hat{y}_t$. To handle open-loop simulation terms, we approximate the recurrence of the RNN dynamics by truncating its *unrolling* into the past at a finite time step $t = m_-$. The DCRNN contains in modeling an internal dynamics by *overshooting* the network dynamics in the sense that, we take $m_+ > 1$ overshooting time-steps into the future where the network uses its own

predictions as the future external inputs [252] (see Figure 34). Still, in the overshooting part, the network gets the system inputs u_t as they also influence the network dynamics. This phenomenon results in the following state-space equations of the DCRNN dynamics

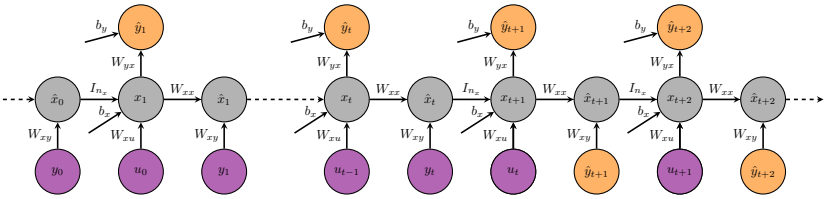


Figure 34: Unrolled recurrent neural network with dynamically consistent overshooting (DCRNN).

$$x_{t+1} = \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}u_t + b_x) \tag{5.4a}$$

$$\hat{y}_t = W_{yx}x_t + b_y \tag{5.4b}$$

$$\text{with} \quad \hat{x}_t = \begin{cases} W_{xx}x_t + W_{xy}y_t, & \forall 0 \leq t \leq m_- \\ W_{xx}x_t + W_{xy}\hat{y}_t, & \forall m_- < t \leq N \end{cases} \tag{5.4c}$$

where $\mathrm{I} \in \mathbb{R}^{n_x \times n_x}$ is an identity matrix. Letting $\ell = (y_t - \hat{y}_t)^2$, $c(z) \triangleq [c_0, c_1, \ldots, c_N] \in \mathbb{R}^m$, $N = m_- + m_+$, the corresponding optimization problem is

$$\min_z \quad f(z) \triangleq \mathcal{R}(x_0, \theta_x, \theta_y) + \sum_{t=0}^{N-1} (y_t - \hat{y}_t)^2 \tag{5.5a}$$

$$\text{s.t.} \quad c_t = 0, \quad \forall t = 0, \ldots, N \tag{5.5b}$$

$$\text{with} \quad c_t \triangleq x_{t+1} - \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}u_t + b_x) \tag{5.5c}$$

$$\hat{x}_t = \begin{cases} W_{xx}x_t + W_{xy}y_t, & \forall 0 \leq t \leq m_- \\ W_{xx}x_t + W_{xy}\hat{y}_t, & \forall m_- < t \leq N-1 \end{cases} \tag{5.5d}$$

5.2.3 FNN for optimal control policy selection

Given a training dataset $\{u_t, y_t\}, t = 0, \ldots, N-1$, after computing the optimal values of the parameter vectors θ_x, θ_y, we construct a FNN on top

of the DCRNN to obtain a RCNN (see Figure 35). In this chapter, a three layer FNN model is considered to obtain new optimal control inputs $\hat{u}_t$, $\forall t > m_-$. The resulting control network together with the DCRNN forms the following RCNN represented by their state-space equations:

$$\hat{u}_t = \sigma_u(V_{uh}\sigma_h(V_{hx}\hat{x}_t + b_h) + b_u), \quad \forall m_- \leq t \leq N \tag{5.6a}$$

$$x_{t+1} = \begin{cases} \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}u_t + b_x), & \forall 0 \leq t < m_- \\ \sigma_x(\mathrm{I}\hat{x}_t + W_{xu}\hat{u}_t + b_x), & \forall m_- < t \leq N \end{cases} \tag{5.6b}$$

$$R_t = G_r\sigma_r(W_{yx}x_t + b_y), \quad \forall m_- < t \leq N \tag{5.6c}$$

$$\text{with} \quad \hat{x}_t = \begin{cases} W_{xx}x_t + W_{xy}y_t, & 0 \leq t \leq m_- \\ W_{xx}x_t + W_{xy}\hat{y}_t, & m_- < t \leq N \end{cases} \tag{5.6d}$$

where $V_{uh} \in \mathbb{R}^{n_u \times n_h}$, $V_{hx} \in \mathbb{R}^{n_h \times n_x}$, $b_h \in \mathbb{R}^{n_h}$, $b_u \in \mathbb{R}^{n_u}$ are the control network parameters to be learnt, and σ_u, σ_h are element-wise activation functions. G_r is a constant, problem-specific matrix, chosen such that with an appropriate choice of the function σ_r, the network output map $R_t(x_t, \hat{y}_t; \hat{u}_t)$ models the problem's reward function. Here, the optimization problem to solve is

$$\max_{\theta_u} \quad \hat{f}(\theta_u) \triangleq \sum_{t=m_-}^{N-1} R_t \tag{5.7}$$

where $\theta_u \triangleq vec([V_{uh}\, V_{hx}\, b_h\, b_u]) \in \mathbb{R}^{n_{\theta_u}}$.

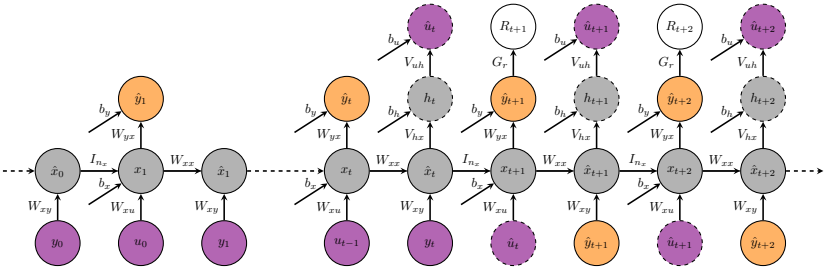


Figure 35: Architecture of recurrent control neural networks (RCNNs).

5.3 Sequential quadratic programming for recurrent learning

In the following, we describe the iSQPRL framework which we use to construct an algorithm to solve the DCRNN learning problems. Consider again problem (5.5) and define the Lagrangian

$$\mathcal{L}(z, \lambda) \triangleq f(z) - \lambda^\top c(z) \tag{5.8}$$

where $\lambda \in \mathbb{R}^m$ is the vector of Lagrange multipliers. The SQP approach for solving problem (5.5) involves iteratively solving the quadratic programming (QP) subproblem

$$d_k^z \in \arg\min_{d_z} \quad \frac{1}{2} d_z^\top H(z_k) d_z + \nabla f(z_k)^\top d_z \tag{5.9a}$$

$$\text{s.t.} \quad \nabla c(z_k) d_z + c(z_k) = 0 \tag{5.9b}$$

at a given approximate solution z_k, $k = 1, 2, \ldots$, where $\nabla c(z) = [\nabla c_0(z) \cdots \nabla c_{N-1}(z)]^\top \in \mathbb{R}^{m \times n}$ and $H(z_k)$ is the Hessian $\nabla^2 \mathcal{L}(z_k, \lambda_k) \triangleq \nabla^2 f(z_k) + \nabla(\nabla c(z_k)^\top \lambda_k)$ of the Lagrangian at step k, or an approximation to it. Suppose at iteration k, the QP solver finds an optimal multiplier $\tilde{\lambda}_k$ (which can change from iteration to iteration). Then, by setting

$$d_k^\lambda = \tilde{\lambda}_k - \lambda_k \tag{5.10}$$

we update (z, λ) using

$$z_{k+1} = z_k + \alpha_k d_k^z, \quad \lambda_{k+1} = \lambda_k + \alpha_k d_k^\lambda \tag{5.11}$$

where α_k is a carefully chosen step-length, or *learning rate*. This iterative process creates a sequence $\{z_k\}$ of approximations which should be made to converge to a solution z^* of (5.5). Next, we consider the equivalent Newton viewpoint of the QP subproblem (5.9), to simplify our analysis of the proposed solution technique.

The necessary (KKT) optimality conditions for the QP subproblem at any step k are given by the nonlinear equation

$$\tilde{\mathcal{F}}(z_k, \lambda_k) \triangleq \begin{bmatrix} \nabla f(z_k) - \nabla c(z_k)^\top \lambda_k \\ c(z_k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{5.12}$$

For simplicity of notation, we denote by $\tilde{\mathcal{F}}_k \triangleq \tilde{\mathcal{F}}(z_k, \lambda_k)$ and similarly for other functional quantities, and set $J_k \triangleq \nabla c_k$ and $g_k \triangleq \nabla f_k$. By applying Newton's method to (5.12), we obtain the *Newton's equation*

$$\tilde{A}_k \tilde{d}_k = -\tilde{\mathcal{F}}_k, \quad \tilde{A}_k = \nabla \tilde{\mathcal{F}}_k = \begin{bmatrix} H_k & -J_k^\top \\ J_k & 0 \end{bmatrix}, \tilde{d}_k = \begin{bmatrix} d_k^z \\ d_k^\lambda \end{bmatrix}. \tag{5.13}$$

By treating λ_{k+1} as an unknown and by letting $\mathcal{F}_k \triangleq [g_k, c_k]^\top, b_k \triangleq -\mathcal{F}_k$, we equivalently write (5.13) as the saddle point system

$$\underbrace{\begin{bmatrix} H_k & J_k^\top \\ J_k & 0 \end{bmatrix}}_{A_k} \underbrace{\begin{bmatrix} d_k^z \\ -\lambda_k \end{bmatrix}}_{d_k} = \underbrace{\begin{bmatrix} -g_k \\ -c_k \end{bmatrix}}_{b_k}. \tag{5.14}$$

The solution d_k to (5.14) constitutes the Newton direction at step k. Depending on the choice of H_k, say $H_k = \nabla^2 \mathcal{L}_k$, and by setting d_k^λ as in (5.10), we choose an appropriate value for the step-length parameter α_k, say $\alpha_k = 1$, and take the step (5.11). Once the system in (5.14) is solved, $\tilde{A}_k$ and $\tilde{d}_k$ satisfying (5.13) can then be obtained accordingly.

5.3.1 Approximating the Lagrangian Hessian

Clearly, the choice $H_k = \nabla^2 \mathcal{L}_k$ in (5.14) would be computationally prohibitive for the DCRNN training problem in most cases. Besides, we often need H_k to be positive-definite in order to ensure that the QP subproblem solved at each step is convex, thereby preserving the convergence properties provided by the primal step direction. However, $\nabla^2 \mathcal{L}_k$ is not positive-definite in general. As in Section **??**, we reduce these computational difficulties by approximating $\nabla^2 \mathcal{L}_k$ by the modified BFGS formula [50, 94, 101, 216] presented in [184]. For completeness, we also summarize the modified BFGS scheme here.

Starting from an initial positive definite matrix H_0, the modified BFGS scheme seeks an approximation H_k to the true Hessian $\nabla^2 \mathcal{L}_k$ as follows. At step k, define

$$\bar{\gamma}_k = \theta_k \gamma_k + (1 - \theta_k) H_k \delta_k, \qquad 0 \leq \theta_k \leq 1 \tag{5.15}$$

where $\gamma_k = \nabla\mathcal{L}_{k+1} - \nabla\mathcal{L}(z_k, \lambda_{k+1})$, $\delta_k \triangleq \alpha_k d_k^z = z_{k+1} - z_k$. Vector $\bar{\gamma}_k$ is geometrically the closest to γ_k such that the inequality

$$\delta_k^\top \bar{\gamma}_k \geq \eta \delta_k^\top H_k \delta_k \tag{5.16}$$

is satisfied, where the parameter η is chosen empirically and typically takes the value 0.2 [184, 185, 186] which is just sufficient for our approximations. The key idea is to gradually approach the true Hessian while accounting for the curvature information captured during the most recent step. Consequently, the convexity parameter θ_k in (5.15) takes the value

$$\theta_k = \begin{cases} 1, & \text{if } \delta_k^\top \gamma_k \geq \eta \delta_k^\top H_k \delta_k \\ \frac{(1-\eta)\delta_k^\top H_k \delta_k}{\delta_k^\top H_k \delta_k - \delta_k^\top \gamma_k}, & \text{otherwise} \end{cases} \tag{5.17}$$

and a rank-two matrix U_k defined by

$$U_k = \frac{\bar{\gamma}_k \bar{\gamma}_k^\top}{\delta_k^\top \bar{\gamma}_k} - \frac{H_k \delta_k \delta_k^\top H_k}{\delta_k^\top H_k \delta_k} \tag{5.18}$$

is selected to update H_k as

$$H_{k+1} = H_k + U_k \tag{5.19}$$

which maintains positive-definiteness of H_k.

5.3.2 Numerical solution of the saddle-point system

As the linear system (5.14) can involve a large number of variables, it becomes natural to present a computationally efficient technique for solving it. In this chapter, we do not wish to impose restrictive assumptions on iSQPRL as we know that the matrix A_k in (5.14) is not positive-definite. Hence, we present a solution approach to the system in a general learning framework. The GMRES algorithm [204], which we briefly describe next, is our method of choice for this purpose. Starting from an arbitrary initial guess $d_{k,0} \in \mathbb{R}^{\hat{n}}, \hat{n} = n + m$, define the initial residual $r_{k,0} \triangleq b_k - A_k d_{k,0}$ and let

$$\mathcal{K}_{\hat{m}}(A_k, r_{k,0}) \triangleq \text{span}\left\{r_{k,0}, A_k r_{k,0}, \ldots, A_k^{\hat{m}-1} r_{k,0}\right\} \tag{5.20}$$

be the $\hat{m}$-dimensional Krylov-subspace generated by A_k and $r_{k,0}$. At the $\hat{m}$-th step, GMRES finds an approximation $d_{k,\hat{m}} \in \mathcal{K}_{\hat{m}}$ to the true solution $d_k = A_k^{-1} b_k$. This approximate solution is the value of d_k that minimizes the norm of the residual $r_{k,\hat{m}} = b_k - A_k d_{k,\hat{m}}$. In its implementation, GMRES does not explicitly form the vectors $r_{k,0}, A_k r_{k,0}, A_k^2 r_{k,0}, \ldots, A_k^{\hat{m}-1} r_{k,0}$, as they may be close to being linearly dependent; instead, it uses the Arnoldi iteration to form an orthonormal basis for $\mathcal{K}_{\hat{m}}(A_k, r_{k,0})$. In particular, if $q_{k,1}, q_{k,2}, \ldots, q_{k,\hat{m}}$ are the orthonormal vectors formed by the Arnoldi iteration, and if these vectors form the matrix $Q_{k,\hat{m}} \in \mathbb{R}^{\hat{n} \times \hat{m}}$, then we can write $d_{k,\hat{m}}$ as $d_{k,\hat{m}} = d_{k,0} + Q_{k,\hat{m}} s_{k,\hat{m}}$, $s_{k,\hat{m}} \in \mathbb{R}^{\hat{m}}$.

Let $\tilde{H}_{k,\hat{m}} \in \mathbb{R}^{(\hat{m}+1) \times \hat{m}}$ be the upper Hessenberg matrix generated from the (modified) Gram-Schmidt orthogonalization step of Arnoldi iteration satisfying

$$A_k Q_{k,\hat{m}} = Q_{k,\hat{m}+1} \tilde{H}_{k,\hat{m}}.$$

and let $\beta_k \triangleq \|r_{k,0}\|$. Then, $Q_{k,\hat{m}+1} e_1 = q_{k,1} = \|r_{k,0}\|^{-1} r_{k,0}$, where $e_1 \triangleq [1\, 0\, 0 \cdots 0]^\top \in \mathbb{R}^{\hat{m}+1}$, and

$$\begin{aligned} r_{k,\hat{m}} &= b_k - A_k d_{k,\hat{m}} \\ &= b_k - A_k (d_{k,0} + Q_{k,\hat{m}} s_{k,\hat{m}}) \\ &= \beta_k q_{k,1} - Q_{k,\hat{m}+1} \tilde{H}_{k,\hat{m}} s_{k,\hat{m}} \\ &= Q_{k,\hat{m}+1} (\beta_k e_1 - \tilde{H}_{k,\hat{m}} s_{k,\hat{m}}). \end{aligned}$$

As $Q_{k,\hat{m}+1}$ has orthonormal columns, we get that

$$\|r_{k,\hat{m}}\| = \|\beta_k e_1 - \tilde{H}_{k,\hat{m}} s_{k,\hat{m}}\|.$$

Therefore, one finds $d_{k,\hat{m}} \equiv d_k$ by solving

$$\min_{s \in \mathbb{R}^{\hat{m}}} \|\beta_k e_1 - \tilde{H}_{k,\hat{m}} s\| \tag{5.21}$$

whose solution can be obtained efficiently by using the QR factorization of $\tilde{H}_{k,\hat{m}}$, which in turn can be cheaply updated from iteration to iteration by exploiting its structure.

While GMRES solves a general linear system, the restarted variant of it, written GMRES($\hat{m}_r$), provides a better convergence speed and helps

to curtail the memory issue linked to the storage of the orthonormal vectors formed in the former. Instead of allowing the storage of the entire Krylov-subspaces whose size grows quadratically with the number $\hat{m}$ of steps, GMRES($\hat{m}_r$) restricts the dimension of the Krylov-subspace to a fixed integer parameter $\hat{m}_r$ and restarts the Arnoldi process using the current approximate solution $d_{k,\hat{m}_r}$ as the new initial guess [204]. As the GMRES($\hat{m}_r$) method only computes an approximate solution d_k of the SQP subproblem, it forms an *inexact SQP method* (cf. [80]). Hence, we consider (5.14) with the residual vector $r_k \equiv r_{k,\hat{m}_r} \triangleq (r_k^z, r_k^\lambda)$ that accounts for the error due to this inexactness:

$$\begin{bmatrix} H_k & J_k^\top \\ J_k & 0 \end{bmatrix} \begin{bmatrix} d_k^z \\ -\lambda_k \end{bmatrix} = \begin{bmatrix} -g_k \\ -c_k \end{bmatrix} + \begin{bmatrix} r_k^z \\ r_k^\lambda \end{bmatrix}. \tag{5.22}$$

The addition of this residual vector is useful for deriving relevant bounds on important terms of the problem, as we shall see. In specific terms, the residual vector characterizes a part of the globalization strategy (Steps 9–13 of Algorithm 4) described in the next subsection. As a first step, we now state a result, established in [204], on the convergence of GMRES($\hat{m}_r$), that provides us with an important property of the residual norm associated with the algorithm.

Proposition 5.3.1 ([204], Proposition 4 and Theorem 5)**.** *Suppose that A_k is diagonalizable so that $A_k = P_k D_k P_k^{-1}$ where D_k is the diagonal matrix of eigenvalues of A_k. Let τ be the number of distinct eigenvalues $\varrho_1, \varrho_2, \ldots, \varrho_\tau$ of A_k with nonpositive real parts and let the other eigenvalues be enclosed in $C_\Theta(\Omega)$, a circle of radius Θ centered at Ω with $\Omega > \Theta > 0$. Then the residual norm of GMRES($\hat{m}_r$) satisfies*

$$\|r_{k,\hat{m}_r}\| \leq \kappa(P_k)\xi_k \|r_{k,0}\| \tag{5.23}$$

where

$$\kappa(P_k) = \|P_k\|\|P_k^{-1}\|, \quad \xi_k = \left(\frac{\bar{\alpha}}{\bar{\beta}}\right)^\tau \left(\frac{\Theta}{\Omega}\right)^{\hat{m}_r - \tau},$$
$$\bar{\alpha} = \max_{\substack{1 \leq i \leq \tau \\ \tau+1 \leq j \leq \hat{n}}} |\varrho_t - \varrho_j| \quad \textit{and} \quad \bar{\beta} = \min_{1 \leq i \leq \tau} |\varrho_t|.$$

5.3.3 Globalization of iSQPRL by a line-search

Many known inexact SQP approaches rely upon a line-search or trust region method to ensure global convergence. For the sake of completeness, we discuss these globalization concerns for iSQPRL with a line-search strategy, and with specific reference to the GMRES($\hat{m}_r$) iterative scheme considered in this chapter. In order to do this, we assume that the sequence $\{H_k\}$ is obtained through the modified BFGS formula described in Section 5.3.1. We also assume that the value of $\hat{m}_r$ is sufficient to ensure convergence of GMRES($\hat{m}_r$), and that the sequence of iterates $\{z_k, \lambda_k\}$ is generated by the described iSQPRL framework. Furthermore, the following conditions hold (using $\mathcal{L}^*$ to denote $\mathcal{L}(z^*, \lambda^*)$):

A.1 $\{z_k, \lambda_k\}$ is contained in a closed, bounded, convex set S, on which the functions f and c are twice continuously differentiable.

A.2 The columns of J_k are linearly independent, for each k.

A.3 The matrices H_k are uniformly positive definite on the null spaces of J_k, that is, $\exists \sigma_1 > 0$ such that for each k, $d^\top H_k d \geq \sigma_1 \|d\|^2$ for all $d \in \mathbb{R}^{\hat{n}}$ satisfying $J_k d = 0$.

A.4 The sequence $\{H_k\}$ is bounded in norm, that is, $\exists \sigma_2 > 0$ such that for each k, $\|H_k\| \leq \sigma_2$.

A.5 The matrices H_k have inverses that are bounded in norm, that is, $\exists \sigma_3 > 0$ such that for each k, H_k^{-1} exists and $\|H_k^{-1}\| \leq \sigma_3$.

A.6 $\nabla^2 \mathcal{L}_k$ is Lipschitz continuous for each k in the neighbourhood of (z^*, λ^*).

A.7 The primal-dual step d_k *locally* satisfies

$$\lim_{k \to \infty} \frac{\|(P_k(H_k - \nabla^2 \mathcal{L}^*)P_k)d_k^z\|}{\|d_k^z\|} = 0$$

where P_k is the projection matrix $P_k = I - J_k(J_k^\top J_k)^{-1} J_k^\top$.

Assumptions A.1 – A.7 are fairly standard for an equality-constrained optimization problem adopting a line-search technique, and in which positive-definiteness of H_k is enforced [55, 54]. Some of the assumptions can however be relaxed to meet specific practical demands. The following condition provides a stronger version of Assumptions A.3 and A.4, and will be imposed for the sake of simplicity of our presentation.

B.1 For all $d \in \mathbb{R}^{\hat{n}}$, $\exists \sigma_1, \sigma_2 > 0$ such that $\sigma_1 \|d\|^2 \leq d^\top H_k d \leq \sigma_2 \|d\|^2$ for each k.

An important consequence of Assumption A.7 is that one can recover a local two-step superlinear convergence with positive definite matrices H_k, requiring only that a projection of each H_k is close to a projection of $\nabla^2 \mathcal{L}^*$ [186], as opposed to the similar convergence result, say for unconstrained problems, that requires $\nabla^2 \mathcal{L}^*$ to be also positive definite with

$$\lim_{k \to \infty} \frac{\|(H_k - \nabla^2 \mathcal{L}^*) d_k^z\|}{\|d_k^z\|} = 0.$$

We formally state the two-step superlinear convergence result for the constrained problem in the following.

Lemma 5.3.2 ([186, Theorem 1]). *Let Assumptions A.1 – A.7 hold for the recurrent learning problem. Then*

$$\lim_{l \to \infty} \frac{\|z_{l+1} - z^*\|}{\|z_{l-1} - z^*\|} = 0.$$

In order to ensure superlinear convergence of the kind shown in Lemma 5.3.2 from an arbitrary starting point, we equip the SQP with a line-search strategy for choosing a value for the step-length parameter α_k that ensures the value of some well-defined *merit function* is sufficiently reduced for an *acceptable*[1] step to be taken. This strategy provides a way to decide when a progress is made towards a solution. One such merit function commonly used to account for feasibility of the constraints c_k is the ℓ_1 merit function defined as

$$\mathcal{M}_1(z_k; \rho_k) = f_k + \frac{\rho_k}{\omega} \|c_k\|_1, \qquad \frac{\rho_k}{\omega} > 0 \tag{5.24}$$

[1] An inexact solution d_k is considered acceptable if, together with some penalty term ρ_k of the merit function, it causes a sufficient reduction in the value of the merit function.

where $\omega > 1$ is selected heuristically, the motivation for which becomes clear in the ρ_k selection rule of (5.34) below. The main goal is to ensure that the (feasible) KKT points of (5.5) are critical points of $\mathcal{M}_1$.

Upon computing an acceptable step d_k and defining the merit function, we choose a step-length α_k which satisfies the so-called *sufficient decrease* (or Armijo [15]) condition

$$\mathcal{M}_1(z_k + \alpha_k d_k^z; \rho_k) \leq \mathcal{M}_1(z_k; \rho_k) + \nu \alpha_k \nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k) \tag{5.25}$$

where $0 < \nu \leq 0.5$ is a small value and $\nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k)$ is the directional derivative of $\mathcal{M}_1$ along d_k^z, which we derive in the following lemma.

Lemma 5.3.3. *Let assumptions A.1 – A.5 hold for the iSQPRL problem. Then the directional derivative of $\mathcal{M}_1(z_k; \rho_k)$ along a step d_k^z satisfies*

$$\nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k) \leq g_k d_k^z - \frac{\rho_k}{\omega} \|c_k - r_k^\lambda\|_1 \tag{5.26a}$$

$$\nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k) \geq g_k d_k^z - \frac{\rho_k}{\omega} \|c_k - r_k^\lambda\|_1. \tag{5.26b}$$

Proof. Applying Taylor's theorem to f and c, we have

$$\begin{aligned}
&\mathcal{M}_1(z_k + \alpha_k d_k^z; \rho_k) - \mathcal{M}_1(z_k; \rho_k) \\
&= f(z_k + \alpha_k d_k^z) - f_k + \frac{\rho_k}{\omega} \|c(z_k + \alpha_k d_k^z)\|_1 - \frac{\rho_k}{\omega} \|c_k\|_1 \\
&\leq \alpha_k g_k^\top d_k^z + \sigma_2 \alpha_k^2 \|d_k^z\|^2 + \frac{\alpha_k \rho_k}{\omega} \|c_k\|_1.
\end{aligned}$$

From (5.22), we have $J_k d_k^z = -c_k + r_k^\lambda$; hence, with $\alpha_k \leq 1$, we get

$$\begin{aligned}
&\mathcal{M}_1(z_k + \alpha_k d_k^z; \rho_k) - \mathcal{M}_1(z_k; \rho_k) \\
&\leq \alpha_k g_k^\top d_k^z + \sigma_2 \alpha_k^2 \|d_k^z\|^2 - \frac{\alpha_k \rho_k}{\omega} \|c_k - r_k^\lambda\|_1 \\
&= \alpha_k \left(g_k^\top d_k^z - \frac{\rho_k}{\omega} \|c_k - r_k^\lambda\|_1 \right) + \sigma_2 \alpha_k^2 \|d_k^z\|^2.
\end{aligned}$$

Similar arguments yield the lower bound

$$\begin{aligned}
&\mathcal{M}_1(z_k + \alpha_k d_k^z; \rho_k) - \mathcal{M}_1(z_k; \rho_k) \\
&\geq \alpha_k \left(g_k^\top d_k^z - \frac{\rho_k}{\omega} \|c_k - r_k^\lambda\|_1 \right) - \sigma_2 \alpha_k^2 \|d_k^z\|^2.
\end{aligned}$$

Taking the limits

$$\lim_{\alpha_k \to 0} \frac{\mathcal{M}_1(z_k + \alpha_k d_k^z; \rho_k) - \mathcal{M}_1(z_k; \rho_k)}{\alpha_k} \triangleq \nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k)$$

proves (5.26). ■

Consequently, the directional derivative of $\mathcal{M}_1$ along d_k^z is given by

$$\nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k) = g_k^\top d_k^z - \frac{\rho_k}{\omega} \|c_k - r_k^\lambda\|_1. \tag{5.27}$$

As with inexact Newton methods for a general nonlinear system of equations [80], inexact SQP methods require that the residual norm is reduced strictly at each step to allow for the update (z_{k+1}, λ_{k+1}) in (5.11). This requirement, in our case, is provided in the following conditions:

M.1 The primal step computed in the iSQPRL framework is a descent direction for $\mathcal{M}_1$.

M.2 The residual r_k satisfies $\|r_k\| \leq \sigma_4 \|\mathcal{F}_k\|, 0 < \sigma_4 < 1$.

Indeed, that condition M.2 holds with the primal steps evaluated via GMRES($\hat{m}_r$), is a simple deduction from Proposition 5.3.1.

Corollary 1 (of Proposition 5.3.1)**.** *The residual norm at step k satisfies condition M.2 with $\sigma_4 = \kappa(P_k)\xi_k$ assuming the initial guess $d_{k,0} = 0$.*

Proof. With $d_{k,0} = 0$, we get from (5.23) and the definition of $r_{k,\hat{m}}$ that $\|r_k\| \leq \kappa(P_k)\xi_k \|r_{k,0}\| = \kappa(P_k)\xi_k \|b_k - A_k d_{k,0}\| = \kappa(P_k)\xi_k \|b_k\| = \kappa(P_k)\xi_k \|\mathcal{F}_k\|$. ■

The following result tells us what conditions are sufficient, in particular the choice of ρ_k, such that condition M.1 holds.

Proposition 5.3.4. *Let Assumptions A.1 – A.5 and B.1 hold for the iSQPRL problem. Then the directional derivative of $\mathcal{M}_1(z_k; \rho_k)$ along a step d_k^z satisfies*

$$\begin{aligned} &\nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k) \leq \\ &- {d_k^z}^\top H_k d_k^z + \left(\frac{\rho_k}{\omega} - \|\tilde{\lambda}_k\|_\infty \right) \left(\|c_k\|_1 - \|r_k^\lambda\|_1 \right) + {d_k^z}^\top r_k^z. \end{aligned} \tag{5.28}$$

Moreover, if we choose

$$\rho_k > \omega \|\tilde{\lambda}_k\|_\infty \tag{5.29}$$

then d_k^z is guaranteed to be a descent direction for $\mathcal{M}_1$, and $\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k) < 0$ at nonstationary points of (5.5).

Proof. From (5.22), we obtain

$$\begin{aligned} g_k^\top d_k^z &= -{d_k^z}^\top H_k d_k^z + {d_k^z}^\top J_k \tilde{\lambda}_k + {d_k^z}^\top r_k^z \\ {d_k^z}^\top J_k \tilde{\lambda}_k &= -c_k^\top \tilde{\lambda}_k + {r_k^\lambda}^\top \tilde{\lambda}_k. \end{aligned}$$

Substituting these into (5.27) gives

$$\begin{aligned} &\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k) = \\ &- {d_k^z}^\top H_k d_k^z + {d_k^z}^\top J_k \tilde{\lambda}_k + {d_k^z}^\top r_k^z - \frac{\rho_k}{\omega}\|c_k - r_k^\lambda\|_1 \\ &= -{d_k^z}^\top H_k d_k^z - (c_k - r_k^\lambda)^T \tilde{\lambda}_k + {d_k^z}^\top r_k^z - \frac{\rho_k}{\omega}\|c_k - r_k^\lambda\|_1. \end{aligned}$$

Using the relation $-(c_k - r_k^\lambda)^\top \tilde{\lambda}_k \le \|\tilde{\lambda}_k\|_\infty \|c_k - r_k^\lambda\|_1$ from Hölder's inequality, we obtain

$$\begin{aligned} &\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k) \le \\ &- {d_k^z}^\top H_k d_k^z - \left(\frac{\rho_k}{\omega} - \|\tilde{\lambda}_k\|_\infty\right)\|c_k - r_k^\lambda\|_1 + {d_k^z}^\top r_k^z \end{aligned} \tag{5.30}$$

which proves (5.28) by applying the reverse triangle inequality on $-\|c_k - r_k^\lambda\|_1$. By assumption B.1, and if $d_k^z \neq 0$ solves (5.9), it is sufficient to show that

$$\Delta\mathcal{M}_1 \triangleq {d_k^z}^\top r_k^z - \left(\frac{\rho_k}{\omega} - \|\tilde{\lambda}_k\|_\infty\right)\|c_k - r_k^\lambda\|_1 \le 0 \tag{5.31}$$

in order to show that d_k^z is a descent direction for $\mathcal{M}_1$, with $\rho_k > \omega\|\tilde{\lambda}_k\|_\infty$. To do this, we define

$$r_k^\top \bar{d}_k = {r_k^\lambda}^\top \tilde{\lambda}_k + {d_k^z}^\top r_k^z = {d_k^z}^\top H_k d_k^z + g_k^\top d_k^z + c_k^\top \tilde{\lambda}_k$$

where $\bar{d}_k = (d_k^z, \tilde{\lambda}_k)$. By noting the relation

$$-\frac{\rho_k}{\omega}\|c_k - r_k^\lambda\|_1 < -\|\tilde{\lambda}_k\|_\infty \|c_k - r_k^\lambda\|_1$$

we have

$$
\begin{aligned}
&{d_k^z}^\top r_k^z - \left(\frac{\rho_k}{\omega} - \|\tilde{\lambda}_k\|_\infty\right)\|c_k - r_k^\lambda\|_1 \\
&= \underbrace{{d_k^z}^\top H_k d_k^z + g_k^\top d_k^z + c_k^\top \tilde{\lambda}_k - {r_k^\lambda}^\top \tilde{\lambda}_k}_{={d_k^z}^\top r_k^z} - \frac{\rho_k}{\omega}\|c_k - r_k^\lambda\|_1 \\
&\quad + \|\tilde{\lambda}_k\|_\infty \|c_k - r_k^\lambda\|_1 \\
&< {d_k^z}^\top r_k^z + \|d_k^z\|_\infty \|r_k^z\|_1 \leq 0.
\end{aligned}
$$

■

Remark 1. *Alternatives to the merit reduction condition* (5.29) *exist in the literature (see, e.g.,* [74]*), with their specificity linked to the choice of merit function. The use of a particular merit reduction condition is however often well-motivated, say, as a way to quantify an appropriate steepness of the directional derivative of the merit function with the residual terms of the inexact SQP step* [55]*.*

Corollary 2. *Let the assumptions of Proposition 5.3.4 hold, and let* d_k^z *be such that* (5.28) *is satisfied. Then the following property holds:*

$$
\rho_k \geq \frac{\omega}{2}\left[\frac{g_k^\top d_k^z + {d_k^z}^\top H_k d_k^z - {d_k^z}^\top r_k^z}{\|c_k\|_1 - \|r_\lambda\|_1} + \|\tilde{\lambda}_k\|_\infty\right]. \tag{5.32}
$$

Proof. From (5.28) and (5.27), we get

$$
g_k^\top d_k^z + {d_k^z}^\top H_k d_k^z + \left(\frac{2\rho_k}{\omega} - \|\tilde{\lambda}_k\|_\infty\right)\|c_k - r_k^\lambda\|_1 - {d_k^z}^\top r_k^z \leq 0
$$

and applying the reverse triangle inequality on $\|c_k - r_k^\lambda\|_1$ gives

$$
\frac{2\rho_k}{\omega} - \|\tilde{\lambda}_k\|_\infty \geq \frac{g_k^\top d_k^z + {d_k^z}^\top H_k d_k^z - {d_k^z}^\top r_k^z}{\|c_k\|_1 - \|r_k^\lambda\|_1}.
$$

Rearranging the terms gives the result. ■

From the property shown in Corollary 2, we see that a necessary requirement for condition (5.29) is that we choose $\omega \geq 2$ (cf. [55, 54]). In order to maintain (5.29) at each step k, thereby ensuring the exactness

of $\mathcal{M}_1$ at convergence, a common approach is to *safeguard* the inequality with some positive constant $\bar{\rho}$ in a way that

$$\rho_k \geq \omega \|\tilde{\lambda}_k\|_\infty + \bar{\rho}. \tag{5.33}$$

The selection rule for ρ_k, proposed in [184], is slightly modified for the inexact SQP problem of this chapter:

$$\rho_k = \begin{cases} \max\{\omega\|\tilde{\lambda}_k\|_\infty + \bar{\rho},\ \frac{1}{\omega}(\rho_{k-1} + \omega\|\tilde{\lambda}_k\|_\infty + \bar{\rho})\}, \forall k \neq 1 \\ \omega\|\tilde{\lambda}_k\|_\infty + \bar{\rho}, \quad \text{if } k = 1. \end{cases} \tag{5.34}$$

The rationale behind (5.34) is that it provides relevant bounds on the sequence $\{\rho_k\}$ and allows, for k large enough and towards convergence, the choice $\rho_k = \rho_{k-1}$, provided that $\rho_{k-1} \geq \omega\|\tilde{\lambda}_k\|_\infty + \bar{\rho}$. Coupled with a backtracking line-search procedure, a step-length α_k such that the Armijo condition (5.25) holds is chosen.

Many known results for the global convergence of line-search SQP algorithms in constrained optimization are based upon the exactness and stationarity of $\mathcal{M}_1$ at KKT optimality points of the problem, and are direct consequences of the descent property of $\mathcal{M}_1$ (when shown to hold) at d_k such that the line-search succeeds for all k. In the same spirit, and with condition M.2 assumed to hold, we obtain the following result (which we prove by following the reasoning in [43, Theorem 17.2]).

Theorem 5.3.5. *Let the assumptions of Proposition 5.3.4 hold for the iSQPRL problem, and let ρ_k be chosen according to the rule (5.34) so that $\rho_k > \omega\|\tilde{\lambda}_k\|_\infty$ holds. Suppose further that the sequence $\{\lambda_k\}$ is bounded. Then if α_k is bounded below by some positive constant, the sequence of iterates $\{z_k\}$, starting from an arbitrary point, converges to a stationary point of $\mathcal{M}_1$.*

Proof. Since $\{\lambda_k\}$ is bounded, and hence $\{\rho_k\}$, then it is easy to show that there exists an index k_1 such that $\rho_k \approx \rho \geq \omega\|\tilde{\lambda}_k\|_\infty + \bar{\rho}$ for all $k \geq k_1$. The adaptation rule (5.34) indeed ensures that ρ_k does not increase unnecessarily in attaining this ρ since then $\{\tilde{\lambda}_k\}$ is bounded. Therefore, by the descent property of $\mathcal{M}_1$ in Proposition 5.3.4, we are ensured $\mathcal{M}_1$ remains exact.

Consider the set of indices $\mathcal{K} \triangleq \{k \geq k_1 \colon \alpha_k < 1\}$. Then each of the step $k \in \mathcal{K}$ causes a sufficient decrease in $\mathcal{M}_1$ by the backtracking

line-search procedure. Moreover, with $\mathcal{M}_1(z_k;\rho) \geq K > -\infty$ for some constant K, the sufficient decrease condition (5.25) shows that

$$\alpha_k \nabla_{d_k^z} \mathcal{M}_1(z_k;\rho_k) \to 0. \tag{5.35}$$

Now let $\bar{\alpha} \leq \alpha_k$ be some constant for which the backtracking line-search procedure fails. Then with $k \in \mathcal{K}$, at least one of the following holds:

$$z_k + \bar{\alpha} d_k^z \notin S \tag{5.36a}$$

$$\mathcal{M}_1(z_k + \bar{\alpha} d_k^z;\rho_k) > \mathcal{M}_1(z_k;\rho_k) + \nu\bar{\alpha}\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k). \tag{5.36b}$$

However, by Assumption A.1 and with $d_k^z \neq 0$, (5.36a) does not hold. Hence, we have (5.36b). From the proof of Lemma 5.3.3 and (5.27), we get

$$\begin{aligned}\mathcal{M}_1(z_k + \bar{\alpha} d_k^z;\rho_k) - \mathcal{M}_1(z_k;\rho_k) \leq \bar{\alpha}\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k) \\ + \sigma_2\bar{\alpha}^2\|d_k^z\|^2\end{aligned}$$

which together with (5.36b) gives

$$-(1-\nu)\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k) \leq \sigma_2\bar{\alpha}\|d_k^z\|^2.$$

From (5.30) and Assumption B.1, we have

$$\sigma_1\|d_k^z\|^2 \leq -\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k)$$

and since $\nu > 1$, we get from the above two inequalities

$$\alpha_k \geq \bar{\alpha} \geq \frac{\sigma_1}{\sigma_2}(1-\nu) > 0. \tag{5.37}$$

Therefore, (5.35) implies $\nabla_{d_k^z}\mathcal{M}_1(z_k;\rho_k) \to 0$. Consequently, by (5.30) and $\rho_k \geq \omega\|\tilde{\lambda}_k\|_\infty + \bar{\rho}$ we have

$$d_k^{z\top} H_k d_k^z \to 0, \quad c_k \to 0.$$

By our assumptions on H_k, the above implies $d_k^z \to 0$ which we know to hold if and only if z_k is a feasible point satisfying the KKT optimality conditions in (5.12). ■

5.3.4 Practical aspects

We discuss two practical issues that are considered in the implementation of our algorithm with the goal of enabling the convergence results discussed so far.

Algorithm 4 GRL($\hat{m}_r$) with nonmonotone line-search

Input: data $\{u_t, y_t\}_{t=0}^{N-1}$, initial guess for $(x_0, \theta_x, \theta_y)$, number of epochs N_e, line-search parameters $0 < \mu < 1, \nu, \bar{\rho}$, initial Lagrangian multiplier vector λ_0, initial BFGS matrix H_0
Output: updated parameters $(x_0, \theta_x, \theta_y)$
Evolve (5.1). Set $z_0 \leftarrow [x_1^\top \cdots x_{N-1}^\top x_0^\top \theta_y^\top \theta_x^\top]^\top$
Compute $f_0, c_0, J_0, g_0, \mathcal{L}_0$
$k \leftarrow 0$
repeat
    Construct KKT system (5.14)
    Solve KKT system with preconditioning (see Section 5.3.2 and Section 5.3.4). Get $(d_k^z, \tilde{\lambda}_k)$. Get d_k^λ as in (5.10)
    Get ρ_k using (5.34). Define the merit function $\mathcal{M}_1$ (5.24). Define m_k (5.39)
    if $\mathcal{M}_1(z_k + d_k^z; \rho_k) \leq m_k + \nu \nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k)$ **then**
        $z_{k+1} \leftarrow z_k + d_k^z$
        $\lambda_{k+1} \leftarrow \lambda_k + d_k^\lambda$
    else
        Construct KKT system (5.41)
        Solve KKT system with preconditioning (see Section 5.3.2 and Section 5.3.4). Get $(\bar{d}_k^z, \bar{\lambda}_k)$
        $\bar{d}_\lambda \leftarrow \bar{\lambda}_k - \lambda_k$
        if $\|\bar{d}_k^z\| > \|d_k^z\|$ **then**
            $\bar{d}_k^z \leftarrow 0, \bar{d}_\lambda \leftarrow 0$
        end if
        $\alpha_k \leftarrow 1$
        while $\mathcal{M}_1(z_k + \alpha_k d_k^z + \alpha_k^2 \bar{d}_k^z) > m_k + \nu \alpha_k \nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k)$ **do**
            $\alpha_k \leftarrow \mu \alpha_k$
        end while
        $z_{k+1} \leftarrow z_k + \alpha_k d_k^z + \alpha_k^2 \bar{d}_k^z$
        $\lambda_{k+1} \leftarrow \lambda_k + \alpha_k d_k^\lambda + \alpha_k^2 \bar{d}_\lambda$
    end if
    Compute $f_{k+1}, c_{k+1}, J_{k+1}, g_{k+1}, \mathcal{L}_{k+1}$
    Get U_k as in (5.18). $H_{k+1} \leftarrow H_k + U_k$
    $k \leftarrow k + 1$
until $k \geq N_e$

Preconditioning for GMRES($\hat{m}_r$)

The (left) preconditioned GMRES($\hat{m}_r$) is the GMRES($\hat{m}_r$) applied to the system

$$M_k^{-1} A_k d_k = M_k^{-1} b_k$$

where $M_k \in \mathbb{R}^{\hat{n}\times\hat{m}}$ is a suitably chosen, problem-dependent preconditioner. In this chapter, we implement GMRES($\hat{m}_r$) for the solution of (5.14) at each step k, left-preconditioned with the non-singular preconditioner

$$M_k = \begin{bmatrix} G_k & J_k^\top \\ J_k & 0 \end{bmatrix} \tag{5.38}$$

where $G_k \neq H_k$ is an $n \times n$ matrix typically chosen so that M_k is invertible. This preconditioning method, known in the literature as *constraint preconditioning* [37], has been used extensively to achieve improved convergence results for problems of the kind (5.14). There are various options for the choice of G_k (see, e.g., [37] for an overview).

Enforcing superlinear convergence

An important requirement for the superlinear convergence of SQP algorithms is that the line-search strategy accepts a unit step-length[2] for all k. This superlinear step, however, may not be accepted for the merit function $\mathcal{M}_1$ thereby inhibiting fast convergence. This condition, known as the *Maratos effect*, is avoided in this chapter by incorporating the nonmonotone line-search procedure introduced in [174] into our algorithm allowing to occasionally take nonmonotone steps towards progress (see Algorithm 4, lines 14 – 25). The approach is based on the observation that with the number m_k defined by

$$m_k \triangleq \max_{j=0,\ldots,h_m} \mathcal{M}_1(z_{k-j}; \rho_k) \tag{5.39}$$

over a nonmonotone horizon h_m (typically, $h_m = 3$), the function $\mathcal{M}_1$ "eventually" satisfies the *modified* sufficient decrease condition

$$\mathcal{M}_1(z_k + \alpha_k d_k^z; \rho_k) \leq m_k + \nu\alpha_k \nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k). \tag{5.40}$$

[2]It is remarked that obtaining a unit step-length does not always hold [186].

In specific terms, it is proposed in [174] to perform the variable update

$$z_{k+1} = z_k + \hat{\alpha}_{k\,k} d_k^z + \hat{\alpha}_k^2 \bar{d}_k^z$$

whenever the merit function does not accept a unit step-length in the test (5.40), where $0 < \hat{\alpha}_k \leq 1$ is the number returned by a backtracking line-search performed on

$$\mathcal{M}_1(z_k + \alpha_k d_k^z + \alpha_k^2 \bar{d}_k^z; \rho_k) \leq m_k + \nu \alpha_k \nabla_{d_k^z} \mathcal{M}_1(z_k; \rho_k)$$

and $\bar{d}_k^z$ is defined by

$$\begin{bmatrix} H_k & J_k^\top \\ J_k & 0 \end{bmatrix} \begin{bmatrix} \bar{d}_k^z \\ -\bar{\lambda}_k \end{bmatrix} = \begin{bmatrix} -g_k \\ -\bar{c}_k \end{bmatrix} \tag{5.41}$$

with $\bar{c}_k = c(z_k + d_k^z)$. It is shown that under suitable conditions (such as those assumed in this chapter), this "correction plus arc-search" procedure globally preserves Q-superlinear convergence of the SQP algorithm, even though now the merit function $\mathcal{M}_1$ is not forced to reduce at each step k [174].

5.4 Off-line learning of the control network

A key element considered in discussing our proposed algorithm for the optimization problem of the second stage of training RCNN is the structure of the Hessian terms which the method efficiently exploits. Although a FNN is often a simpler network to train than a recurrent network, it is still desirable to exploit curvature information about the training data to improve learning. In Chapter 2, we presented a GGN algorithm that ultimately exploits the idea of self-concordance [164] for approximating batch-wise parameter updates in an unconstrained optimization problem. We also studied this algorithm in Chapter 4 for optimizing feedforward neural networks. In this chapter, we adapt the algorithm for training the control network as described next.

Suppose that in stage two of learning the RCNN we process mini-batches of sample streams in a multi-step ahead fashion, composed of

the current and future stacks $(\hat{u}_k, \hat{y}_k)$, $\hat{u}_k \in \mathbb{R}^{n_u \cdot n_b}, \hat{y}_k \in \mathbb{R}^{n_y \cdot n_b}$, where n_b is the mini-batch size, $1 < n_b < n_{\theta_u}$ (possibly $1 \ll n_b \ll n_{\theta_u}$), $k = 1, 2, \ldots, \lceil \frac{N_u}{n_b} \rceil$, $N_u \geq n_b$ is a desired number of training samples. Let $\bar{r} \colon \mathbb{R}^{n_{\theta_u}} \to \mathbb{R}$ be a self-concordant function[3] with parameter $M_{\bar{r}}$, that is, the inequality

$$\left| \left\langle v, \left(\nabla^3 \bar{r}(\theta)[v] \right) v \right\rangle \right| \leq 2M_{\bar{r}} \left\langle v, \nabla^2 \bar{r}(\theta) v \right\rangle^{3/2} \tag{5.42}$$

holds for any θ in the closed convex set $\mathbb{W} \subseteq \mathbb{R}^{n_{\theta_u}}$ and $v \in \mathbb{R}^{\theta_u}$, where $\nabla^3 \bar{r}(\theta)[v] \in \mathbb{R}^{n_{\theta_u} \times n_{\theta_u}}$ denotes the limit

$$\nabla^3 \bar{r}(\theta)[v] \triangleq \lim_{t \to 0} \frac{1}{s} \left[\nabla^2 \bar{r}(\theta + sv) - \nabla^2 \bar{r}(\theta) \right], \quad s \in \mathbb{R}.$$

Note that any self-concordant function can be scaled to have $M_{\bar{r}} = 1$ in (5.42) [163, Corollary 5.1.3].

Let us regularize problem (5.7) with $\bar{r}(\theta_u)$ and a penalty parameter $\lambda_{\bar{r}} \in \mathbb{R}$ to have the regularized minimization problem[4]

$$\min_{\theta_u} \quad \bar{f}(\theta_u) + \lambda_{\bar{r}} \bar{r}(\theta_u) \tag{5.43}$$

where $\bar{f} \equiv -\hat{f}$. Then, letting $\bar{z} \triangleq \theta_u$ and $n_d \triangleq n_y \cdot n_b$, the mini-batch GGN update rule for optimizing over θ_u is

$$\bar{z}_{k+1} = \bar{z}_k - \left(J_{\hat{y}}^\top Q_{\bar{f}} J_{\hat{y}} + \lambda_{\bar{r}} \bar{H}_k \right)^{-1} J_{\hat{y}}^\top g_{\bar{f}} \tag{5.44}$$

where at step k, $\bar{H}_k \in \mathbb{R}^{n_{\theta_u} \times n_{\theta_u}}$ is the Hessian of $\bar{r}$ with respect to θ_u, $J_{\hat{y}} \in \mathbb{R}^{n_d \times n_{\theta_u}}$ is the Jacobian of $\hat{y}_k$ with respect to θ_u, $g_{\bar{f}} \in \mathbb{R}^{n_d}$ is the residual vector defined as the gradient of $\bar{f}$ with respect to $\hat{y}_k$, and $Q_{\bar{f}} \in \mathbb{R}^{n_d \times n_d}$ is the Hessian of $\bar{f}$ with respect to $\hat{y}_k$. Note that dependence of the terms on k is ignored for notational convenience.

[3] For details on self-concordant functions, we refer the interested reader to the references provided in [4].

[4] Note that $\bar{f} \equiv \hat{f}$ is used if the problem defines R_t in (5.7) as a cost to be minimized rather than a reward to be maximized.

Clearly, the computations involved in (5.44) are highly prohibitive. In [4], the authors use a generalized inverse identity on (5.44) to derive a more computationally convenient way to update $\bar{z}$ as follows:

$$\bar{B}_k \triangleq \lambda_{\bar{r}} \mathrm{I} + \bar{Q}_k \bar{J}_k \bar{H}_k^{-1} \bar{J}_k^\top \tag{5.45a}$$

$$\bar{z}_{k+1} = \bar{z}_k - \bar{H}_k^{-1} \bar{J}_k \bar{B}_k^{-1} \bar{e}_k \tag{5.45b}$$

where $\mathrm{I} \in \mathbb{R}^{n_d \times n_d}$ is an identity matrix, $\bar{J}_k \in \mathbb{R}^{(n_d+1)\times n_{\theta_u}}$ is $J_{\hat{y}}$ augmented with $\lambda_{\bar{r}} \bar{g}_k$ the gradient of $\lambda_{\bar{r}} \bar{r}$ with respect to θ_u, $\bar{e}_k \in \mathbb{R}^{n_d+1}$ is $g_{\bar{f}}$ augmented with a unit term, $\bar{Q}_k \in \mathbb{R}^{(n_d+1)\times(n_d+1)}$ is a diagonal matrix with diagonal terms defined by $Q_{\bar{f}}$ with an additional zero diagonal term. That is,

$$\bar{J}_k^\top = \left[\begin{array}{c:c} J_{\hat{y}}^\top & \lambda_{\bar{r}} \bar{g}_k \end{array}\right], \bar{Q}_k = \left[\begin{array}{c:c} Q_{\bar{f}} & 0 \\ \hdashline 0 & 0 \end{array}\right], \bar{e}_k = \left[\begin{array}{c} g_{\bar{f}} \\ \hdashline 1 \end{array}\right].$$

We note that the Hessian $\bar{H}_k$ is a diagonal matrix with diagonal elements that can be quite cheap to obtain, and hence its inverse $\bar{H}_k^{-1}$ can clearly be computed efficiently.

Furthermore, with motivation from the *Newton decrement* framework in convex optimization, a learning rate $\sigma/(1 + M_{\bar{r}} \bar{\eta}_k)$ was introduced in (5.45) to obtain the full GGN-SCORE update step

$$\bar{B}_k \triangleq \lambda_{\bar{r}} \mathrm{I} + \bar{Q}_k \bar{J}_k \bar{H}_k^{-1} \bar{J}_k^\top \tag{5.46a}$$

$$\bar{z}_{k+1} = \bar{z}_k - \frac{\sigma}{1 + M_{\bar{r}} \bar{\eta}_k} \bar{H}_k^{-1} \bar{J}_k \bar{B}_k^{-1} \bar{e}_k \tag{5.46b}$$

where $0 < \sigma \leq 1$ and $\bar{\eta}_k = \left\langle \bar{g}_k, \bar{H}_k^{-1} \bar{g}_k \right\rangle^{1/2}$. Hence, the self-concordance of $\bar{r}$ helps to control the rate at which its second-derivatives change within the region of convergence, thereby giving the problem an affine-invariant structure [4], as well as to select a learning rate at step k without a line-search technique.

An efficient way to compute vector $\bar{B}_k^{-1} \bar{e}_k$ in (5.46b), is to solve a linear system, say, by QR factorization, and not by a direct matrix inversion. However, in our adaptation of the algorithm, it is observed that, by construction, the diagonal matrix $\underline{B}_k \in \mathbb{R}^{n_{\theta_u} \times n_{\theta_u}}$, with diagonal terms

defined by

$$\underline{\mathrm{b}}_{i,j} = \bar{\delta}_{i,j}/\bar{b}_{i,i} \quad i,j = 1,\ldots,n_{\theta_u}$$

provides a good approximation to $\bar{B}_k^{-1}$, where $\bar{\delta}_{i,j}$ is the Kronecker delta function and $\bar{b}_{i,i}$ are the diagonal entries of $\bar{B}_k$. Hence, for the training of the control network in this chapter, we propose the approximation $\bar{B}_k^{-1} \approx \underline{\mathrm{B}}_k$ so that (5.46b) becomes:

$$\bar{z}_{k+1} = \bar{z}_k - \frac{\sigma}{1 + M_{\bar{r}}\bar{\eta}_k}\bar{H}_k^{-1}\bar{J}_k\underline{\mathrm{B}}_k\bar{e}_k.$$

Once the RCNN is trained on the representative DCRNN model to a desired accuracy, the control network parameters become fixed and are used to perform simulation tasks on the true system. Simulation results obtained from a classical RL problem and a model predictive control (MPC) steady-state regulation problem are presented in Section 5.6 to demonstrate the efficiency of our approach.

5.5 Complexity analysis

In this section, we estimate the computational complexity of the method proposed in this chapter. The proposed GRL($\hat{m}_r$) algorithm is summarized into three main steps, viz.: constructing the Newton-KKT system, solving the Newton-KKT system, and updating the optimization variables.

Constructing the KKT system

Constructing the Newton-KKT system at each iteration k involves evaluating the constraints and their gradients with a combined worst-case complexity of $\mathcal{O}(2mn)$. Hence, in addition to evaluating the modified BFGS Hessian matrix, constructing the KKT system has an overall complexity of $\mathcal{O}(n^2 + 2mn)$ per iteration in the worst-case scenario.

Solving the KKT system

The KKT system is solved using the restarted GMRES scheme which depends on both the number of restarts required to reach a given tolerance

and the restart parameter $\hat{m}_r$. Let us denote by N_1, the number of GMRES restarts required to reach a specified tolerance. Then the matrix-vector multiplications and vector operations in the restarted GMRES procedure have a (fixed) worst-case complexity of $\mathcal{O}(N_1 n \hat{m}_r^2)$ which involves the construction of the upper Hessenberg matrix $\tilde{H}_{k,\hat{m}_r}$ in the Arnoldi iteration. Here, we assume the use of an updating QR-algorithm in the solution of (5.21). Note that $\hat{m}_r$ is typically small, say between $10 - 50$, compared to n.

Updating the optimization variables

The line-search method using the Armijo update rule with the ℓ_1 merit function has a worst-case complexity of $\mathcal{O}(n)$ per iteration.

Overall, the complexity of the proposed GRL($\hat{m}_r$) algorithm for training the RNN model considered in this chapter is estimated as $\mathcal{O}(2n^2 N_e + 2nN_e + 2mnN_e)$. Since the number of iterations, N_e, is generally small for GRL($\hat{m}_r$), which also provides a better solution quality, a compromise is made on computational complexity in favour of GRL($\hat{m}_r$) for convergence speed in terms of small iteration number and solution quality, against benchmark (first-order) BPTT algorithms.

As we mentioned before, we may use any convenient gradient-based algorithm for learning the control FNN. Hence, the effective complexity involved in training the full RCNN will be the complexity of the GRL($\hat{m}_r$) algorithm plus the complexity of the algorithm used to train the control FNN.

5.6 Numerical examples

In the following examples, the RCNN is trained with σ_x and σ_h set, respectively, to the hyperbolic tangent ($\tanh$) and the rectified linear unit (ReLU) functions. The RCNN is structured as described in Section 5.2. The entries of the DCRNN weights are initialized to a normally-distributed number randomly generated by the Mersenne-Twister pseudorandom number generator, while the bias vectors, the hidden states and the Lagrangian parameter are all initialized to the zero array. $\bar{\rho} = 0.8, \omega = 2, N_e = 200, \mu =$

$0.8, \nu = 0.5$. We set $\mathcal{R}(x_0, \theta_x, \theta_y) = \frac{Q_\alpha}{2}\|x_0\|_2^2 + \frac{Q_\beta}{2}\left(\|\theta_x\|_2^2 + \|\theta_y\|_2^2\right)$ with $Q_\alpha = 1.0, Q_\beta = 10^{-5}$. The restart parameter $\hat{m}_r$ in GRL($\hat{m}_r$) is upper bounded by 50 for solving the resulting saddle point systems with $G_k = \mathrm{I}$, the identity matrix, in (5.38). The control network weights V_{uh}, V_{hx} are initialized according to the Kaiming uniform initialization scheme described in [112], that is, we initialize each entry of the matrices to a number generated randomly from a uniform distribution in the interval $[-d, d]$ where $d = 4\sqrt{3/\hat{n}_w}$, and $\hat{n}_w$ represents n_u or n_h accordingly. The bias vectors here are initialized to the zero array. The control network is trained with $\bar{r}$ set to the pseudo-Huber function $r_{\bar{\mu}}$ parameterized by $\bar{\mu}$, $r_{\bar{\mu}}(\theta_u) \triangleq \sqrt{\bar{\mu}^2 + \theta_u^2} - \bar{\mu}$, and $\lambda_{\bar{r}} = 10^{-2}, \sigma = M_{\bar{r}} = 1$. All bound constrained continuous variables $a_i \leq x_i \leq b_i$ are transformed to [176]:

$$x_i = t_{a_i,b_i}(\hat{x}_i) = \frac{b_i + a_i}{2} + \frac{b_i - a_i}{2}\left(\frac{2\hat{x}_i}{1 + \hat{x}_i}\right) \tag{5.47}$$

$i = 1, 2, \ldots, n$.

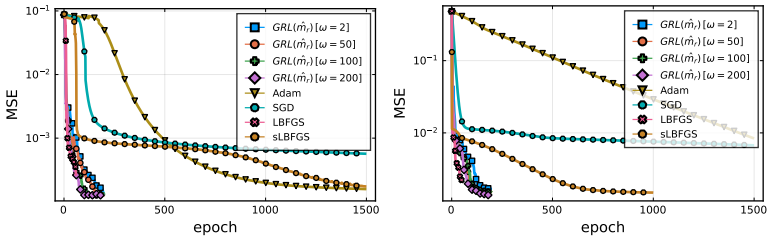
5.6.1 Classical reinforcement learning example

The RL problem considered in this example is the classical mountain car problem described in [224]. The task is to drive an underpowered car to the top of a steep mountain. The continuous observables here are the states of the car given by $y_t = [\overline{\boldsymbol{y}}_t, \dot{\overline{\boldsymbol{y}}}_t]$, where $\overline{\boldsymbol{y}}_t$ is the car's position and $\dot{\overline{\boldsymbol{y}}}_t$ is its velocity at any given time t. For each time the car reaches the top of the mountain, it gets a positive reward R_t

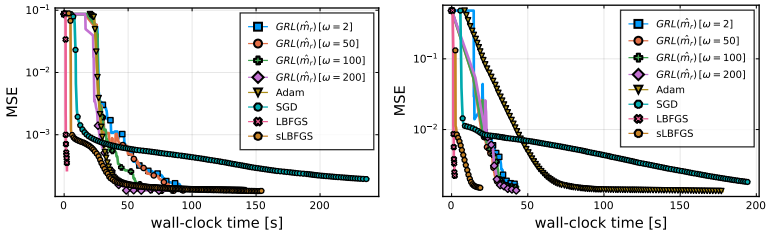
$$R_t = \mathrm{logistic}_{10}(\hat{y}_t - y^{ref}) \tag{5.48}$$

in which σ_r has been set to a steep logistic function $\mathrm{logistic}_{10}(x) \triangleq \frac{1}{1+e^{-10x}}$ [207], and $y^{ref} = 0.5$. The function R_t returns a number that is approximately equal to 1 whenever $\hat{y}_t > y^{ref}$ and 0 otherwise. The control action is the applied force $u_t \in \{-1, 0, 1\}$. The car's motion is described by the set of equations

$$\begin{aligned}
\overline{\boldsymbol{y}}_{t+1} &= \overline{\boldsymbol{y}}_t + \dot{\overline{\boldsymbol{y}}}_t \\
\dot{\overline{\boldsymbol{y}}}_{t+1} &= \dot{\overline{\boldsymbol{y}}}_t + 0.001u_t - 0.0025\cos(3\overline{\boldsymbol{y}}_t)
\end{aligned}$$



(a) Mountain car **(b)** Ethilene oxidation



(c) Mountain car **(d)** Ethilene oxidation

Figure 36: Convergence curves for the function approximation in DCRNN training using GRL($\hat{m}_r$) with $\bar{\rho} = 0.8$, first-order methods (Adam [128] and SGD [188]), limited-memory BFGS (LBFGS; best memory size: 1) with line-search [169, Section 3.5] (training stopped at line-search failure), and stochastic LBFGS (sLBFGS; best memory size: 10) with adaptive step-size [240]. Hidden state vector is initialized to zero. MSE stands for the mean squared error.

with the bound constraints $-1.2 \leq \overline{\boldsymbol{y}}_t \leq 0.5$ and $-0.07 \leq \dot{\overline{\boldsymbol{y}}}_t \leq 0.07$. The task is episodic in that, when $\overline{\boldsymbol{y}}_t$ has reached its right bound, the goal was reached and an episode is completed. However, when it reached its left bound, $\dot{\overline{\boldsymbol{y}}}_t$ was reset to zero.

Each episode starts from an initial random position $\overline{\boldsymbol{y}}_t \in [-0.6, -0.4]$ and a zero velocity. We trained the RCNN using the algorithms proposed in this chapter, where the control inputs u_t and environment observables y_t were given to the network as inputs and targets accordingly. Here, $n_u = 1, n_y = 2$, and we used $m_- = m_+ = 50, n_x = 2, n_h = 128, N_u = 1000, n_b = 64$.

Given the characteristics of the problem, we chose σ_u as the $\tanh$

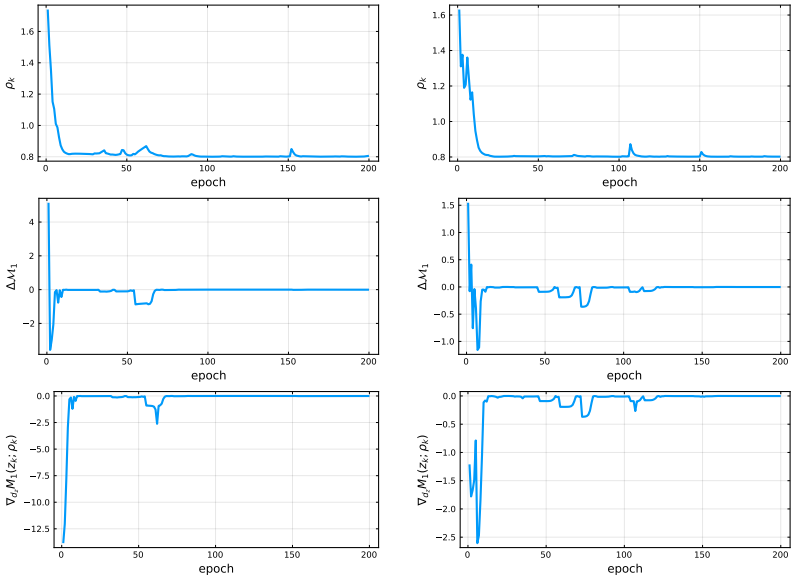


Figure 37: Convergence of Algorithm 4 for terms defined in Proposition 5.3.4 and Theorem 5.3.5 ($\bar{\rho} = 0.8, \omega = 2$). **Left**: Mountain car dynamics. **Right**: Ethylene oxidation process.

Table 7: Solution comparison between the proposed method and SGD, Adam, LBFGS and sLBFGS (mountain car problem).

BPTT algorithm	Approximation error (MSE)	Iter	CPU time [s]
SGD	5.6828×10^{-4}	1500	5.6584×10
Adam	1.5821×10^{-4}	1500	4.9532×10
LBFGS	7.1389×10^{-4}	30	2.8670×10^{0}
sLBFGS	3.9884×10^{-4}	1000	2.8071×10
GRL($\hat{m}_r$), $\omega = 2$	1.3451×10^{-4}	200	6.4956×10
GRL($\hat{m}_r$), $\omega = 50$	$\mathbf{1.2702 \times 10^{-4}}$	200	1.0091×10^{2}
GRL($\hat{m}_r$), $\omega = 100$	1.3164×10^{-4}	200	5.6220×10
GRL($\hat{m}_r$), $\omega = 200$	1.2707×10^{-4}	200	1.1914×10^{2}

function that keeps the variables $\hat{u}_t$ in the interval $[-1, 1]$ during training. However, for our simulations, we follow the standard approach of buck-

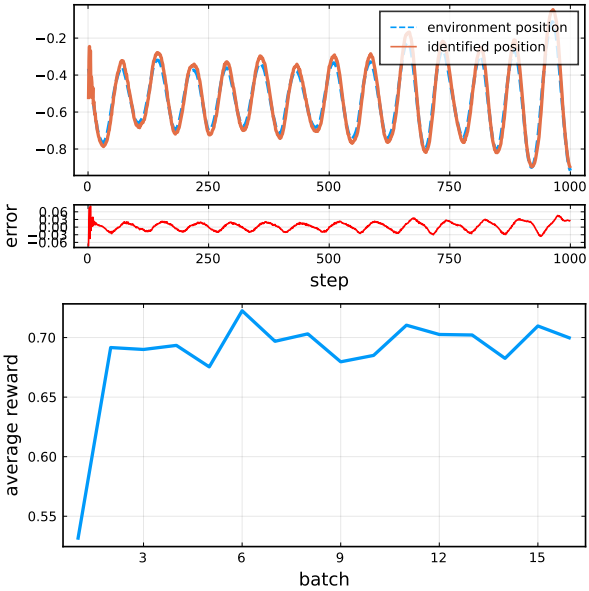


Figure 38: RCNN off-line training results for the mountain car environment. **Top**: DCRNN identified model in stage I (testing via open-loop simulation); $error = y - \hat{y}$. **Bottom**: Stage II training of the RCNN.

eting the continuous outputs of σ_u into the discrete set $\{-1, 0, 1\}$. This approach for ensuring the input bound constraints are satisfied can be used in different problem settings by using an appropriate σ_u or by using the variable transform (5.47). Further, we set $G_r = [1, 0]$ and R_t is defined in (5.48).

The RCNN training results are shown in Figure **38**. In the identification stage, the *effective* DCRNN model was trained on 3 randomly generated datasets with trained hidden states from one dataset used in the next initialization[5]. In the second stage, the control network was trained with 1000 noisy observations (5% Gaussian noise over 10 seeds). Simulation performance on the true system with 10000 noisy observations (5% Gaussian noise over 10 seeds) is shown in Figure **40**. Adding noise

[5]This way, we are able to generate a good start value for the hidden state to improve the representation power of the DCRNN.

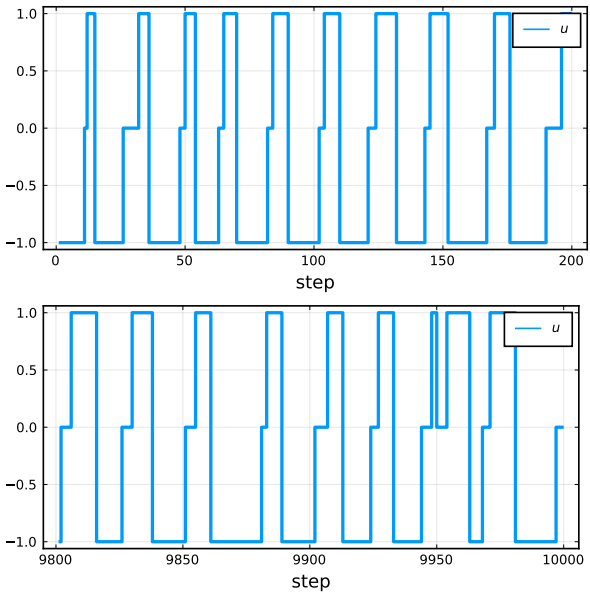


Figure 39: Real-time RCNN control actions $u_t \in \{-1, 0, 1\}$. **Top**: Control actions for the first 200 steps. **Bottom**: Control actions for the last 200 steps.

at these stages creates a more realistic situation [113, 85]. As displayed, we were able to train the RCNN with our algorithms to keep the agent's rewards and steps taken per episode within a reasonably good threshold in the true environment.

5.6.2 Chemical process example

The ethylene oxidation process in a nonisothermal continuously stirred tank reactor is considered in this example. The oxidation process is

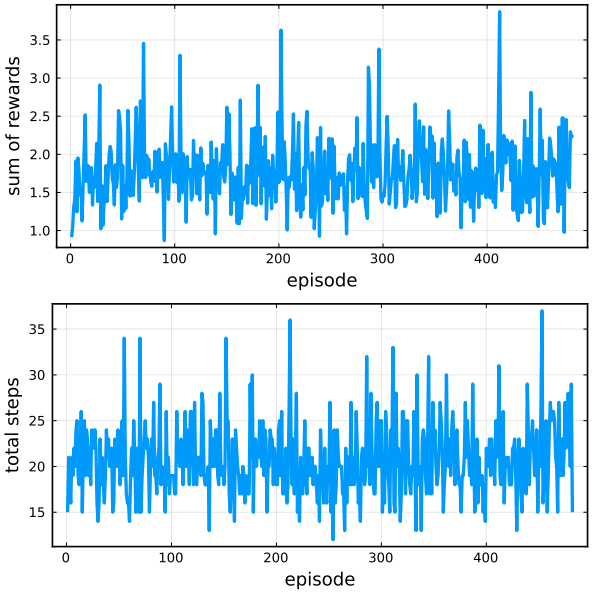


Figure 40: Real-time performance of the trained RCNN control actions in the mountain car environment.

Table 8: Solution comparison between the proposed method and SGD, Adam, LBFGS and sLBFGS (ethylene oxidation).

BPTT algorithm	Approximation error (MSE)	Iter	CPU time [s]
SGD	6.7298×10^{-3}	1500	5.3596×10
Adam	8.3706×10^{-3}	1500	4.4071×10
LBFGS	1.5866×10^{-3}	46	1.6200×10^{0}
sLBFGS	1.4902×10^{-3}	1000	1.9693×10
GRL($\hat{m}_r$), $\omega = 2$	1.6720×10^{-3}	200	4.3819×10
GRL($\hat{m}_r$), $\omega = 50$	1.4948×10^{-3}	200	4.2956×10
GRL($\hat{m}_r$), $\omega = 100$	1.5043×10^{-3}	200	4.4190×10
GRL($\hat{m}_r$), $\omega = 200$	$\mathbf{1.3642 \times 10^{-3}}$	200	4.5000×10

described by the following dimensionless equations [89]:

$$\dot{\overline{\boldsymbol{y}}}_1 = u_1(1 - \overline{\boldsymbol{y}}_1\overline{\boldsymbol{y}}_4)$$

$$\dot{\overline{\boldsymbol{y}}}_2 = u_1(u_2 - \overline{\boldsymbol{y}}_2\overline{\boldsymbol{y}}_4) - A_1 e^{\gamma_1/\overline{\boldsymbol{y}}_4}(\overline{\boldsymbol{y}}_2\overline{\boldsymbol{y}}_4)^{0.5} - A_2 e^{\gamma_2/\overline{\boldsymbol{y}}_4}(\overline{\boldsymbol{y}}_2\overline{\boldsymbol{y}}_4)^{0.25}$$

$$\dot{\overline{\boldsymbol{y}}}_3 = -u_1\overline{\boldsymbol{y}}_3\overline{\boldsymbol{y}}_4 + A_1 e^{\gamma_1/\overline{\boldsymbol{y}}_4}(\overline{\boldsymbol{y}}_2\overline{\boldsymbol{y}}_4)^{0.5} - A_3 e^{\gamma_3/\overline{\boldsymbol{y}}_4}(\overline{\boldsymbol{y}}_3\overline{\boldsymbol{y}}_4)^{0.5}$$

$$\dot{\overline{\boldsymbol{y}}}_4 = \frac{u_1}{\overline{\boldsymbol{y}}_1}(1 - \overline{\boldsymbol{y}}_4) + \frac{B_1}{\overline{\boldsymbol{y}}_1} e^{\gamma_1/\overline{\boldsymbol{y}}_4}(\overline{\boldsymbol{y}}_2\overline{\boldsymbol{y}}_4)^{0.5} + \frac{B_2}{\overline{\boldsymbol{y}}_1} e^{\gamma_2/\overline{\boldsymbol{y}}_4}(\overline{\boldsymbol{y}}_2\overline{\boldsymbol{y}}_4)^{0.25}$$

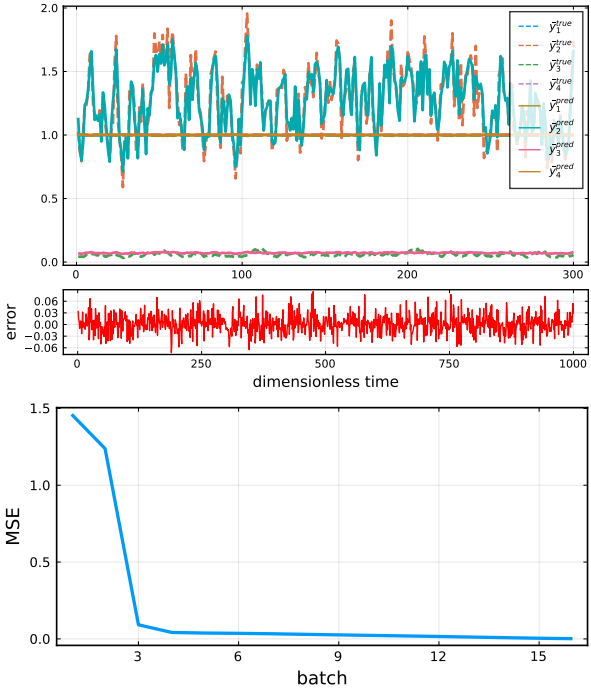


Figure 41: RCNN training results for ethylene oxidation process dynamics. **Top**: DCRNN identified model in stage I (testing via open-loop simulation); $error = y - \hat{y}$. **Bottom**: Stage II training of the RCNN. MSE stands for the mean squared error.

where the continuous observables are given by the dimensionless state variables $y_t = [\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \overline{\boldsymbol{y}}_3, \overline{\boldsymbol{y}}_4]$, $\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \overline{\boldsymbol{y}}_3, \overline{\boldsymbol{y}}_4$ represent the gas density, ethylene concentration, ethylene oxide concentration, and temperature in the reactor, respectively, at any given time t. The control action is $u_t = [u_1, u_2]$, where u_1 is the feed volumetric flow rate and u_2 is the concentration of ethylene in the feed, with bound constraints $0.0704 \leq u_1 \leq 0.7042$, $0.2465 \leq u_2 \leq 2.4648$. The parameters in the equations above are given values adopted from [89] (see Table **10**). The goal of the process is to maximize the production of ethylene oxide for a limited reactant feedstock, starting from an initial state $y_0 = [0.997, 1.264, 0.209, 1.004]$

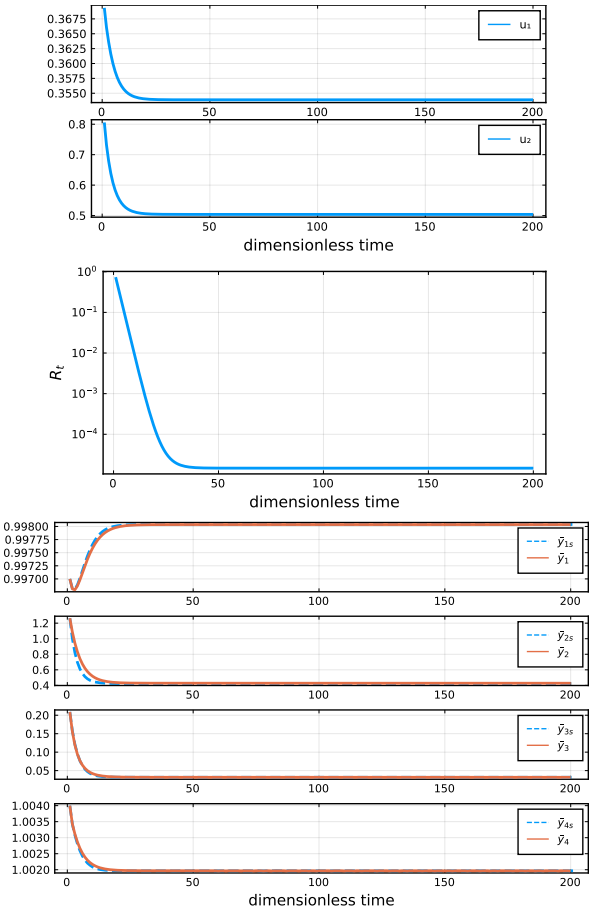


Figure 42: Performance of the trained RCNN control actions in ethylene oxidization process. **Top-left**: RCNN control inputs $u_t = [u_1, u_2]$ with bound constraints $0.0704 \leq u_1 \leq 0.7042$, $0.2465 \leq u_2 \leq 2.4648$. **Bottom-left**: Simulation squared errors $R_t = (\hat{y}_t - y^{ref})^2$. **Right**: $\hat{y}_t = [\overline{\boldsymbol{y}}_1, \overline{\boldsymbol{y}}_2, \overline{\boldsymbol{y}}_3, \overline{\boldsymbol{y}}_4]$ are the RCNN predictions of the steady-state $y^{ref} = [\overline{\boldsymbol{y}}_{1s}, \overline{\boldsymbol{y}}_{1s}, \overline{\boldsymbol{y}}_{3s}, \overline{\boldsymbol{y}}_{4s}]$.

with a sampling period of $\Delta t = 9.36$. Again, we trained the RCNN using the algorithms proposed in this chapter, where the control inputs u_t and environment observables y_t were given to the network as

Table 9: Comparison between the proposed method and selected RNN structures in the mountain car problem.

Method	Approximation error (MSE)	Iter	CPU time [s]
Vanilla RNN + SGD	2.0034×10^{-4}	8000	1.0377×10^{2}
DRRRNN + SGD + ℓ_1 [215]	3.0245×10^{-4}	8000	1.5338×10^{2}
DRRGRU + SGD + ℓ_1 [215]	1.7761×10^{-4}	8000	1.6378×10^{2}
Ours (GRL($\hat{m}_r$), $\omega = 50$)	1.3689×10^{-4}	**250**	5.4477×10
Ours (GRL($\hat{m}_r$), $\omega = 200$)	$\mathbf{1.3435 \times 10^{-4}}$	**250**	$\mathbf{5.3771 \times 10}$

Table 10: Parameters used in the ethylene oxidation process [89].

Parameter	Value	Parameter	Value
γ_1	-8.13	B_1	7.32
γ_2	-7.12	B_2	10.39
γ_3	-11.07	B_3	2170.57
A_1	92.80	B_4	7.02
A_2	12.66	T_c	1.0
A_3	2412.71		

inputs and targets accordingly. Here, $n_u = 2, n_y = 4$, and we used $m_- = m_+ = 40, n_x = 2, n_h = 128, N_u = 1000, n_b = 64$. Here, σ_u was taken to be the identity map with bound constraints enforced through (5.47), $G_r = [1, 1, 1, 1]$ and R_t is defined by taking the squared difference between $\hat{y}_t$ and a reference point y^{ref}, with σ_r and σ_y both set to the identity map so that we have

$$R_t = (\hat{y}_t - y^{ref})^2$$

where we set $y^{ref} \equiv [\overline{\boldsymbol{y}}_{1s}, \overline{\boldsymbol{y}}_{2s}, \overline{\boldsymbol{y}}_{3s}, \overline{\boldsymbol{y}}_{4s}] = [0.998, 0.424, 0.032, 1.002]$, the open-loop asymptotically stable steady-state of the process provided in [89]. Note that in this problem, R_t is specified as a cost to be minimized.

The training and identification results are displayed in Figure **41**. In the first identification stage, the *effective* DCRNN model was trained on 5 randomly generated datasets with trained hidden states from the first

dataset used in the second initialization. Simulation performance on the true system is shown in Figure **42**. As shown, the constant control inputs reported in [89] to bring the process states to the asymptotically stable steady-state value resembles with what is obtained by our approach, that is, $u_s = [u_{1s}, u_{2s}] = [0.35, 0.5]$ in [89].

Regarding the comparison between the proposed GRL($\hat{m}_r$) algorithm for training the DCRNN with respect to first-order SGD and Adam optimizers and two quasi-Newton methods (LBFGS and sLBFGS), in Figure **36** we compare the corresponding convergence curves. Although shown to sometimes converge faster than GRL($\hat{m}_r$), LBFGS (for which we report the best results obtained after careful tuning) yields less numerically robust function approximation results in our examples, as the backtracking line-search fails most of the time. In particular, failure occurs frequently when the memory size is as large as 4. The sLBFGS method seems numerically more robust than the LBFGS in our experiments for a small memory size, but its performance depends heavily on selecting a good adaptive step-size. For a large memory size, and with the adaptive step-size technique adopted in our experiments, sLBFGS tends to converge as slow as a first-order method such as SGD. In summary, GRL($\hat{m}_r$) compares favorably overall with respect to the alternative state-of-the-art methods tested.

In Table **9**, we also compare our approach with the vanilla RNN and recently proposed RNN structures and training methods. In particular, we compare with two variants of the recently proposed DRRNets [215]: DRRRNN and DRRGRU. In order to have a feel of the low-rank regularization proposed in [215], which leads to a (convex-)cardinality problem, we trained the respective DRRNets with an ℓ_1 regularization (i.e., DRRNets + SGD + ℓ_1). Our choice of these variants for comparison is based on the results and discussions in the DRRNets paper. As shown, our approach produces smaller approximation error in fewer number of iterations and reduced runtime.

All parts of the experiments, including the RCNN models and the proposed algorithms were implemented in Julia v1.7.2 on a laptop with dual (2.30GHz + 2.30GHz) Intel Core i7-11800H CPU and 16GB RAM.

Gradient and Jacobian computations were performed by Julia's open-source `ForwardDiff.jl` [192] package. The first-order algorithms used for comparison in Figure **36** were provided by Julia's open-source `Flux.jl` [119, 120] package. The continuous dynamical system simulations in example Section 5.6.2 were performed within Julia's open-source `DynamicalSystems.jl` [76] library.

Acknowledgment

We thank Mario Zanon (IMT Lucca) for providing very useful comments on an initial version of this chapter's content which helped to improve the overall presentation.

Part IV

Conclusion and future work

Chapter 6

Conclusion

In this thesis, we have presented several practical quasi-Newton frameworks and algorithms for solving large-scale optimization problems in machine learning and control. Under practical settings, we provided theoretical guarantees on the convergence of the proposed algorithms and demonstrated their effectiveness through numerical experiments, which reveal their competitiveness and superiority over existing state-of-the-art methods. First, in Chapter 1, we provided a general setup of the optimization problems considered and presented a motivation for the proposed algorithms. The main contributions of this thesis were presented in Chapters 2–5, which are organized into two main parts.

In Chapter 2, we proposed GGN-SCORE, a generalized Gauss-Newton algorithm for solving unconstrained regularized minimization problems, where the regularization function is considered to be self-concordant. In this generalized setting, we employed a matrix approximation scheme that significantly reduces the computational overhead associated with the method. Unlike existing techniques that impose self-concordance on the problem's objective function, our analysis involves a less restrictive condition from a practical point of view but similarly benefits from the idea of self-concordance by considering scaled optimization step-lengths that depend on the self-concordant parameter of the regularization function. We proved a quadratic local convergence rate for our method under

certain conditions, and validate its efficiency in numerical experiments that involve both strongly convex problems and the general non-convex problems that arise when training neural networks. In both cases, our method compare favourably against Adam, SGD, and L-BFGS methods in terms of per-iteration convergence speed, as well as some machine learning techniques used in binary classification, in terms of solution quality.

As we highlighted in Chapter 2, (generalized) self-concordant optimization provides very useful tools for implementing and analyzing Newton-type methods for unconstrained problems. This helps to reconcile the geometric properties of Newton-type methods with their implementations, while providing convergence guarantees. In the presence of constraints or nonsmooth terms in the objective functions, it becomes natural to extend these methods via proximal schemes. However, when the (generalized) self-concordant property is uncheckable for the objective functions, these methods are no longer applicable and the convergence guarantees become difficult to prove. In addition to other related computational issues, Chapter 3 addresses this with a self-concordant smoothing notion which combines and synthesizes different regularization/smoothing phenomena, namely: inf-conv regularization, self-concordant regularization (SCORE), and Moreau-Yosida regularization. This approach, leading to two algorithms (`Prox-N-SCORE` and `Prox-GGN-SCORE`), is able to utilize certain properties of generalized self-concordant functions in the selection of adaptive step-lengths and a simple variable-metric in the proximal Newton-type scheme. We proved convergence guarantees for our approach. As demonstrated in numerical simulations, in most cases, our approach compares favourably against other state-of-the-art first- and second-order approaches from the literature.

In Chapter 4, we proved a non-asymptotic guarantee for the optimization of a two-layer neural network with explicit regularization using the generalized Gauss-Newton method. We considered the class of generalized self-concordant regularizers under which we quantified the decay of the objective function throughout the training iterations. The class of

regularizers share certain epigraphical properties with commonly-used neural network regularizers, giving a more general perspective on the global convergence of the Gauss-Newton beyond what, say, ℓ_2 regularizer provides. Numerical simulations revealed several generalization and stability properties of the optimized neural network model with this class of regularizers.

In Chapter 5, we proposed an efficient training approach for RCNNs, a recurrent neural network architecture for data-efficient RL and control. We presented an approximate Newton framework for training the state-space model of DCRNNs with convergence guarantee, through the lens of inexact SQP and numerical linear algebra techniques. The proposed GRL($\hat{m}_r$) algorithm efficiently addresses one of the main shortcomings associated with benchmark BPTT algorithms, viz. requiring excessive number of iterations to converge to the true solution. The method does this by exploiting approximate second-order information about the training data to speed up convergence, with minimal parameters to tune—we provided a detailed insight on the choice of the main parameter ω in our algorithm. Numerical examples have shown the efficiency of our proposed method.

6.1 Future work

The proposed algorithms in this thesis have shown promising results in solving large-scale optimization problems in machine learning and control. However, there are several directions for future work that can be explored to further improve the efficiency and robustness of the proposed algorithms. Some of these directions include:

- **More general constraints**: One of the most important extensions of the proposed algorithms is to consider more general types of constraints in the optimization problems. Specifically, we are considering problems with general (nonlinear) equality and inequality constraints, possibly with nonsmooth regularization terms. One approach is to develop augmented Lagrangian-based methods (and their proximal variants) that can handle these types of constraints. The versatility of the proposed

algorithms in this thesis and that of the augmented Lagrangian methods allow for the development of efficient algorithms that can handle a wide range of constraints. The main difficulty in this case is in the design of non-conservative, problem-independent, and efficient strategies for updating the penalty parameter and the Lagrange multipliers in the augmented Lagrangian methods. Developing such strategies can help to broaden the applicability of the proposed algorithms to a wider range of optimization problems.

- **Adaptive step-size selection**: While the proposed algorithms in this thesis use adaptive step-size selection strategies, there is still room for improvement in the design of these strategies. Future work will focus on developing more sophisticated step-size selection strategies that can use additional information about the optimization problem to adapt to the local geometry of the optimization landscape. This can help to further improve the convergence properties of the algorithms and make them more robust to different types of optimization problems.

- **Adaptive smoothing parameters**: The proposed algorithms in Chapter 3 use fixed smoothing parameters for the regularization terms. This restricts the flexibility of the algorithms, as fixed smoothing parameters work well only under certain conditions. In future research, we will explore efficient strategies for adjusting the smoothing parameters during the optimization process. The use of adaptive smoothing parameters (particularly those that become smaller as the optimization progresses) will help discourage the destruction of certain properties that we aim to preserve in the solution of the optimization problem. This in turn will help to improve the convergence properties of the algorithms.

- **Parallel and distributed optimization**: The proposed algorithms can be extended to exploit parallel and distributed computing environments to speed up the optimization process. One of such use cases is in federated learning. In future work, we will adapt the proposed algorithms to work in a federated learning setting, where the optimization process is performed in parallel on multiple devices, and the results are

aggregated to obtain a global model. This will help to improve the scalability of the algorithms and make them more suitable for large-scale optimization problems.

- **Application to real-world problems**: Beyond the simulated environments and datasets used in this thesis, the proposed algorithms can be applied to real-world optimization problems in machine learning and control. This will help to validate the effectiveness of the algorithms in practical settings and demonstrate their competitiveness with existing state-of-the-art methods. Applying the algorithms to real-world problems will also help to identify potential limitations of the algorithms and guide future research directions.

- **Theoretical analysis**: In future work, the convergence analysis of the proposed algorithms will be extended to cover more general classes of optimization problems. This will help to provide more practical insights into the convergence properties of the algorithms, thereby establishing their reliability in real-world applications.

Bibliography

[1] Jos U Abubakar and AD Adeoye. "Effects of radiative heat and magnetic field on blood flow in an inclined tapered stenosed porous artery". In: *Journal of Taibah University for Science* 14.1 (2020), pp. 77–86.

[2] Adeyemi D Adeoye and Alberto Bemporad. *SC-Reg: Training Overparameterized Neural Networks under Self-Concordant Regularization*. Tech. rep. IMT School for Advanced Studies Lucca, 2021.

[3] Adeyemi D. Adeoye and Alberto Bemporad. "An Inexact Sequential Quadratic Programming Method for Learning and Control of Recurrent Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 36.2 (2025), pp. 2762–2776. DOI: 10.1109/TNNLS.2024.3354855.

[4] Adeyemi D. Adeoye and Alberto Bemporad. "SCORE: approximating curvature information under self-concordant regularization". In: *Computational Optimization and Applications* 86.2 (2023), pp. 599–626.

[5] Adeyemi D. Adeoye and Alberto Bemporad. "Self-concordant Smoothing for Large-Scale Convex Composite Optimization". In: *arXiv preprint arXiv:2309.01781* (2024). Submitted.

[6] Adeyemi D. Adeoye, Philipp Christian Petersen, and Alberto Bemporad. "Regularized Gauss-Newton for Optimizing Overparameterized Neural Networks". In: *arXiv preprint arXiv:2404.14875* (2024). Submitted.

[7] Adeyemi Damilare Adeoye. "Blood Flow in an Inclined Tapered Stenosed Porous Artery under the Influence of Magnetic Field and Heat Transfer". MA thesis. African Institue for Mathematical Sciences, Cameroon, 2018.

[8] Adeyemi Damilare Adeoye and Philipp Petersen. *A Deep Neural Network Optimization Method Via A Traffic Flow Model*. Tech. rep. African Institue for Mathematical Sciences, Rwanda, 2021.

[9] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. "Learning and generalization in overparameterized neural networks, going beyond two layers". In: *Advances in neural information processing systems* 32 (2019).

[10] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A convergence theory for deep learning via over-parameterization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 242–252.

[11] Shun-Ichi Amari. "Natural gradient works efficiently in learning". In: *Neural computation* 10.2 (1998), pp. 251–276.

[12] Galen Andrew and Jianfeng Gao. "Scalable training of ℓ_1-regularized log-linear models". In.

[13] Michael Arbel. "Rethinking Gauss-Newton for learning over-parameterized models". In: *arXiv preprint arXiv:2302.02904* (2023).

[14] Michael Arbel et al. "Kernelized wasserstein natural gradient". In: *arXiv preprint arXiv:1910.09652* (2019).

[15] Larry Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives". In: *Pacific Journal of mathematics* 16.1 (1966), pp. 1–3.

[16] Sanjeev Arora et al. "Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 322–332.

[17] Sanjeev Arora et al. "On exact computation with an infinitely wide neural net". In: *Advances in Neural Information Processing Systems* 32 (2019).

[18] Amir F Atiya and Alexander G Parlos. "New results on recurrent network training: unifying the algorithms and accelerating convergence". In: *IEEE transactions on neural networks* 11.3 (2000), pp. 697–709.

[19] Francis Bach. "Self-concordant analysis for logistic regression". In: *Electronic Journal of Statistics* 4.none (2010), pp. 384–414. DOI: 10.1214/09-EJS521. URL: https://doi.org/10.1214/09-EJS521.

[20] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. "Kernels as features: On kernels, margins, and low-dimensional mappings". In: *Machine Learning* 65 (2006), pp. 79–94.

[21] John GP Barnes. "An algorithm for solving non-linear equations based on the secant method". In: *The Computer Journal* 8.1 (1965), pp. 66–72.

[22] Heinz H Bauschke, Jérôme Bolte, and Marc Teboulle. "A descent lemma beyond Lipschitz gradient continuity: first-order methods revisited and applications". In: *Mathematics of Operations Research* 42.2 (2017), pp. 330–348.

[23] Heinz H Bauschke and Jonathan M Borwein. "Legendre functions and the method of random Bregman projections". In: *Journal of convex analysis* 4.1 (1997), pp. 27–67.

[24] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Vol. 408. Springer, 2011.

[25] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202.

[26] Amir Beck and Marc Teboulle. "Gradient-based algorithms with applications to signal recovery". In: *Convex optimization in signal processing and communications* (2009), pp. 42–88.

[27] Amir Beck and Marc Teboulle. "Smoothing and first order methods: A unified framework". In: *SIAM Journal on Optimization* 22.2 (2012), pp. 557–580.

[28] Stephen Becker and Jalal Fadili. "A quasi-Newton proximal splitting method". In: *Advances in neural information processing systems* 25 (2012).

[29] Stephen Becker, Jalal Fadili, and Peter Ochs. "On quasi-Newton forward-backward splitting: proximal calculus and convergence". In: *SIAM Journal on Optimization* 29.4 (2019), pp. 2445–2481.

[30] Sue Becker and Yann le Cun. "Improving the Convergence of Back-Propagation Learning with Second Order Methods". In: (1988).

[31] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

[32] Alberto Bemporad. "An L-BFGS-B Approach for Linear and Nonlinear System Identification Under ℓ_1 and Group-Lasso Regularization". In: *IEEE Transactions on Automatic Control* (2025).

[33] Alberto Bemporad. "Recurrent neural network training with convex loss and regularization functions by extended Kalman filtering". In: *IEEE Transactions on Automatic Control* (2023).

[34] Alberto Bemporad. "Training recurrent neural networks by sequential least squares and the alternating direction method of multipliers". In: *Automatica* 156 (2023), p. 111183.

[35] Alberto Bemporad et al. "The explicit linear quadratic regulator for constrained systems". In: *Automatica* 38.1 (2002), pp. 3–20.

[36] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[37] Michele Benzi, Gene H Golub, and Jörg Liesen. "Numerical solution of saddle point problems". In: *Acta numerica* 14 (2005), pp. 1–137.

[38] Alberto Bernacchia, Máté Lengyel, and Guillaume Hennequin. "Exact natural gradient in deep linear networks and application to the nonlinear case". In: NIPS. 2019.

[39] Julius Berner et al. "The modern mathematics of deep learning". In: *arXiv preprint arXiv:2105.04026* (2021), pp. 86–114.

[40] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*. Vol. 4. Athena Scientific, 2012.

[41] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98.

[42] Katharina Bieker, Bennet Gebken, and Sebastian Peitz. "On the treatment of optimization problems with l1 penalty terms via multiobjective continuation". In: *arXiv preprint arXiv:2012.07483* (2020).

[43] Joseph-Frédéric Bonnans et al. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[44] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[45] Aleksandar Botev, Hippolyt Ritter, and David Barber. "Practical gauss-newton optimisation for deep learning". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 557–565.

[46] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer. 2010, pp. 177–186.

[47] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *Siam Review* 60.2 (2018), pp. 223–311.

[48] Charles G Broyden. "A class of methods for solving nonlinear simultaneous equations". In: *Mathematics of computation* 19.92 (1965), pp. 577–593.

[49] Charles G Broyden. "Quasi-Newton methods and their application to function minimisation". In: *Mathematics of Computation* 21.99 (1967), pp. 368–381.

[50] Charles George Broyden. "The convergence of a class of double-rank minimization algorithms 1. General considerations". In: *IMA Journal of Applied Mathematics* 6.1, 3 (1970), pp. 76–90.

[51] Sébastien Bubeck and Mark Sellke. "A Universal Law of Robustness via Isoperimetry". In: *arXiv preprint arXiv:2105.12806* (2021).

[52] James V Burke and Tim Hoheisel. "Epi-convergence properties of smoothing by infimal convolution". In: *Set-Valued and Variational Analysis* 25 (2017), pp. 1–23.

[53] James V Burke and Tim Hoheisel. "Epi-convergent smoothing with applications to convex composite functions". In: *SIAM Journal on Optimization* 23.3 (2013), pp. 1457–1479.

[54] Richard H Byrd, Frank E Curtis, and Jorge Nocedal. "An inexact Newton method for nonconvex equality constrained optimization". In: *Mathematical programming* 122.2 (2010), pp. 273–299.

[55] Richard H Byrd, Frank E Curtis, and Jorge Nocedal. "An inexact SQP method for equality constrained optimization". In: *SIAM Journal on Optimization* 19.1 (2008), pp. 351–369.

[56] Richard H Byrd et al. "On the use of stochastic hessian information in optimization methods for machine learning". In: *SIAM Journal on Optimization* 21.3 (2011), pp. 977–995.

[57] Tianle Cai et al. “Gram-gauss-newton method: Learning overparameterized neural networks for regression problems”. In: *arXiv preprint arXiv:1905.11675* (2019).

[58] Wenzhe Cai et al. “Learning a World Model With Multitimescale Memory Augmentation”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[59] Emmanuel J Candès, Justin Romberg, and Terence Tao. “Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information”. In: *IEEE Transactions on information theory* 52.2 (2006), pp. 489–509.

[60] Andrea Caponnetto and Ernesto De Vito. “Optimal rates for the regularized least-squares algorithm”. In: *Foundations of Computational Mathematics* 7 (2007), pp. 331–368.

[61] Yair Carmon et al. “Acceleration with a ball optimization oracle”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19052–19063.

[62] Augustin Cauchy et al. “Méthode générale pour la résolution des systemes d’équations simultanées”. In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pp. 536–538.

[63] Antonin Chambolle. “An algorithm for total variation minimization and applications”. In: *Journal of Mathematical imaging and vision* 20 (2004), pp. 89–97.

[64] Lai-Wan Chan and Chi-Cheong Szeto. “Training recurrent network with block-diagonal approximated Levenberg-Marquardt algorithm”. In: *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. Vol. 3. IEEE. 1999, pp. 1521–1526.

[65] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: a library for support vector machines”. In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), pp. 1–27.

[66] Wing-Fai Chang and Man-Wai Mak. “A conjugate gradient learning algorithm for recurrent neural networks”. In: *Neurocomputing* 24.1-3 (1999), pp. 173–189.

[67] Pierre Charbonnier et al. “Deterministic edge-preserving regularization in computed imaging”. In: *IEEE Transactions on image processing* 6.2 (1997), pp. 298–311.

[68] Pei Chen. “Hessian matrix vs. Gauss–Newton hessian matrix”. In: *SIAM Journal on Numerical Analysis* 49.4 (2011), pp. 1417–1435.

[69] Xi Chen et al. “Graph-structured multi-task regression and an efficient optimization method for general fused lasso”. In: *arXiv preprint arXiv:1005.3579* (2010).

[70] Xi Chen et al. “Smoothing proximal gradient method for general structured sparse regression”. In: (2012).

[71] Xiaojun Chen. “Smoothing methods for nonsmooth, nonconvex minimization”. In: *Mathematical programming* 134 (2012), pp. 71–99.

[72] Lenaic Chizat and Francis Bach. “Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 1305–1338.

[73] Lenaic Chizat, Edouard Oyallon, and Francis Bach. “On lazy training in differentiable programming”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[74] Rémi Choquet and Jocelyne Erhel. “Some convergence results for the Newton-GMRES algorithm”. PhD thesis. INRIA, 1993.

[75] Amit Daniely. “SGD learns the conjugate kernel class of the network”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[76] George Datseris. “DynamicalSystems.jl: A Julia software library for chaos and nonlinear dynamics”. In: *Journal of Open Source Software* 3.23 (Mar. 2018), p. 598. DOI: 10.21105/joss.00598. URL: https://doi.org/10.21105/joss.00598.

[77] Yann N Dauphin et al. “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization”. In: *Advances in neural information processing systems* 27 (2014).

[78] WC Davidon. “Variable metric method for minimization”. In: *AEC Research and Development Report ANL-5990* (1959). DOI: 10.2172%2F4252678.

[79] A.R. De Pierro and A.N. Iusem. “A relaxed version of Bregman’s method for convex programming”. In: *Journal of Optimization Theory and Applications* 51 (1986), pp. 421–440.

[80] Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. “Inexact Newton methods”. In: *SIAM Journal on Numerical analysis* 19.2 (1982), pp. 400–408.

[81] Nikita Doikov and Yurii Nesterov. “Gradient Regularization of Newton Method with Bregman Distances”. In: *arXiv preprint arXiv:2112.02952* (2021).

[82] David L Donoho. “Compressed sensing”. In: *IEEE Transactions on information theory* 52.4 (2006), pp. 1289–1306.

[83] Simon Du et al. “Gradient descent finds global minima of deep neural networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1675–1685.

[84] Simon S Du et al. “Gradient descent provably optimizes over-parameterized neural networks”. In: *arXiv preprint arXiv:1810.02054* (2018).

[85] Yan Duan et al. “Benchmarking deep reinforcement learning for continuous control”. In: *International conference on machine learning*. PMLR. 2016, pp. 1329–1338.

[86] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).

[87] William Jolly Duncan. “LXXVIII. Some devices for the solution of large sets of simultaneous linear equations: With an appendix on the reciprocation of partitioned matrices”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 35.249 (1944), pp. 660–670.

[88] Peter K Dunn and Gordon K Smyth. *Generalized linear models with examples in R*. U.S.A.: Springer, 2018.

[89] Helen Durand, Matthew Ellis, and Panagiotis D Christofides. “Economic model predictive control designs for input rate-of-change constraint handling and guaranteed economic performance”. In: *Computers & Chemical Engineering* 92 (2016), pp. 18–36.

[90] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning”. In: *Neural Networks* 107 (2018), pp. 3–11.

[91] Murat A Erdogdu and Andrea Montanari. “Convergence rates of sub-sampled newton methods”. In: *arXiv preprint arXiv:1508.02810* (2015).

[92] Tolga Ergen and Suleyman Serdar Kozat. “Efficient online learning algorithms based on LSTM neural networks”. In: *IEEE transactions on neural networks and learning systems* 29.8 (2017), pp. 3772–3783.

[93] Brecht Evens et al. “Neural Network Training as an Optimal Control Problem:—An Augmented Lagrangian Approach—”. In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 5136–5143.

[94] Roger Fletcher. “A new approach to variable metric algorithms”. In: *The computer journal* 13.3 (1970), pp. 317–322.

[95] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[96] Jaroslav M Fowkes, Nicholas IM Gould, and Chris L Farmer. “A branch and bound algorithm for the global optimization of Hessian Lipschitz continuous functions”. In: *Journal of Global Optimization* 56.4 (2013), pp. 1791–1815.

[97] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. “Regularization paths for generalized linear models via coordinate descent”. In: *Journal of statistical software* 33.1 (2010), p. 1.

[98] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. “A note on the group lasso and a sparse group lasso”. In: *arXiv preprint arXiv:1001.0736* (2010).

[99] Xingang Fu et al. “Training recurrent neural networks with the Levenberg–Marquardt algorithm for optimal control of a grid-connected converter”. In: *IEEE transactions on neural networks and learning systems* 26.9 (2015), pp. 1900–1912.

[100] Jezabel R Garcia et al. “Fisher-Legendre (FishLeg) optimization of deep neural networks”. In: *The Eleventh International Conference on Learning Representations*. 2022.

[101] Donald Goldfarb. “A family of variable-metric methods derived by variational means”. In: *Mathematics of computation* 24.109 (1970), pp. 23–26.

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

[103] Robert Gower et al. “RSN: randomized subspace Newton”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[104] I. Grattan-Guinness. *Landmark Writings in Western Mathematics 1640-1940*. Elsevier Science, 2005. ISBN: 9780080457444. URL: https://books.google.it/books?id=UdGBy8iLpocC.

[105] Suriya Gunasekar et al. “Characterizing implicit bias in terms of optimization geometry”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1832–1841.

[106] Louis Guttman. “Enlargement methods for computing the inverse matrix”. In: *The annals of mathematical statistics* (1946), pp. 336–343.

[107] David Ha and Jürgen Schmidhuber. “Recurrent world models facilitate policy evolution”. In: *Advances in neural information processing systems* 31 (2018).

[108] Martin T Hagan and Mohammad B Menhaj. “Training feedforward networks with the Marquardt algorithm”. In: *IEEE transactions on Neural Networks* 5.6 (1994), pp. 989–993.

[109] Stephen Hanson and Lorien Pratt. “Comparing biases for minimal network construction with back-propagation”. In: *Advances in Neural Information Processing Systems* 1 (1988).

[110] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge, United Kingdom: Cambridge University Press, ISBN: 0521540518, 2004.

[111] S.S. Haykin. *Neural Networks and Learning Machines*. Pearson International Edition. Pearson, 2009. ISBN: 9780131293762. URL: https://books.google.it/books?id=KCwWOAAACAAJ.

[112] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[113] Verena Heidrich-Meisner and Christian Igel. “Variable metric reinforcement learning methods applied to the noisy mountain car problem”. In: *European Workshop on Reinforcement Learning*. Springer. 2008, pp. 136–150.

[114] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. New York, U.S.A.: SIAM, 2002.

[115] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Grundlehren Text Editions. Springer Science & Business Media, 2004.

[116] Sepp Hochreiter. “Recurrent neural net learning and vanishing gradient”. In: *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems* 6.2 (1998), pp. 107–116.

[117] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[118] Yasutoshi Ida, Yasuhiro Fujiwara, and Hisashi Kashima. "Fast sparse group lasso". In: *Advances in neural information processing systems* 32 (2019).

[119] Michael Innes et al. "Fashionable Modelling with Flux". In: *CoRR* abs/1811.01457 (2018). arXiv: 1811.01457. URL: https://arxiv.org/abs/1811.01457.

[120] Mike Innes. "Flux: Elegant Machine Learning with Julia". In: *Journal of Open Source Software* (2018). DOI: 10.21105/joss.00602.

[121] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in Neural Information Processing Systems* 31 (2018).

[122] Ziwei Ji and Matus Telgarsky. "The implicit bias of gradient descent on nonseparable data". In: *Conference on Learning Theory*. PMLR. 2019, pp. 1772–1798.

[123] Rie Johnson and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in neural information processing systems* 26 (2013), pp. 315–323.

[124] Ryo Karakida and Kazuki Osawa. "Understanding approximate fisher information for fast convergence of natural gradient descent in wide neural networks". In: *arXiv preprint arXiv:2010.00879* (2020).

[125] Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. "Global linear convergence of Newton's method without strong-convexity or Lipschitz gradients". In: *arXiv preprint arXiv:1806.00413* (2018).

[126] Anna Kerekes, Anna Mészáros, and Ferenc Huszár. "Depth Without the Magic: Inductive Bias of Natural Gradient Descent". In: *arXiv preprint arXiv:2111.11542* (2021).

[127] Seyoung Kim, Kyung-Ah Sohn, and Eric P Xing. "A multivariate regression approach to association analysis of a quantitative trait network". In: *Bioinformatics* 25.12 (2009), pp. i204–i212.

[128] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[129] Mikalai Korbit et al. “Exact Gauss-Newton Optimization for Training Deep Neural Networks”. In: *arXiv preprint arXiv:2405.14402* (2024). Submitted.

[130] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[131] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[132] Yann LeCun et al. “A theoretical framework for back-propagation”. In: *Proceedings of the 1988 connectionist models summer school*. Vol. 1. 1988, pp. 21–28.

[133] Jason D Lee, Yuekai Sun, and Michael A Saunders. “Proximal Newton-type methods for minimizing composite functions”. In: *SIAM Journal on Optimization* 24.3 (2014), pp. 1420–1443.

[134] Kenneth Levenberg. “A method for the solution of certain nonlinear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[135] Qianxiao Li, Long Chen, Cheng Tai, et al. “Maximum principle based algorithms for deep learning”. In: *arXiv preprint arXiv:1710.09513* (2017).

[136] Xudong Li, Defeng Sun, and Kim-Chuan Toh. “A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems”. In: *SIAM Journal on Optimization* 28.1 (2018), pp. 433–458.

[137] Yuanzhi Li and Yingyu Liang. “Learning overparameterized neural networks via stochastic gradient descent on structured data”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[138] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. “Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations”. In: *Conference On Learning Theory*. PMLR. 2018, pp. 2–47.

[139] Zhan Li and Shuai Li. “Neural network model-based control for manipulator: An autoencoder perspective”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[140] Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. “What Happens after SGD Reaches Zero Loss?–A Mathematical Framework”. In: *arXiv preprint arXiv:2110.06914* (2021).

[141] Tengyuan Liang and Alexander Rakhlin. "Just interpolate: Kernel "Ridgeless" regression can generalize". In: *The Annals of Statistics* 48.3 (2020), pp. 1329–1347. DOI: 10.1214/19-AOS1849. URL: https://doi.org/10.1214/19-AOS1849.

[142] Pierre-Louis Lions and Bertrand Mercier. "Splitting algorithms for the sum of two nonlinear operators". In: *SIAM Journal on Numerical Analysis* 16.6 (1979), pp. 964–979.

[143] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[144] Mitchell M Livstone, Jay A Farrell, and Walter L Baker. "A computationally efficient algorithm for training recurrent connectionist networks". In: *1992 American Control Conference*. IEEE. 1992, pp. 555–561.

[145] Yves Lucet. "Faster than the fast Legendre transform, the linear-time Legendre transform". In: *Numerical Algorithms* 16 (1997), pp. 171–185.

[146] Donald W Marquardt. "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[147] James Martens et al. "Deep learning via hessian-free optimization." In: *ICML*. Vol. 27. 2010, pp. 735–742.

[148] James Martens. "New insights and perspectives on the natural gradient method". In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5776–5851.

[149] James Martens and Roger Grosse. "Optimizing neural networks with kronecker-factored approximate curvature". In: *International conference on machine learning*. PMLR. 2015, pp. 2408–2417.

[150] James Martens and Ilya Sutskever. "Learning recurrent neural networks with hessian-free optimization". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 1033–1040.

[151] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[152] Naoki Marumo, Takayuki Okuno, and Akiko Takeda. "Constrained Levenberg-Marquardt method with global complexity bound". In: *arXiv preprint arXiv:2004.08259* (2020).

[153] Si Yi Meng et al. "Fast and furious convergence: Stochastic second order methods under interpolation". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 1375–1386.

[154] Derrick Mirikitani and Nikolay Nikolaev. "Recursive Bayesian Levenberg-Marquardt training of recurrent neural networks". In: *2007 International Joint Conference on Neural Networks*. IEEE. 2007, pp. 282–287.

[155] Konstantin Mishchenko. "Regularized Newton Method with Global $O(1/k^2)$ Convergence". In: *arXiv preprint arXiv:2112.02089* (2021).

[156] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[157] Vidya Muthukumar et al. "Harmless interpolation of noisy data in regression". In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (2020), pp. 67–83.

[158] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[159] Eugene Ndiaye et al. "Gap safe screening rules for sparse-group lasso". In: *Advances in neural information processing systems* 29 (2016).

[160] Eugene Ndiaye et al. "Gap safe screening rules for sparsity enforcing penalties". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 4671–4703.

[161] Yu Nesterov. "Smooth minimization of non-smooth functions". In: *Mathematical programming* 103 (2005), pp. 127–152.

[162] Yurii Nesterov. "Barrier subgradient method". In: *Mathematical programming* 127.1 (2011), pp. 31–56.

[163] Yurii Nesterov et al. *Lectures on convex optimization*. Vol. 137. Switzerland: Springer, 2018.

[164] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. Philadelphia: SIAM, 1994.

[165] Yurii Nesterov and Boris T Polyak. "Cubic regularization of Newton method and its global performance". In: *Mathematical Programming* 108.1 (2006), pp. 177–205.

[166] Yurii E Nesterov. "A method for solving the convex programming problem with convergence rate O (1/kˆ 2)". In: *Dokl. akad. nauk Sssr*. Vol. 269. 1983, pp. 543–547.

[167] Jorge Nocedal. "Updating quasi-Newton matrices with limited storage". In: *Mathematics of computation* 35.151 (1980), pp. 773–782.

[168] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[169] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. New York, NY: Springer, 1999.

[170] Alexander Ororbia et al. "Continual learning of recurrent neural networks by locally aligning distributed representations". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.10 (2020), pp. 4267–4278.

[171] Antonio Orvieto et al. "Explicit regularization in overparametrized models via noise injection". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 7265–7287.

[172] Dmitrii M Ostrovskii and Francis Bach. "Finite-sample analysis of M-estimators using self-concordance". In: *Electronic Journal of Statistics* 15.1 (2021), pp. 326–391.

[173] Art B Owen. "Self-concordance for empirical likelihood". In: *Canadian Journal of Statistics* 41.3 (2013), pp. 387–397.

[174] Eliane R Panier and André L Tits. "Avoiding the Maratos effect by means of a nonmonotone line search I. General constrained problems". In: *SIAM Journal on Numerical Analysis* 28.4 (1991), pp. 1183–1195.

[175] Neal Parikh, Stephen Boyd, et al. "Proximal algorithms". In: *Foundations and Trends® in Optimization* 1.3 (2014), pp. 127–239.

[176] Stephen Keith Park. *A transformation method for constrained-function minimization*. Tech. rep. L-10178, NASA-TN-D-7983. NASA Langley Research Center Hampton, VA, United States, 1975.

[177] Razvan Pascanu and Yoshua Bengio. "Revisiting natural gradient for deep networks". In: *arXiv preprint arXiv:1301.3584* (2013).

[178] Adam Paszke et al. “Automatic differentiation in pytorch”. In: (2017).

[179] Michael Patriksson. “A unified framework of descent algorithms for nonlinear programs and variational inequalities”. PhD thesis. Linköping University Linköping, Sweden, 1993.

[180] Michael Patriksson. “Cost approximation: a unified framework of descent algorithms for nonlinear programs”. In: *SIAM Journal on Optimization* 8.2 (1998), pp. 561–582.

[181] Panagiotis Patrinos and Alberto Bemporad. “Proximal Newton methods for convex composite optimization”. In: *52nd IEEE Conference on Decision and Control*. IEEE. 2013, pp. 2358–2363.

[182] Panagiotis Patrinos, Lorenzo Stella, and Alberto Bemporad. “Forward-backward truncated Newton methods for convex composite optimization”. In: *arXiv preprint arXiv:1402.6655* (2014).

[183] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[184] Michael JD Powell. “A fast algorithm for nonlinearly constrained optimization calculations”. In: *Numerical analysis*. Springer, 1978, pp. 144–157.

[185] Michael JD Powell. “Algorithms for nonlinear constraints that use Lagrangian functions”. In: *Mathematical programming* 14.1 (1978), pp. 224–248.

[186] Michael JD Powell. “The convergence of variable metric methods for nonlinearly constrained optimization calculations”. In: *Nonlinear programming 3*. Elsevier, 1978, pp. 27–63.

[187] Gintaras V Puskorius and Lee A Feldkamp. “Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks”. In: *IEEE Transactions on neural networks* 5.2 (1994), pp. 279–297.

[188] Ning Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural networks* 12.1 (1999), pp. 145–151.

[189] Ali Rahimi and Benjamin Recht. “Random features for large-scale kernel machines”. In: *Advances in Neural Information Processing Systems* 20 (2007).

[190] Anant Raj and Francis Bach. “Explicit regularization of stochastic gradient methods through duality”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1882–1890.

[191] Yi Ren and Donald Goldfarb. "Efficient subsampled Gauss-Newton and natural gradient methods for training neural networks". In: *arXiv preprint arXiv:1906.02353* (2019).

[192] J. Revels, M. Lubin, and T. Papamarkou. "Forward-Mode Automatic Differentiation in Julia". In: *arXiv:1607.07892 [cs.MS]* (2016). URL: https://arxiv.org/abs/1607.07892.

[193] Stefan Richter, Colin N Jones, and Manfred Morari. "Real-time input-constrained MPC using fast gradient methods". In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE. 2009, pp. 7387–7393.

[194] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[195] R Tyrrell Rockafellar. "Monotone operators and the proximal point algorithm". In: *SIAM journal on control and optimization* 14.5 (1976), pp. 877–898.

[196] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*. Vol. 317. Springer Science & Business Media, 2009.

[197] Joseph D Romano et al. "PMLB v1.0: an open source dataset collection for benchmarking machine learning methods". In: *arXiv preprint arXiv:2012.00058v2* (2021).

[198] Farbod Roosta-Khorasani and Michael W Mahoney. "Sub-sampled newton methods ii: Local convergence rates". In: *arXiv preprint arXiv:1601.04738* (2016).

[199] Nicolas Roux, Pierre-Antoine Manzagol, and Yoshua Bengio. "Topmoumoute online natural gradient algorithm". In: *Advances in Neural Information Processing Systems* 20 (2007).

[200] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[201] Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. "Less is more: Nyström computational regularization". In: *Advances in Neural Information Processing Systems* 28 (2015).

[202] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[203] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[204] Youcef Saad and Martin H Schultz. "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems". In: *SIAM Journal on scientific and statistical computing* 7.3 (1986), pp. 856–869.

[205] Euripedes P dos Santos and Fernando J Von Zuben. "Improved second-order training algorithms for globally and partially recurrent neural networks". In: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. Vol. 3. IEEE. 1999, pp. 1501–1506.

[206] Anton Maximilian Schaefer, Steffen Udluft, and Hans-Georg Zimmermann. "A recurrent control neural network for data efficient reinforcement learning". In: *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE. 2007, pp. 151–157.

[207] Anton Maximilian Schäfer. "Reinforcement learning with recurrent neural networks". PhD thesis. Osnabrück, Germany: Universität Osnabrück, Oct. 2008.

[208] Jiirgen Schmidhuber. *Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments*. Tech. rep. TR FKI-126-90. Department of Computer Science, Technical University of Munich, 1990.

[209] Jürgen Schmidhuber. "An on-line algorithm for dynamic reinforcement learning and planning in reactive environments". In: *1990 IJCNN international joint conference on neural networks*. IEEE. 1990, pp. 253–258.

[210] Jürgen Schmidhuber. "Reinforcement learning in Markovian and non-Markovian environments". In: *Advances in neural information processing systems* 3 (1990).

[211] Mark Schmidt, Glenn Fung, and Rmer Rosales. "Fast optimization methods for l1 regularization: A comparative study and two new approaches". In: *European Conference on Machine Learning*. Springer. 2007, pp. 286–297.

[212] Nicol N Schraudolph. "Fast curvature matrix-vector products for second-order gradient descent". In: *Neural computation* 14.7 (2002), pp. 1723–1738.

[213] Shayle R Searle. *Matrix algebra useful for statistics*. United States: John Wiley & Sons, 1982.

[214] Ivan W Selesnick and Ilker Bayram. "Sparse signal estimation by maximally sparse convex optimization". In: *IEEE Transactions on Signal Processing* 62.5 (2014), pp. 1078–1092.

[215] Dongjing Shan et al. "DRRNets: Dynamic Recurrent Routing via Low-Rank Regularization in Recurrent Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[216] David F Shanno. "Conditioning of quasi-Newton methods for function minimization". In: *Mathematics of computation* 24.111 (1970), pp. 647–656.

[217] Noah Simon et al. "A sparse-group lasso". In: *Journal of computational and graphical statistics* 22.2 (2013), pp. 231–245.

[218] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[219] Lorenzo Stella, Andreas Themelis, and Panagiotis Patrinos. "Forward–backward quasi-Newton methods for nonsmooth optimization problems". In: *Computational Optimization and Applications* 67.3 (2017), pp. 443–487.

[220] Lorenzo Stella et al. "A simple and efficient algorithm for nonlinear model predictive control". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 1939–1944.

[221] Thomas Strömberg. "A study of the operation of infimal convolution". PhD thesis. Luleå Tekniska Universitet, 1994.

[222] Defeng Sun. "The strong second-order sufficient condition and constraint nondegeneracy in nonlinear semidefinite programming and their implications". In: *Mathematics of Operations Research* 31.4 (2006), pp. 761–776.

[223] Tianxiao Sun and Quoc Tran-Dinh. "Generalized self-concordant functions: a recipe for Newton-type methods". In: *Mathematical Programming* 178.1 (2019), pp. 145–213.

[224] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[225] Salma Tarmoun et al. "Understanding the dynamics of gradient flow in overparameterized linear models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10153–10161.

[226] Andreas Themelis, Lorenzo Stella, and Panagiotis Patrinos. "Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone linesearch algorithms". In: *SIAM Journal on Optimization* 28.3 (2018), pp. 2274–2303.

[227] Robert Tibshirani et al. "Strong rules for discarding predictors in lasso-type problems". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 74.2 (2012), pp. 245–266.

[228] Quoc Tran-Dinh, Anastasios Kyrillidis, and Volkan Cevher. "Composite self-concordant minimization". In: *J. Mach. Learn. Res.* 16.1 (2015), pp. 371–416.

[229] Quoc Tran-Dinh, Yen-Huan Li, and Volkan Cevher. "Barrier smoothing for nonsmooth convex minimization". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 1503–1507.

[230] Paul Tseng. "On accelerated proximal gradient methods for convex-concave optimization". In: *submitted to SIAM Journal on Optimization* 2.3 (2008). URL: https://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf.

[231] Anne Van Mulders et al. "Two nonlinear optimization methods for black box identification compared". In: *Automatica* 46.10 (2010), pp. 1675–1681.

[232] Vladimir Vapnik and Alexey Chervonenkis. *Theory of pattern recognition*. 1974.

[233] AA Vartak, Michael Georgiopoulos, and Georgios C Anagnostopoulos. "On-line Gauss–Newton-based learning for fully recurrent neural networks". In: *Nonlinear Analysis: Theory, Methods & Applications* 63.5-7 (2005), e867–e876.

[234] Oriol Vinyals and Daniel Povey. "Krylov subspace descent for deep learning". In: *Artificial Intelligence and Statistics*. PMLR. 2012, pp. 1261–1268.

[235] Jie Wang and Jieping Ye. "Two-layer feature reduction for sparse-group lasso via decomposition of convex sets". In: *Advances in Neural Information Processing Systems* 27 (2014).

[236] Sheng-De Wang, Te-Son Kuo, and Chen-Fa Hsu. "Trace bounds on the solution of the algebraic matrix Riccati and Lyapunov equation". In: *IEEE Transactions on Automatic Control* 31.7 (1986), pp. 654–656.

[237] Xiaoyu Wang and Yong Huang. "Convergence study in extended Kalman filter-based training of recurrent neural networks". In: *IEEE Transactions on Neural Networks* 22.4 (2011), pp. 588–600.

[238] Colin Wei et al. "Regularization matters: Generalization and optimization of neural nets vs their induced kernel". In: *Advances in Neural Information Processing Systems* 32 (2019).

[239] Ronald J Williams. "Training recurrent networks using the extended Kalman filter". In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. Vol. 4. IEEE. 1992, pp. 241–246.

[240] Adrian Wills and Thomas Schön. "Stochastic quasi-Newton with adaptive step lengths for large-scale problems". In: *arXiv preprint arXiv:1802.04310* (2018).

[241] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[242] Greg Yang. "Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation". In: *arXiv preprint arXiv:1902.04760* (2019).

[243] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. "On early stopping in gradient descent learning". In: *Constructive Approximation* 26 (2007), pp. 289–315.

[244] Zhewei Yao et al. "ADAHESSIAN: An adaptive second order optimizer for machine learning". In: *proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 10665–10673.

[245] Haishan Ye, Luo Luo, and Zhihua Zhang. "Nesterov's Acceleration for Approximate Newton." In: *J. Mach. Learn. Res.* 21 (2020), pp. 142–1.

[246] Yao-Liang Yu. "On decomposing the proximal map". In: *Advances in neural information processing systems* 26 (2013).

[247] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[248] Chiyuan Zhang et al. "Understanding deep learning (still) requires rethinking generalization". In: *Communications of the ACM* 64.3 (2021), pp. 107–115.

[249] Guodong Zhang, James Martens, and Roger Grosse. "Fast convergence of natural gradient descent for overparameterized neural networks". In: *arXiv preprint arXiv:1905.10961* (2019).

[250] Yangjing Zhang et al. "An efficient Hessian based algorithm for solving large-scale sparse group Lasso problems". In: *Mathematical Programming* 179 (2020), pp. 223–263.

[251] Yiying Zhang. "Neural network algorithm with reinforcement learning for parameters extraction of photovoltaic models". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[252] Hans-Georg Zimmermann et al. "Identification and forecasting of large dynamical systems by dynamical consistent neural networks". In: *New Directions in Statistical Signal Processing: From Systems to Brain* (2006), pp. 203–242.

[253] Difan Zou et al. "Stochastic gradient descent optimizes over-parameterized deep relu networks". In: *arXiv preprint arXiv:1811.08888* (2018).